Tripal Documentation

Release 7.x-3.x

Stephen Ficklin, Lacey Sanderson, Bradford Condon et al

Sep 11, 2023

Contents:

CHAPTER 1

Tripal v3 User's Guide

Note: Looking for the Tripal v2 User's Guide?

1.1 What Is Tripal?

Tripal is a toolkit that facilitates construction of online genomic, genetic, breeding or other biological databases. It uses Drupal, a popular Content Management System (CMS), and by default integrates with Chado, a community-derived relational database schema for storage of biological data. However, Tripal v3 is designed such that it can support other data storage solutions, including no-SQL options. The goal of Tripal is to simplify construction of a community genomics, genetics or biological website to enable individual labs or research communities to construct a high-quality, standards-based website for data sharing and collaboration. A secondary goal is to support community development.

Tripal is open-source and provides an Application Program Interface (API). This allows anyone to customize or create their own tools and extensions. These tools and extensions can in turn be shared with others. Thus, anyone who adopts Tripal will use common tools and community standards, and can participate with a larger distributed group of developers all using the same infrastructure.

1.2 Install Tripal

1.2.1 Pre-planning

1.2.1.1 IT Infrastructure

Tripal requires a server with adequate resources to handle the expected load and systems administration skills to get the machine up and running, applications installed and everything properly secure. Tripal requires a PostgreSQL databases server, Apache (or equivalent) web server, PHP5 and several configuration options to make it all work. However, once these prerequisites are met, working with Drupal and Tripal are quite easy.

There are several ways you could setup a Tripal site:

- **Option #1** In-house dedicated servers: You may have access to servers in your own department or group which you have administrative control and wish to install Tripal on these.
- **Option #2** Institutional IT support: Your institution may provide IT servers and would support your efforts to install a website with database backend.
- **Option #3** Commercial web-hosting: If options #1 and #2 are not available to you, commercial web-hosting is an affordable option. For large databases you may require a dedicated server. Bluehost.com is a web hosting service that provides hosting compatible with Drupal, Tripal and its dependencies.
- Option #4 In the Cloud: Tripal is a part of the GMOD in the cloud Amazon AWS image created by GMOD. It is also accompanied by other GMOD tools such as GBrowse2, JBrowse, Apollo and WebApollo.

After selection of one of the options above you can arrange your database/webserver in the following ways:

- **Arrangement #1**: The database and web server are housed on a single server. This is the approach taken by this course. It is necessary to gain access to a machine with enough memory (RAM), hard disk speed and space, and processor power to handle both services.
- **Arrangement #2**: The database and web server are housed on different servers. This provides dedicated resources to each service (i.e. web and database).

Selection of an appropriate machine

Databases are typically bottle-necked by RAM and disk speed. Selection of the correct balance of RAM, disk speed, disk size and CPU speed is important and dependent on the size of the data. The best advice is to consult an IT professional who can recommend a server installation tailored for the expected size of your data.

Note: Tripal does require command-line access to the web server with adequate local file storage for loading of large data files. Be sure to check with your service provider to make sure command-line access is possible.

1.2.1.2 Technical Skills

Depending on your needs, you may need additional Technical support. Use the following questions to help determine those needs:

Tripal already supports my data, what personnel do I need to maintain it?

Someone to install/setup the IT infrastructure Someone who understands the data to load it properly

Tripal does not yet support all of my data, but I want to use what's been done and expand on it....?

Someone to install/setup the IT infrastructure Someone who understands the data to load it properly PHP/HTML/CSS/JavaScript programmer(s) to write your custom extensions

1.2.1.3 Development and Production Instances

It is recommended that you have separate development and production instances of Tripal. The staging or development instance allows you to test new functionality, add customization, or test modification or additions to data without disturbing the production instance. The production instance serves content to the rest of the world. Once you are certain that customizations and new functionality will work well on the development instance you can easily re-implement or copy these over to the production site. Sometimes it may take a few trials to load data in the way you want. A development sites lets you take time to test data loading prior to making it public. The development site can be password-protected to only allow access to site administrators, developers or collaborators.

1.2.2 DRUPAL_HOME Variable

An important convention in this document is the use of the $DRUPAL_HOME environment variable. If you are new to UNIX/Linux you can learn about environment variables here. Drupal is a necessary dependency of Tripal. The setup and installation sections describe how to install Drupal. If you follow the instructions exactly as described in this User's Guide you will install Drupal into /var/www/html. However, some may desire to install Drupal elsewhere. To ensure that all command-line examples in this guide can be cut-and-pasted you **must** set the $DRUPAL_HOME variable. You can set the variable in the following way:

```
DRUPAL_HOME=/var/www/html
```

Be sure to change the path /var/www/html to the location where you have installed Drupal. If you have never installed Drupal and you intend on following this guide step-by-step then use the command-line above to get started.

Note: You will have to set the $DRUPAL_HOME environment variable anytime you open a new terminal window.

1.2.3 Server Setup

Before installation of Tripal, a web server must be configured and ready, and Tripal requires the following dependencies:

1. A UNIX-based server (e.g. Ubuntu Linux or CentOS are the most popularly used).
2. Web server software. The Apache web server is most commonly used.
3. PHP version 5.6 or higher (the most recent version is recommended).
4. PostgreSQL 9.3 or higher (9.5 required for Chado 1.2 to 1.3 upgrade)
5. Drush 7 or higher
6. Drupal 7.

Warning: PHP 7.2 is not fully compatible with Drupal.

The following sections provide step-by-step instructions to help setup either an Ubuntu or CentOS system.

1.2.3.1 Server Setup (Ubuntu 18.04 LTS)

Note: Tripal can be installed on multiple UNIX/Linux based systems. These instructions are for just Ubuntu 18.04. However, this guide provides instructions for a several systems. Please choose the one that best suits your needs. If you install on a different platform please consider sharing your notes and experience to add to this guide!

The following instructions are for setup of Tripal on an Ubuntu version 18.04 LTS Desktop edition. Ubuntu v18.04 is a long-term support version of Ubuntu, meaning that Ubuntu guarantees patches and security fixes for up to five years. These instructions do not provide guidance for proper configuration settings for the server to handle load, nor for security settings. Consult the software documentation for proper load handling and security settings before your site is made public.

Ubuntu Installation

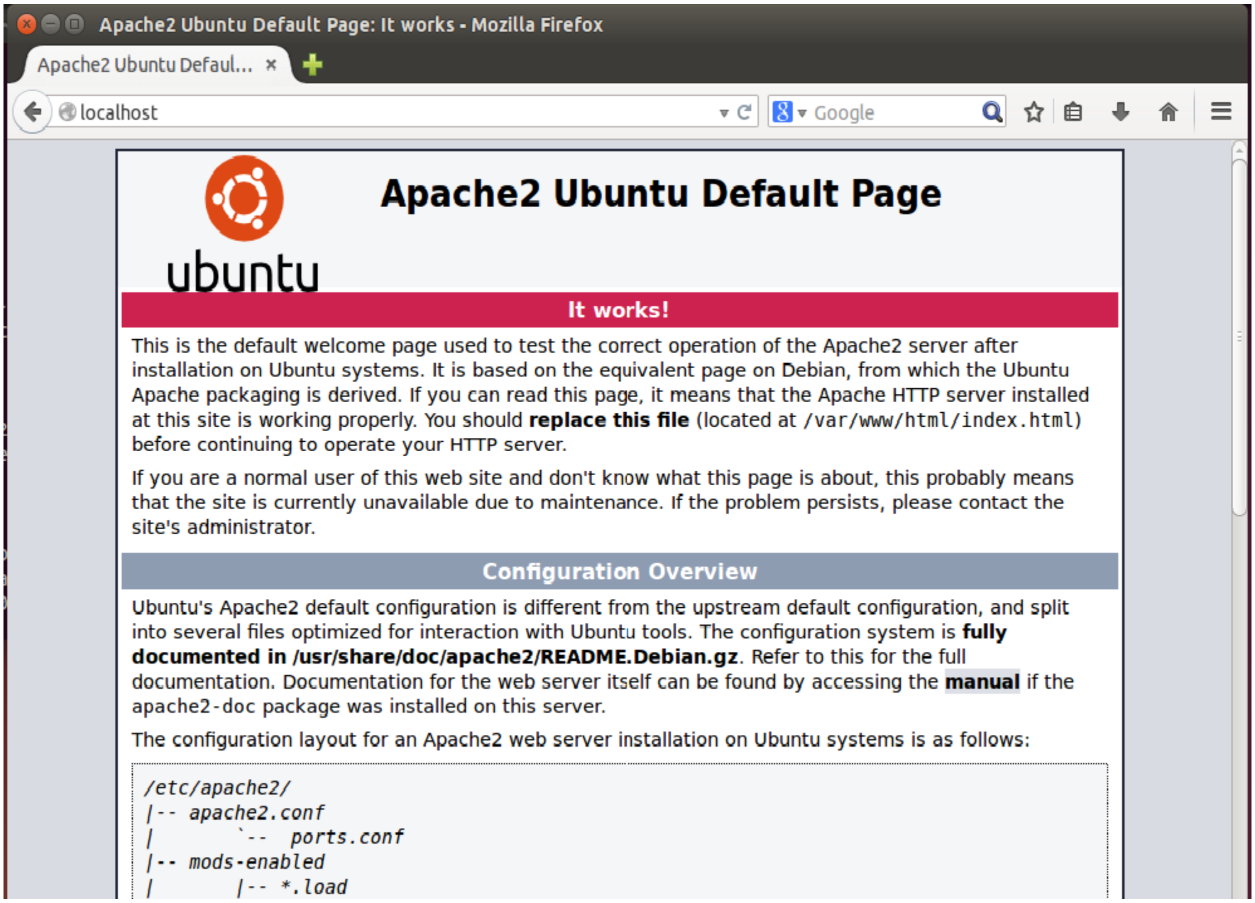
Please follow the online instructions for download and installation of Ubuntu 18.04 Desktop edition. Please be sure to install the 'Desktop' edition rather than the 'Server' edition. The tutorial below will provide the necessary steps to install the server components needed. If you are using this tutorial to test Tripal you can use a virtual machine such as the Oracle VirtualBox or VMWare. The virtual machine allows you to install Ubuntu as a "guest" operating system within your existing "host" operating system (e.g. Windows).

Apache Setup

Apache is the web server software. Ubuntu simplifies the installation of Apache using the 'apt-get' utility. To do so, simply issue the following command:

```
sudo apt-get install apache2
```

Apache should now be installed. On the Ubuntu server, navigate to your new website using this address: http://localhost/. You should see the following page:


Apache2 Ubuntu Default Page: It works - Mozilla Firefox
Apache2 Ubuntu Defaul... ×
localhost
Google
ubuntu

Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
```

Drupal works best with the Apache rewrite module enabled. Within the Ubuntu server, pluggable modules are available in the Apache configuration directory named /etc/apache2/mods-available. Only those modules that are also available in the /etc/apache2/mods-enabled directory will be loaded and used by Apache. Therefore, we want to create a symbolic link for the rewrite module that essentially adds it to the /etc/apache2/mods-enabled directory. To do this execute the following on the command-line:

```
cd /etc/apache2/mods-enabled
sudo ln -s ../mods-available/rewrite.load
```

Next we need to edit the web site configuration file. The configuration file specific for the default website is found here: /etc/apache2/sites-available/000-default.conf. Drupal needs permission to override some default restrictions set by the Apache web server, but it only needs to do so in the directory where it will be installed. By default in Ubuntu 18.04, the web document root is the /var/www/html directory. This is where all web files will be placed. Therefore, we need to adjust the default settings for that directory for Drupal. To do so, edit this file using the 'gedit' graphical text editor that comes with Ubuntu. Because this file is owned by the 'root' user, we must use the 'sudo' command to run 'gedit' with administrative privileges:

```
sudo gedit /etc/apache2/sites-available/000-default.conf
```

Add the following Directory inside the VirtualHost stanza.

```
<Directory /var/www/html>
   Options Indexes FollowSymLinks MultiViews
   AllowOverride All
   Order allow,deny
   allow from all
</Directory>
```

Now restart your Apache again.

```
sudo service apache2 restart
```

Setup PHP

Drupal uses PHP. In Ubuntu 18.04 there are two different instances of PHP that will be installed: a version for apache and another for use on the command-line. To install PHP we can use Ubuntu's apt-get utility.

```
sudo apt-get install php php-dev php-cli libapache2-mod-php php7.2-mbstring
```

You may notice that installing the libapach2-mod-php module will automatically restart the Apache web server which will allow it to parse PHP files. Next, we need a few additional extension modules for PHP that support connection to a PostgreSQL database server, JSON and the GD graphics library:

```
sudo apt-get install php-pgsql php-gd php-xml
```

PHP is now installed both for Apache and for use on the command-line. Before continuing we must make a few changes to the PHP configuration files. PHP will limit the amount of memory that a script can consume. By default this limit is too low the Apache configuration of PHP. For Tripal we need that limit to be higher. To change it, edit the /etc/php5/apache2/php.ini configuration file:

```
sudo gedit /etc/php/7.2/apache2/php.ini
```

Within that file, find the setting titled, memory_limit, and change it to something larger than 128M. For this tutorial we will set the limit to be 2048M, but be sure not to exceed physical memory of your machine:

```
memory_limit = 2048M
```

Now, restart the webserver so that it picks up the new changes to the PHP settings.

```
sudo service apache2 restart
```

PostgreSQL Server

PostgreSQL is the database software that will be used to house both the Drupal and Tripal databases. PostgreSQL can be installed on Ubuntu 18.04 simply by issuing the following command.

```
sudo apt-get install postgresql
```

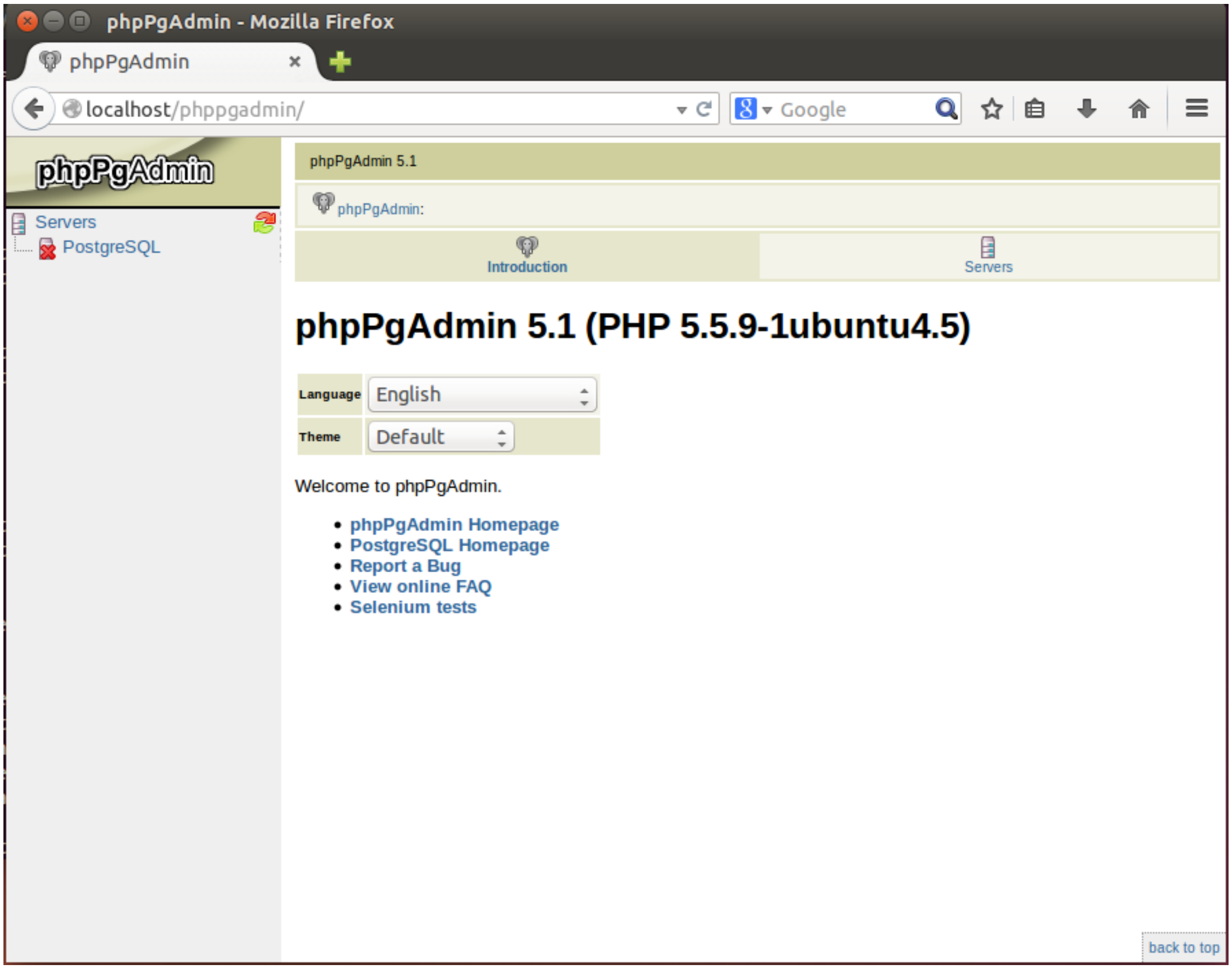
PostgreSQL database server is now installed and setup with default options.

Install phpPgAdmin (Optional)

phpPgAdmin is a web-based utility for easy administration of a PostgreSQL database. PhpPgAdmin is not required for successful operation of Tripal but is very useful. It can be easily installed with an 'apt-get' command:

```
sudo apt-get install phppgadmin
```

Now navigate to the URL http://localhost/phppgadmin and you should see the following:



phpPgAdmin 5.1 (PHP 5.5.9-1ubuntu4.5)

Language: English

Theme: Default

Welcome to phpPgAdmin.

- phpPgAdmin Homepage
- PostgreSQL Homepage
- Report a Bug
- View online FAQ
- Selenium tests

Now, phpPgAdmin is available for access only on the local installation of the machine. It will not be available via remote connections.

1.2.3.2 Server Setup (Ubuntu 16.04 LTS)

Note: Tripal can be installed on multiple UNIX/Linux based systems. These instructions are for just Ubuntu 16.04. However, this guide provides instructions for a several systems. Please choose the one that best suits your needs. If you install on a different platform please consider sharing your notes and experience to add to this guide!

The following instructions are for setup of Tripal on an Ubuntu version 16.04 LTS Desktop edition. Ubuntu v16.04 is a long-term support version of Ubuntu, meaning that Ubuntu guarantees patches and security fixes for up to five years. These instructions do not provide guidance for proper configuration settings for the server to handle load, nor for security settings. Consult the software documentation for proper load handling and security settings before your site is made public.

Ubuntu Installation

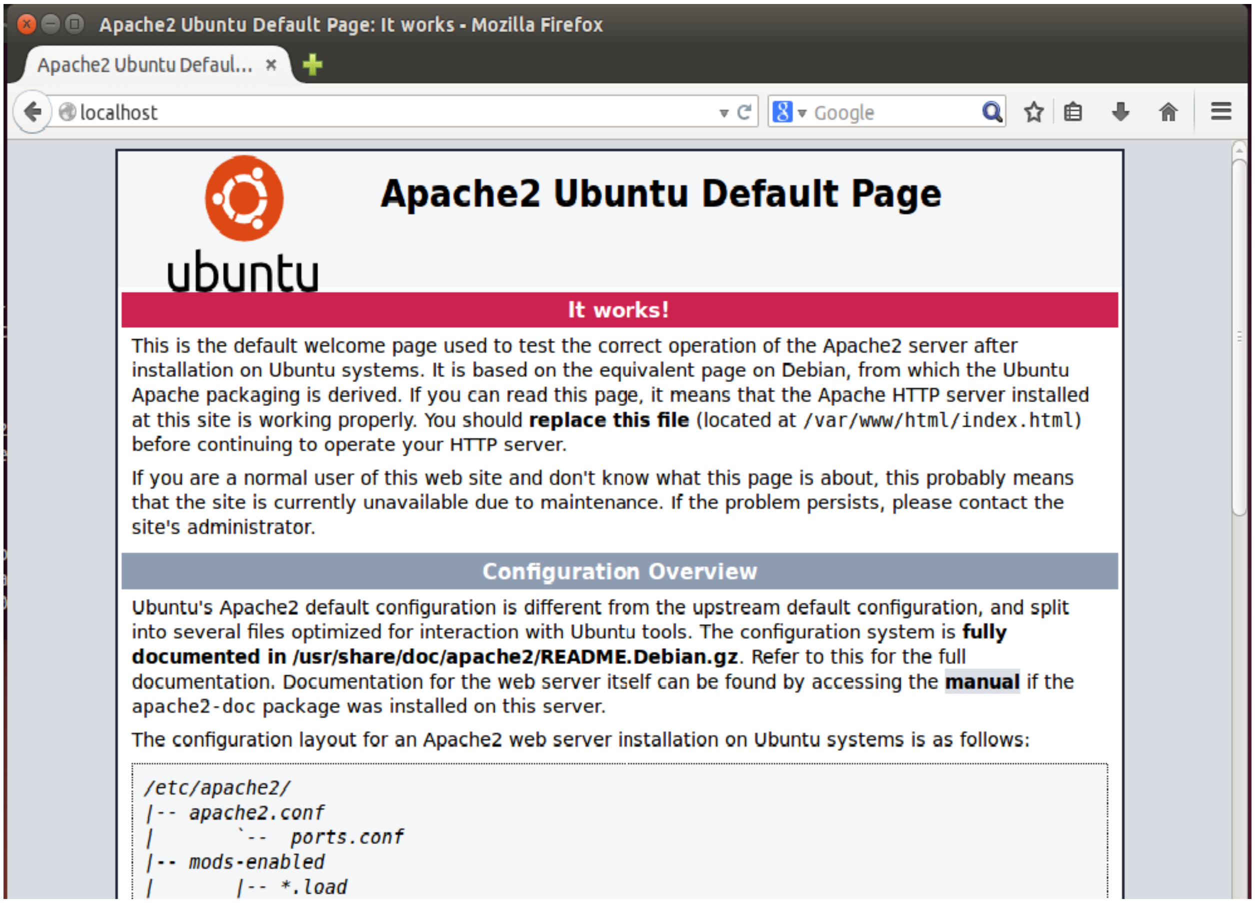
Please follow the online instructions for download and installation of Ubuntu 16.04 Desktop edition. Please be sure to install the 'Desktop' edition rather than the 'Server' edition. The tutorial below will provide the necessary steps to install the server components needed. If you are using this tutorial to test Tripal you can use a virtual machine such as the Oracle VirtualBox or VMWare. The virtual machine allows you to install Ubuntu as a "guest" operating system within your existing "host" operating system (e.g. Windows).

Apache Setup

Apache is the web server software. Ubuntu simplifies the installation of Apache using the 'apt-get' utility. To do so, simply issue the following command:

```
sudo apt-get install apache2
```

Apache should now be installed. On the Ubuntu server, navigate to your new website using this address: http://localhost/. You should see the following page:



Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at /var/www/html/index.html) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
```

Drupal works best with the Apache rewrite module enabled. Within the Ubuntu server, pluggable modules are available in the Apache configuration directory named /etc/apache2/mods-available. Only those modules that are also available in the /etc/apache2/mods-enabled directory will be loaded and used by Apache. Therefore, we want to create a symbolic link for the rewrite module that essentially adds it to the /etc/apache2/mods-enabled directory. To do this execute the following on the command-line:

```
cd /etc/apache2/mods-enabled
sudo ln -s ../mods-available/rewrite.load
```

Next we need to edit the web site configuration file. The configuration file specific for the default website is found here: /etc/apache2/sites-available/000-default.conf. Drupal needs permission to override some default restrictions set by the Apache web server, but it only needs to do so in the directory where it will be installed. By default in Ubuntu 16.04, the web document root is the /var/www/html directory. This is where all web files will be placed. Therefore, we need to adjust the default settings for that directory for Drupal. To do so, edit this file using the 'gedit' graphical text editor that comes with Ubuntu. Because this file is owned by the 'root' user, we must use the 'sudo' command to run 'gedit' with administrative privileges:

```
sudo gedit /etc/apache2/sites-available/000-default.conf
```

Add the following Directory inside the VirtualHost stanza.

```
<Directory /var/www/html>
   Options Indexes FollowSymLinks MultiViews
   AllowOverride All
   Order allow,deny
   allow from all
```

(continues on next page)

(continued from previous page)

```
</Directory>
```

Now restart your Apache again.

```
sudo service apache2 restart
```

Setup PHP

Drupal uses PHP. In Ubuntu 16.04 there are two different instances of PHP that will be installed: a version for apache and another for use on the command-line. To install PHP we can use Ubuntu's apt-get utility.

```
sudo apt-get install php php-dev php-cli libapache2-mod-php
```

You may notice that installing the libapach2-mod-php module will automatically restart the Apache web server which will allow it to parse PHP files. Next, we need a few additional extension modules for PHP that support connection to a PostgreSQL database server, JSON and the GD graphics library:

```
sudo apt-get install php-pgsql php-gd php-xml-parser
```

PHP is now installed both for Apache and for use on the command-line. Before continuing we must make a few changes to the PHP configuration files. PHP will limit the amount of memory that a script can consume. By default this limit is too low the Apache configuration of PHP. For Tripal we need that limit to be higher. To change it, edit the /etc/php5/apache2/php.ini configuration file:

```
sudo gedit /etc/php/7.0/apache2/php.ini
```

Within that file, find the setting titled, memory_limit, and change it to something larger than 128M. For this tutorial we will set the limit to be 2048M, but be sure not to exceed physical memory of your machine:

```
memory_limit = 2048M
```

Now, restart the webserver so that it picks up the new changes to the PHP settings.

```
sudo service apache2 restart
```

PostgreSQL Server

PostgreSQL is the database software that will be used to house both the Drupal and Tripal databases. PostgreSQL can be installed on Ubuntu 16.04 simply by issuing the following command.

```
sudo apt-get install postgresql
```

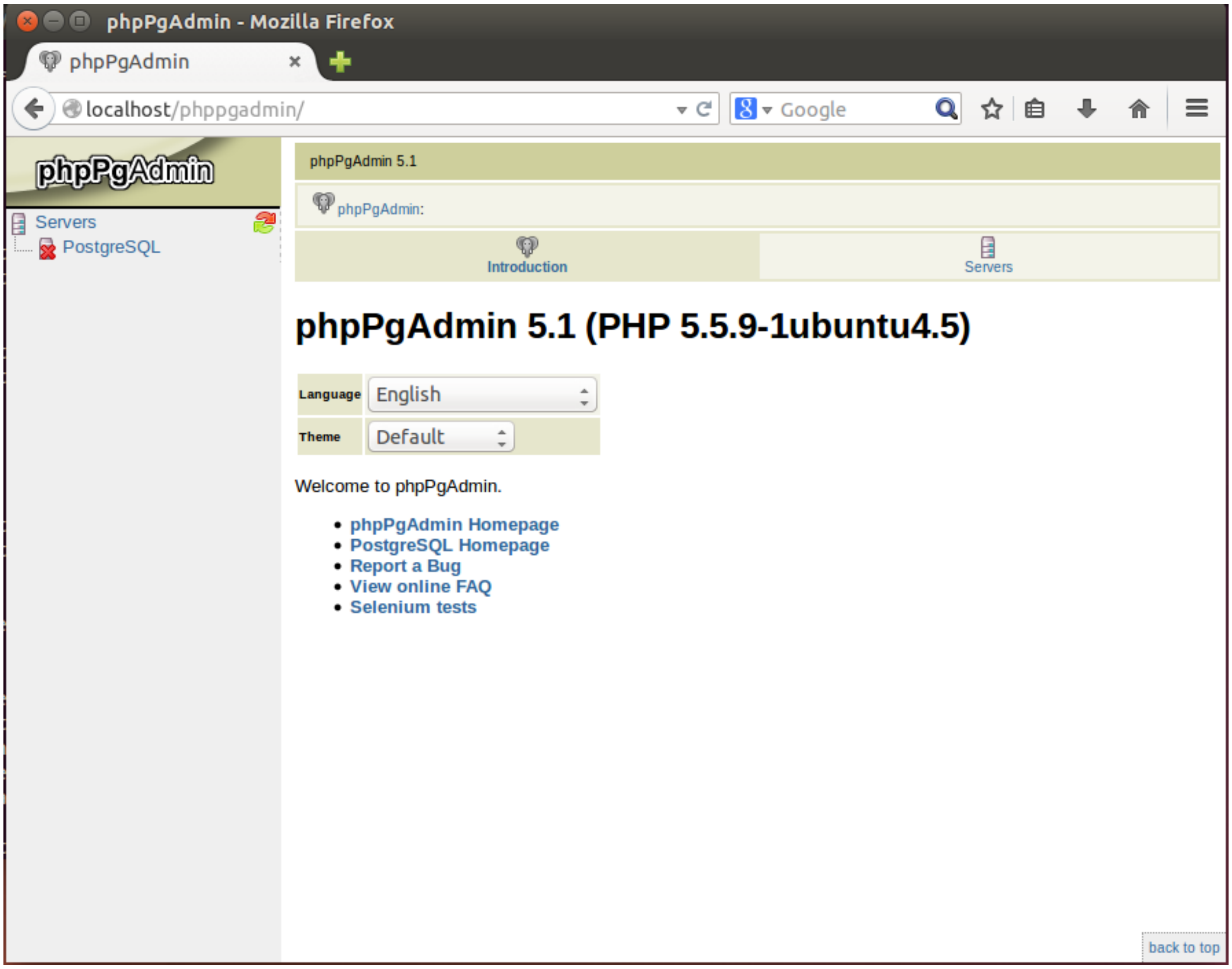
PostgreSQL database server is now installed and setup with default options.

Install phpPgAdmin (Optional)

phpPgAdmin is a web-based utility for easy administration of a PostgreSQL database. PhpPgAdmin is not required for successful operation of Tripal but is very useful. It can be easily installed with an 'apt-get' command:

```
sudo apt-get install phppgadmin
```

Now navigate to the URL http://localhost/phppgadmin and you should see the following:



phpPgAdmin 5.1 (PHP 5.5.9-1ubuntu4.5)

Language	English
Theme	Default

Welcome to phpPgAdmin.

- phpPgAdmin Homepage
- PostgreSQL Homepage
- Report a Bug
- View online FAQ
- Selenium tests

Now, phpPgAdmin is available for access only on the local installation of the machine. It will not be available via remote connections.

1.2.3.3 Server Setup (CentOS 7)

Note: Tripal can be installed on multiple UNIX/Linux based systems. These instructions are for just CentOS 7. However, this guide provides instructions for a several systems. Please choose the one that best suits your needs. If you install on a different platform please consider sharing your notes and experience to add to this guide!

The following sections provide step-by-step instructions for installation of Tripal on a CentOS 7 server. They provide details for setup of the server, including installation of the Apache web server and the PostgreSQL database server; installation of Drupal; installation of Drush, the command-line interface for Drupal; prerequisites for Tripal; and installation of Tripal within Drupal.

CentOS 7 Installation

Please follow the online instructions for download and installation of CentOS 7. The CentOS installation must have the following:

- A user account with administrative access (i.e. sudo privilege).
- A graphical desktop.
- An installed web browser.
- The "Development Tools" should be installed. To install the development tools issue this command:

  ```
  sudo yum groupinstall "Development Tools"
  ```

- A graphical text editor such as 'gedit'. To Install gedit issue this command:

  ```
  sudo yum install gedit
  ```

- The 'wget' utilty for retrieving files from remote web services. To install wget issue this command:

  ```
  sudo yum install wget
  ```

The tutorial below will provide the necessary steps to install the server components needed. If you are using this tutorial to test Tripal you can use a virtual machine such as the Oracle VirtualBox or VMWare. The virtual machine allows you to install Ubuntu as a "guest" operating system within your existing "host" operating system (e.g. Windows).

Apache Installation

Apache is the web server software. CentOS simplifies the installation of Apache using the 'yum' utility. To do so, simply issue the following command:

```
sudo yum install httpd
```

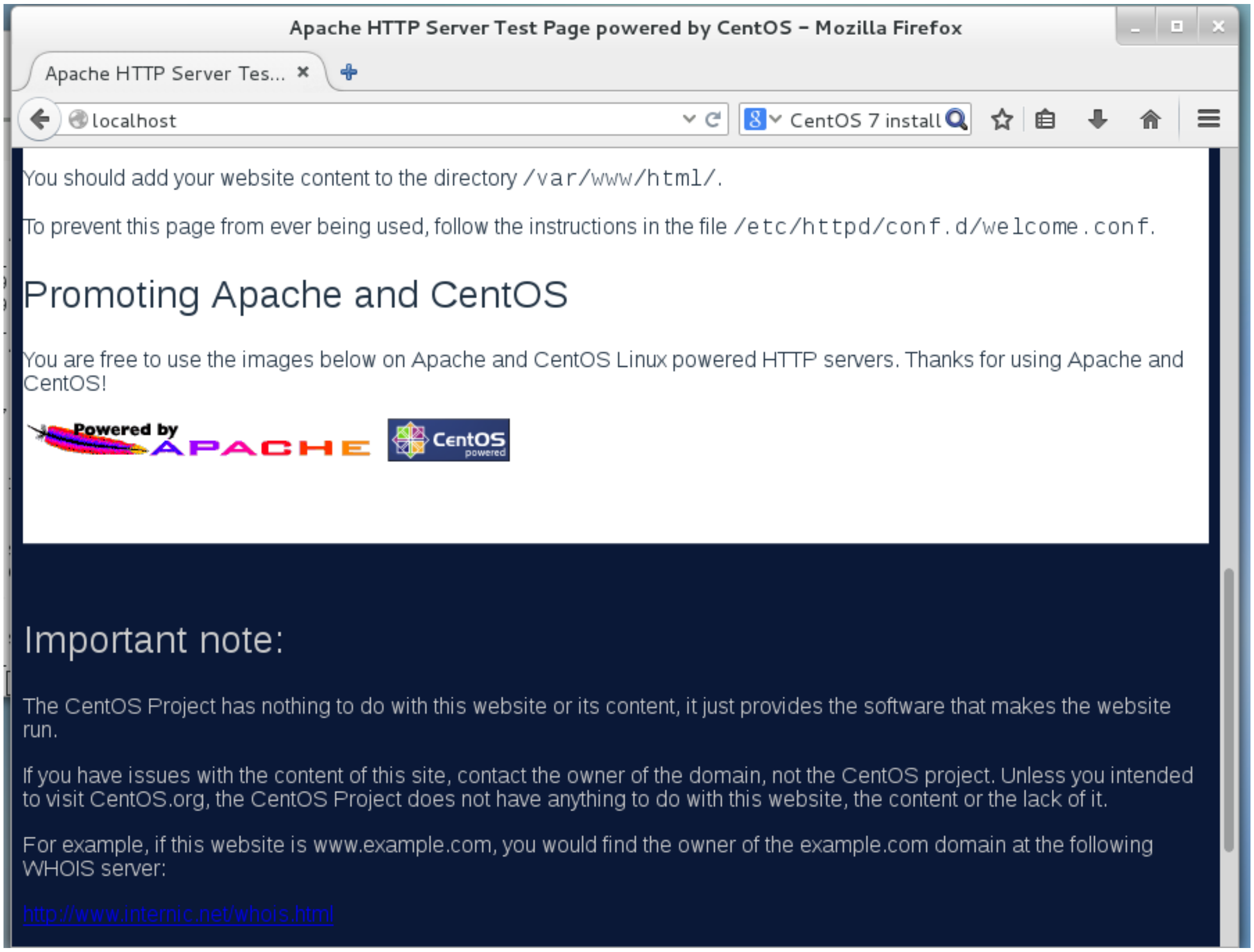
Then start the web server

```
sudo systemctl start httpd.service
```

Execute the following to ensure that the Apache server is started automatically if the server reboots:

```
sudo systemctl enable httpd.service
```

Apache should now be installed. On the server, navigate to your new website using this address: http://localhost/. You should see the following page:


Apache HTTP Server Test Page powered by CentOS - Mozilla Firefox
Apache HTTP Server Tes...
localhost
CentOS 7 install
Powered by APACHE
CentOS powered

You should add your website content to the directory /var/www/html/.

To prevent this page from ever being used, follow the instructions in the file /etc/httpd/conf.d/welcome.conf.

Promoting Apache and CentOS

You are free to use the images below on Apache and CentOS Linux powered HTTP servers. Thanks for using Apache and CentOS!

Important note:

The CentOS Project has nothing to do with this website or its content, it just provides the software that makes the website run.

If you have issues with the content of this site, contact the owner of the domain, not the CentOS project. Unless you intended to visit CentOS.org, the CentOS Project does not have anything to do with this website, the content or the lack of it.

For example, if this website is www.example.com, you would find the owner of the example.com domain at the following WHOIS server:

http://www.internic.net/whois.html

Next we need to edit the web site configuration file. The configuration file specific for the default website is found here: /etc/httpd/conf/httpd.conf. Drupal needs permission to override some default restrictions set by the Apache web server, but it only needs to do so in the directory where it will be installed. By default in CentOS 7, the web document root is the /var/www/html directory. This is where all web files will be placed. Therefore, we need to adjust the default settings for that directory for Drupal. To do so, edit this file using the 'gedit' graphical text editor (should be previously installed). Because this file is owned by the 'root' user, we must use the 'sudo' command to run 'gedit' with administrative privileges:

```
sudo gedit /etc/httpd/conf/httpd.conf
```

Find the Directory stanza for the /var/www/html directory and edit it so that it looks like the following:

```
<Directory /var/www/html>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride All
    Order allow,deny
    allow from all
</Directory>
```

Now restart your Apache again.

```
sudo systemctl restart httpd.service
```

Setup PHP

Drupal uses PHP. In CentOS there are two different instances of PHP that will be installed: a version for apache and another for use on the command-line. To install PHP we can use 'yum' utility:

```
yum install php
```

Next, we need a few additional extension modules for PHP that support connection to a PostgreSQL database server, the GD graphics library and a few others:

```
sudo yum install php-gd php-pgsql php-mbstring php-xml
```

PHP is now installed. Before continuing we must make a few changes to the PHP configuration file. PHP will limit the amount of memory that a script can consume. By default this limit is too low the Apache configuration of PHP. For Tripal we need that limit to be higher. To change it, edit the /etc/php.ini configuration file:

```
sudo gedit /etc/php.ini
```

Within that file, find the setting titled, memory_limit, and change it to something larger than 128M. For this tutorial we will set the limit to be 2048M, but be sure not to exceed physical memory of your machine:

```
memory_limit = 2048M
```

Now, restart the webserver so that it picks up the new changes to the PHP settings.

```
sudo systemctl restart httpd.service
```

PostgreSQL Server

PostgreSQL is the database software that will be used to house both the Drupal and Tripal databases. PostgreSQL can be installed on CentOS 7 simply by issuing the following command.

```
sudo yum install postgresql-server
```

Next, initialize the PostgreSQL database:

```
sudo postgresql-setup initdb
```

PostgreSQL database server is now installed and setup with default options. However, it currently does not allow connections. We want to allow at least connections from the local machine. To do this, edit the /var/lib/pgsql/data/pg_hba.conf file:

```
sudo gedit /var/lib/pgsql/data/pg_hba.conf
```

And set the following to allow connections from the localhost:

```
# IPv4 local connections:
host    all             all             127.0.0.1/32            md5
# IPv6 local connections:
host    all             all             ::1/128                 md5
```

Be sure that any previous "host" lines are commented out by adding a '#' symbol in front. Next, start up the PostgreSQL server

```
systemctl start postgresql.service
```

Finally execute the following to ensure that the PostgreSQL server is started automatically if the server reboots:

```
systemctl enable postgresql.service
```

SE-Linux Configuration

SE-Linux is short for Security Enhanced Linux. It comes installed in RedHat flavors of Linux (such as CentOS). It provides access control mechanisms. If your operating system comes with SE-Linux you will want to change the security context for the web files and associate those with the web server. The following command can be executed to allow that:

```
sudo chcon -R -t httpd_sys_content_rw_t /var/www/html
```

Additionally, we need to allow web scripts and modules to connect to database servers.

```
setsebool -P httpd_can_network_connect_db on
```

Install phpPgAdmin (Optional)

phpPgAdmin is a web-based utility for easy administration of a PostgreSQL database. PhpPgAdmin is not required for successful operation of Tripal but is very useful. First, we need to install the Extra Packages for Enterprise Linux (EPEL) library. This library contains many compatible packages including phpPgAdmin. This can be done with the following command:

```
rpm -Uvh http://mirror.pnl.gov/epel/7/x86_64/e/epel-release-7-5.noarch.rpm
```

Next, phpPgAdmin can be easily installed with a 'yum' command:

```
sudo yum install phpPgAdmin
```

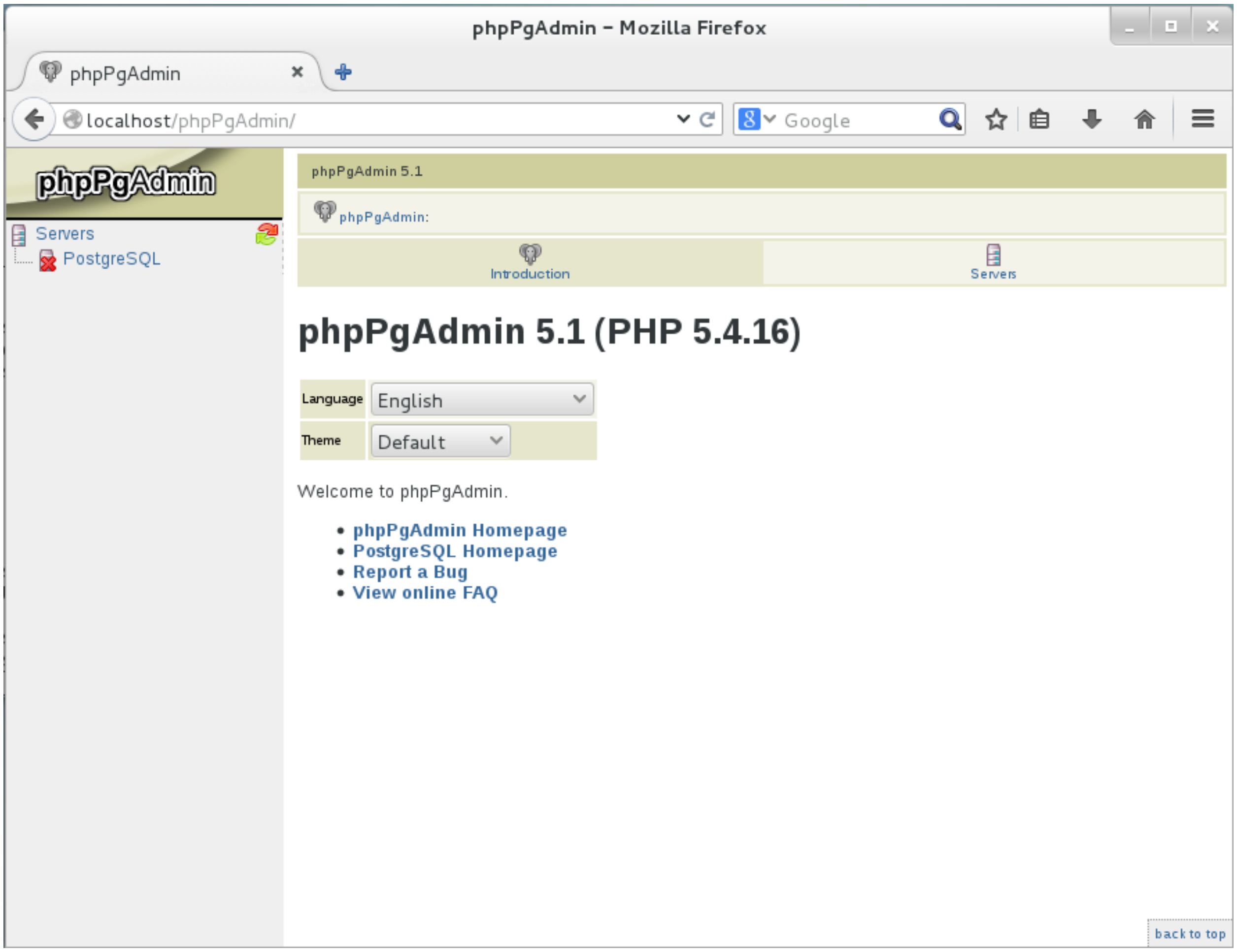
Next, we need to configure phpPgAdmin. To do this, Edit the /etc/phpPgAdmin/config.inc.php file.

```
sudo gedit /etc/phpPgAdmin/config.inc.php
```

Within this file, add "localhost" in the following server parameter:

```
$conf['servers'][0]['host'] = 'localhost';
```

Now navigate to the URL http://localhost/phpPgAdmin and you should see the following:



phpPgAdmin 5.1 (PHP 5.4.16)

Language English

Theme Default

Welcome to phpPgAdmin.

- phpPgAdmin Homepage
- PostgreSQL Homepage
- Report a Bug
- View online FAQ

Now, phpPgAdmin is available for access only on the local installation of the machine. It will not be available via remote connections.

1.2.4 Drush Installation

Drush is a command-line utility that allows for non-graphical access to the Drupal website. You can use it to automatically download and install themes and modules, clear the Drupal cache, upgrade the site and more. Tripal v3 supports Drush. For this tutorial we will use Drush and therefore we want the most recent, Drupal7-compatible version installed: we recommend Drush 8.x (see compatibility chart below.)

Drush Version	Drush Branch	PHP	Compatible Drupal versions
Drush 9	master	5.6+	D8.4+
Drush 8	8.x	5.4.5+	D6, D7, D8.3
Drush 7	7.x	5.3.0+	D6, D7
Drush 6	6.x	5.3.0+	D6, D7
Drush 5	5.x	5.2.0+	D6, D7

As you can see from the above table, the newest version of Drupal which supports Drupal 7 is Drush 8.

1.2.4.1 Install Drush

The official documentation for installing Drush 8 can be found here: https://docs.drush.org/en/8.x/install/.

> **Warning:** Don't accidentally follow the Drupal 8 installation method for your Drupal 7 site! The "site-local" Drush installation won't work for Drupal 7.

1.2.5 Installation Method #1: Rapid Installation

Note: Remember you must set the `$DRUPAL_HOME` environment variable if you want to cut-and-paste the commands below. See *DRUPAL_HOME Variable*

Before installing via the rapid installation process please ensure drush is installed, and the server is setup. Rapid Installation works with Tripal v3.0-rc2 (release candidate 2) and later. If you are using a previous version of Tripal, please proceed to the step-by-step instructions.

1.2.5.1 Database Setup

Before we can install Tripal we must have a database ready for it. In the server setup instructions were provided to set up a PostgreSQL database server. Now, we need to create the Drupal database. To do so we must first become the PostgreSQL user.

```
sudo su - postgres
```

Next, create the new 'drupal' user account. This account will not be a "superuser" nor allowed to create new roles, but should be allowed to create a database.

```
createuser -P drupal
```

When requested, enter an appropriate password. Finally, create the new database:

```
createdb drupal -O drupal
```

We no longer need to be the postgres user so exit

```
exit
```

1.2.5.2 Tripal Installation

Navigate to your Drupal install directory.

```
cd $DRUPAL_HOME
```

Note: Make sure you have write permissions within this directory.

Clone the tripal_install project using the `git` command and move the contents up one level into the web document directory:

```
git clone https://github.com/tripal/tripal_install.git
mv tripal_install/* ./
```

Begin the installation for a generic installation with the following command:

```
drush --include=. tripal-generic-install
```

From this point onward, you will be asked a series of questions in the terminal window. First you will be asked for the name of the site (this will appear at the top of your site after creation), the site administrator's email address, a username for the administrator to log on, and the password for the administrator:

```
Name of the site : Tripal
Admin email for the site : admin@gmail.com
Name for your admin user on the site : admin
Password for the admin user, needs to be complex including numbers and characters,␣
↪example P@55w0rd: P@55w0rd

These are the site settings provided, please review and confirm they are correct
   Site Name: Tripal
   Site email address: admin@gmail.com
   Administrator username: admin
   Administrator password: P@55w0rd
Is this information correct? (y/n): y
```

Next, you will be asked for the database information: database name, database username, database user password, host, and port. The database name and user should match what you created in the previous section (i.e. database name = 'drupal' and database user = 'drupal'). The 'host' is the name of the server or its IP address, and the port is a numerical value that PostgreSQL uses for communication. By default PostgreSQL uses the port 5432. If a mistake is made you can make corrections as shown below:

```
Now we need to setup Drupal to connect to the database you want to use. These␣
↪settings are added to Drupal`s settings.php file.

database name: database
postgres username: drupal
postgres password: drupal
host, like localhost or 127.0.0.1: 127.0.01
port, usually S432: 5432
This is the database information provided, please review and confirm it is correct:
Database name: database
Database username: drupal
Database user password: drupal
Database host: 127.0.01
Database port: 5432
Is this information correct? (Y/n): n

Now we need to setup Drupal to connect to the database you want to use. These␣
↪settings are added to Drupal`s settings.php file.

database name: database
postgres username: drupal
postgres password: drupal
host, like localhost or 127.0.0.1: 127.0.0.1
port, usually S432: 5432
This is the database information provided, please review and confirm it is correct:
Database name: database
Database username: drupal
```

(continues on next page)

(continued from previous page)

```
Database user password: drupal
Database host: 127.0.0.1
Database port: 5432
Is this information correct? (Y/n): y
```

After site information and database credentials are provided, Drupal will be installed. You will see this in the terminal:

```
Now installing Drupal.

--2017-09-20 12:29:16-- https://www.drupal.org/files/projects/drupal-7.56.tar.gz

Resolving www.drupal.org (www.drupal.org)... 151.101.5.175
Connecting to www.drupal.org (www.drupal.org)|151.101.5.175|:443... connected.
HTTP request sent, awaiting response... 200 OK

Length: 3277833 (3.1M) [application/x-gzip]
Saving to: ‘drupal-7.56.tar.gz'

drupal-7.56.tar.gz 100%[::::::::::::::::::::::::::::::::::::::::::::::::>] 3.13M 1.
↪82MB/s in 1.75

2017-09-20 12:29:20 (1.82 MB/S) - ‘drupal-7.56.tar.gz' saved [3277833/3277833]

You are about to create a /var/www/html/sites/default/settings.php file and DROP all␣
↪tables in your ‘database` database. Do you want to continue? (y/n): y

Starting Drupal installation. This takes a while. Consider using the --notify global␣
↪option.
Installation complete. User name: admin User password: P@55word
```

Next, the required modules will be downloaded:

```
Downloading modules.

Project field_group (7.x-1.5) downloaded to /var/www/html/sites/all/modules/field_
↪group.
Project field_group_table (7.x-1.6) downloaded to /var/www/html/sites/all/modules/
↪field_group_table.
Project field_formatter_class (7.x-1.1) downloaded to /var/www/html/sites/all/modules/
↪field_formatter_class.
Project field_formatter_settings (7.x-1.1) downloaded to /var/www/html/sites/all/
↪modules/field_formatter_settings.
Project ctools (7.x-1.12) downloaded to /var/www/html/sites/all/modules/ctools.␣
↪[success]
Project ctools contains 10 modules: ctools_custom_content, stylizer, ctools_plugin_
↪example, views_content, ctools_ajax_sample, term_depth, ctools_access_ruleset, page_
↪manager, bulk_export, ctools.
Project date (7.x-2.10) downloaded to /var/www/html/sites/all/modules/date.
Project date contains 11 modules: date_context, date_migrate_example, date_migrate,␣
↪date_popup, date_tool
repeat, date_views, date_all_day, date_api, date_repeat_field, date.
Project devel (7.x-1.5) downloaded to /var/www/html/sites/all/modules/devel.
Project devel contains 3 modules: devel_generate, devel, devel_node_access.
Project ds (7.x-2.14) downloaded to /var/www/html/sites/all/modules/ds.
Project ds contains 7 modules: ds_forms, ds_ui, ds_devel, ds_format, ds_extras, ds_
↪search, ds.
Project link (7.x-1.4) downloaded to /var/www/html/sites/all/modules/link.
```

(continues on next page)

(continued from previous page)

```
Project entity (7.x-1.8) downloaded to /var/www/html/sites/all/modules/entity.
Project entity contains 2 modules: entity_token, entity.
Project libraries (7.x-2.3) downloaded to /var/www/html/sites/all/modules/libraries.
redirect (7.x-1.0-rc3) downloaded to /var/www/html/sites/all/modules/redirect.
Project token (7.x-1.7) downloaded to /var/www/html/sites/all/modules/token.
Project tripal (7.x-3.1) downloaded to /var/www/html/sites/all/modules/tripal.
Project tripal contains 24 modules: tripal_daemon, tripal, tripal_chado, tripal_ws,␣
↪tripal_bulk_loader, tripal_chado_views, tripal_ds, tripal_contact, tripal_natural_
↪diversity, tripal_views, tripal_core, tripal_library, tripal_organism, tripal_
↪featuremap, tripal_genetic, tripal_db, tripal_analysis, tripal_phenotype, tripal_
↪pub, tripal_stock, tripal_project, tripal_cv, tripal_phylogeny, tripal_feature.
Project uuid (7.x-1.0) downloaded to /var/www/html/sites/all/modules/uuid.
Project uuid contains 4 modules: uuid_services, uuid_path, uuid_services_example,␣
↪uuid_path
Project jquery_update (7.x-2.7) downloaded to /var/www/html/sites/all/modules/jquery_
↪update.
Project views (7.x-3.18) downloaded to /var/www/html/sites/all/modules/views.␣
↪[success]
Project views contains 2 modules: views_ui, views.
Project webform (7.x-4.15) downloaded to /var/www/html/sites/all/modules/webform.␣
↪[success]
```

Then those modules will be enabled:

```
Enabling modules.
The following extensions will be enabled: ctools, date, devel, ds, link, entity,␣
↪libraries, redirect, tok
en, uuid, jquery_update, views, webform, field_group, field_group_table, field_
↪formatter_class, field_for
matter_settings, views_ui, date_api
Do you really want to continue? (Y/n): y
webform was enabled successfully.
ctools was enabled successfully.
date was enabled successfully.
webform defines the following permissions: access all webform results, access own␣
↪webform results, edit a
ll webform submissions, delete all webform submissions, access own webform␣
↪submissions, edit own webform
submissions, delete own webform submissions, edit webform components
ctools defines the following permissions: use ctools import
date_api was enabled successfully.
entity was enabled successfully.
field_formatter_class was enabled successfully.
field_formatter_settings was enabled successfully.
field_group_table was enabled successfully.
jquery_update was enabled successfully.
libraries was enabled successfully.
link was enabled successfully.
token was enabled successfully.
uuid was enabled successfully.
views_ui was enabled successfully.
ds was enabled successfully.
field_group was enabled successfully.
views was enabled successfully.
iredirect was enabled successfully.
uuid defines the following permissions: administer uuid
ds defines the following permissions: admin_display_suite
```

(continues on next page)

(continued from previous page)

```
field_group defines the following permissions: administer fieldgroups
views defines the following permissions: administer views, access all views
jdevel was enabled successfully.
The Date API requires that you set up the site timezone and first day of week␣
↪settings and the date format settings to function correctly.
redirect defines the following permissions: administer redirects
devel defines the following permissions: access devel information, execute php code,␣
↪switch users
```

Patches are then applied:

```
Applying patches.

--2017-09-20 12:29:48-- https2//drupal.org/files/drupal.pgsql-bytea.27.patch
Resolving drupal.org (drupal.org)... 151.101.129.175, 151.101.1.175, 151.101.193.175,
Connecting to drupal.org (drupal.org)|151.101.129.175|2443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https2//www.drupal.org/files/drupal.pgsql-bytea.27.patch [following]
--2017-09-20 12:29:49-- https2//www.drupal.org/files/drupal.pgsql-bytea.27.patch
Resolving www.drupal.org (www.drupal.org)... 151.101.5.175
Connecting to www.drupal.org (www.drupal.org)|151.101.5.175|2443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1613 (1.6K) [text/plain]
Saving to: ‘drupal.pgsql-bytea.27.patch'

drupal.pgsql-bytea.27.patch 100%[==========================================>]    1.58K␣
↪    --.-KB/s
  in 0s

2017-09-20 12:29:49 (98.4 MB/s) - ‘drupal.pgsql-bytea.27.patch' saved [1613/1613]
```

and Tripal will be enabled:

```
Enabling Tripal modules.

The following extensions will be enabled: tripal, tripal_chado, tripal_ds, tripal_ws,␣
↪php, tripal_chado_views
Do you really want to continue? (Y/n): y
php was enabled successfully.
php defines the following permissions: use PHP for settings
tripal was enabled successfully.
tripal defines the following permissions: administer tripal, access tripal content␣
↪overview, manage tripal content types, upload files, view dev helps
tripal_chado was enabled successfully.
tripal_chado defines the following permissions: install chado, view chado_ids
tripal_chado_views was enabled successfully.
tripal_chado_views defines the following permissions: manage tripal_views_integration
tripal_ds was enabled successfully.
tripal_ws was enabled successfully.
A PHP code text format has been created.

Clear cache.
`all` cache was cleared.
```

Next, you will be prompted to choose the Chado version you would like to install. Unless you need an earlier version for a specific reason, it is best to select the most recent version. In this case, Chado v1.3:

```
Installing Chado.
Which version of Chado would you like installed?
[0] : Cancel
[1] : Install Chado v1.3
[2] : Install Chado v1.2
[3] : Install Chado v1.11
Job `Install Chado v1.3` submitted.

2017-09-21 03:29:24
Tripal Job Launcher
Running as user `admin`
-------------------
2017-09-21 032292242 There are 1 jobs queued.
2017-09-21 032292242 Calling2 tripal_chado_install_chado(Install Chado v1.3)
Creating `chado` schema
Loading sites/all/modules/tripal/tripal_chado/chado_schema/default_schema-1.3.sql...
Install of Chado v1.3 (Step 1 of 2) Successful!
Loading sites/all/modules/tripal/tripal_chado/chado_schema/initialize-1.3.sql...
Install of Chado v1.3 (Step 2 of 2) Successful.
Installation Complete
```

Next, the site will be prepared and content types created:

```
Now preparing the site by creating content types.
Job `Prepare Chado` submitted.

2017-09-21 03:56:30
Tripal Job Launcher
Running as user `shawna`
-------------------
2017-09-21 03:56:30: There are 1 jobs queued.
2017-09-21 03:56:30: Calling: tripal_chado_prepare_chado()
Creating Tripal Materialized Views and Custom Tables...
Loading Ontologies...
Loading ontology: Taxonomic Rank (3)...
Downloading URL http://purl.obolibrary.org/obo/taxrank.obo, saving to /tmp/obo_RxmcoM
Percent complete: 100.00%. Memory: 32,394,440 bytes.
Updating cvtermpath table. This may take a while...
Loading ontology: Tripal Contact (4)...
Loading ontology: Tripal Publication (S)...68 bytes.
Loading ontology: Sequence Ontology (6)...424 bytes.
Downloading URL http://purl.obolibrary.org/obo/so.obo, saving to /tmp/obo_S40JJr
Percent complete: 100.00%. Memory: 33,718,672 bytes.
Updating cvtermpath table. This may take a while...
Making semantic connections for Chado tables/fields...
Map Chado Controlled vocabularies to Tripal Terms...
Examining analysis...
Examining biomaterial...
Examining contact...
Examining control...
Examining cvterm...
Examining feature...
Examining featuremap...
Examining genotype...
Examining library...
Examining organism...
Examining phenotype...
```

(continues on next page)

(continued from previous page)

```
Examining phylotree...
Examining project...
Examining protocol...
Examining protocolparam...
Examining pub...
Examining stock...
Examining stockcollection...
Examining studyfactor...
Examining synonym...

Done.
Creating common Tripal Content Types...

NOTE: Loading of publications is performed using a database transaction.
{If the load fails or is terminated prematurely then the entire set of
Einsertions/updates is rolled back and will not be found in the database

Custom table, `tripal_gff_temp` , created successfully.
Custom table, `tripal_gffcds_temp` , created successfully.
Custom table, `tripal_gffprotein_temp` , created successfully.
Custom table, `organism_stock_count` , created successfully.
Materialized view `organism_stock_count` created
Custom table, `library_feature_count` , created successfully.
Materialized view `library_feature_count` created
Custom table, `organism_feature_count` , created successfully.
Materialized view `organism_feature_count` created
Custom table, `analysis_organism` , created successfully.
Materialized view `analysis_organism` created
Custom table, `cv_root_mview` , created successfully.
Materialized view `cv_root_mview` created
```

The final step is to add permissions for the site administrator to view, edit, create, and delete the content types created in the previous step.

```
Adding permissions for the administrator to View, edit, create, and delete all the␣
↪newly created content types.
Added "View bio_data_1" to "administrator"
Added "create bio_data_1" to "administrator"
Added "edit bio_data_1" to "administrator"
Added "delete bio_data_1" to "administrator"
Added "View bio_data_2" to "administrator"
Added "create bio_data_2" to "administrator"
Added "edit bio_data_2" to "administrator"
Added "delete bio_data_2" to "administrator"
Added "View bio_data_3" to "administrator"
Added "create bio_data_3" to "administrator"
Added "edit bio_data_3" to "administrator"
Added "delete bio_data_3" to "administrator"
Added "View bio_data_4" to "administrator"
Added "create bio_data_4" to "administrator"
Added "edit bio_data_4" to "administrator"
Added "delete bio_data_4" to "administrator"
Added "View bio_data_5" to "administrator"
Added "create bio_data_5" to "administrator"
Added "edit bio_data_5" to "administrator"
Added "delete bio_data_5" to "administrator"
```

(continues on next page)

(continued from previous page)

```
Added "View bio_data_6" to "administrator"
Added "create bio_data_6" to "administrator"
Added "edit bio_data_6" to "administrator"
Added "delete bio_data_6" to "administrator"
Added "View bio_data_7" to "administrator"
Added "create bio_data_7" to "administrator"
Added "edit bio_data_7" to "administrator"
Added "delete bio_data_7" to "administrator"
"all" cache was cleared.

Installation is now complete. You may navigate to your new site. For more information␣
↪on using Tripal please see the installation guide on tripal.info.
```

The installation is now finished! Navigate to your new site by entering it's URL in a browser. For this example the URL is: http://localhost/.

1.2.6 Installation Method #2: Step-by-Step Manual Installation

Tripal is designed to be highly configurable in order to support a wide variety of community databases with different data types and styles. This results in a series of steps for installation to set the site as desired. The following provides step-by-step instructions for installation of Tripal. Users should use these instructions if a more in-depth understanding of installation steps is desired. Otherwise, it is recommend that new users follow the Rapid Installation instructions.

1.2.6.1 Drupal Installation

Database Setup

Drupal can use a MySQL or PostgreSQL database but Chado prefers PostgreSQL so that is what we will use for both Drupal and Chado. We need to create the Drupal database. The following command can be used to create a new database user and database.

First, become the 'postgres' user:

```
sudo su - postgres
```

Next, create the new 'drupal' user account. This account will not be a "superuser' nor allowed to create new roles, but should be allowed to create a database.

```
createuser -P drupal
```

When requested, enter an appropriate password. Finally, create the new database:

```
createdb drupal -O drupal
```

We no longer need to be the postgres user so exit

```
exit
```

Software Installation

Note: Remember you must set the $DRUPAL_HOME environment variable if you want to cut-and-paste the commands below. See *DRUPAL_HOME Variable*

Before we can install Drupal we must ensure that that we are allowed to add files into the root directory. Select a user account that will be the owner of all web files and change the owner of the $DRUPAL_HOME directory to that user:

```
sudo chown -R [user] $DRUPAL_HOME
```

Substitute [user] for the name of the user that will own the web files.

Note: The apache web server runs as the user 'www-data'. For security reasons you should chose a user other than 'www-data' to be the owner of the Drupal root directory.

Tripal 3.x requires version 7.x of Drupal. Drupal can be freely downloaded from the http://www.drupal.org website. At the writing of this Tutorial the most recent version of Drupal 7 is version 7.69. The software can be downloaded manually from the Drupal website through a web browser or we can use the wget command to retrieve it:

```
cd $DRUPAL_HOME
wget http://ftp.drupal.org/files/projects/drupal-7.69.tar.gz
```

Next, we want to install Drupal. We will use the tar command to uncompress the software:

```
tar -zxvf drupal-7.69.tar.gz
```

Notice that we now have a drupal-7.69 directory with all of the Drupal files. We want the Drupal files to be in our document root, not in a 'drupal-7.69' subdirectory. So, we'll move the contents of the directory up one level:

```
mv drupal-7.69/* ./
mv drupal-7.69/.htaccess ./
```

If an index.html file is present (as is the case with Ubuntu installations) you can move it out of the way so that it does not interfere with Drupal by executing the following:

```
mv index.html index.html.orig
```

Note: It is extremely important the the hidden file .htaccess is also moved (note the second mv command above. Check to make sure this file is there:

```
ls -l .htaccess
```

Configuration File

Next, we need to tell Drupal how to connect to our database. To do this we have to setup a configuration file. Drupal comes with an example configuration file which we can borrow.

First navigate to the location where the configuration file should go:

```
cd $DRUPAL_HOME/sites/default/
```

Next, copy the example configuration that already exists in the directory to be our actual configuration file by renaming it to settings.php.

```
cp default.settings.php settings.php
```

Now, we need to edit the configuration file to tell Drupal how to connect to our database server. To do this we'll use an easy to use text editor **gedit**.

```
gedit settings.php
```

Find the following line

```
$databases = array();
```

and then insert the following array just after the above line:

```
$databases['default']['default'] = array(
  'driver' => 'pgsql',
  'database' => 'drupal',
  'username' => 'drupal',
  'password' => '********',
  'host' => 'localhost',
  'prefix' => '',
);
```

Replace the text '****' with your database password for the user 'drupal' created previously. Save the configuration file and close the editor.

Files Directory Creation

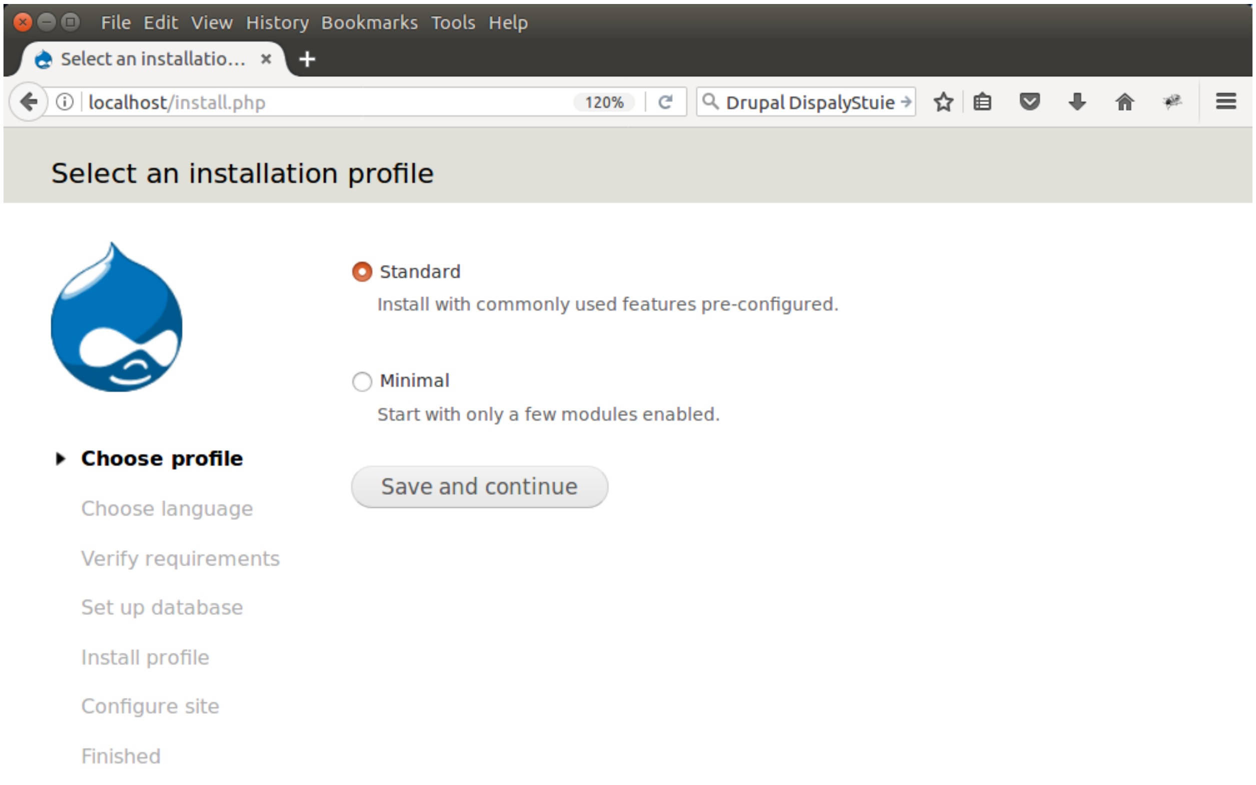
Finally, we need to create the directory where Drupal will have write-access to add files. By default, Drupal expects to have write permission in the `$DRUPAL_HOME/sites/default/files` directory. Therefore, we will set group ownership of the directory to the group used by the Apache web server. This will be the user that Drupal uses to write files.

```
mkdir -p $DRUPAL_HOME/sites/default/files
sudo chgrp [group] $DRUPAL_HOME/sites/default/files
sudo chmod g+rw $DRUPAL_HOME/sites/default/files
```

Substitute [group] for the name of the web server's group. In Ubuntu this is www-data in CentOS this is apache. The above commands creates the directory, sets the group ownership for group, and gives read/write permissions to the group on the directory.

Web-based Steps

Navigate to the installation page of our new web site http://localhost/install.php

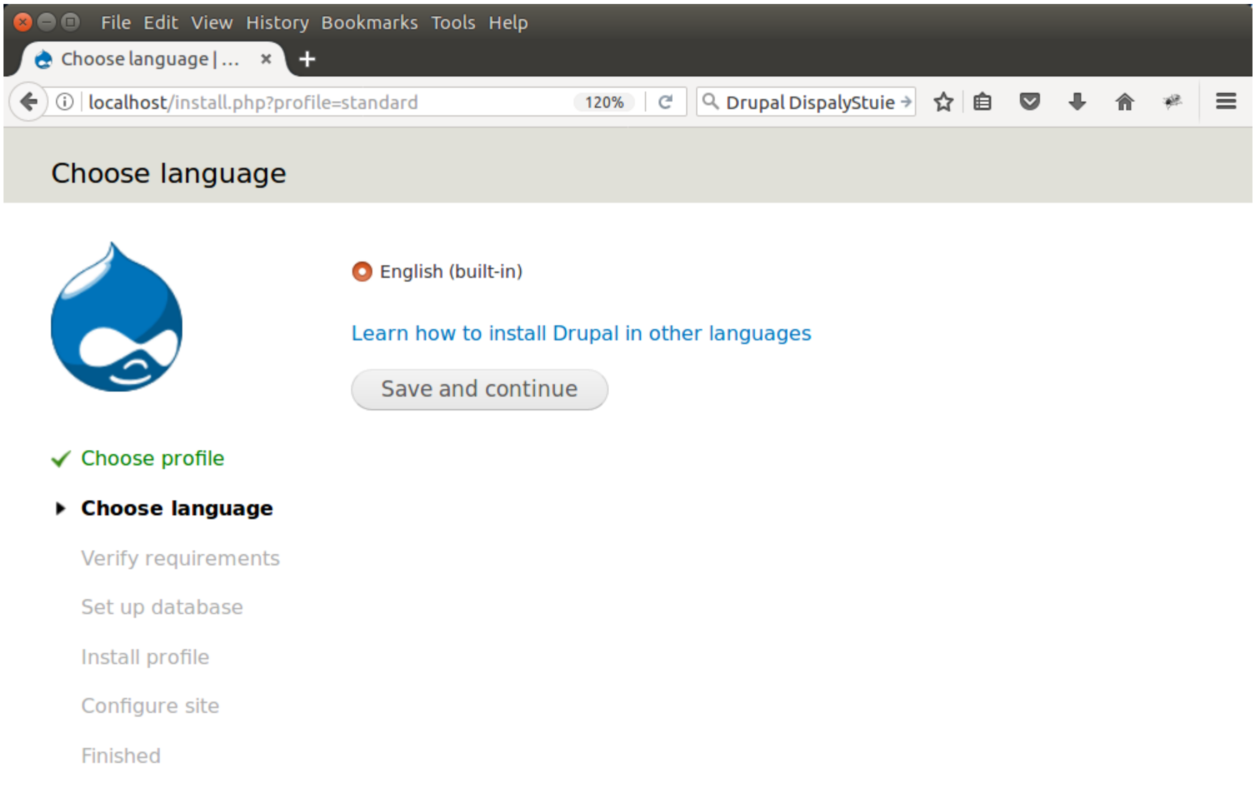


Select an installation profile

● Standard
Install with commonly used features pre-configured.

○ Minimal
Start with only a few modules enabled.

Ensure that Standard is selected and click **Save and Continue**. You will next be asked to select the language you want to use. Choose **English**:


File Edit View History Bookmarks Tools Help
Choose language | ...
localhost/install.php?profile=standard
120%
Drupal DispalyStuie
Save and continue

Choose language

● English (built-in)

Learn how to install Drupal in other languages

✓ Choose profile

Choose language

Verify requirements

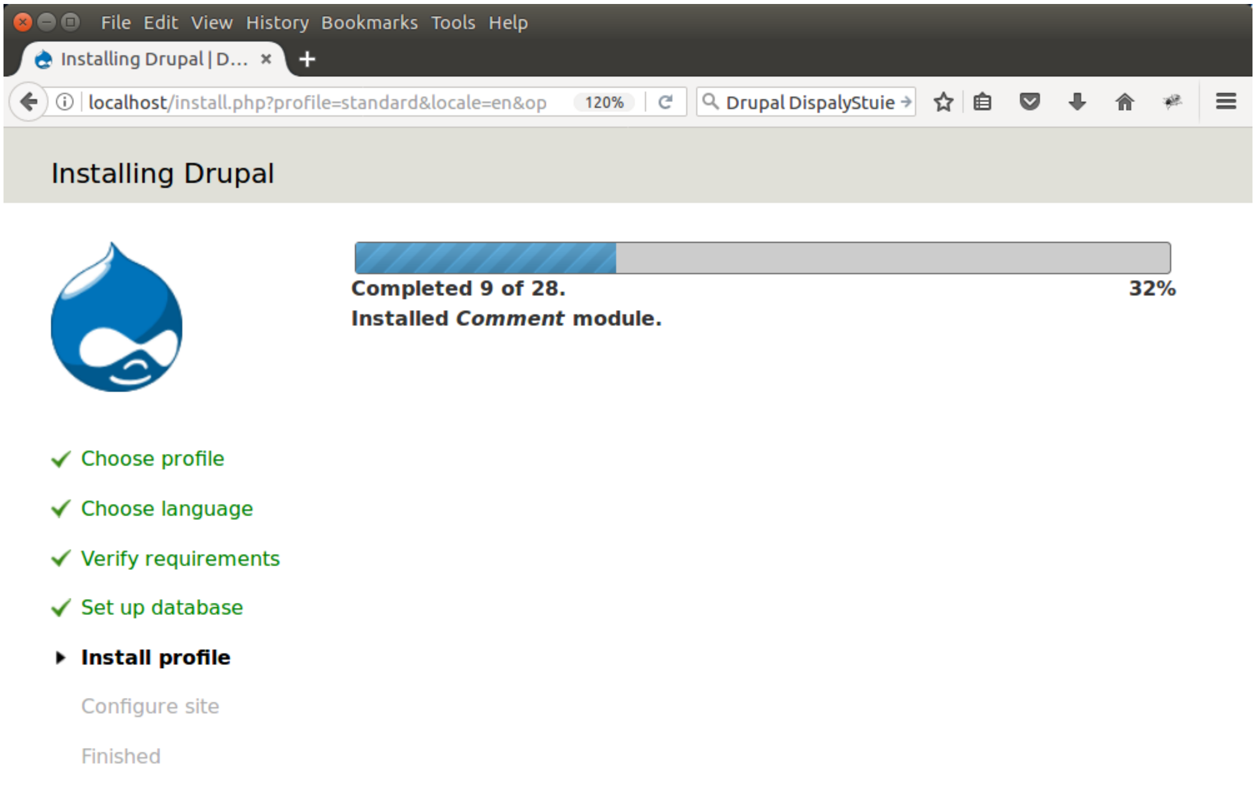
Set up database

Install profile

Configure site

Finished

Next, you will see a progress bar as Drupal is installed.


File Edit View History Bookmarks Tools Help
Installing Drupal | D…
localhost/install.php?profile=standard&locale=en&op
120%
Drupal DispalyStuie

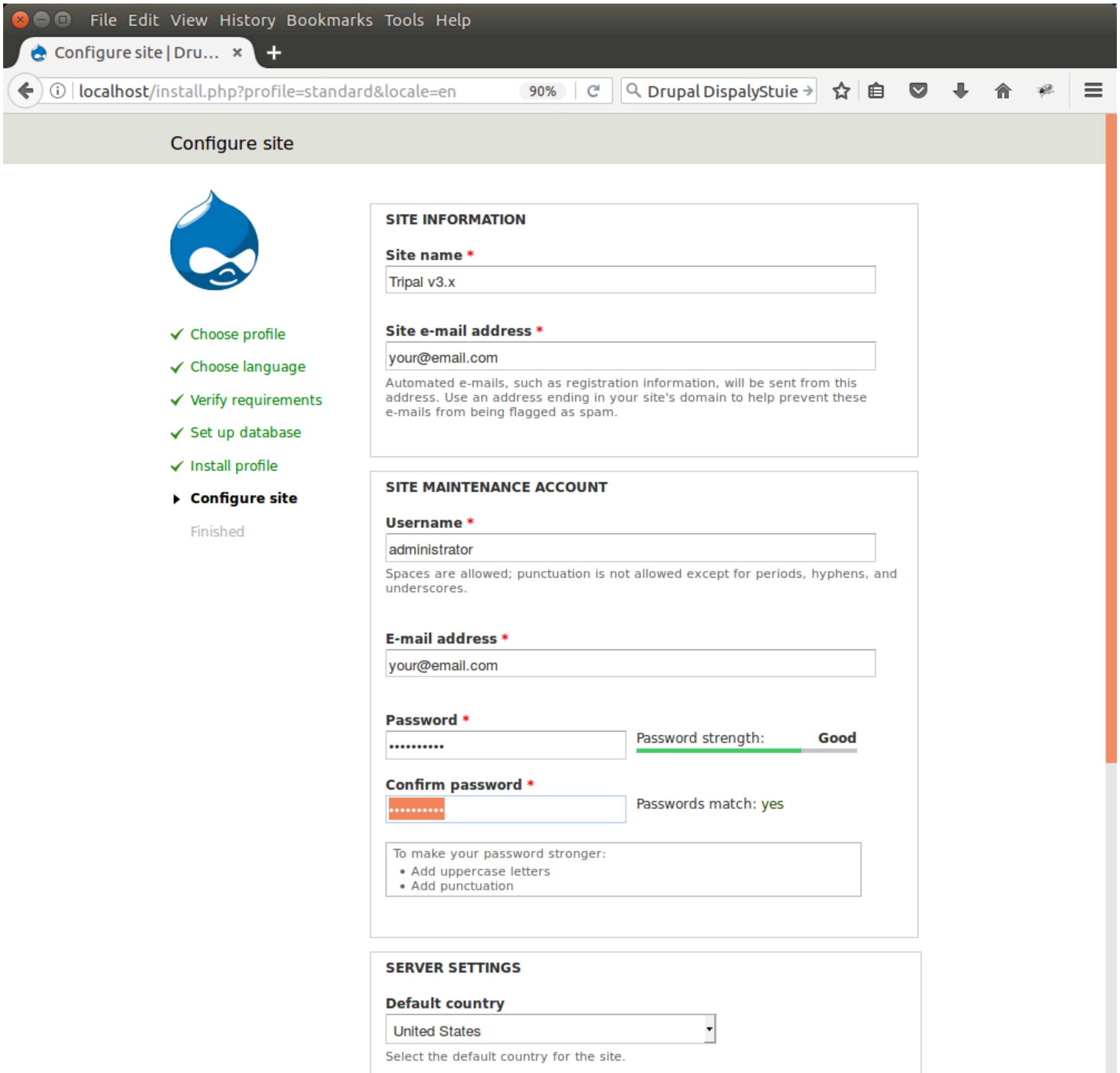
Installing Drupal

Completed 9 of 28. 32%

Installed *Comment* module.

- ✓ Choose profile
- ✓ Choose language
- ✓ Verify requirements
- ✓ Set up database
- ▸ **Install profile**
- Configure site
- Finished

Once it completes, a configuration page with some final settings will be visible.



Configure site

SITE INFORMATION

Site name *

Tripal v3.x

Site e-mail address *

your@email.com

Automated e-mails, such as registration information, will be sent from this address. Use an address ending in your site's domain to help prevent these e-mails from being flagged as spam.

SITE MAINTENANCE ACCOUNT

Username *

administrator

Spaces are allowed; punctuation is not allowed except for periods, hyphens, and underscores.

E-mail address *

your@email.com

Password *

••••••••••

Password strength: Good

Confirm password *

••••••••••

Passwords match: yes

To make your password stronger:
- Add uppercase letters
- Add punctuation

SERVER SETTINGS

Default country

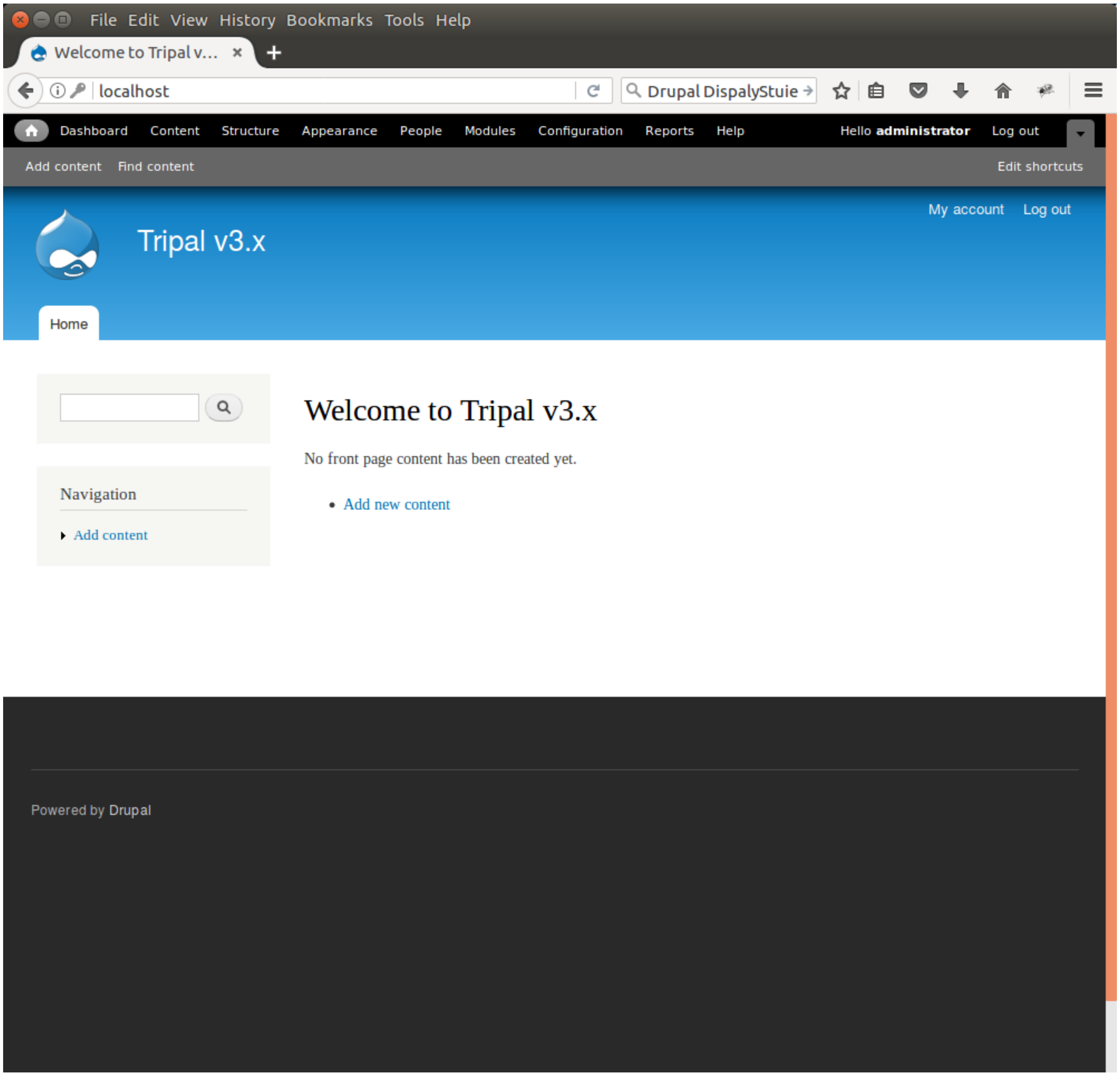
United States

Select the default country for the site.

Here you will provide details appropriate for your site, including your site name and administrative password. If you are experimenting with Tripal the following values can be used:

- Site Information - Site Name: Tripal 3.x - Site email: Your email address
- Site Maintenance Account - Username: administrator (all lower-case) - Email: Your email address - Password: ****
- Server Settings - Default country: (wherever the site is located) - Default time zone: (your time zone)
- Update Notifications (both boxes checked)

Now, click the **Save and Continue** button. You will see a message about unable to send an email. This is safe to ignore for the tutorial, but for a production site you will need that your server can send emails to a service provider. Now, your site is enabled. Click the link Your new site:



Welcome to Tripal v3.x

No front page content has been created yet.

- Add new content

1.2.6.2 Tripal Prerequisites

Note: Remember you must set the `$DRUPAL_HOME` environment variable if you want to cut-and-paste the commands below. See *DRUPAL_HOME Variable*

Tripal v3.x requires several Drupal modules. These include Entity, Views, CTools, Display Suite, Field Group, Field Group Table, Field Formatter Class and Field Formatter Settings modules. Modules can be installed using the graphical Drupal website by clicking on the Modules link in the top adminstrative menu bar. Instructions for instaling Modules via the web-interface can be found here: https://www.drupal.org/documentation/install/modules-themes/modules-7. However, Drush can be quicker for module installation. The following instructions will show how to install a module using the Drush command-line tool.

First, install the Entity module. We will download the current version using the drush command. On the command-line, execute the following:

```
cd $DRUPAL_HOME/sites/all/modules
drush pm-download entity
```

Typically for all module installation we should check the README for additional installation instructions. Next, enable the module using a drush command:

```
drush pm-enable entity
```

For basic Tripal functionality you must also enable the Views and CTools modules. You can specify as many module as desired on one line:

```
drush pm-download views ctools
drush pm-enable views views_ui ctools
```

Finally, Tripal works best when it can provide default display layouts. To support default layouts you must also enable the remaining dependencies:

```
drush pm-download ds field_group field_group_table field_formatter_class field_
↪formatter_settings
drush pm-enable ds field_group field_group_table field_formatter_class field_
↪formatter_settings
```

Optionally, you can install the ckeditor module. This module provides a nice WYSIWYG editor that allows you to edit text on your site using a graphical editor. Otherwise, if you need images or formatting (e.g. underline, bold, headers) you would be required to write HTML. It is recommended that this module be installed to improve the user experience:

```
drush pm-download ckeditor
drush pm-enable ckeditor
```

Finally, we need an more recent version of JQuery that what comes with Drupal. We can get this by installing the JQuery update module.

```
drush pm-download jquery_update
drush pm-enable jquery_update
```

1.2.6.3 Tripal Installation

Note: Remember you must set the `$DRUPAL_HOME` environment variable if you want to cut-and-paste the commands below. See *DRUPAL_HOME Variable*

Before installation of Tripal, you must first have a working Drupal installation. Please see the previous section of this tutorial for step-by-step examples for server setup and Drupal installation instructions. After installation of Tripal, you may install any Tripal extension modules you may want.

Download Tripal

The easiest way to download Tripal is to use the Drush command-line interface for Drupal. If you do not have Drush please see the Drush installation section of this tutorial. To download using drush follow these steps:

- Chage directories to your Drupal installation

- Execute the following drush command

  ```
  drush pm-download tripal
  ```

Alternatively, you can download Tripal using the Drupal web interface as per the instructions provided on the Installing modules documentation at Drupal.org. The Tripal project page at Drupal.org can be found here: https://www.drupal.org/project/tripal.

Apply Patches

A bug exists in Drupal related to the bytea data type in PostgreSQL. At the writing of this document, a fix is not yet incorporated into Drupal, but a patch has been provided. Execute the following commands to patch Drupal:

```
cd $DRUPAL_HOME
wget --no-check-certificate https://drupal.org/files/drupal.pgsql-bytea.27.patch
patch -p1 < drupal.pgsql-bytea.27.patch
```

There is also a bug in the Drupal Views 3.0 code that prevents Tripal's administrative and search data views from functioning. The patch is provided within the tripal_views module. To apply the patch execute the following:

```
cd $DRUPAL_HOME/sites/all/modules/views
patch -p1 < ../tripal/tripal_chado_views/views-sql-compliant-three-tier-naming-
↪1971160-30.patch
```

Install Tripal

The process for enabling the Tripal modules is the same as for the **ctools** and **views** modules that were enabled previously. To install the Tripal package, enter the following command:

```
drush pm-enable tripal
```

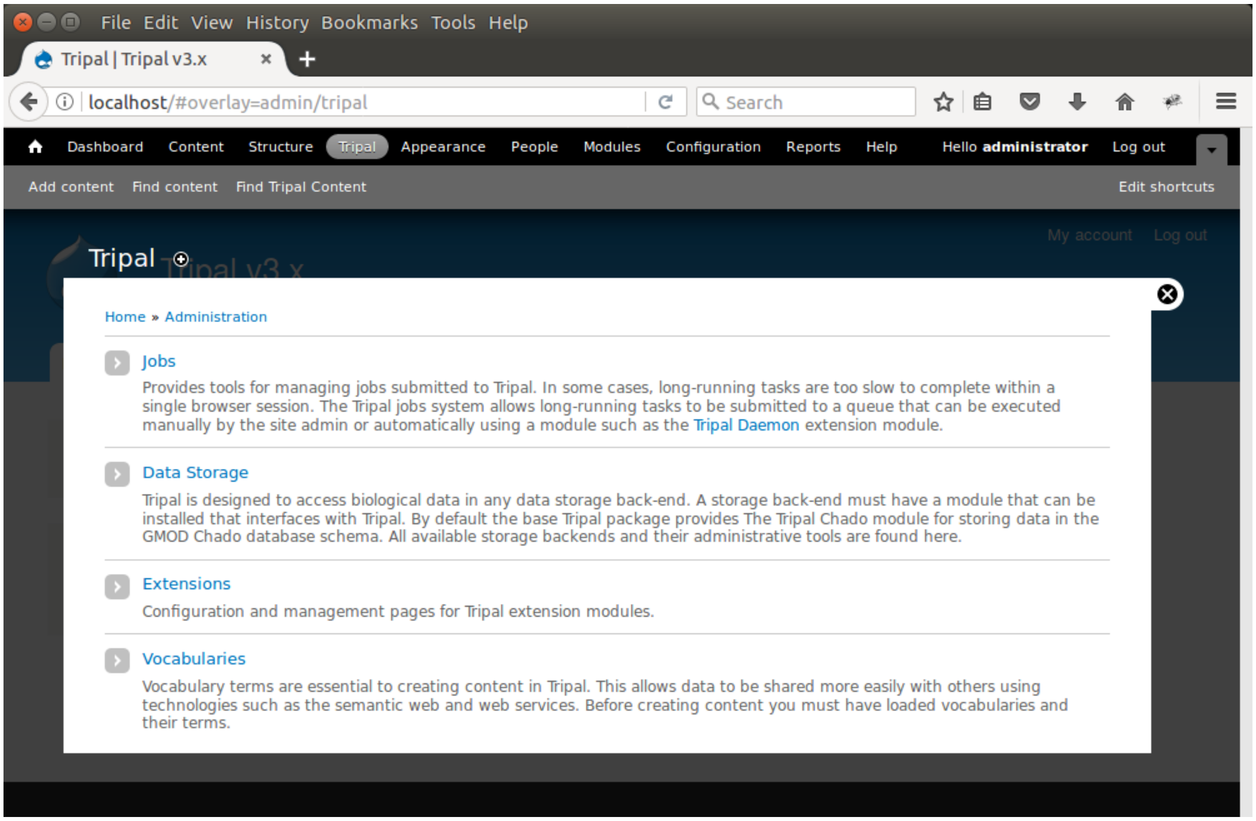
Tripal v3 is designed to be data store agnostic, therefore any data store with a compatible module can be used. By default, Tripal supports Chado and a Tripal Chado module is provided. Perform the same steps as before to enable the Tripal Chado module:

```
drush pm-enable tripal_chado
```

There are two more important Tripal modules: **tripal_ds** and **tripal_ws**. These modules provide default layouts for content (tripal_ds) and RESTful web services (tripal_ws). Neither of these modules are required, however without the default layouts content pages will be less attractive without manual organization. With web services you can share the content of your site that will allow remote software developers to access data programmatically. The Tripal Web Services module also will allow integration of data from other Tripal sites with this site. To enable both default layouts and web services use the following command:

```
drush pm-enable tripal_ds tripal_ws
```

Returning to the website, a new **Tripal** menu item appears at the top in the Administrative menu. Clicking the **Tripal** menu item reveals the Tripal administrative menu which contains four sections: Jobs, Data Storage, Extensions and Vocabularies. Each section will be described later in this guide.


File Edit View History Bookmarks Tools Help
Tripal | Tripal v3.x
localhost/#overlay=admin/tripal
Search
Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help Hello administrator Log out
Add content Find content Find Tripal Content Edit shortcuts
My account Log out
Tripal
Tripal v3.x
Home » Administration

Jobs

Provides tools for managing jobs submitted to Tripal. In some cases, long-running tasks are too slow to complete within a single browser session. The Tripal jobs system allows long-running tasks to be submitted to a queue that can be executed manually by the site admin or automatically using a module such as the Tripal Daemon extension module.

Data Storage

Tripal is designed to access biological data in any data storage back-end. A storage back-end must have a module that can be installed that interfaces with Tripal. By default the base Tripal package provides The Tripal Chado module for storing data in the GMOD Chado database schema. All available storage backends and their administrative tools are found here.

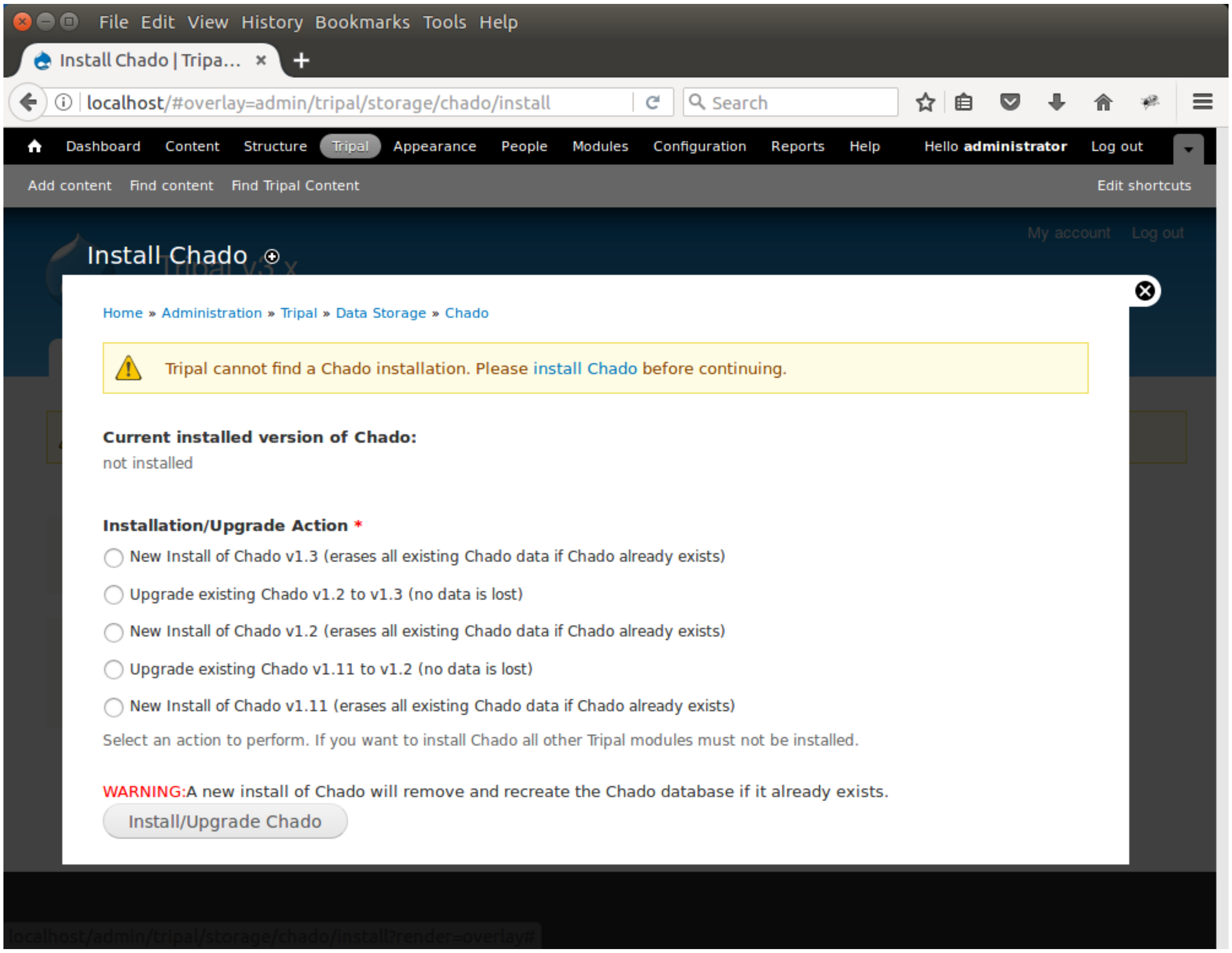
Extensions

Configuration and management pages for Tripal extension modules.

Vocabularies

Vocabulary terms are essential to creating content in Tripal. This allows data to be shared more easily with others using technologies such as the semantic web and web services. Before creating content you must have loaded vocabularies and their terms.

Because we have the Tripal Chado module enabled we will find a link to manage the Chado setup under the **Tripal** → **Data Storage** section. Notice the warning message indicating the Chado installation cannot be found. This is because the Chado schema has not yet been installed. The Chado schema is not automatically installed into the relational database (i.e. PostgreSQL). This is because Chado can be installed separately outside of Tripal and therefore Tripal does not try to overwrite it if it already exists. It is left to the site developer to consciously install Chado. To install Chado, navigate to **Tripal** → **Data Storage** → **Chado** → **Install Chado**. For this User's Guide it is assumed that Chado is not installed. Select the option to install Chado v1.3 and click the button Install/Upgrade Chado.


File Edit View History Bookmarks Tools Help
Install Chado | Tripa...
localhost/#overlay=admin/tripal/storage/chado/install
Search
Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help
Hello administrator Log out
Add content Find content Find Tripal Content
Edit shortcuts
My account Log out

Install Chado

Home » Administration » Tripal » Data Storage » Chado

Tripal cannot find a Chado installation. Please install Chado before continuing.

Current installed version of Chado:
not installed

Installation/Upgrade Action *

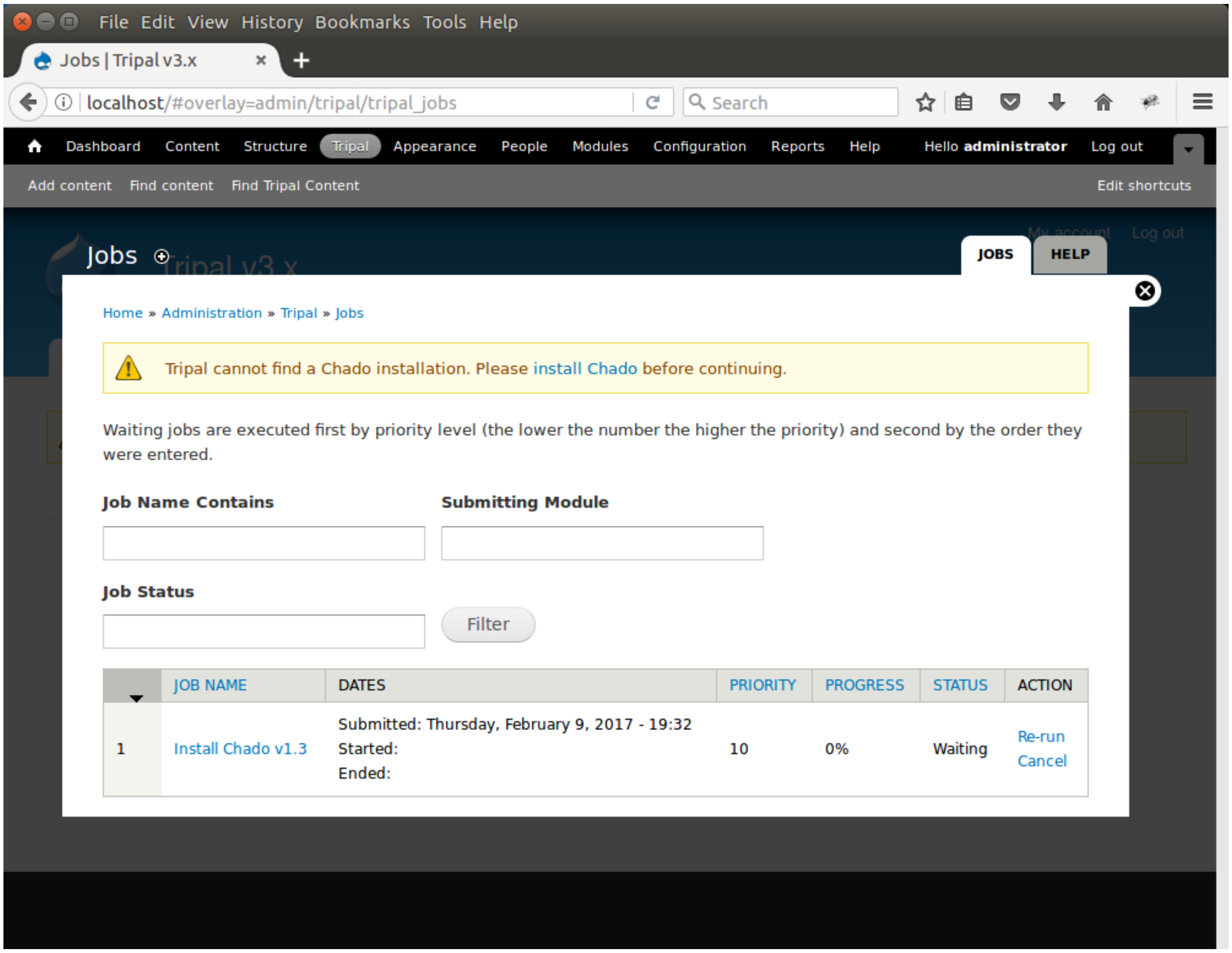
○ New Install of Chado v1.3 (erases all existing Chado data if Chado already exists)
○ Upgrade existing Chado v1.2 to v1.3 (no data is lost)
○ New Install of Chado v1.2 (erases all existing Chado data if Chado already exists)
○ Upgrade existing Chado v1.11 to v1.2 (no data is lost)
○ New Install of Chado v1.11 (erases all existing Chado data if Chado already exists)

Select an action to perform. If you want to install Chado all other Tripal modules must not be installed.

WARNING:A new install of Chado will remove and recreate the Chado database if it already exists.

Install/Upgrade Chado
localhost/admin/tripal/storage/chado/install?render=overlay#

After the button is clicked a message will appear stating “Job ‘Install Chado v1.3’ submitted.”. Click the jobs page link to see the job that was submitted:


File Edit View History Bookmarks Tools Help
Jobs | Tripal v3.x
localhost/#overlay=admin/tripal/tripal_jobs
Search
Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help
Hello administrator Log out
Add content Find content Find Tripal Content
Edit shortcuts
Jobs
JOBS
HELP

Home » Administration » Tripal » Jobs

Tripal cannot find a Chado installation. Please install Chado before continuing.

Waiting jobs are executed first by priority level (the lower the number the higher the priority) and second by the order they were entered.

Job Name Contains

Submitting Module

Job Status

Filter

	JOB NAME	DATES	PRIORITY	PROGRESS	STATUS	ACTION
1	Install Chado v1.3	Submitted: Thursday, February 9, 2017 - 19:32 Started: Ended:	10	0%	Waiting	Re-run Cancel

The job is waiting in the queue until the Tripal jobs system wakes and tries to run the job. The jobs management subsystem allows modules to submit long-running jobs, on behalf of site administrators or site visitors. Often, long running jobs can time out on the web server and fail to complete. The jobs system runs separately in the background. In the example above we now see a job for installing Chado. The job view page provides details such as the name of the job, dates that the job was submitted and job status.

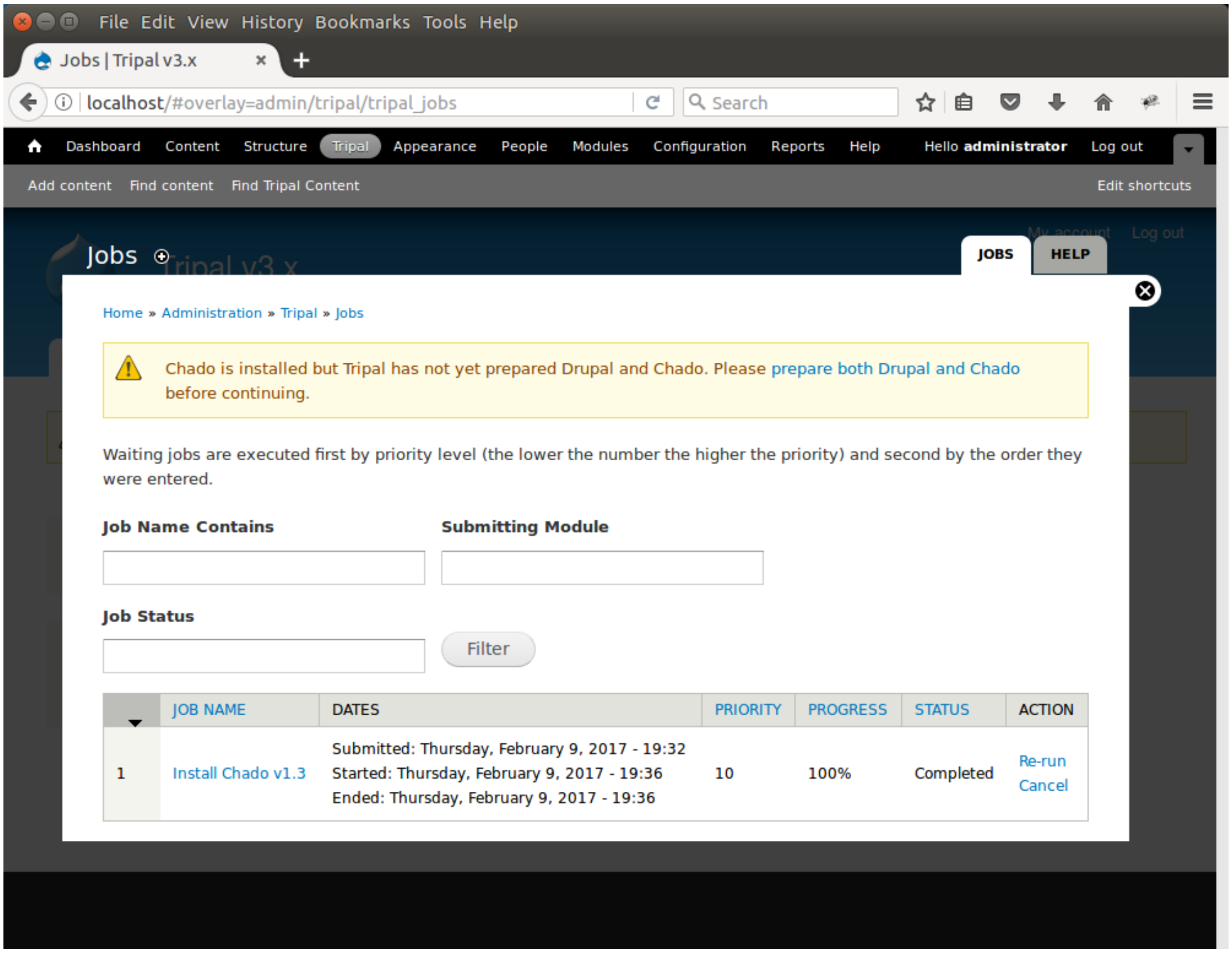
Jobs in the queue can be executed using drush to manually launch the job:

```
drush trp-run-jobs --username=administrator --root=$DRUPAL_HOME
```

As the installation of Chado proceeds, we should see output on the terminal console indicating the progress of the installation. You should see output similar to the following:

```
Tripal Job Launcher
Running as user 'administrator'
-------------------
2018-06-29 16:28:38: There are 1 jobs queued.
2018-06-29 16:28:38: Job ID 1.
2018-06-29 16:28:38: Calling: tripal_chado_install_chado(Install Chado v1.3)
Creating 'chado' schema
Loading sites/all/modules/tripal/tripal_chado/chado_schema/default_schema-1.3.sql...
Install of Chado v1.3 (Step 1 of 2) Successful!
Loading sites/all/modules/tripal/tripal_chado/chado_schema/initialize-1.3.sql...
Install of Chado v1.3 (Step 2 of 2) Successful.
Installation Complete
```

We now see that the job has completed when refreshing the jobs management page:


File Edit View History Bookmarks Tools Help
Jobs | Tripal v3.x
localhost/#overlay=admin/tripal/tripal_jobs
Search
Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help
Hello administrator Log out
Add content Find content Find Tripal Content
Edit shortcuts
Jobs
JOBS HELP
Home » Administration » Tripal » Jobs

Chado is installed but Tripal has not yet prepared Drupal and Chado. Please prepare both Drupal and Chado before continuing.

Waiting jobs are executed first by priority level (the lower the number the higher the priority) and second by the order they were entered.

Job Name Contains

Submitting Module

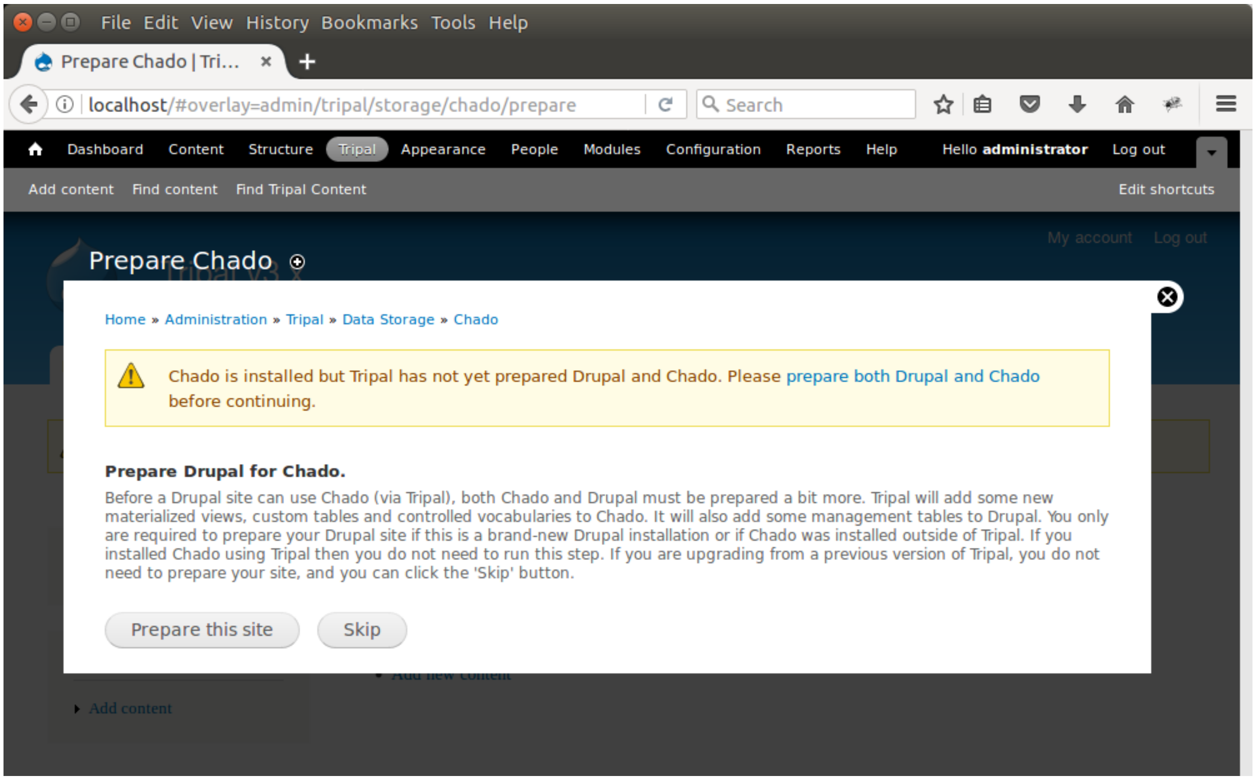
Job Status

Filter

	JOB NAME	DATES	PRIORITY	PROGRESS	STATUS	ACTION
1	Install Chado v1.3	Submitted: Thursday, February 9, 2017 - 19:32 Started: Thursday, February 9, 2017 - 19:36 Ended: Thursday, February 9, 2017 - 19:36	10	100%	Completed	Re-run Cancel

Prepare Chado and Drupal

To complete the installation of Chado we must prepare it for use with Tripal. Notice in the screen shot above the message indicates that "Chado is installed by Tripal has not yet prepared Drupal and Chado. . . .". We must prepare Chado and Drupal before continuing. To do this, click the link titled **prepare both Drupal and Chado**. The following page appears:


File Edit View History Bookmarks Tools Help
Prepare Chado | Tri...
localhost/#overlay=admin/tripal/storage/chado/prepare
Search
Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help
Hello administrator Log out
Add content Find content Find Tripal Content
Edit shortcuts
My account Log out
Prepare Chado
Prepare this site
Skip
Add content

Home » Administration » Tripal » Data Storage » Chado

Chado is installed but Tripal has not yet prepared Drupal and Chado. Please prepare both Drupal and Chado before continuing.

Prepare Drupal for Chado.

Before a Drupal site can use Chado (via Tripal), both Chado and Drupal must be prepared a bit more. Tripal will add some new materialized views, custom tables and controlled vocabularies to Chado. It will also add some management tables to Drupal. You only are required to prepare your Drupal site if this is a brand-new Drupal installation or if Chado was installed outside of Tripal. If you installed Chado using Tripal then you do not need to run this step. If you are upgrading from a previous version of Tripal, you do not need to prepare your site, and you can click the 'Skip' button.

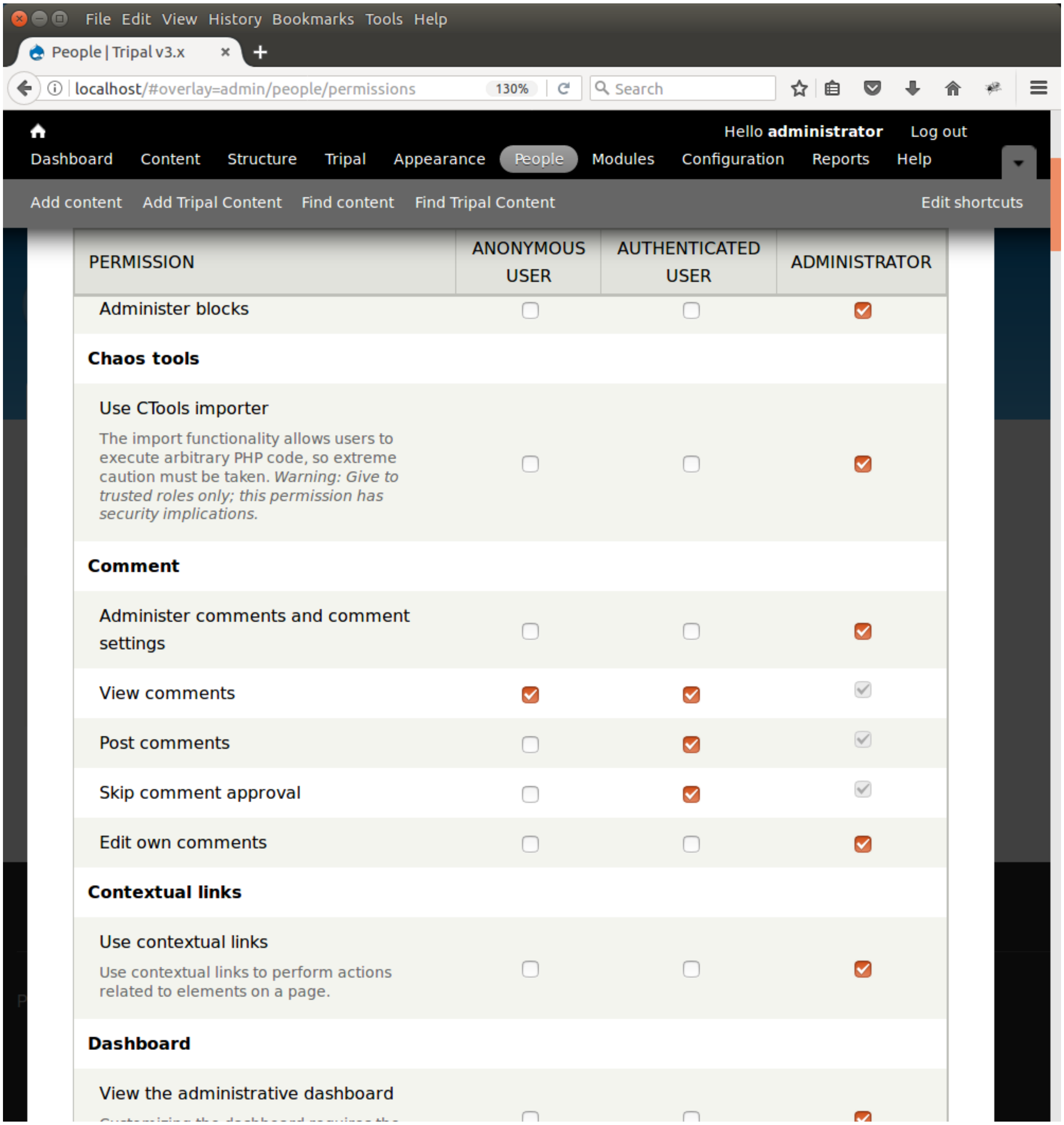
To prepare the site click the button Prepare this site. A new job is added to the jobs queue. Jobs in the queue can be executed using drush to manually launch the job:

```
drush trp-run-jobs --username=administrator --root=$DRUPAL_HOME
```

Note: Preparing Chado may take several minutes to complete. This is because the Sequence Ontology is automatically downloaded and installed into Chado as well as a few other vocabularies.

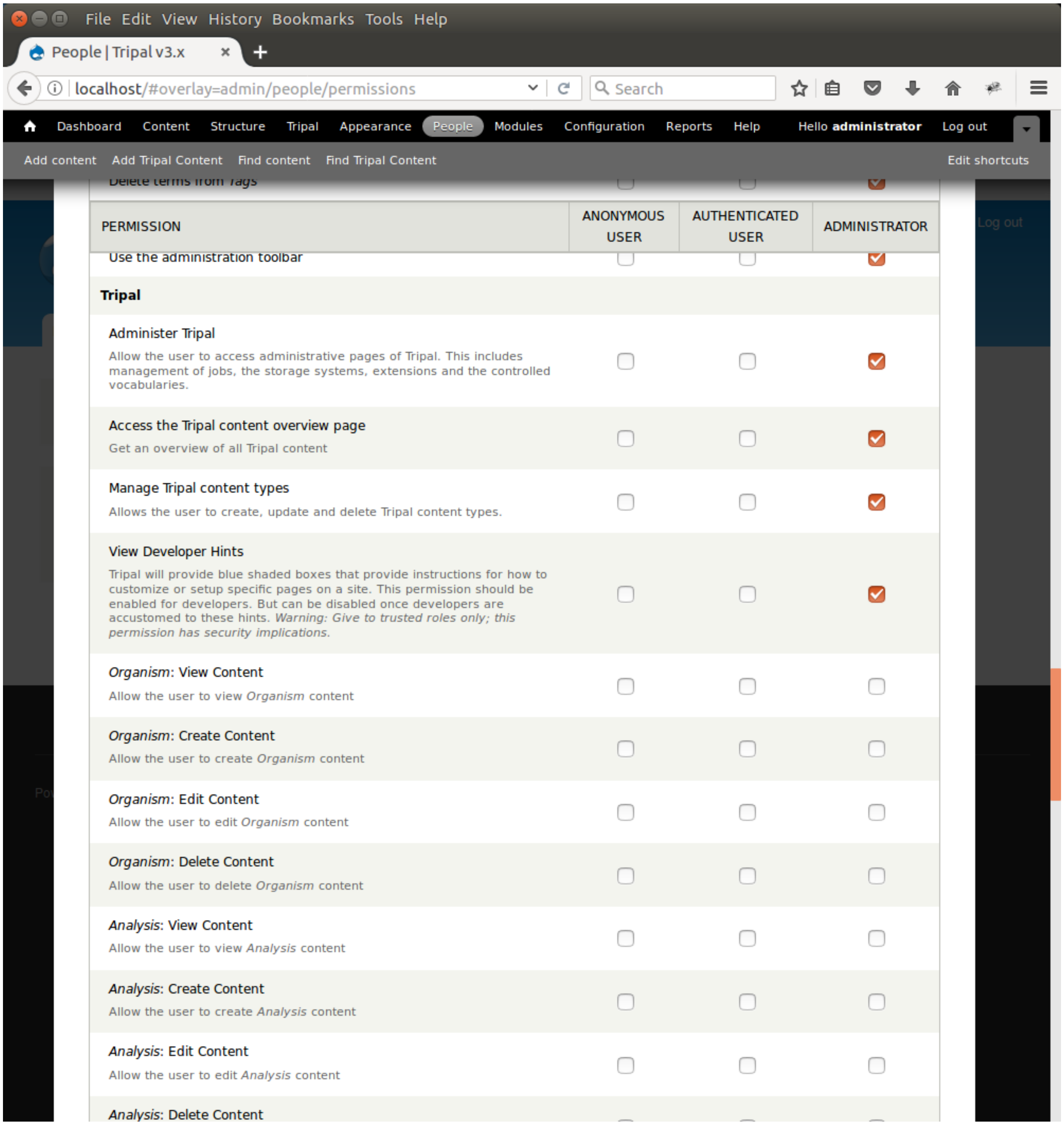
Set Permissions

Because we are logged on to the site as the administrator user we are able to see all content. However, Drupal provides user management and permissions tools that allows the site admin to set which types of users can view the content on the site. By default there are three types of users anonymous, authenticated and the administrator. For this tutorial we want to set permissions so that anonymous visitors to the site can see the genomics content. To do this, navigate to **People** → **Permissions**. Here you will see permissions for all types of content.



PERMISSION	ANONYMOUS USER	AUTHENTICATED USER	ADMINISTRATOR
Administer blocks	☐	☐	☑
Chaos tools			
Use CTools importer The import functionality allows users to execute arbitrary PHP code, so extreme caution must be taken. *Warning: Give to trusted roles only; this permission has security implications.*	☐	☐	☑
Comment			
Administer comments and comment settings	☐	☐	☑
View comments	☑	☑	☑
Post comments	☐	☑	☑
Skip comment approval	☐	☑	☑
Edit own comments	☐	☐	☑
Contextual links			
Use contextual links Use contextual links to perform actions related to elements on a page.	☐	☐	☑
Dashboard			
View the administrative dashboard Customizing the dashboard requires the	☐	☐	☑

Preparing Chado and Drupal in a previous step resulted in the automatic creation of some commonly used content types such as Organism, Analysis, Gene, mRNA, Map, Publication, and others. You can control who can view, create, edit and delete these types of content types, as well as set some administrative permissions if needed. On the Permission page, scroll down to the Tripal section. Here you will see permissions that you can set per type of user:


File Edit View History Bookmarks Tools Help
People | Tripal v3.x
localhost/#overlay=admin/people/permissions
Search
Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help
Hello **administrator** Log out
Add content Add Tripal Content Find content Find Tripal Content
Edit shortcuts

PERMISSION	ANONYMOUS USER	AUTHENTICATED USER	ADMINISTRATOR
Delete terms from *Tags*	☐	☐	☑
Use the administration toolbar	☐	☐	☑
Tripal			
Administer Tripal Allow the user to access administrative pages of Tripal. This includes management of jobs, the storage systems, extensions and the controlled vocabularies.	☐	☐	☑
Access the Tripal content overview page Get an overview of all Tripal content	☐	☐	☑
Manage Tripal content types Allows the user to create, update and delete Tripal content types.	☐	☐	☑
View Developer Hints Tripal will provide blue shaded boxes that provide instructions for how to customize or setup specific pages on a site. This permission should be enabled for developers. But can be disabled once developers are accustomed to these hints. *Warning: Give to trusted roles only; this permission has security implications.*	☐	☐	☑
Organism: View Content Allow the user to view *Organism* content	☐	☐	☐
Organism: Create Content Allow the user to create *Organism* content	☐	☐	☐
Organism: Edit Content Allow the user to edit *Organism* content	☐	☐	☐
Organism: Delete Content Allow the user to delete *Organism* content	☐	☐	☐
Analysis: View Content Allow the user to view *Analysis* content	☐	☐	☐
Analysis: Create Content Allow the user to create *Analysis* content	☐	☐	☐
Analysis: Edit Content Allow the user to edit *Analysis* content	☐	☐	☐
Analysis: Delete Content			

Review these permissions and set them according to how you want content to be managed. Typically, the administrator user receives all permissions, and anonymous and authenticated users receive 'View' permissions for all content types. If you desire to create other types of users, Drupal allows you to do this by creating new types of roles. For example, if you know that some users will be responsible for curating content, then you may add a curator role by clicking the **Roles** link in the top right corner of this permissions page. After the new role is created you can return to the permission page to set the permissions accordingly.

1.2.7 Upgrade from Tripal v2 to v3

Note: If you are installing Tripal v3 for the first time you can ignore these upgrade instructions.

1.2.7.1 Step 1: Upgrade Tripal

1. Please note the following before upgrading:

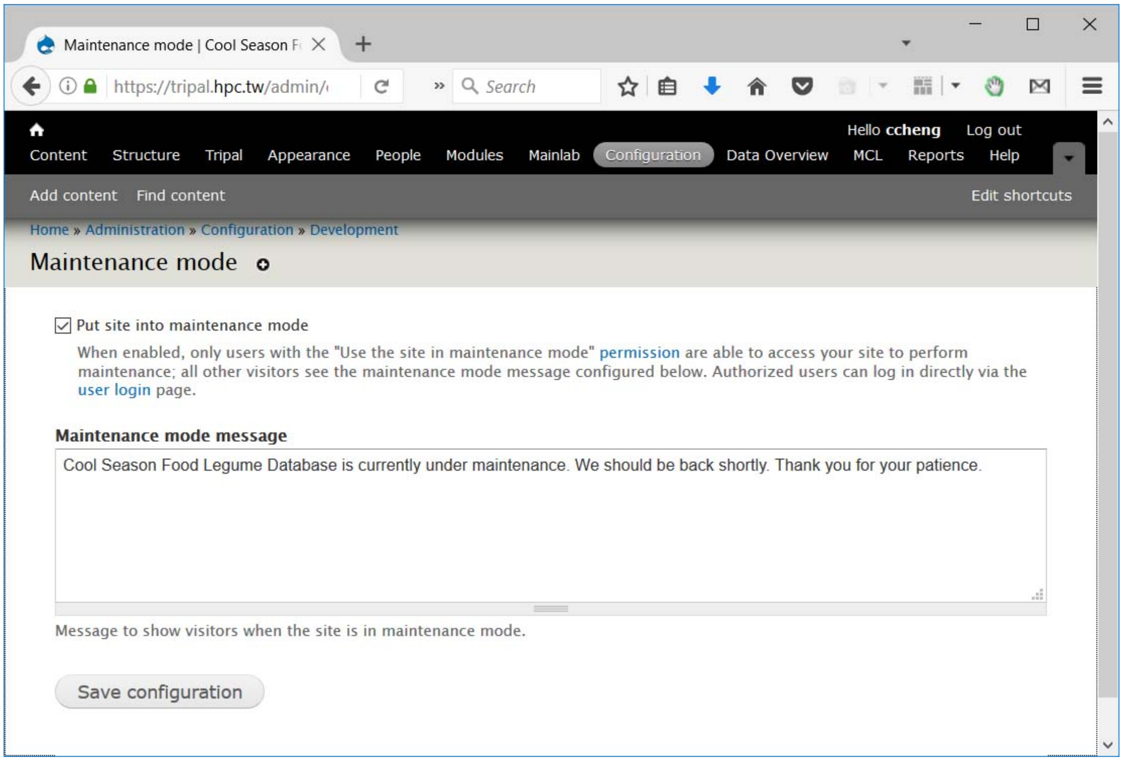
   > **Warning:** If you have created custom extension modules for your site please note that deprecated API functions from Tripal v1.x have been removed from Tripal v3. Therefore, use of deprecated API functions in templates or custom modules may cause a "white screen of death" (WSOD). Check the server logs if this occurs to find where deprecated functions may be used.

   > **Warning:** Remember to perform a full backup prior to any upgrade. It is recommended to test upgrade on a copy or development version of your site prior to performing on the production site.

   > **Warning:** Upgrade can only be performed using 'drush' command.

   > **Warning:** If you have made customizations to Chado you may encounter problems during the upgrade. It is not recommended to ever change any of the existing tables of Chado. However, if you have and if you do encounter such issues, please use the Tripal Issue queue to request help: https://github.com/tripal/tripal/issues
   >
   > If you have custom Drupal fields attached to Tripal nodes then the content in those fields will not automatically be migrated to the new Tripal v3 entities. Bradford Condon has provided some instructions to help migrate these fields after the site has been upgrade. You can find those instructions here.

2. Put the site in maintenance mode. Before completing any upgrade you should put your site into "maintenance mode". This ensures that users are isolated from any temporary error messages generated through the process. To put the site in maintenance mode, navigate to **Administration > Configuration > Maintenance Mode** . Then click the **Put site into maintenance mode** checkbox and click **Save Configuration**. Additionally, there is a text area on this page that allows you to customize the message displayed to your users while your site is in maintenance mode.



Home » Administration » Configuration » Development

Maintenance mode

☑ Put site into maintenance mode

When enabled, only users with the "Use the site in maintenance mode" permission are able to access your site to perform maintenance; all other visitors see the maintenance mode message configured below. Authorized users can log in directly via the user login page.

Maintenance mode message

Cool Season Food Legume Database is currently under maintenance. We should be back shortly. Thank you for your patience.

Message to show visitors when the site is in maintenance mode.

You can also put your site into “Maintenance mode” using drush be executing the following command:

```
drush vset site_offline 1
```

3. Disable tripal modules.

   Before updating the Tripal codebase, you should disable all Tripal modules. This ensures that Tripal is not actively trying to access files that you are changing, as well as, clears any cached information for these modules. When using drush, disabling the core module will disable all other Tripal modules:

   ```
   drush pm-disable tripal_core
   ```

4. The Tripal modules must also be downloaded and updated. To do this, delete the old Tripal v2 modules directories, located in `sites/all/modules` from your Drupal root: for example `/var/www/html/sites/all/modules` (be sure you have a backup before removing). The following command will retrieve the Tripal 3 version:

   ```
   drush pm-download tripal-7.x-3.5
   ```

5. Enable the tripal module

   ```
   drush pm-enable tripal
   ```

6. Enable the tripal_chado module

   ```
   drush pm-enable tripal_chado
   ```

7. Enable Tripal v2 Legacy modules. Tripal v2 modules are now called ‘legacy modules’. these are the modules that were disabled in step #2. For backwards compatibility, you should re-enable these modules:

   ```
   drush pm-enable tripal_core, tripal_views, tripal_db, tripal_cv, tripal_
   ↪analysis, tripal_organism, tripal_feature, tripal_pub, tripal_stock
   ```

Be sure to enable any additional modules not included in the example drush command above. The easiest way to ensure you have re-enabled all the modules disabled above is to copy the list drush provided when asking you to confirm disabling tripal_core above.

8. (Optional but Recommended) Enable the Tripal DS (provides default themeing for Tripal 3.x) and Tripal Web Services.

    - Tripal DS: Tripal 3.x provides complete integration with Drupal's Display UI allowing you to re-order fields and customize display using Drupal Extension modules. The Tripal DS module provides Tripal Panes similar to those in Tripal 2.x (except that more then one pane can be open at a time) and groups fields by default to make the display less overwhelming.
    - Tripal Web Services: Tripal Web services provide a way for Tripal sites to share data with each other and with their community in a programmatic manner. Your web services will show the same content available through your Tripal site using the RDF Specification.

    ```
    drush pm-enable tripal_ds tripal_ws
    ```

9. Tripal Daemon provides automatic job execution and was previously a tripal extension module but is now part of the main Tripal package. If you had Tripal Daemon installed with Tripal 2 and you would like to continue using it follow these instructions. First, disable the module and remove the module directory.

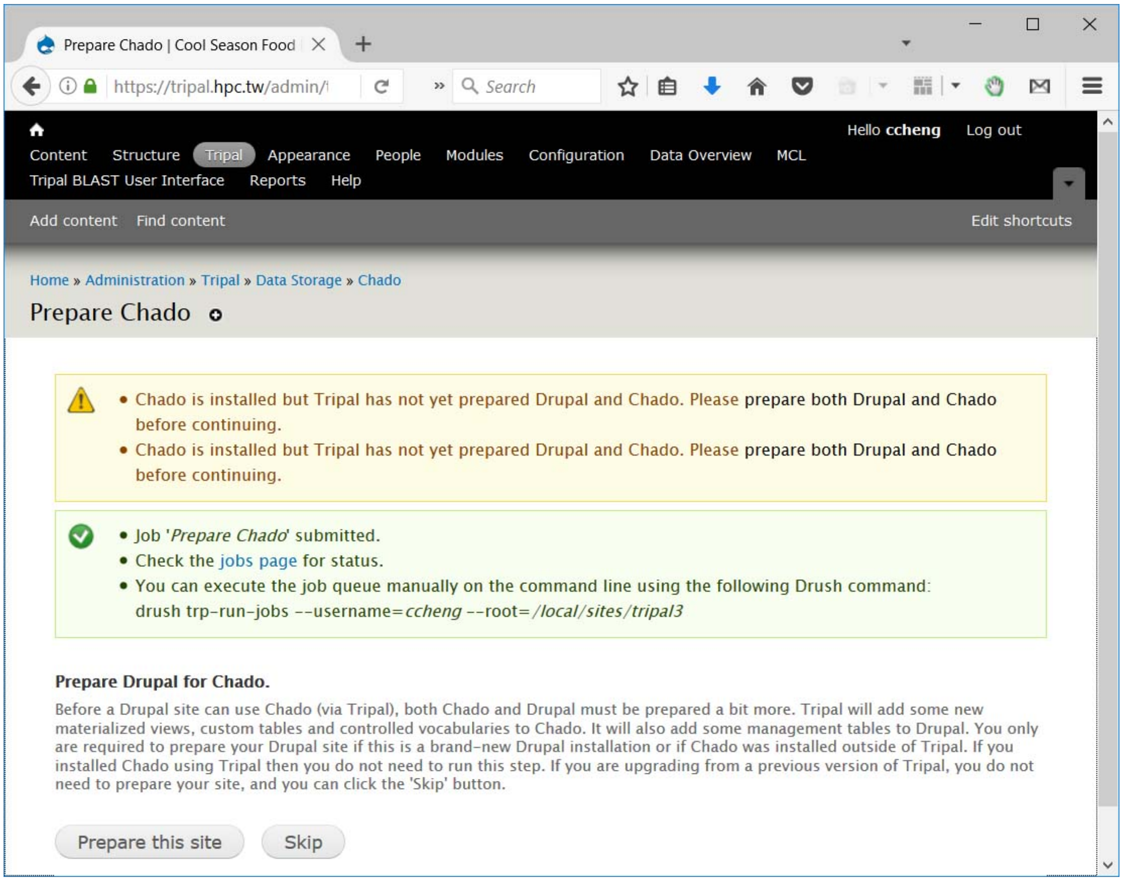
    ```
    drush pm-disable tripal_daemon
    ```

    Next remove the original tripal_daemon module from the sites/all/modules directory of your site. If you have had the Tripal Daemon installed for Tripal 2 then you should have all the necessary prerequisites and you can simply re-enable the module:

    ```
    drush pm-enable tripal_daemon
    ```

    Note: Remember to restart the tripal_daemon once you have completed the upgrade.

10. Return to your Tripal site, and click the link that appears for preparing Chado and launch the job.



- Chado is installed but Tripal has not yet prepared Drupal and Chado. Please prepare both Drupal and Chado before continuing.
- Chado is installed but Tripal has not yet prepared Drupal and Chado. Please prepare both Drupal and Chado before continuing.

- Job '*Prepare Chado*' submitted.
- Check the jobs page for status.
- You can execute the job queue manually on the command line using the following Drush command: drush trp-run-jobs --username=*ccheng* --root=*/local/sites/tripal3*

Prepare Drupal for Chado.

Before a Drupal site can use Chado (via Tripal), both Chado and Drupal must be prepared a bit more. Tripal will add some new materialized views, custom tables and controlled vocabularies to Chado. It will also add some management tables to Drupal. You only are required to prepare your Drupal site if this is a brand-new Drupal installation or if Chado was installed outside of Tripal. If you installed Chado using Tripal then you do not need to run this step. If you are upgrading from a previous version of Tripal, you do not need to prepare your site, and you can click the 'Skip' button.

Note: You may see the message "Please update the database using "drush updatedb" before continuing" You can safely ignore this message and it should disappear after preparing Chado.

11. Next, navigate to the permissions page at **Administration > People > Permissions** and ensure that all new Tripal permissions are set appropriately for your site roles.

Note: Tripal v3 adds a variety of new permissions so please check the permissions carefully.

12. You can now bring your site out of maitenence mode. This can be done by either reversing the your actions through the interface in #1 or through drush with the following command:

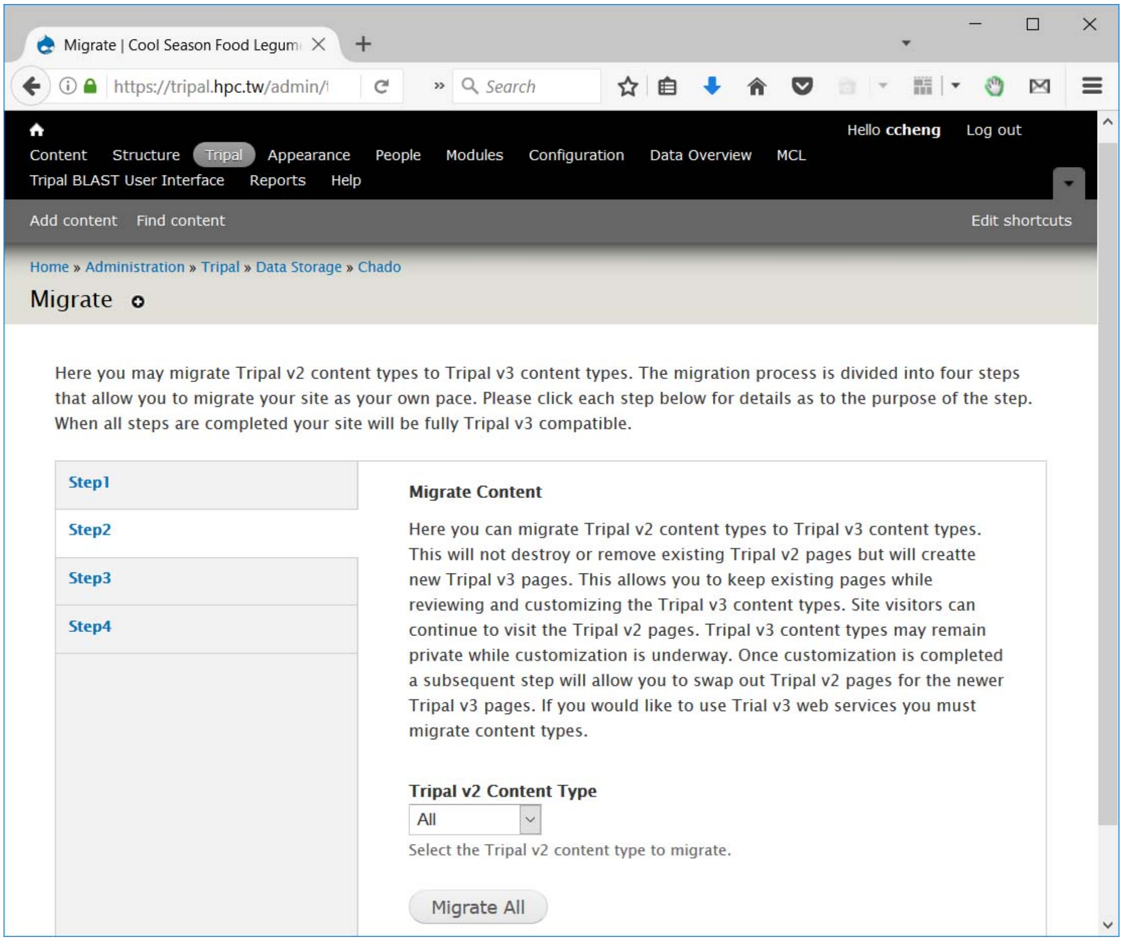
```
drush vset site_offline 0
```

13. Software Upgrade Complete! At this point your site is running Tripal 3. You currently have all your Tripal 2 pages (known as nodes) living happily inside your upgraded Tripal 3 site. This is known as "legacy mode". The upgrade process was designed to allow you to upgrade to Tripal 3 first and then migrate your "nodes" slowly to the new "entities" as you are able. Migrating from nodes to entities provides greater flexibilty and access to newer Tripal 3 features.

1.2.7.2 Step 2: Migrate Content

The process allows you to create Tripal 3 content types exposing the same data as your Tripal 2 nodes. Data is not duplicated as it resides in Chado but rather mappings are made between Chado records and your new Tripal 3 entities just as they were made to Tripal 2 nodes. This step will not remove or destroy existing Tripal v2 nodes/pages but will create new Tripal v3 entities/pages. This allows you to keep existing pages while reviewing and customizing the Tripal v3 content types. Site visitors can continue to visit the Tripal v2 pages. Tripal v3 content types may remain private while customization is underway. Once customization is completed a subsequent step will allow you to swap out Tripal v2 pages for the newer Tripal v3 pages. Once this step is complete, you will also be able to expose your data via Tripal 3 Web Services immediately.

1. Navigate to **Administration > Tripal > Data Storage > Chado** and click on Step 2.



Migrate

Here you may migrate Tripal v2 content types to Tripal v3 content types. The migration process is divided into four steps that allow you to migrate your site as your own pace. Please click each step below for details as to the purpose of the step. When all steps are completed your site will be fully Tripal v3 compatible.

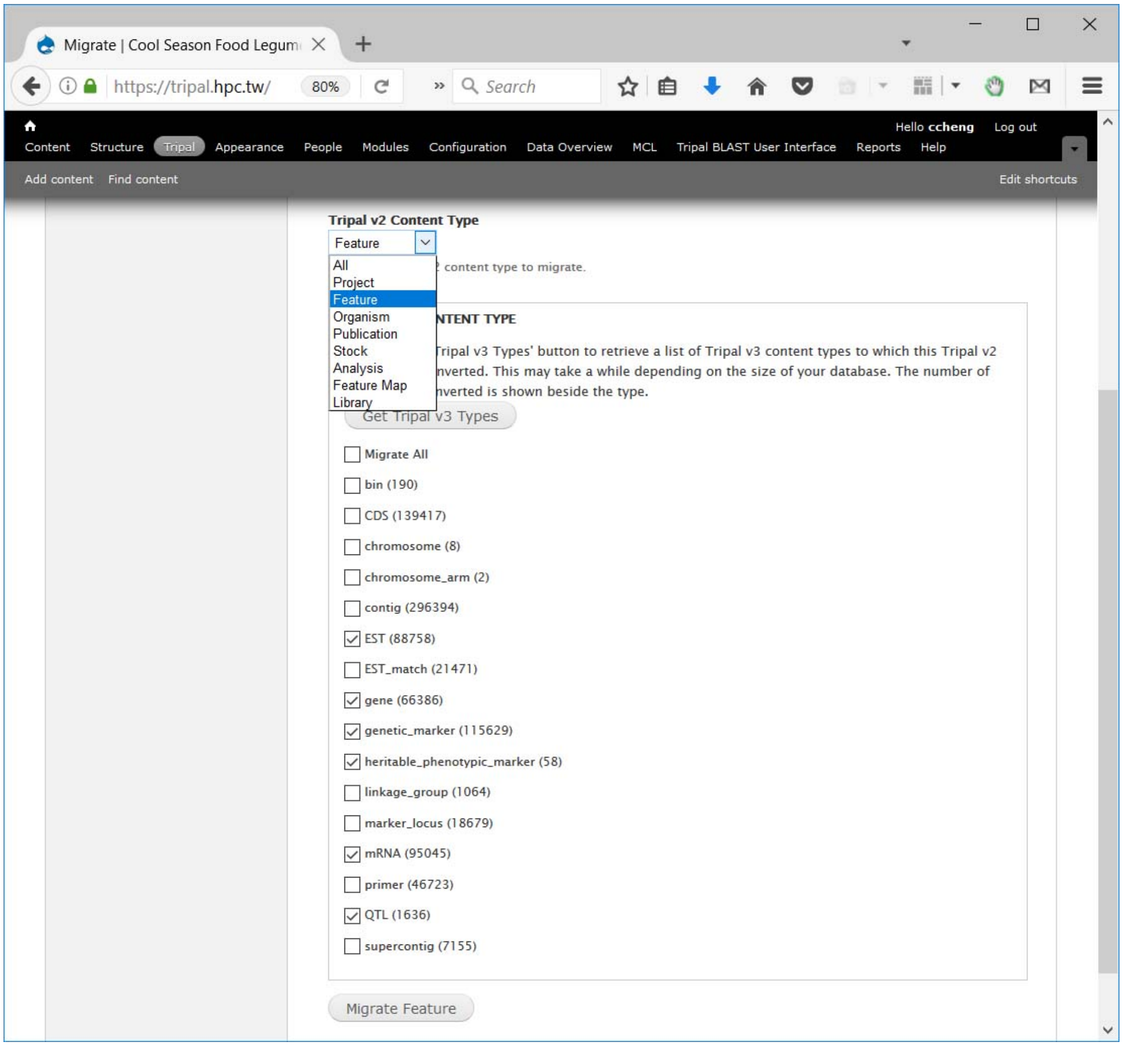
Migrate Content

Here you can migrate Tripal v2 content types to Tripal v3 content types. This will not destroy or remove existing Tripal v2 pages but will creatte new Tripal v3 pages. This allows you to keep existing pages while reviewing and customizing the Tripal v3 content types. Site visitors can continue to visit the Tripal v2 pages. Tripal v3 content types may remain private while customization is underway. Once customization is completed a subsequent step will allow you to swap out Tripal v2 pages for the newer Tripal v3 pages. If you would like to use Trial v3 web services you must migrate content types.

Tripal v2 Content Type: All

Select the Tripal v2 content type to migrate.

2. Select an individual content type to migrate from the Tripal v2 Content Type drop-down.



Tripal v2 Content Type

Feature

All
Project
Feature
Organism
Publication
Stock
Analysis
Feature Map
Library

2 content type to migrate.

NTENT TYPE

Tripal v3 Types' button to retrieve a list of Tripal v3 content types to which this Tripal v2 nverted. This may take a while depending on the size of your database. The number of nverted is shown beside the type.

Get Tripal v3 Types

☐ Migrate All

☐ bin (190)

☐ CDS (139417)

☐ chromosome (8)

☐ chromosome_arm (2)

☐ contig (296394)

☑ EST (88758)

☐ EST_match (21471)

☑ gene (66386)

☑ genetic_marker (115629)

☑ heritable_phenotypic_marker (58)

☐ linkage_group (1064)

☐ marker_locus (18679)

☑ mRNA (95045)

☐ primer (46723)

☑ QTL (1636)

☐ supercontig (7155)

Migrate Feature

3. Click the 'Get Tripal v3 Types' button to retrieve a list of Tripal v3 content types to which this Tripal v2 type can be converted. This may take a while depending on the size of your database.

4. Select the checkbox beside each Tripal v3 type you would like to create. The number of entities/pages that will be created for that content type is shown in brackets beside the name.

5. Then click the "Migrate [Tripal v2 Type]" button. This will submit a Tripal job to create the requested content. Submit this job manually on the command-line as follows (note we `cd` to the project root at `/var/www/html`: please navigate to wherever your site is installed):

```
cd $DRUPAL_HOME
drush trp-run-jobs --user=administrator
```

6. Now repeat 1-5 for each content type. Since this step simply creates new Tripal v3 content without touching the existing Tripal v2 content, there really is no reason not to migrate all your content types. Especially since the Tripal v3 content remains private and thus hidden from your users.

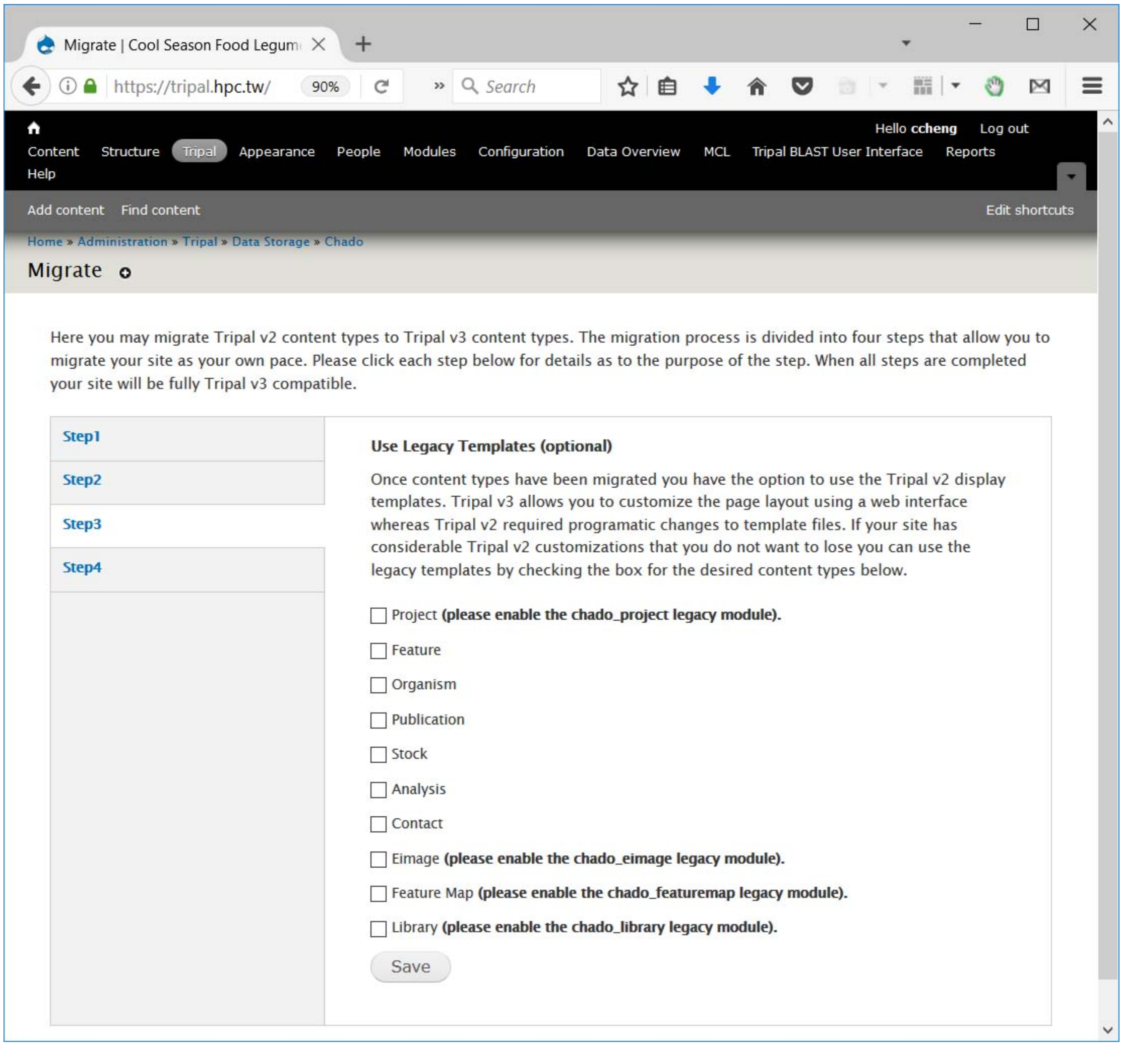
1.2.7.3 Step 3: Use Legacy Templates (optional)

This step is completely optional and not necessarily recommended. It was provided to aid the upgrade process for Tripal sites with lots of customizations who may not have the developers or time to create new Tripal 3 fields to replace their old templates.

All customizations involving re-ordering or re-naming of existing fields can now be done through the Drupal "Manage Fields" Admin interface found under **Administration > Structure > Tripal Content Types > [Type you are interested in] > "manage fields"**. You can also use this interface to switch from Tripal Panes to a long listing of content, fieldsets, tables, tabs, accordions, etc. I suggest playing around with this new interface and looking into Drupal Field Group and/or Display Suite to explore your options for customizing page display through the interface, since this will ease the transition to Drupal 8.

That said, if you decide to stick with your current customized templates, the following instructions will show you how. Keep in mind this is done on a per content type basis allowing you to do use the new interface on less customized content while still relying on your templates for highly customized content.

1. Navigate to **Administration > Tripal > Data Storage > Migrate** and click on Step 3



Migrate

Here you may migrate Tripal v2 content types to Tripal v3 content types. The migration process is divided into four steps that allow you to migrate your site as your own pace. Please click each step below for details as to the purpose of the step. When all steps are completed your site will be fully Tripal v3 compatible.

Step1
Step2
Step3
Step4

Use Legacy Templates (optional)

Once content types have been migrated you have the option to use the Tripal v2 display templates. Tripal v3 allows you to customize the page layout using a web interface whereas Tripal v2 required programatic changes to template files. If your site has considerable Tripal v2 customizations that you do not want to lose you can use the legacy templates by checking the box for the desired content types below.

□ Project **(please enable the chado_project legacy module).**

□ Feature

□ Organism

□ Publication

□ Stock

□ Analysis

□ Contact

□ Eimage **(please enable the chado_eimage legacy module).**

□ Feature Map **(please enable the chado_featuremap legacy module).**

□ Library **(please enable the chado_library legacy module).**

Save

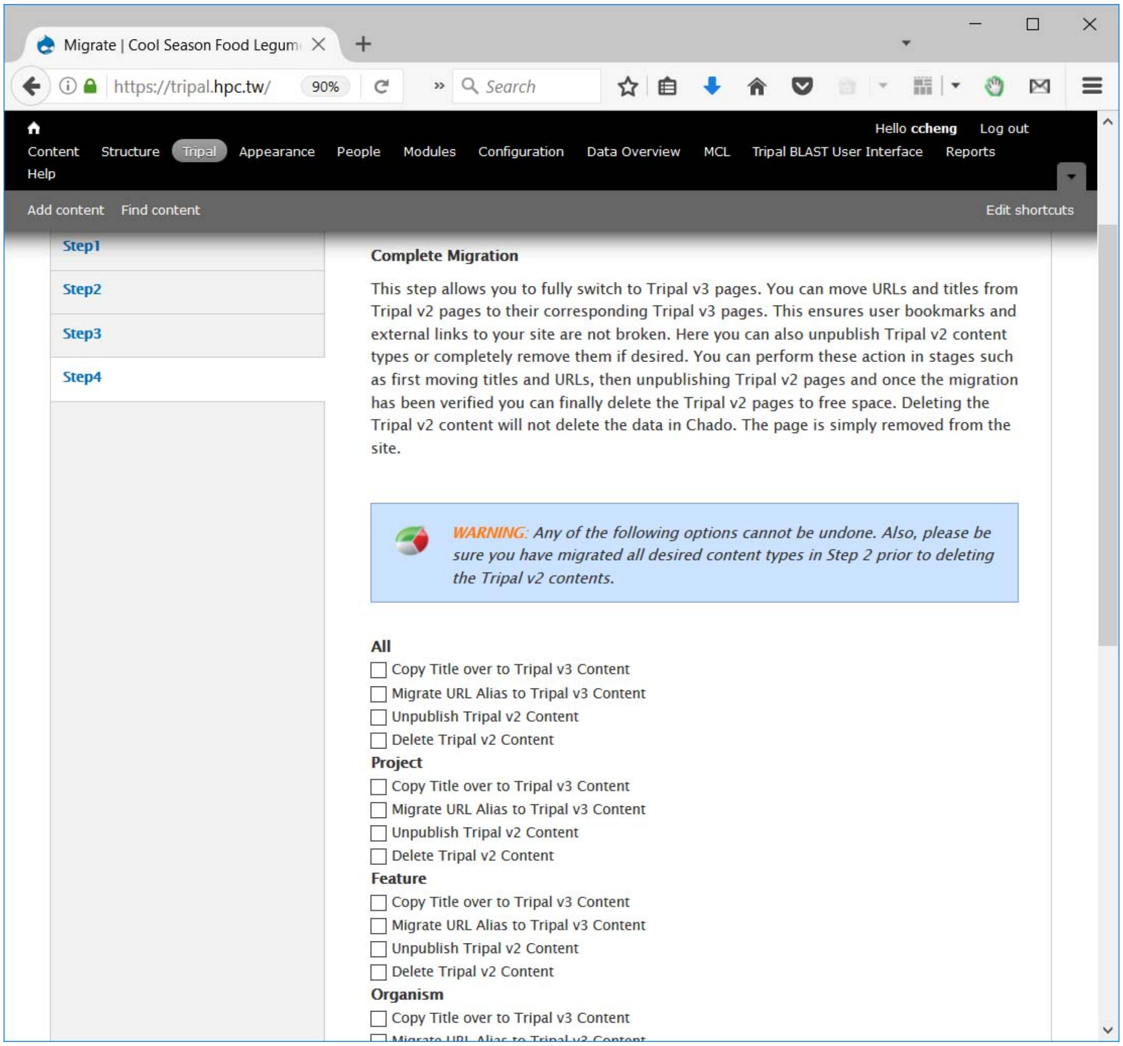
2. Click the checkbox for the Tripal v2 content types you want to keep your old templates for. Unchecked content types will use the new Tripal 3 interface.

3. Click Save.

1.2.7.4 Step 4: Delete Tripal v2 Content and Publish Tripal v3 Content

This final step allows you to fully switch to Tripal v3 pages. You can move URLs and titles from Tripal v2 pages to their corresponding Tripal v3 pages. This ensures user bookmarks and external links to your site are not broken. Here you can also unpublish Tripal v2 content types or completely remove them if desired. You can perform these actions in stages such as first moving titles and URLs, then unpublishing Tripal v2 pages and once the migration has been verified you can finally delete the Tripal v2 pages to free space. Deleting the Tripal v2 content will not delete the data in Chado. The page is simply removed from the site.

1. Navigate to **Administration > Tripal > Data Storage > Migrate** and click on Step 4



Complete Migration

This step allows you to fully switch to Tripal v3 pages. You can move URLs and titles from Tripal v2 pages to their corresponding Tripal v3 pages. This ensures user bookmarks and external links to your site are not broken. Here you can also unpublish Tripal v2 content types or completely remove them if desired. You can perform these action in stages such as first moving titles and URLs, then unpublishing Tripal v2 pages and once the migration has been verified you can finally delete the Tripal v2 pages to free space. Deleting the Tripal v2 content will not delete the data in Chado. The page is simply removed from the site.

WARNING: Any of the following options cannot be undone. Also, please be sure you have migrated all desired content types in Step 2 prior to deleting the Tripal v2 contents.

All

□ Copy Title over to Tripal v3 Content
□ Migrate URL Alias to Tripal v3 Content
□ Unpublish Tripal v2 Content
□ Delete Tripal v2 Content

Project

□ Copy Title over to Tripal v3 Content
□ Migrate URL Alias to Tripal v3 Content
□ Unpublish Tripal v2 Content
□ Delete Tripal v2 Content

Feature

□ Copy Title over to Tripal v3 Content
□ Migrate URL Alias to Tripal v3 Content
□ Unpublish Tripal v2 Content
□ Delete Tripal v2 Content

Organism

□ Copy Title over to Tripal v3 Content
□ Migrate URL Alias to Tripal v3 Content

2. Once you have confirmed that you are happy with the Tripal v3 pages for a given content type, check the desired check boxes for that content type.

3. Then click submit –This step cannot be reversed!

You have now completed the migration process and can safely disable the Tripal v2.x Legacy modules assuming no extension modules still depend on them.

Note: If you are a developer of Tripal extension modules, then the Tripal API is completely backwards compatible so any extension modules that do not interact with nodes directly can safely be made Tripal v3.x compatible by changing the module to depend on **tripal** rather then **tripal_core** (can be done in the modules .info file).

1.2.7.5 Troubleshooting

1. Dealing with `stack depth limit exceeded` on Step 4 of the Migration.

When there is a large number of nodes, Drupal's search module fails to update the search_total table and gives the following error:

```
Uncaught exception thrown in shutdown function. PDOException: SQLSTATE[54001]: Statement too complex: 7 ERROR:  stack depth limit exceeded
HINT:  Increase the configuration parameter &amp;quot;max_stack_depth
```

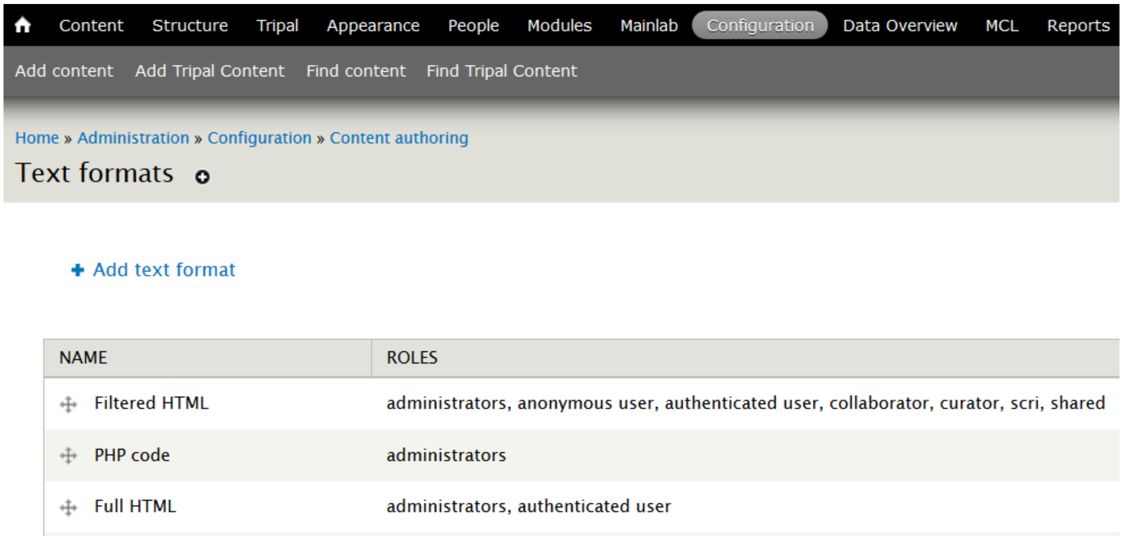
You can avoid this problem by clearing out the Drupal search tables byu executing the following SQL commands:

```
TRUNCATE search_total;
TRUNCATE search_index;
```

2. For sites that have upgrading from Drupal 6:

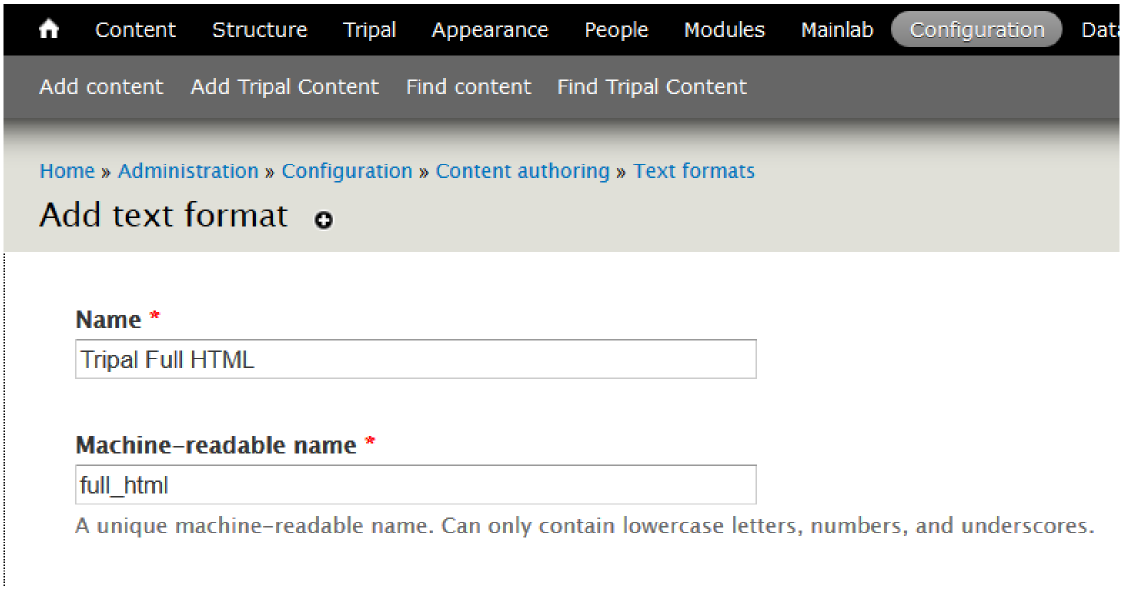
   If your site was upgraded from Drupal 6, you'll need to add a new text format with a machine name called 'full_html' as this is the default formatter that Tripal v3 uses. As in Drupal 6, the 'Full HTML' text format has a numeric machine name (usually '2') that was later changed to 'full_html' in Drupal 7.

   To do this, go to **Configuration > Text formats** in your administrative menu and click on the 'Add text format' link:



NAME	ROLES
Filtered HTML	administrators, anonymous user, authenticated user, collaborator, curator, scri, shared
PHP code	administrators
Full HTML	administrators, authenticated user

   Make sure its machine-readable_name is 'full_html' and save the configuration.



Add text format

Name *

Tripal Full HTML

Machine-readable name *

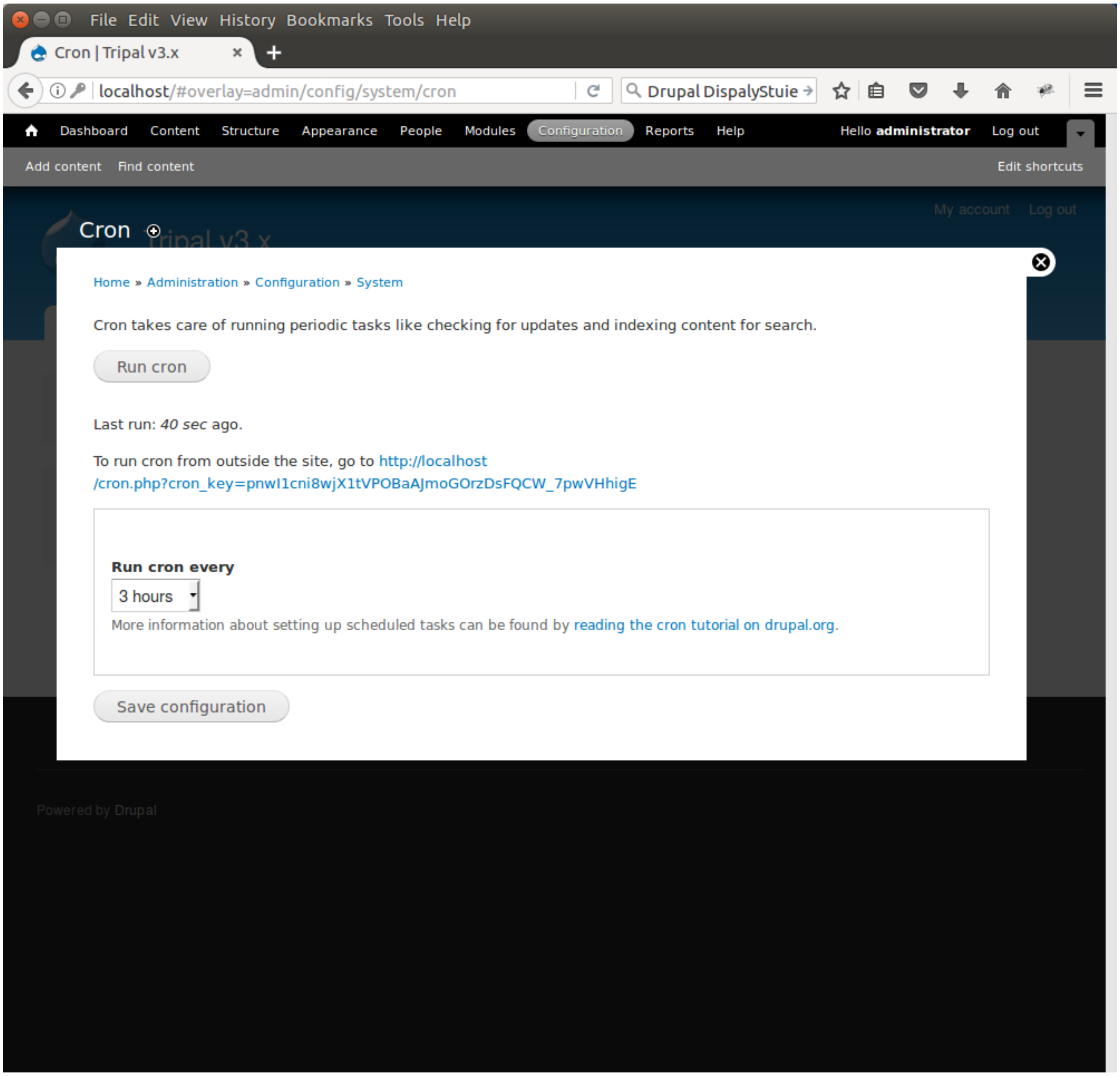
full_html

A unique machine-readable name. Can only contain lowercase letters, numbers, and underscores.

1.2.8 Automating Job Execution

Note: Remember you must set the `$DRUPAL_HOME` environment variable if you want to cut-and-paste the commands below. See *DRUPAL_HOME Variable*

The Drupal cron is used to automatically execute necessary Drupal housekeeping tasks on a regular interval. You should *always* setup the Drupal cron to ensure your site checks for updates and security issues. To do this, we want to integrate Drupal cron with the UNIX cron facility. The UNIX cron will automatically execute commands on set regular intervals. First, we must get the appropriate URL for the cron by navigating to **Configuration** → **Cron**. On this page you will see a link that we will use for cron:


File Edit View History Bookmarks Tools Help
Cron | Tripal v3.x
localhost/#overlay=admin/config/system/cron
Drupal DispalyStuie
Dashboard Content Structure Appearance People Modules Configuration Reports Help
Hello administrator Log out
Add content Find content
Edit shortcuts
My account Log out
Cron
Tripal v3.x
Powered by Drupal

Home » Administration » Configuration » System

Cron takes care of running periodic tasks like checking for updates and indexing content for search.

Run cron

Last run: *40 sec* ago.

To run cron from outside the site, go to http://localhost/cron.php?cron_key=pnwI1cni8wjX1tVPOBaAJmoGOrzDsFQCW_7pwVHhigE

Run cron every

3 hours

More information about setting up scheduled tasks can be found by reading the cron tutorial on drupal.org.

Save configuration

Warning: Be sure to edit the settings on the page and set the drop down value titled Run Cron Every to Never and save the configuration. If we do not set this to Never then Drupal will run cron when user's visit the site and that may cause slowness.

Also, on that page is the URL for cron. In this example the URL is http://localhost/cron.php?cron_key=pnwI1cni8wjX1tVPOBaAJmoGOrzDsFQCW_7pw.

To add an entry to the UNIX cron we must use the crontab tool:

```
sudo crontab -e
```

Add this line to the crontab:

```
0,30 * * * * /usr/bin/wget -O - -q http://localhost/cron.php?cron_
↪key=pnwI1cni8wjX1tVPOBaAJmoGOrzDsFQCW_7pwVHhigE
```

Now save the changes. We have now added a UNIX cron job that will occur every 30 minutes that will execute the cron.php script and cause Drupal to perform housekeeping tasks.

1.2.8.1 Automating Tripal Tasks

Many of the tasks that Tripal needs to perform can take some time to run. Examples include loading of ontologies, publishing content, and importing data. It is not practical for these tasks to run within the web browser. Therefore, Tripal provides a jobs management system. When long-running tasks need execution a job is submitted and it waits in a queue until it is executed. There are several methods that can be used to help ensure jobs can be executed in a timely manner.

Option #1: Manual Execution of Jobs

Any job that is added to the Job's system can be run manually on the command line using a Drush command. Throughout this tutorial instructions are provided to execute jobs manually. Jobs in the queue can be executed using a Drush command similar to the following:

```
drush trp-run-jobs --username=administrator --root=$DRUPAL_HOME
```

Remember to change the username from **administrator** to the name of the administrator on your site.

Option #2: Additional Cron Entry

If you do not want to manually run every job that is added to Tripal's job system you can automate execution of the job by using the same cron system that Drupal uses for housekeeping steps. To do so, use the crontab Command to add a new line to the bottom of the cron setup:

```
udo crontab -e
```

Add this line to the crontab:

```
0,5,10,15,20,25,30,35,40,45,50,55 * * * * drush trp-run-jobs --username=administrator␣
↪--root=$DRUPAL_HOME
```

Here, job execution will occur every 5 minutes.

Option #3: Tripal Daemon Setup

Tripal version 3 has incorporated the Tripal Daemon module. This module was previously an extension but is now part of the Tripal package. The Tripal Daemon module will allow jobs to execute immediately upon submission, rather than waiting on the time set in the cron setup of option #2. This can be especially useful when end-users submit jobs such as with the Tripal Blast UI module. To enable the Tripal Daemon module use the following Drush command within your Drupal installation directory:

```
drush pm-enable tripal_daemon
```

Further documentation for setup of the Tripal Daemon will appear here in the future. For now, please see the *Job Management* page for usage instructions.

1.2.9 Theming your Site

1.2.9.1 Changing the Look-and-Feel

Drupal makes it easy to change the look-and-feel of your site by providing Themes. There are hundreds of contributed themes on the Drupal.org website that can be downloaded and installed on any Drupal site to instantly change the look. Drupal comes with several themes with the default theme called Bartik. Once your Tripal site is available you can see the enabled theme and view other available themes by clicking the **Appearance** link in the administrator's menu.

1.2.9.2 Customizing a Theme

If you want to make customizations to the theme you should create your own sub theme. A sub theme borrows from an existing **base theme** (e.g. Bartik) and allows you to make your customizations. Thus, when updates for a theme are released you can easily upgrade your base theme without losing your changes. To create a sub theme, follow the instructions on the Creating a sub-theme page on the Drupal website. Alternatively, completely custom themes do not borrow from any other theme, you can create your own full-blown theme by following the Theming instructions at the Drupal website.

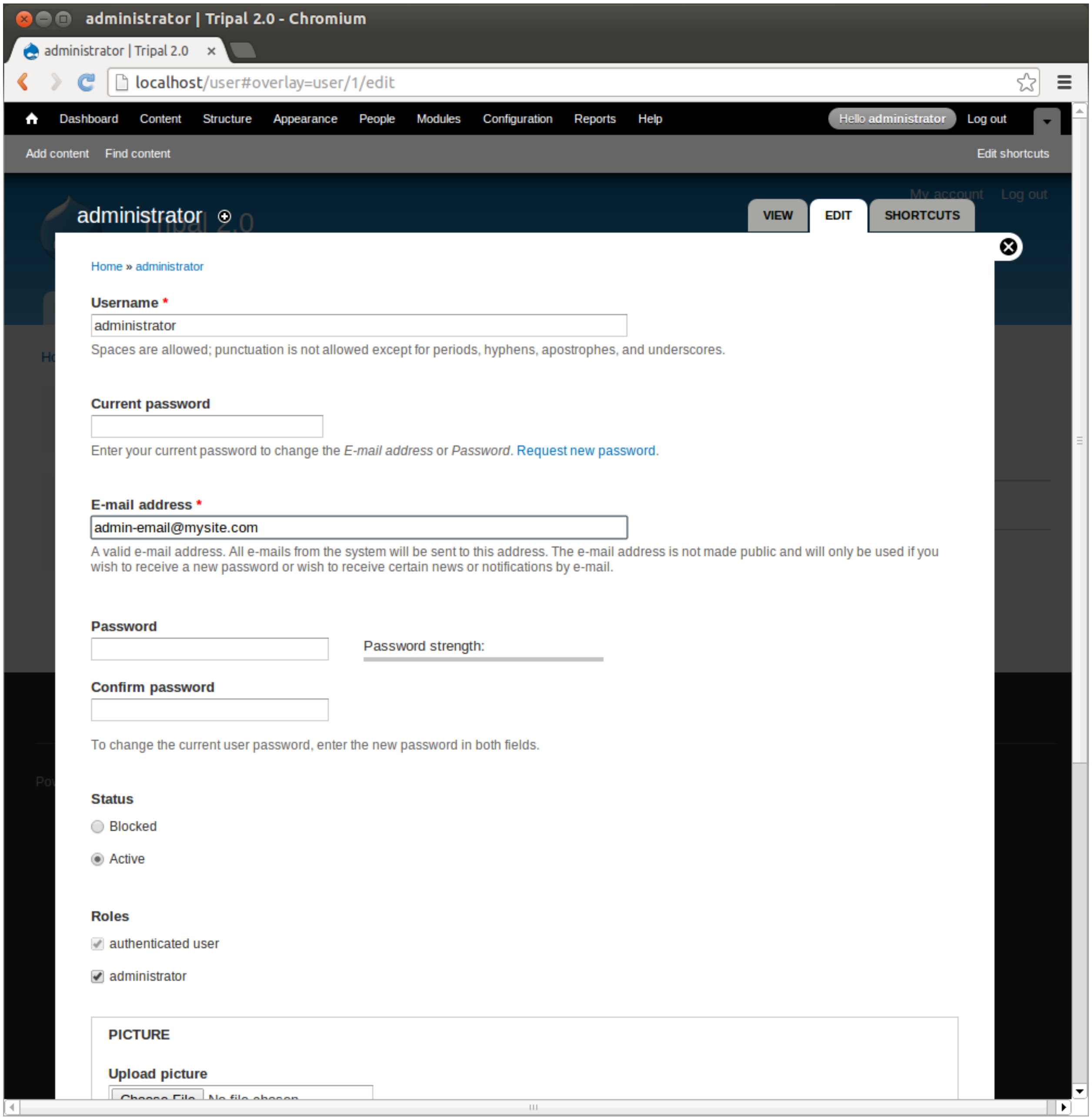
Note: You may want to explore your Tripal site first to become familiar with the functionality and where you might want to make customizations before attempting to create a custom sub theme.

1.3 Brief Drupal Overview

Note: Some of the images used on this page refer to Tripal v2. However, this Drupal overview applies equally well to both versions.

1.3.1 User Account Page

All users have an account page. When you first install Drupal, you are logged in as the administrator. The account page is simple for now. Click the **My account** link on the left sidebar. You'll see a brief history for the user and an **Edit** tab. Users can edit their own information using the edit interface:



Home » administrator

Username *

administrator

Spaces are allowed; punctuation is not allowed except for periods, hyphens, apostrophes, and underscores.

Current password

Enter your current password to change the *E-mail address* or *Password*. Request new password.

E-mail address *

admin-email@mysite.com

A valid e-mail address. All e-mails from the system will be sent to this address. The e-mail address is not made public and will only be used if you wish to receive a new password or wish to receive certain news or notifications by e-mail.

Password

Password strength:

Confirm password

To change the current user password, enter the new password in both fields.

Status

○ Blocked

● Active

Roles

☑ authenticated user

☑ administrator

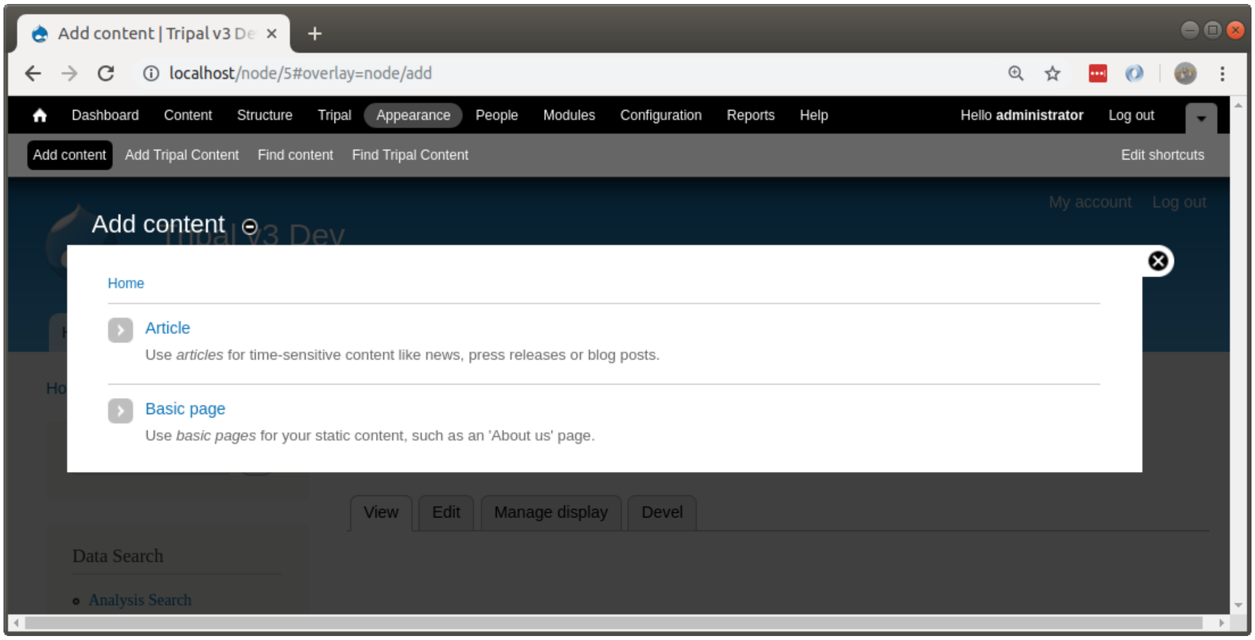
PICTURE

Upload picture

Choose File No file chosen

1.3.2 Creating Content

Creation of content in Drupal is very easy. Click the **Add content** link on the top administrative menu.



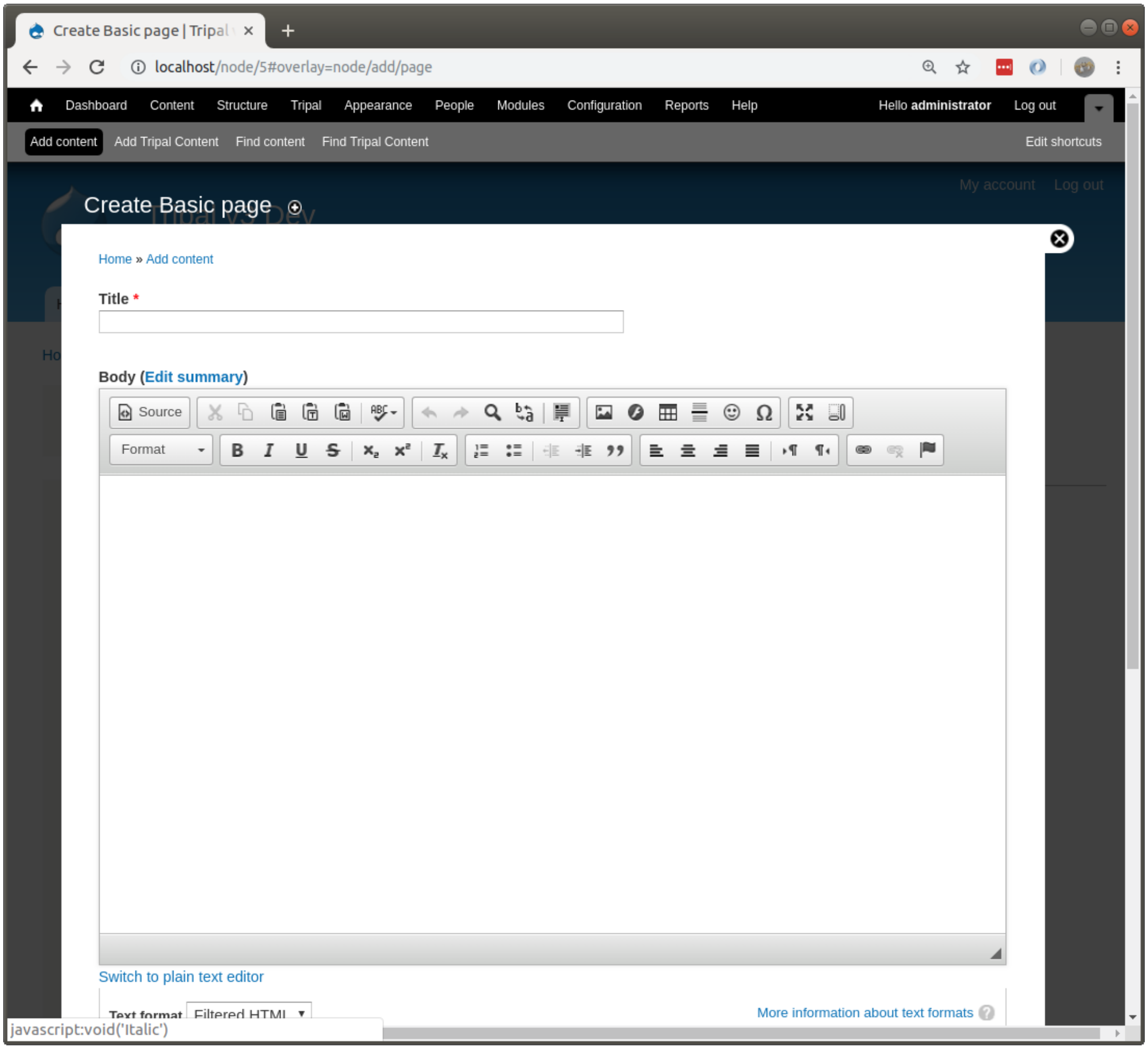
Article

Use *articles* for time-sensitive content like news, press releases or blog posts.

Basic page

Use *basic pages* for your static content, such as an 'About us' page.

You'll see two content types that come default with Drupal: Article and Basic Page. Here is where a user can add simple new pages to the website without knowledge of HTML or CSS. Click the **Basic Page** content type to see the interface for creating a new page:



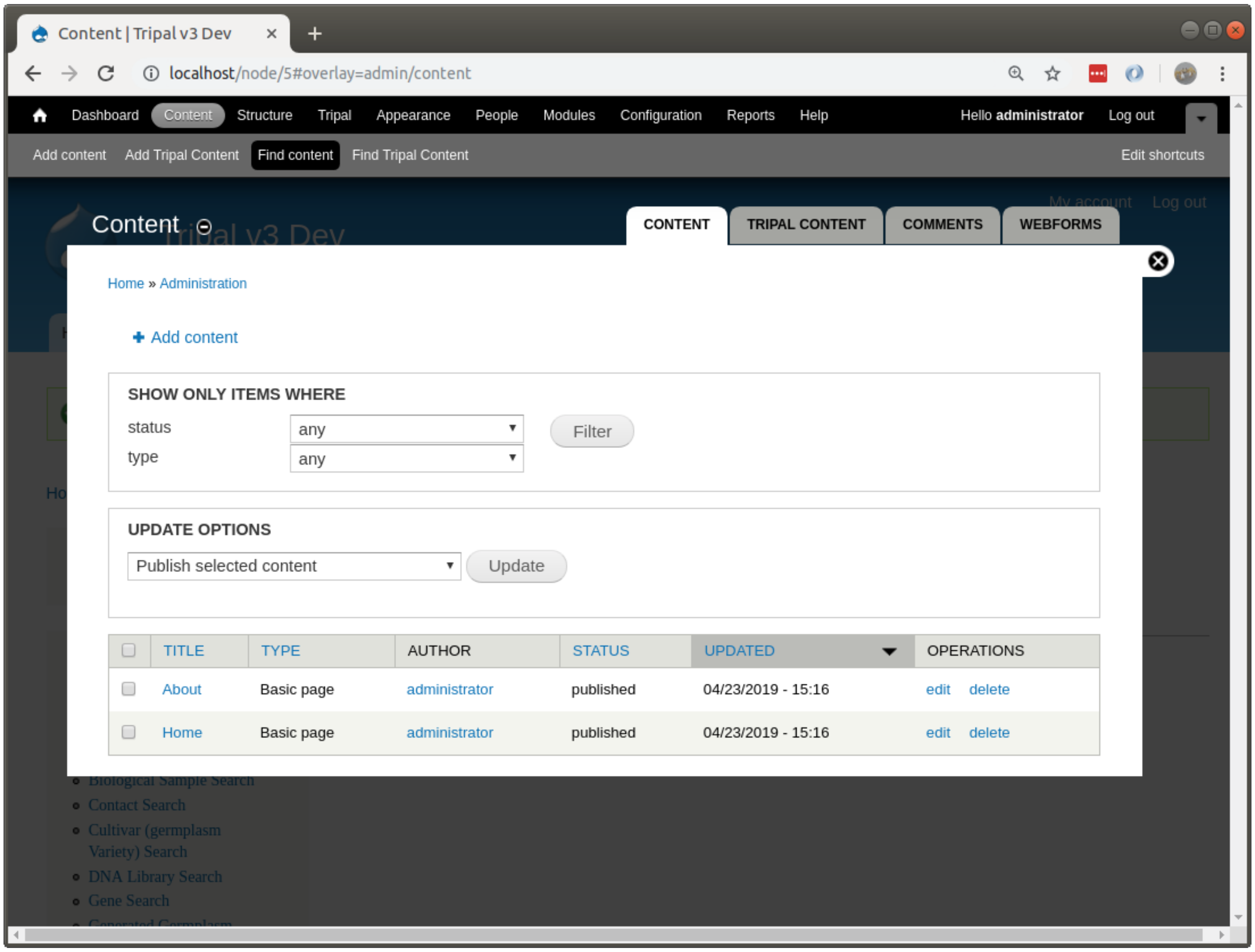
You'll notice at the top a **Title** field and a **Body** text box. All pages require a title and typically have some sort of content entered in the body. Additionally, there are other options that allow someone to enter HTML if they would like, save revisions of a page to preserve a history and to set authoring and publishing information.

For practice, try to create two new pages. A **Home** page and an **About** page for our site. First, create the home page and second create the about page. Add whatever text you like for the body.

In the screenshots above, you may have noticed the link **Add Tripal Content**. Tripal content is different from the typical Drupal content types: **Basic Page** or **Article**. Instead the content that Tripal provides is the biological and ancillary data that your site will provide to users. Later this tutorial will describe how to add new biological data.

1.3.3 Finding Content

To find any content that has been created on the site, click the **Find Content** link on the administrative menu at the top. The page shows all content available on the site. You will see the **About** and **Home** pages you created previously:



SHOW ONLY ITEMS WHERE

status: any

type: any

UPDATE OPTIONS

Publish selected content

□	TITLE	TYPE	AUTHOR	STATUS	UPDATED	OPERATIONS
□	About	Basic page	administrator	published	04/23/2019 - 15:16	edit delete
□	Home	Basic page	administrator	published	04/23/2019 - 15:16	edit delete

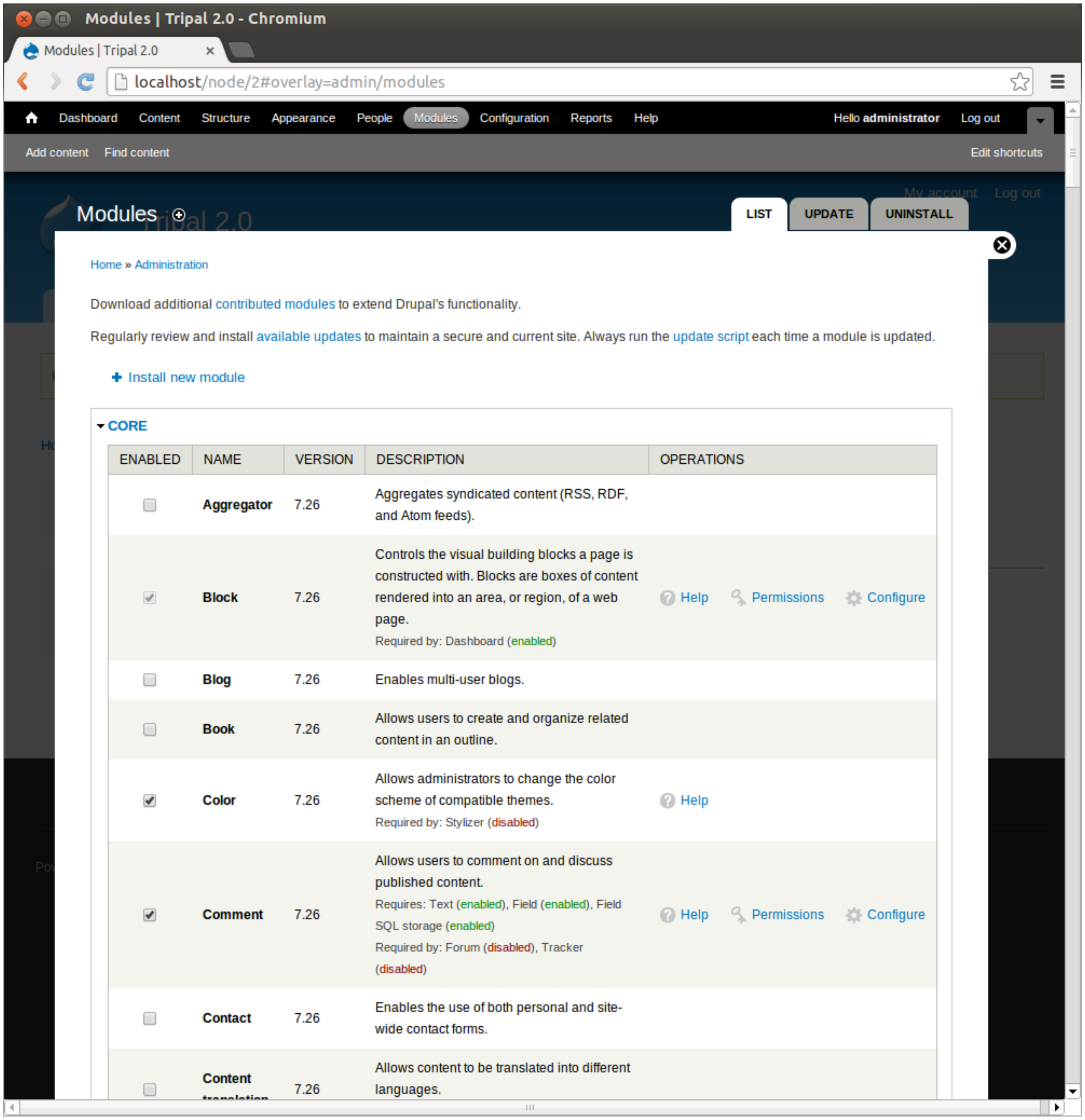
You'll also notice a set of drop down boxes for filtering the content. For sites with many different content types and pages this helps to find content. You can use this list to click to view each page or to edit.

Later in this guide, instructions will be provided for finding Tripal Content. Tripal content is different from the typical Drupal content types: **Basic Page** or **Article**. Instead the content that Tripal provides is the biological and ancillary data that your site will provide to users. It is also accessible via a link named **Find Tripal Content** found on the same menu bar as **Find Content**

1.3.4 Site Administration

1.3.4.1 Modules

Click the **Modules** link on administrative menu at the top of the page:



Home » Administration

Download additional contributed modules to extend Drupal's functionality.

Regularly review and install available updates to maintain a secure and current site. Always run the update script each time a module is updated.

✚ Install new module

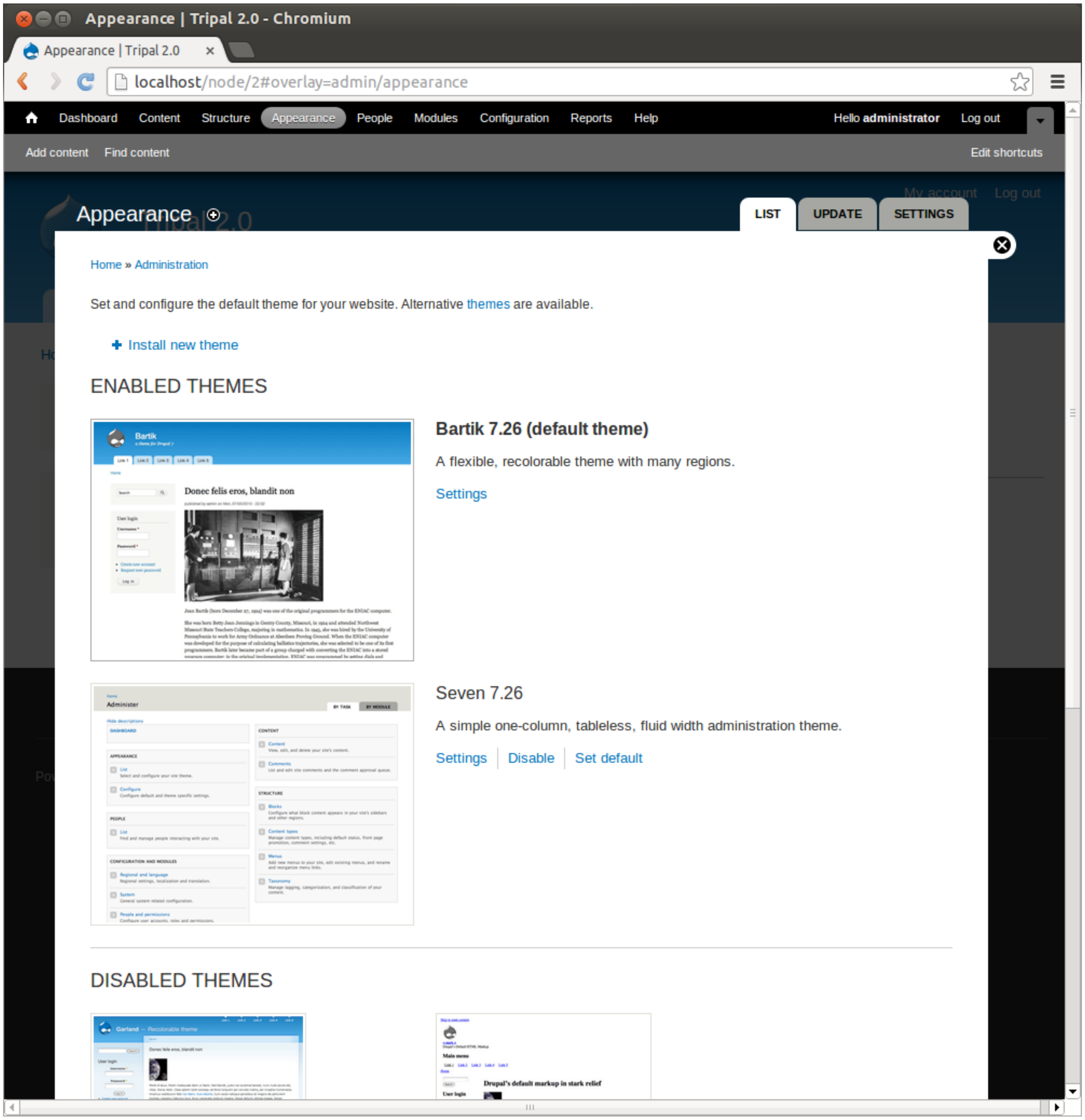
▾ CORE

ENABLED	NAME	VERSION	DESCRIPTION	OPERATIONS
□	**Aggregator**	7.26	Aggregates syndicated content (RSS, RDF, and Atom feeds).	
☑	**Block**	7.26	Controls the visual building blocks a page is constructed with. Blocks are boxes of content rendered into an area, or region, of a web page. Required by: Dashboard (enabled)	Help Permissions Configure
□	**Blog**	7.26	Enables multi-user blogs.	
□	**Book**	7.26	Allows users to create and organize related content in an outline.	
☑	**Color**	7.26	Allows administrators to change the color scheme of compatible themes. Required by: Stylizer (disabled)	Help
☑	**Comment**	7.26	Allows users to comment on and discuss published content. Requires: Text (enabled), Field (enabled), Field SQL storage (enabled) Required by: Forum (disabled), Tracker (disabled)	Help Permissions Configure
□	**Contact**	7.26	Enables the use of both personal and site-wide contact forms.	
□	**Content translation**	7.26	Allows content to be translated into different languages.	

Here is where you see the various modules that make up Drupal. Take a minute to scroll through the list and read some of the descriptions. The modules you see here are core modules that come with Drupal. Those that are checked come pre-enabled. Those that are not checked we will need to install them if we want to use them. To enable or "turn on" a module, check the box next to the desired module, then scroll to the bottom and click 'Save configuration'. Your site will now have the functionality provided by that module. Alternatively, you can search for modules that may be useful to your intended site design at the Drupal module repository, https://drupal.org/project/project_module, and install them by clicking the **Install New Module** link. Finally, a 3rd method to install modules is by use of the drush tool. We will use drush for this tutorial.

1.3.4.2 Themes

Next, click the **Appearance** link on the administrative menu at the top of the page:



Set and configure the default theme for your website. Alternative themes are available.

ENABLED THEMES

Bartik 7.26 (default theme)

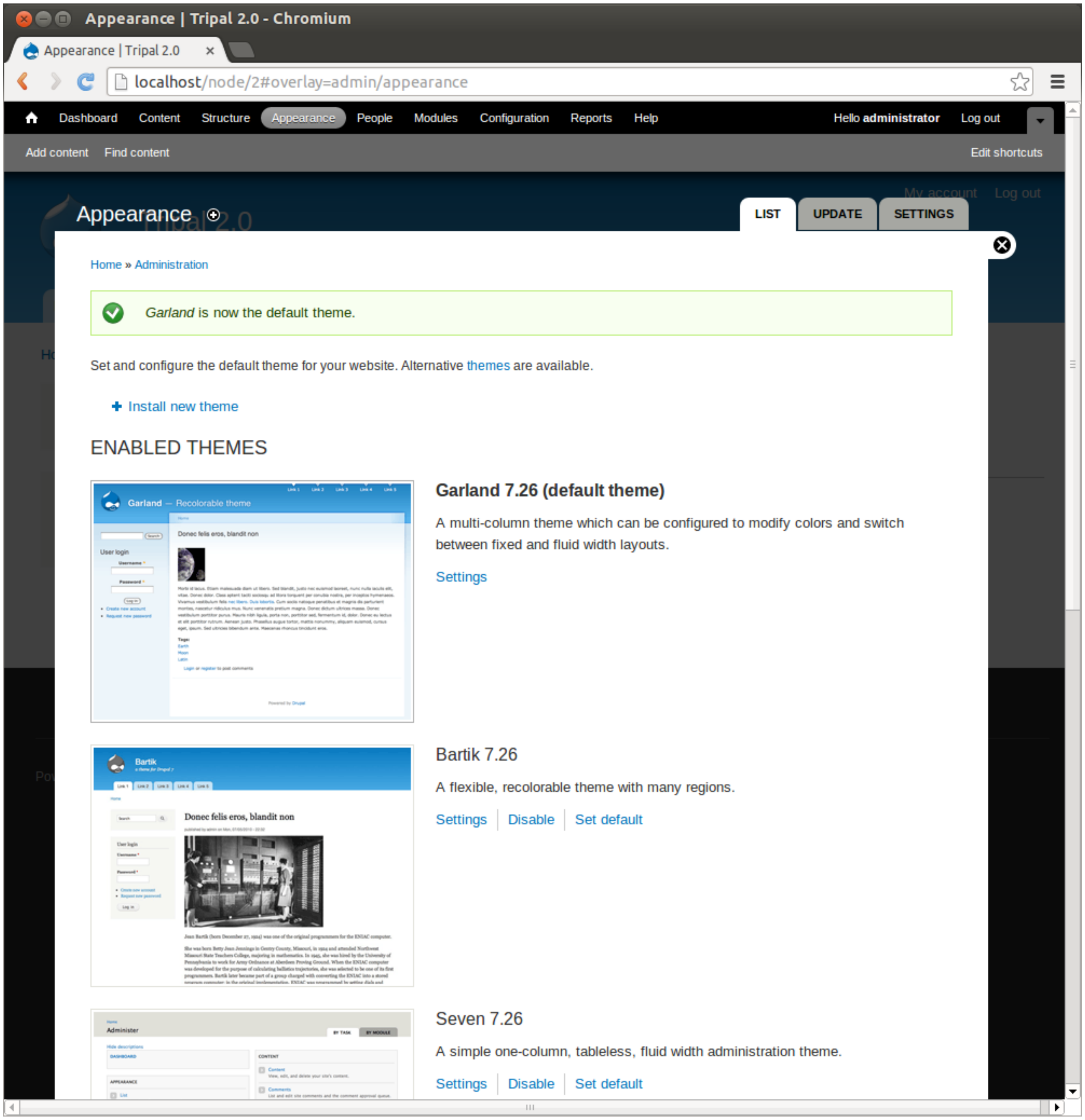
A flexible, recolorable theme with many regions.

Seven 7.26

A simple one-column, tableless, fluid width administration theme.

DISABLED THEMES

Here, you'll see a list of themes that come with Drupal by default. Here you will see the **default theme** is called **Bartik**. This theme controls the appearance of all content on the site. You can easily change the way the site looks by changing the default theme to another theme. For this tutorial, we would like to use the **Garland** theme. If you scroll down you'll see that one theme named Garland. click the link in the Garland theme section titled **Enable and set default**. The current look of the site is using the Garland theme.



Appearance

Home » Administration

Garland is now the default theme.

Set and configure the default theme for your website. Alternative themes are available.

✚ Install new theme

ENABLED THEMES

Garland 7.26 (default theme)

A multi-column theme which can be configured to modify colors and switch between fixed and fluid width layouts.

Settings

Bartik 7.26

A flexible, recolorable theme with many regions.

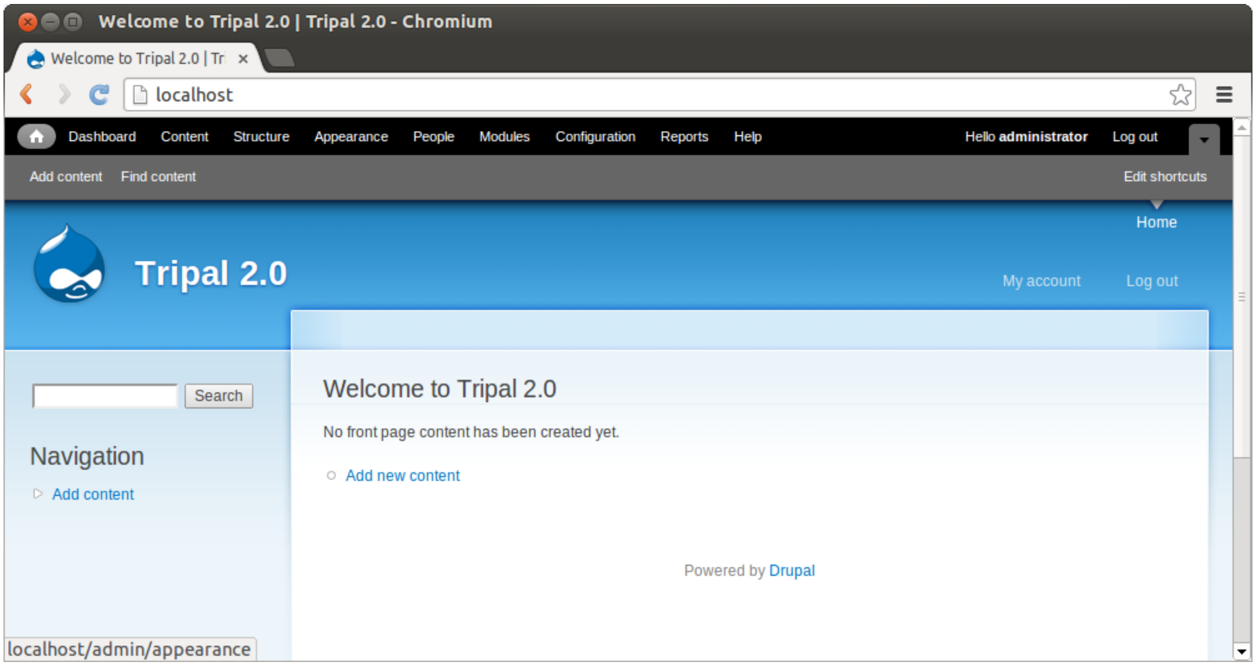
Settings | Disable | Set default

Seven 7.26

A simple one-column, tableless, fluid width administration theme.

Settings | Disable | Set default

Now, click the house icon in the top left. Our home page now uses the Garland theme:



Welcome to Tripal 2.0

No front page content has been created yet.

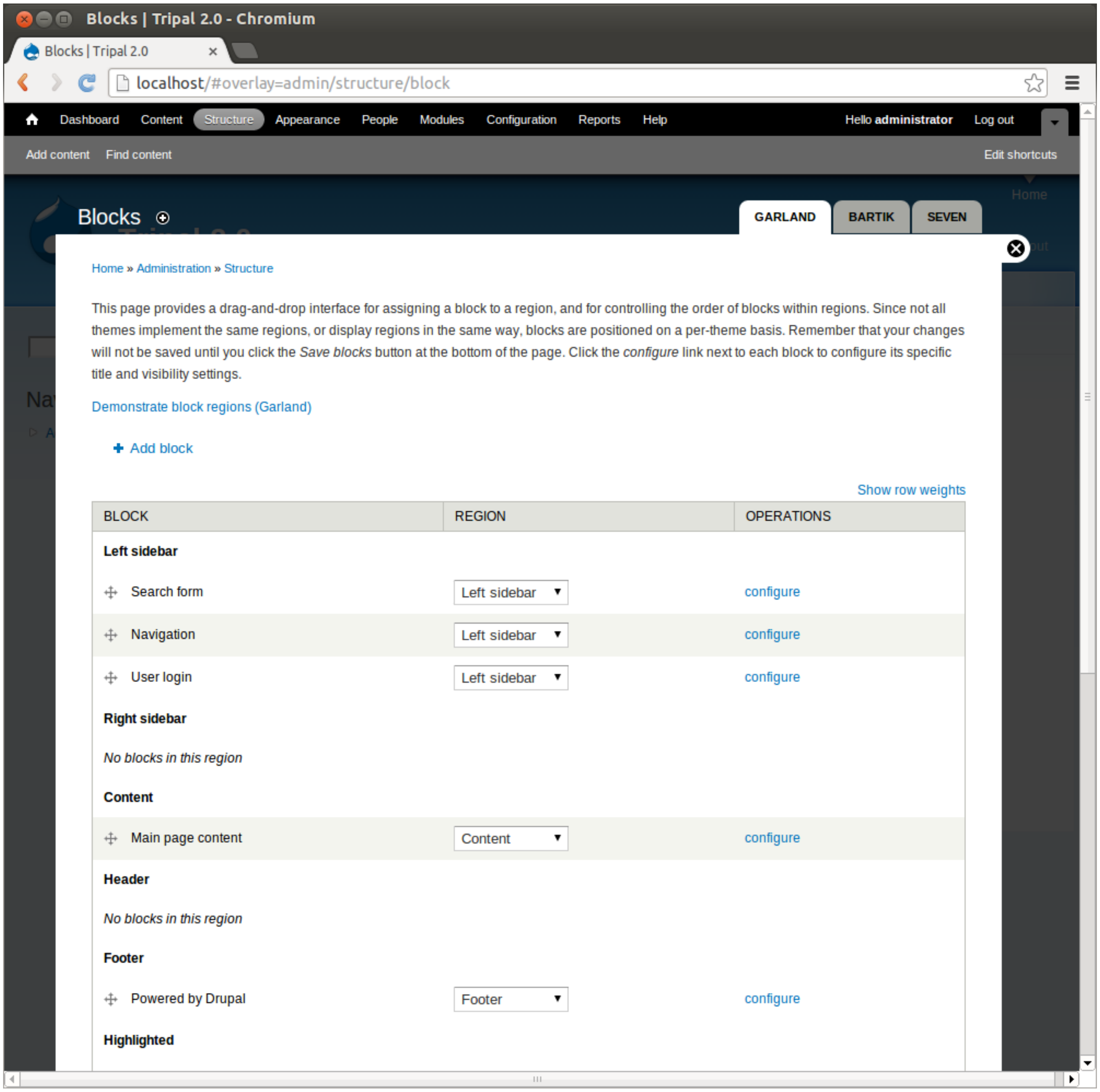
○ Add new content

Powered by Drupal

1.3.4.3 Blocks

Blocks in Drupal are used to provide additional content to any page that already exists. Examples of blocks might be a short overview of recent news items, Twitter feeds, links, recently added content, etc. The blocks interface can be found by navigating to **Structure** → **Blocks** using the top administrative menu.

On this page you'll see a list of available blocks and where they are located within the site.



Home » Administration » Structure

This page provides a drag-and-drop interface for assigning a block to a region, and for controlling the order of blocks within regions. Since not all themes implement the same regions, or display regions in the same way, blocks are positioned on a per-theme basis. Remember that your changes will not be saved until you click the *Save blocks* button at the bottom of the page. Click the *configure* link next to each block to configure its specific title and visibility settings.

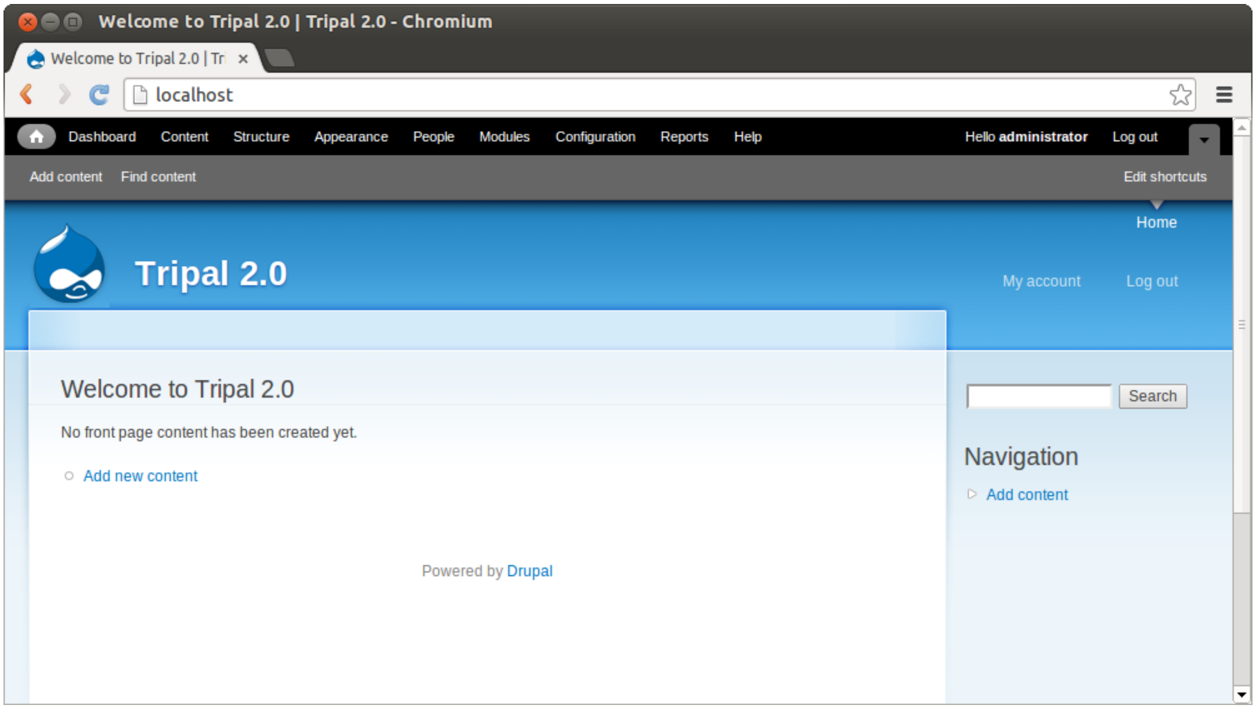
Demonstrate block regions (Garland)

+ Add block

Show row weights

BLOCK	REGION	OPERATIONS
Left sidebar		
Search form	Left sidebar ▾	configure
Navigation	Left sidebar ▾	configure
User login	Left sidebar ▾	configure
Right sidebar		
No blocks in this region		
Content		
Main page content	Content ▾	configure
Header		
No blocks in this region		
Footer		
Powered by Drupal	Footer ▾	configure
Highlighted		

Here you can see that the **Search form**, **Navigation**, and **User Login** blocks are all on the left sidebar of Garland theme. There are also a list of other regions available that do not have any blocks and there are many blocks which are Disabled but could be added to a region on the page. For this tutorial, we would like for blocks to appear on the right sidebar rather than the left sidebar. Therefore, change the **Search form**, **Navigation**, and **User Login** to all use the right sidebar by changing the drop down box next to each one. When done, click the **Save Blocks** button at the bottom. Now when we view our home page the navigation links, search form and user login box (not shown while logged in) all appear on the right side:



Welcome to Tripal 2.0

No front page content has been created yet.

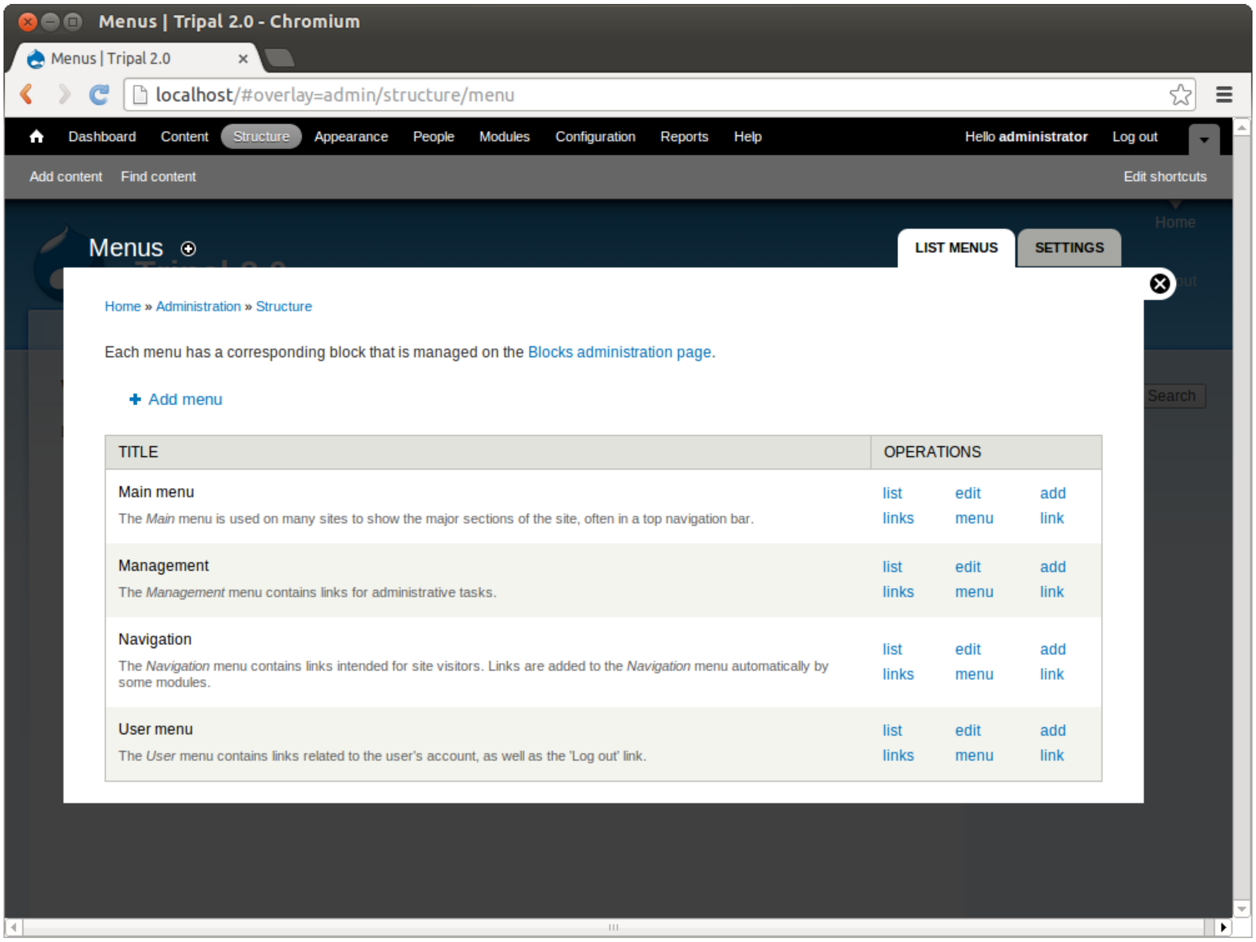
- Add new content

Powered by Drupal

1.3.4.4 Menus

For this tutorial, we want to add new links in the **Main Menu** to our new Home and About pages we created earlier. In the Garland theme, the main menu appears in the top right corner and currently only has the link ‘Home’. We want to change this link to direct to our new home page. But first, we need to find the path for our home page. The path for a page can be found in the address bar for the page. In Drupal pages of content are generally referred to as **nodes**. We can find the new home and about pages using the **Find content** link in the top administrative menu. If we click the link for our home page you’ll see the address is http://localhost/node/1. Our about page is http://localhost/node/2 (i.e the first and second pages we created).

Drupal provides an interface for working with menus, including adding new menu items to an existing menu or for creating new menus. You can find the interface for working with menus by navigating to **Structure** → **Menus** via the administrative top menu:



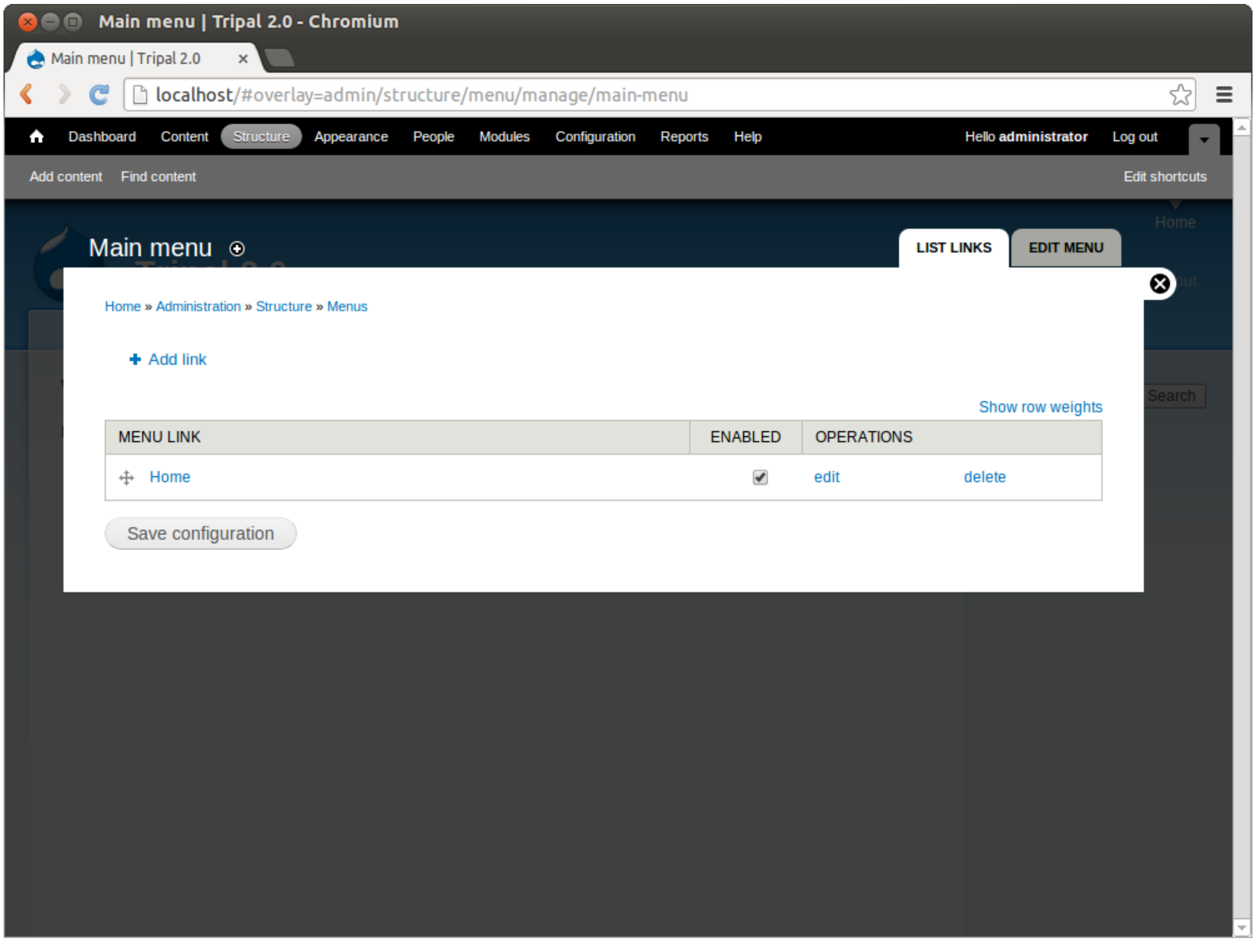
Home » Administration » Structure

Each menu has a corresponding block that is managed on the Blocks administration page.

+ Add menu

TITLE	OPERATIONS		
Main menu The *Main* menu is used on many sites to show the major sections of the site, often in a top navigation bar.	list links	edit menu	add link
Management The *Management* menu contains links for administrative tasks.	list links	edit menu	add link
Navigation The *Navigation* menu contains links intended for site visitors. Links are added to the *Navigation* menu automatically by some modules.	list links	edit menu	add link
User menu The *User* menu contains links related to the user's account, as well as the 'Log out' link.	list links	edit menu	add link

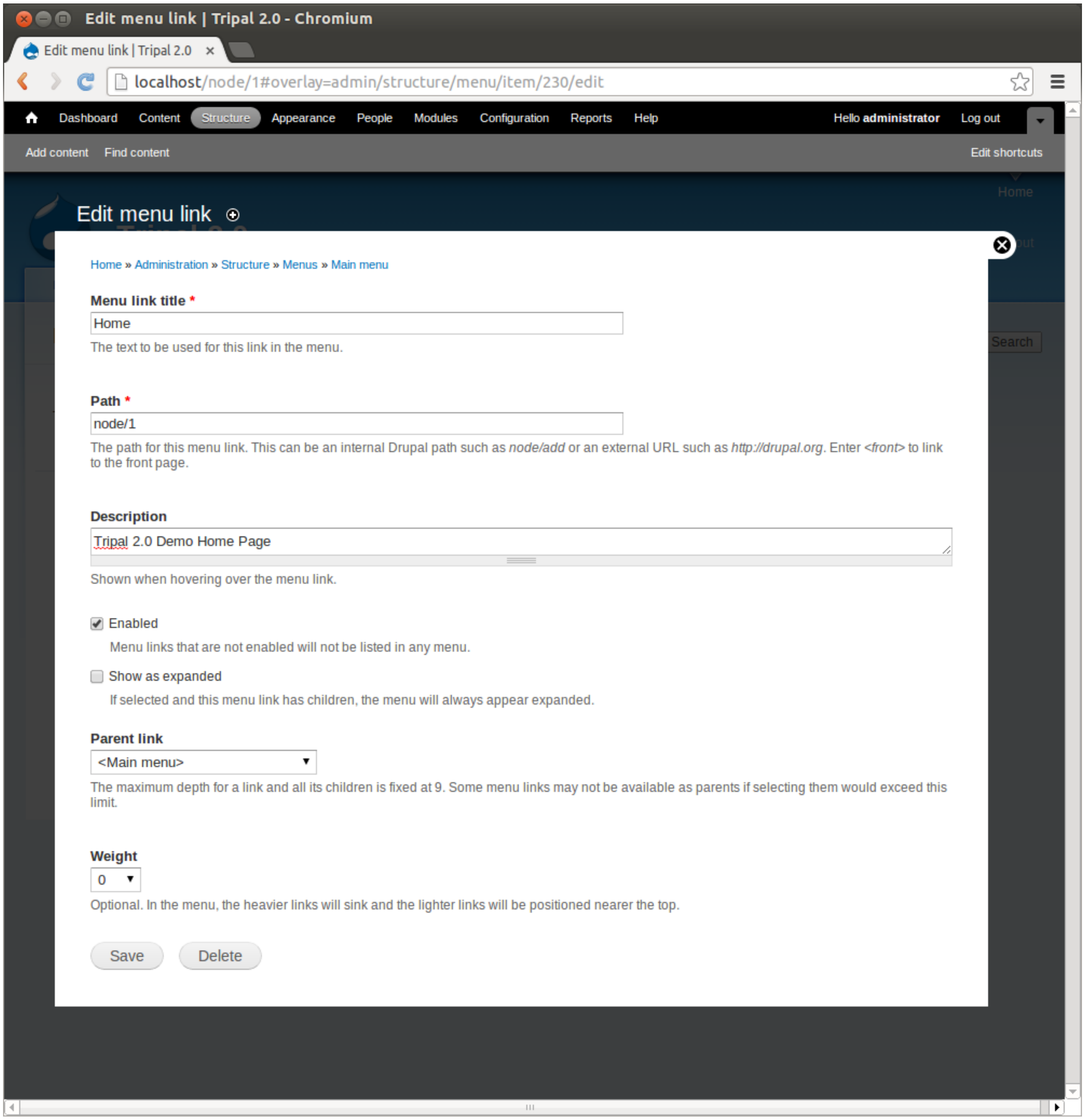
Click the link list links in the operations section for the **Main Menu**. Here we see that the **Home** link already exists:



Click edit to change the location of the Home menu item. In the form that appears, we need to set the path for our new home page. The path for each of these nodes is **node/1** and **node/2**. Fill out the form fields with these values

Form element	Value
Menu Link Title	Home
Path	node/1
Description	Tripal 2.0 Demo Home Page
Enabled	checked
Show as Expanded	no check
Parent item	<Main menu>
Weight	0

The resulting page appears as follows:



Edit menu link | Tripal 2.0 - Chromium

localhost/node/1#overlay=admin/structure/menu/item/230/edit

Edit menu link

Home » Administration » Structure » Menus » Main menu

Menu link title *

Home

The text to be used for this link in the menu.

Path *

node/1

The path for this menu link. This can be an internal Drupal path such as *node/add* or an external URL such as *http://drupal.org*. Enter *<front>* to link to the front page.

Description

Tripal 2.0 Demo Home Page

Shown when hovering over the menu link.

☑ Enabled

Menu links that are not enabled will not be listed in any menu.

☐ Show as expanded

If selected and this menu link has children, the menu will always appear expanded.

Parent link

<Main menu>

The maximum depth for a link and all its children is fixed at 9. Some menu links may not be available as parents if selecting them would exceed this limit.

Weight

0

Optional. In the menu, the heavier links will sink and the lighter links will be positioned nearer the top.

The settings above will give the menu link a title of **Home** and put it on the Main menu menu. If we then click the **Save** button at the bottom our **Home** menu item now redirects us to our new home page. Now, we also want to add a new menu item for the **About** page. Return to the **Main menu** configuration page and add a new link with the following values:

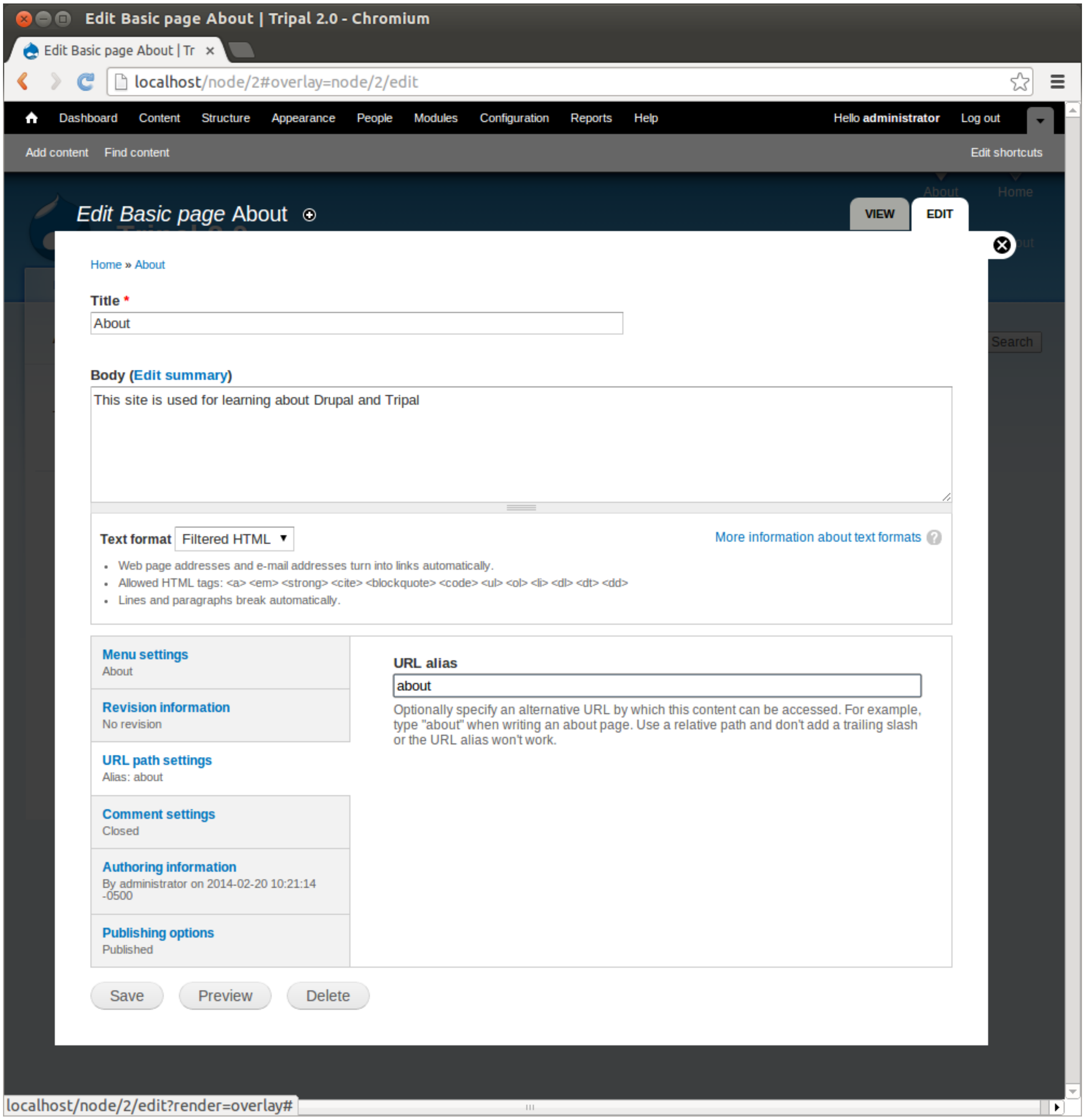
Form element	Value
Menu Link Title	About
Path	node/2
Description	About this site
Enabled	checked
Show as Expanded	no check
Parent item	<Main menu>
Weight	0

Click **Save** and a new menu item should appear. You can then change the order of the menu items by dragging and dropping the link using the cross-hairs next to each menu item.

1.3.4.5 URL Path

As mentioned previously, the URL paths for our pages have node/1 and node/2 in the address. This is not very intuitive for site visitors.

To set a path, click on our new **About** page in the new menu link at the top and click the **Edit** tab (you may have to close the overlay to see the menu item). Scroll to the bottom of the edit page and you'll see a section titled **URL path setting**. click to open this section. Since this is our about page, we simply want the URL to be http://localhost/about. To do this, just add the word "about" in the text box and click the **Save** button. You will now notice that the URL for this page is no longer http://localhost/node/2 but now http://localhost/about. Although, both links will still get you to our About page.



Edit Basic page About

Title *

About

Body (Edit summary)

This site is used for learning about Drupal and Tripal

Text format Filtered HTML

More information about text formats

- Web page addresses and e-mail addresses turn into links automatically.
- Allowed HTML tags: <a> <em> <strong> <cite> <blockquote> <code> <ul> <ol> <li> <dl> <dt> <dd>
- Lines and paragraphs break automatically.

Menu settings
About

Revision information
No revision

URL path settings
Alias: about

Comment settings
Closed

Authoring information
By administrator on 2014-02-20 10:21:14 -0500

Publishing options
Published

URL alias

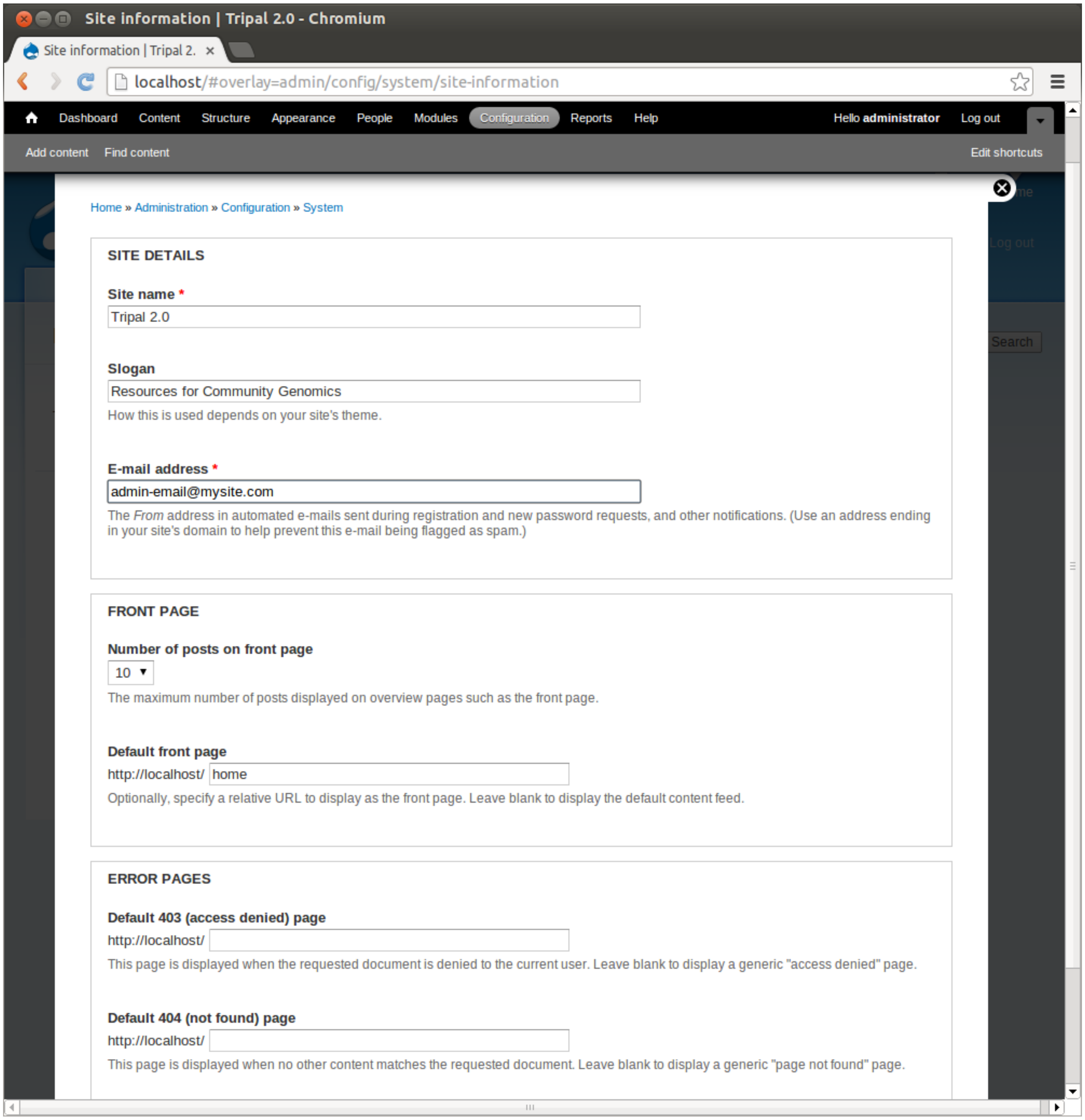
about

Optionally specify an alternative URL by which this content can be accessed. For example, type "about" when writing an about page. Use a relative path and don't add a trailing slash or the URL alias won't work.

Now, use the instructions described above to set a path of ‘home’ for our home page.

1.3.4.6 Site Configuration

There are many options under the **Configuration** link of the administrative menu at the top. Here we will only look at one of these at the moment–the **Site Information** page. Here you will find the settings we made when installing the site. You can change the site name, add a slogan, mission and footer text to the. The section titled **Front Page** is where we can tell Drupal to use our new **Home** page we created as the first page visitors see when they view the site. We want this to be the same as the home page we created and added a link for in the **Main menu**. In this text box enter the text **node/1**. Notice there is no preceding forward slash. Alternatively we could have used the URL path we added in the previous step. Let’s add a slogan: **Resources for Community Genomics**.



Home » Administration » Configuration » System

SITE DETAILS

Site name *
Tripal 2.0

Slogan
Resources for Community Genomics
How this is used depends on your site's theme.

E-mail address *
admin-email@mysite.com
The *From* address in automated e-mails sent during registration and new password requests, and other notifications. (Use an address ending in your site's domain to help prevent this e-mail being flagged as spam.)

FRONT PAGE

Number of posts on front page
10 ▾
The maximum number of posts displayed on overview pages such as the front page.

Default front page
http://localhost/ home
Optionally, specify a relative URL to display as the front page. Leave blank to display the default content feed.

ERROR PAGES

Default 403 (access denied) page
http://localhost/
This page is displayed when the requested document is denied to the current user. Leave blank to display a generic "access denied" page.

Default 404 (not found) page
http://localhost/
This page is displayed when no other content matches the requested document. Leave blank to display a generic "page not found" page.

Now, click the **Save configuration** button at the bottom. You'll now see the slogan now at the top of the page. Also, if you click the site name or the home icon at the top left we are now redirected to the new home page.

1.3.5 User Accounts

For this tutorial, we will not discuss in depth the user management infrastructure except to point out:

- User accounts can be created
- Users are assigned to various roles
- Permissions for those roles can be set to allow groups of users certain administrative rights or access to specific data.

Explore the Drupal **User Management** menu to see how users can be created, added to roles with specific permissions.

1.4 Learn Chado

Note: While you can use Tripal out-of-the-box for a whole genome or transcriptome based website, You will need a good understanding of Chado to expand into other data types and to take full advantage of Tripal

The primary data store for Tripal is Chado. Chado is an open-source database schema managed by GMOD. Chado was selected for Tripal because it is open-source, it is maintained by the community in which anyone can provide input, and use of Chado encourages common data storage between online biological sites which decreases duplication of effort.

Chado is meant to be installed into a PostgreSQL database and is designed to house a variety of biological data. For example, Tripal comes with a variety of content types. However, if you want to create new content types you must know how that data will be stored in Chado. Additionally, use of the Bulk Loader (a tab-delimited data loader for custom data formats) requires a good understanding of Chado. Finally, creating extensions to Tripal requires an understanding of Chado to write SQL and or new Tripal fields. The following links provide training for Chado.

Resource	Link
Chado Home Page	http://gmod.org/wiki/Chado>
Chado Tutorial	http://gmod.org/wiki/Main_Page>
Chado Table List	http://gmod.org/wiki/Chado_Tables>
Chado Best Practices	http://gmod.org/wiki/Chado_Best_Practices>

1.5 Working with Content Types

New in Tripal v3 is the ability to create your own content types and manage their display! In previous versions of Tripal a bit of PHP programming was necessary. This is no longer the case. Tripal v3 comes pre-populated with a variety of content types and provides a web interface that allows you to create your own. This section provides details for working with content types in Tripal.

1.5.1 Creating Content Types

Note: Prior to creating a new content type you should understand the structure of Chado and how others use Chado to store similar types of data.

Tripal v3 comes with some pre-defined content types, however you have the ability to create new Content Types through the Administrative user interface! In Tripal v3, all content types are defined by Controlled Vocabulary (CV) terms. This has a number of advantages:

1. Facilitates sharing between Tripal sites.
2. Provides a clear indication of what content is available on your site.
3. Makes content creation more intuitive from Tripal v2 (add a “Gene” rather then a “feature”).
4. Allows complete customization of what data types your site provides.
5. Integrates tightly with web services allowing Tripal to adhere to RDF specifications.

1.5.1.1 Find a Controlled Vocabulary (CV) Term

Before creating a new content type for your site you must identify a CV term that best matches the content type you would like to create. CVs are plentiful and at times selection of the correct term from the right vocabulary can be challenging. If there is any doubt about what term to use, then it is best practice to reach out to others to confirm your selection. The Tripal User community is a great place to do this by posting a description of your content type and your proposed term on the Tripal Issue Queue. Confirming your term with others will also encourage re-use across Tripal sites and improve data exchange capabilities.

The EBI's Ontology Lookup Service is a great place to locate terms from public vocabularies. At this site you can search for terms for your content type. If you can not find an appropriate term in a public vocabulary or via discussion with others then you create a new **local** term within the **local** vocabulary that comes with Tripal.

> **Warning:** Creation of **local** terms is discouraged but sometimes necessary. When creating local terms, be careful in your description.

1.5.1.2 How to Add a CV Term

Loading From an OBO File

Once you've chosen a term to describe your content type, you may need to add the term to Tripal if it is not already present. Many CVs use the OBO file format to define their terms. If the term belongs to a controlled vocabulary with a file in OBO format then you can load all the terms of the vocabulary using Tripal's OBO Loader at **Tripal** → **Data Loaders** → **Chado Vocabularies** → **Chado OBO Loader**.

Manually Adding a Term

Alternatively, you can add terms one at a time. To add a single term either from an existing vocabulary or a new local term, navigate to **Tripal** → **Data Loaders** → **Chado Vocabularies** → **Manage Chado CVs** and search to see if the vocabulary already exists. If it does you do not need to add the vocabulary. If it does not exist, click the **Add Vocabulary** link to add the vocabulary for your term. Then navigate to **Tripal** → **Data Loaders** → **Chado Vocabularies** → **Mange Chado CV Terms** then click the **Add Term link** to add the term.

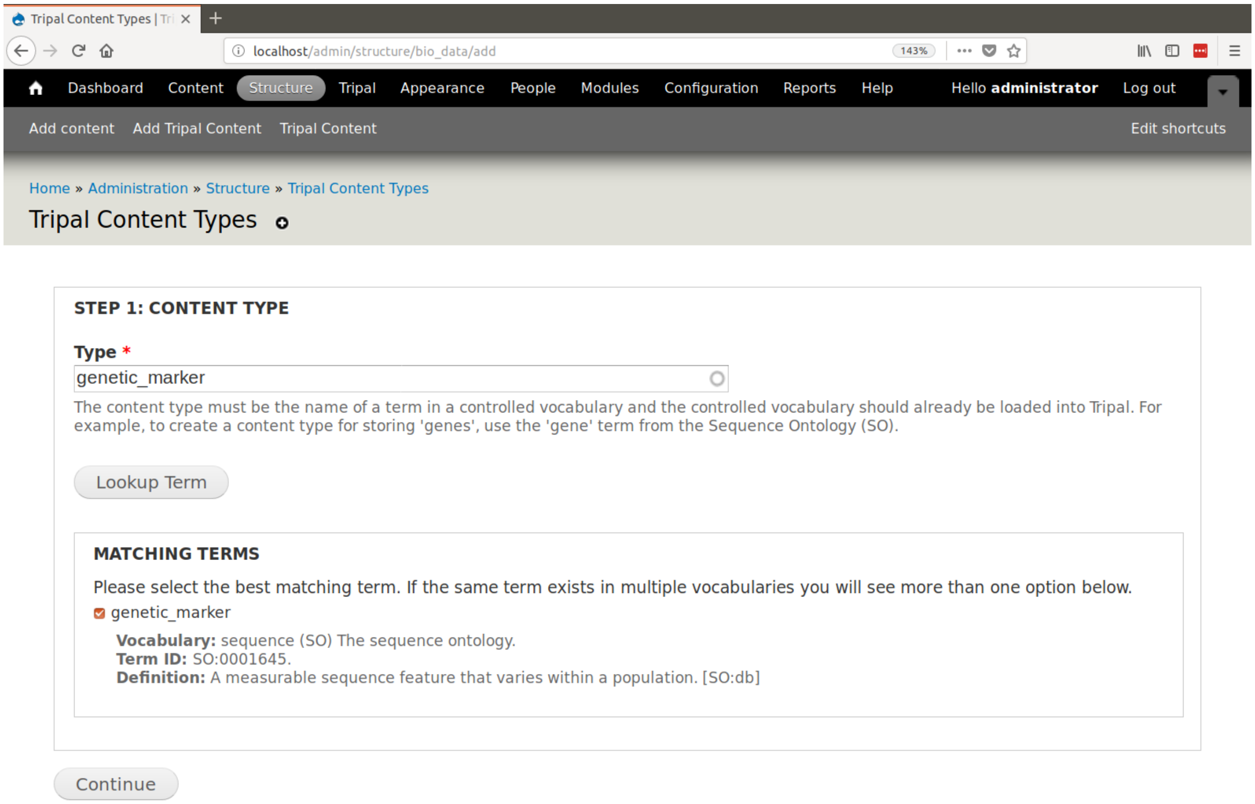
1.5.1.3 Create a Tripal Content Type

Creation of a new content type requires familiarity with Chado. This is because data records used by content types must be mapped to actual data and the data lives in Chado. Tripal's interface for creating content types allows you to provide the CV term for the type and then indicate where in Chado the data is/will be stored. Chado is a flexible relational database schema. Thus, it is possible for different sites to store data in different ways. It is best practice however to follow community standards when storing data. Therefore, please review the online documentation for Chado. If you are unclear how data for your content type should be stored in Chado please consider emailing the Chado mailing list or posting an issue on the Chado GitHub issue queue to ask for help or add a request for help on the Tripal issue queue.

To add a new content type, start by navigating to **Structure** → **Tripal Content Types** and click on the **Add Tripal Content Type** link at the top. This will take you to a web form that leads you through the process of creating a custom Tripal Content Type.

Genetic Marker Example

To demonstrate how to create a new content type we will use the example of a genetic marker. First, enter for the name of the term you would like to use to describe your content in the Content Type autocomplete textbox. Then, click **Lookup Term**. This should bring up a list of matching terms from which you can select the specific term you would like to use. Sometimes the same term exists in multiple vocabularies and you can select the proper one.



Tripal Content Types

STEP 1: CONTENT TYPE

Type *

genetic_marker

The content type must be the name of a term in a controlled vocabulary and the controlled vocabulary should already be loaded into Tripal. For example, to create a content type for storing 'genes', use the 'gene' term from the Sequence Ontology (SO).

Lookup Term

MATCHING TERMS

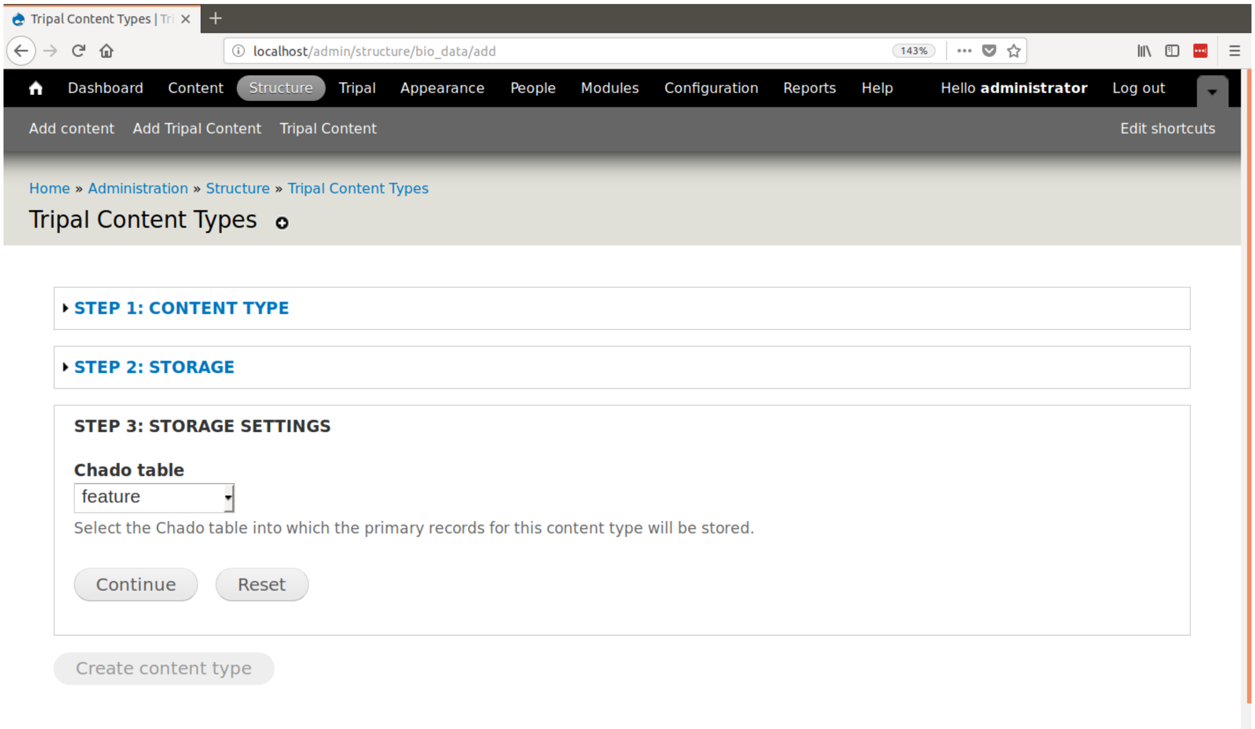
Please select the best matching term. If the same term exists in multiple vocabularies you will see more than one option below.

☑ genetic_marker

Vocabulary: sequence (SO) The sequence ontology.
Term ID: SO:0001645.
Definition: A measurable sequence feature that varies within a population. [SO:db]

Continue

During content type creation there is as a section to specify which Chado tables will store your data. Chado is typically structured with primary **base** tables (e.g. organism, feature, stock, project, etc) and a set of linker and property tables that contain ancillary data related to the base records. Here you must first choose the base table where primary records for your data type are stored. For our example, because genetic markers are sequence features, they are stored in the Chado **feature** table.



Tripal Content Types | Tri
localhost/admin/structure/bio_data/add
Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help Hello administrator Log out
Add content Add Tripal Content Tripal Content Edit shortcuts
Home » Administration » Structure » Tripal Content Types

Tripal Content Types

▸ STEP 1: CONTENT TYPE

▸ STEP 2: STORAGE

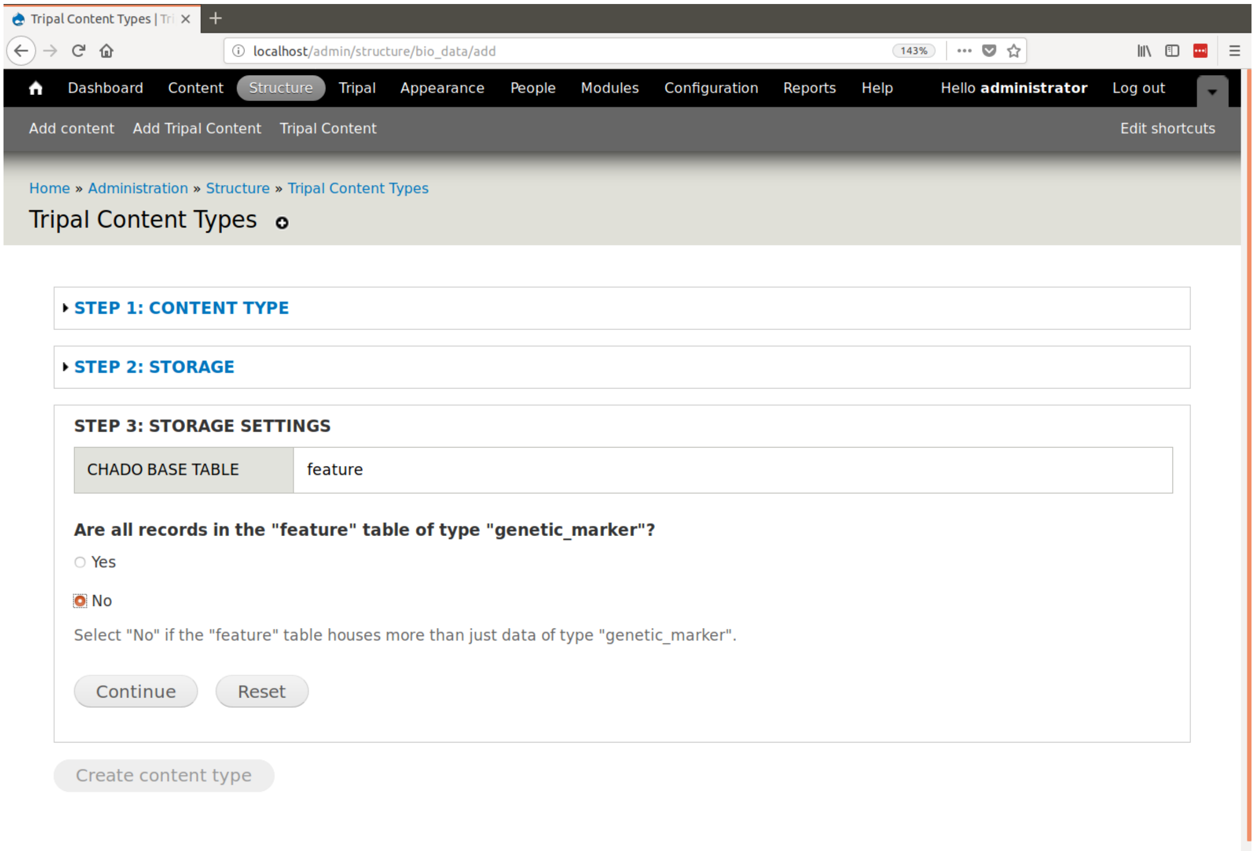
STEP 3: STORAGE SETTINGS

Chado table: feature

Select the Chado table into which the primary records for this content type will be stored.

Continue
Reset
Create content type

Next, you will be asked if all of the records in the selected table are of the desired content type. Usually the answer to this is "No", especially if the Chado table has a **type_id** column. In our case. The **feature** table does have a **type_id** column so we must select "No".



Tripal Content Types | Tri
localhost/admin/structure/bio_data/add
Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help Hello administrator Log out
Add content Add Tripal Content Tripal Content Edit shortcuts
Home » Administration » Structure » Tripal Content Types

Tripal Content Types

▸ STEP 1: CONTENT TYPE

▸ STEP 2: STORAGE

STEP 3: STORAGE SETTINGS

CHADO BASE TABLE	feature

Are all records in the "feature" table of type "genetic_marker"?

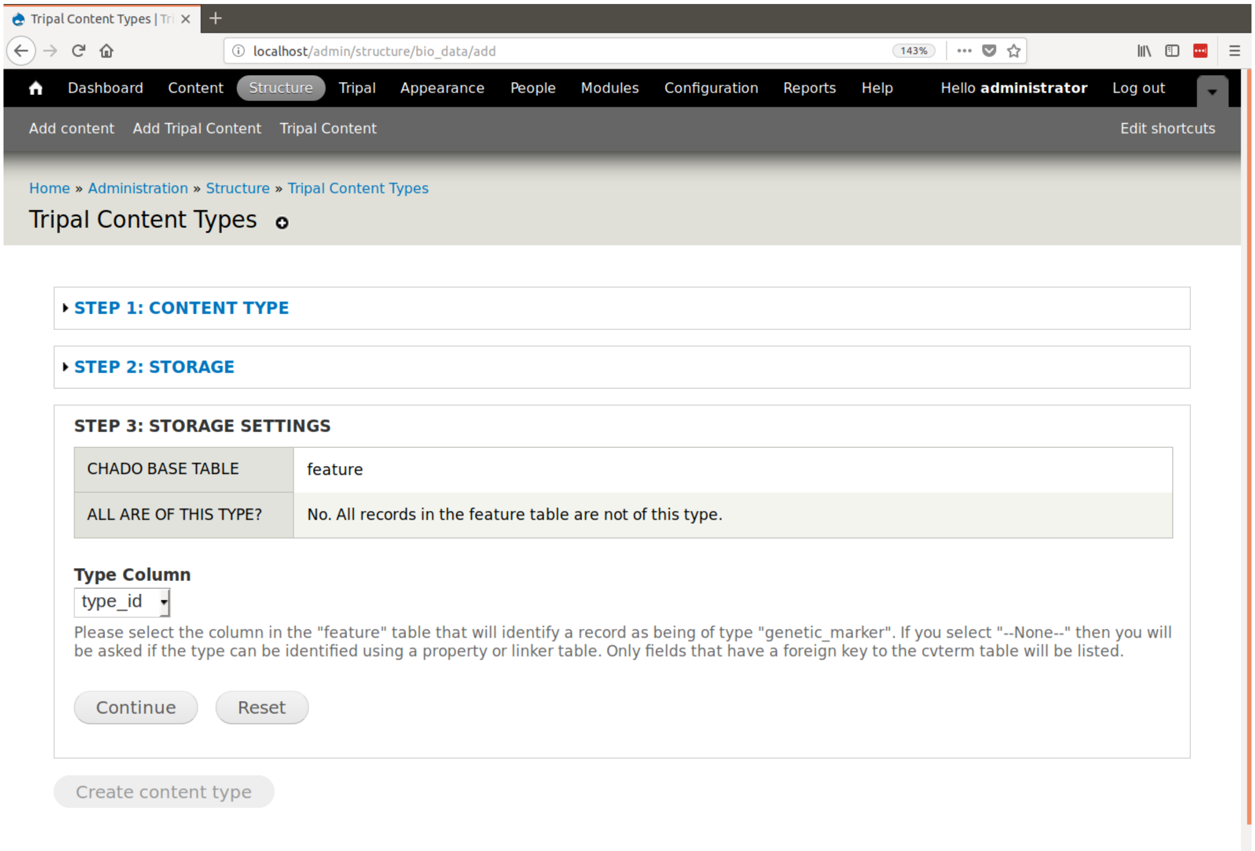
○ Yes

● No

Select "No" if the "feature" table houses more than just data of type "genetic_marker".

Continue
Reset
Create content type

If all of the records in the selected table do not belong to the content type then the form knows enough about each Chado table to offer you appropriate options for how to store data for your content type. The form knows that the **feature** table has a **type_id** column so it asks if we can differentiate records for our content type using the **type_id** field. For our example genetic marker we can do so.



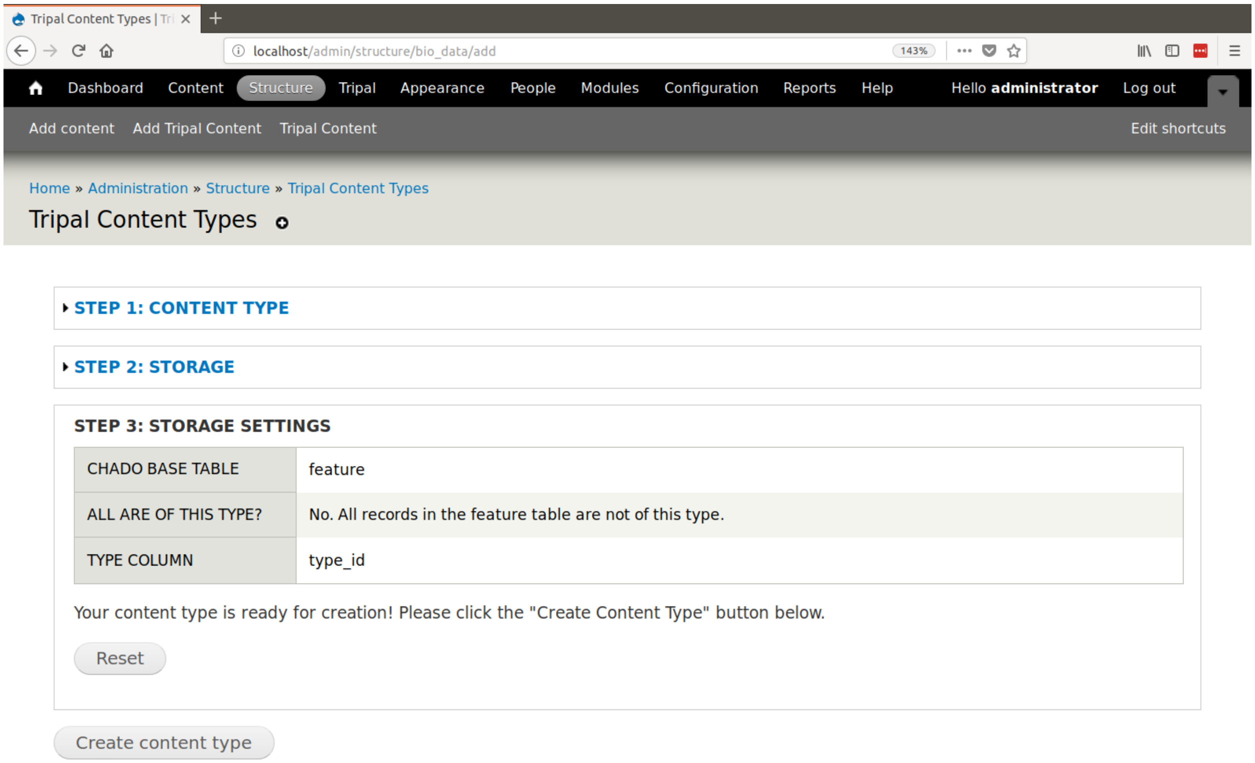
CHADO BASE TABLE	feature
ALL ARE OF THIS TYPE?	No. All records in the feature table are not of this type.

Type Column

type_id

Please select the column in the "feature" table that will identify a record as being of type "genetic_marker". If you select "--None--" then you will be asked if the type can be identified using a property or linker table. Only fields that have a foreign key to the cvterm table will be listed.

Just prior to creating our content type we are provided a summary of our selection to review:



Tripal Content Types

▸ STEP 1: CONTENT TYPE

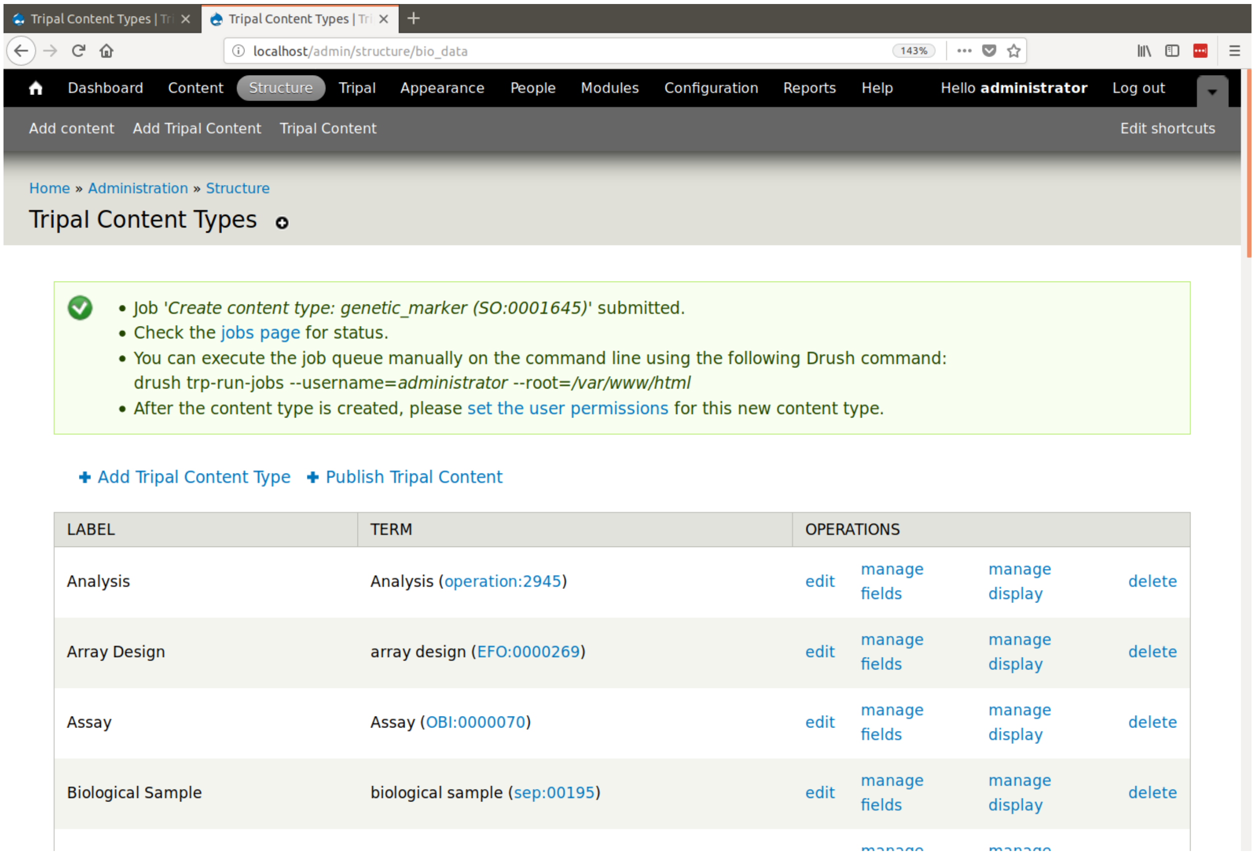
▸ STEP 2: STORAGE

STEP 3: STORAGE SETTINGS

CHADO BASE TABLE	feature
ALL ARE OF THIS TYPE?	No. All records in the feature table are not of this type.
TYPE COLUMN	type_id

Your content type is ready for creation! Please click the "Create Content Type" button below.

Finally, click the **Create Content Type** to create a custom genetic marker content type. We are provided with a message indicating that a job has been added for creation of the content type. Depending how large the Chado database is, creation of a contnet type may take awhile, hence we must run it as a job.



Home » Administration » Structure

Tripal Content Types

- Job '*Create content type: genetic_marker (SO:0001645)*' submitted.
- Check the jobs page for status.
- You can execute the job queue manually on the command line using the following Drush command: drush trp-run-jobs --username=*administrator* --root=*/var/www/html*
- After the content type is created, please set the user permissions for this new content type.

✚ Add Tripal Content Type ✚ Publish Tripal Content

LABEL	TERM	OPERATIONS			
Analysis	Analysis (operation:2945)	edit	manage fields	manage display	delete
Array Design	array design (EFO:0000269)	edit	manage fields	manage display	delete
Assay	Assay (OBI:0000070)	edit	manage fields	manage display	delete
Biological Sample	biological sample (sep:00195)	edit	manage fields	manage display	delete

Once the content type is created, you can create pages for site visitors. This will be described later in this User's Guide. In short, you can manually create new records through brand new web forms that are created automatically for your content type, or you can use a data loader to import your data directly to Chado, then **Publish** those records through the Tripal interface.

Note: Each time you create a new content type, you get several new things:

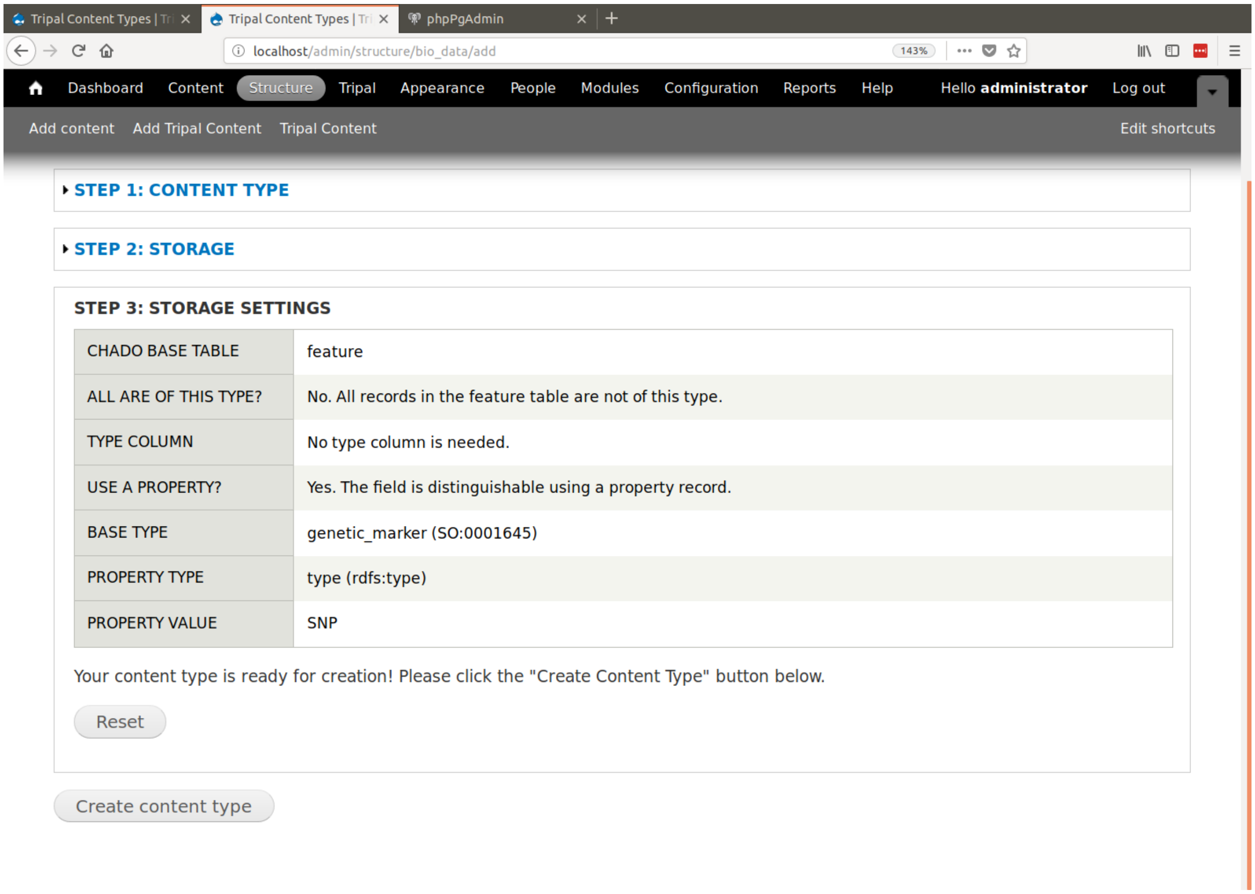
- A new search tool will be created automatically for the content type.
- A new set of permissions to help you control access is created.

SNP Example

Perhaps we want to be more specific with our genetic marker pages and create pages for each type of genetic marker (e.g. SNP, RFLP, etc. pages). Suppose for this example that we continue to store genetic markes in the feature table and use the genetic_marker term in **type_id** as in the previous example. To differentiate between different markers, we store a record in the **featureprop** table where the **featureprop.type_id** indicates the that the property provides the marker type and the **featureprop.value** column houses the string for the marker type (e.g. "SNP"). Thus, any genetic marker that has a property with this type of featureprop should form part of our SNP content type.

To accomplish this we can walk through the content type creation form and set the following values:

Field	Value
Content Type	SNP (SO:0000694)
Storage Backend	Chado
Chado Table	feature
Are all records in the `feature` table of type 'genetic_marker'?	No
Type column	–None–
Do you want to use the `featureprop` table to distinguish between content types?	Yes
Base Type	'genetic_marker' (SO:0001645)
Property Type	type (rdfs:type)
Property Value	SNP



STEP 3: STORAGE SETTINGS

CHADO BASE TABLE	feature
ALL ARE OF THIS TYPE?	No. All records in the feature table are not of this type.
TYPE COLUMN	No type column is needed.
USE A PROPERTY?	Yes. The field is distinguishable using a property record.
BASE TYPE	genetic_marker (SO:0001645)
PROPERTY TYPE	type (rdfs:type)
PROPERTY VALUE	SNP

Your content type is ready for creation! Please click the "Create Content Type" button below.

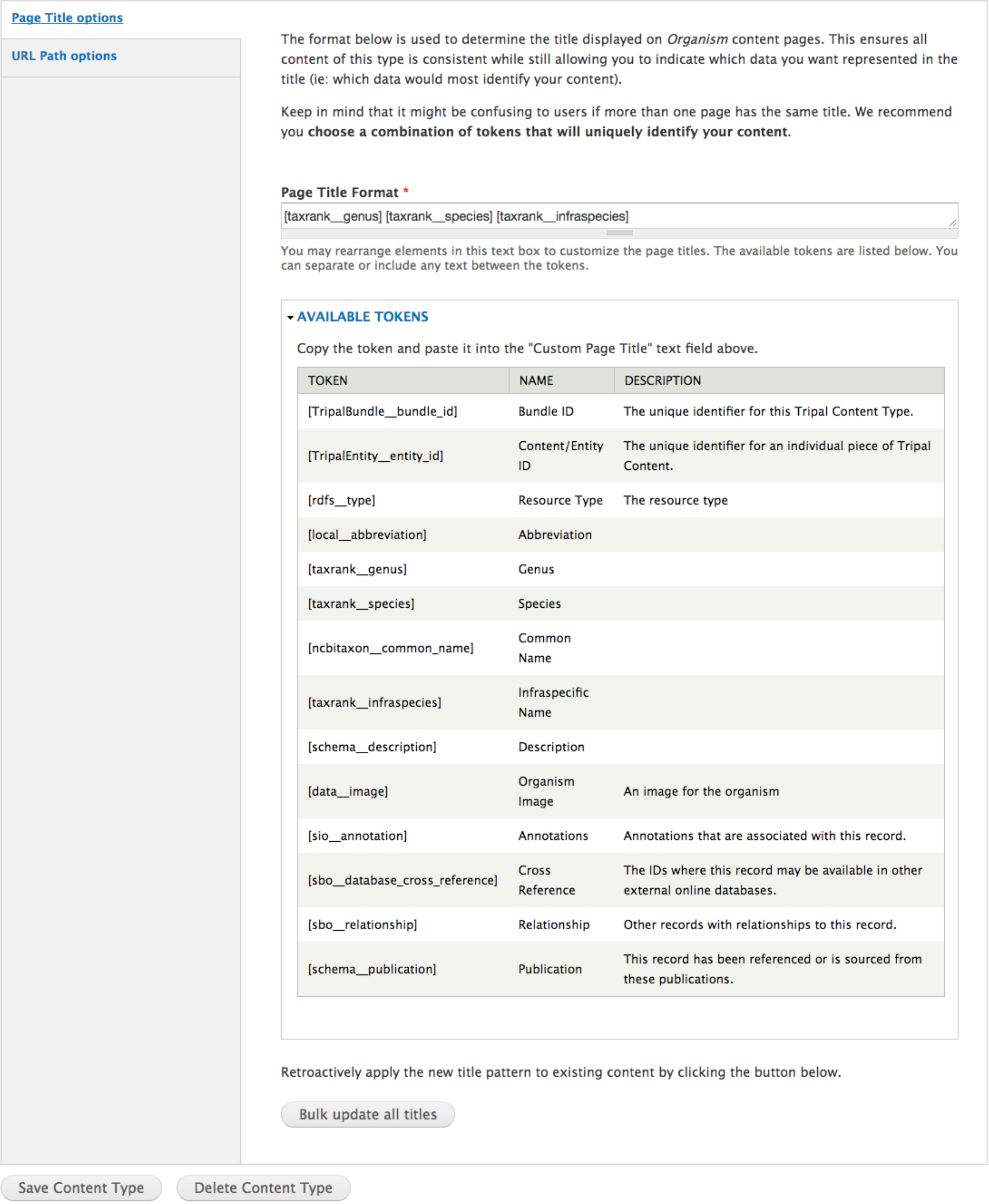
After clicking the **Create content type** button a job will be submitted and we will have a new SNP content type whose data is saved to both the feature and featureprop tables.

1.5.2 Setting Page Titles and URLs

Tripal allows for Page Titles and URLs to be set within the Tripal Content type editing interface. This provides the ability to construct consistent url patterns and titles across your site.

1.5.2.1 Setting Page Titles

Page titles can be set within the edit mechanism of the Tripal Content type. This can be found on the **Structure** → **Tripal Content Types** and click the **edit** link for the desired content type. Scroll to the bottom of the page to the **Page Title options** tab.



The format below is used to determine the title displayed on *Organism* content pages. This ensures all content of this type is consistent while still allowing you to indicate which data you want represented in the title (ie: which data would most identify your content).

Keep in mind that it might be confusing to users if more than one page has the same title. We recommend you **choose a combination of tokens that will uniquely identify your content**.

Page Title Format *

[taxrank__genus] [taxrank__species] [taxrank__infraspecies]

You may rearrange elements in this text box to customize the page titles. The available tokens are listed below. You can separate or include any text between the tokens.

▾ AVAILABLE TOKENS

Copy the token and paste it into the "Custom Page Title" text field above.

TOKEN	NAME	DESCRIPTION
[TripalBundle__bundle_id]	Bundle ID	The unique identifier for this Tripal Content Type.
[TripalEntity__entity_id]	Content/Entity ID	The unique identifier for an individual piece of Tripal Content.
[rdfs__type]	Resource Type	The resource type
[local__abbreviation]	Abbreviation	
[taxrank__genus]	Genus	
[taxrank__species]	Species	
[ncbitaxon__common_name]	Common Name	
[taxrank__infraspecies]	Infraspecific Name	
[schema__description]	Description	
[data__image]	Organism Image	An image for the organism
[sio__annotation]	Annotations	Annotations that are associated with this record.
[sbo__database_cross_reference]	Cross Reference	The IDs where this record may be available in other external online databases.
[sbo__relationship]	Relationship	Other records with relationships to this record.
[schema__publication]	Publication	This record has been referenced or is sourced from these publications.

Retroactively apply the new title pattern to existing content by clicking the button below.

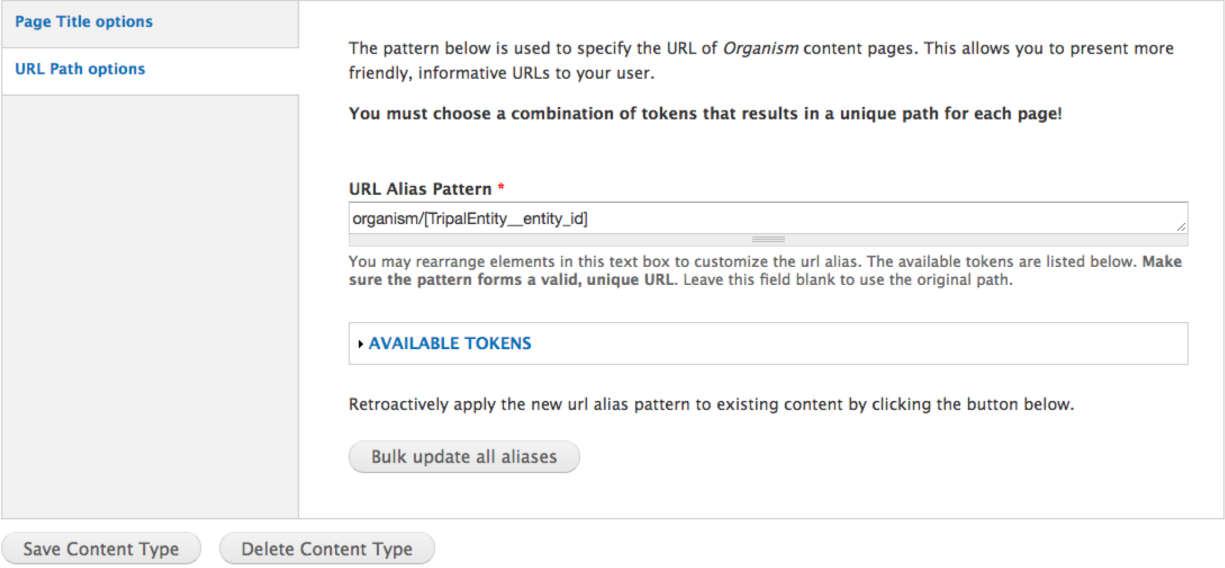
The page title format can be generated using a combination of token. When titles are generated for a new page, the tokens are replaced with the appropriate field values to which they refer. A list of available tokens can be found under the **Available Tokens** link.

Note: We recommend you choose a combination of tokens that will uniquely identify your content.

If you already have content within your site and need to update all page titles you can choose to **Bulk update all titles**. This will update all titles for the existing content that belong to this type. If your title is used to build your alias you will also need to **Bulk update all aliases**.

1.5.2.2 Setting URLs

URLs, also known as aliases, can be found just by selecting the **Page Title options** tab. The URL pattern can be generated using a combination of token. The tokens can be found under the **Available Tokens** link. If you already have content within your site and need to update all URLs you can choose to **Bulk update all aliases**. This will update all existing URLs for all pages of the content type . It will also create redirects from the old URL to the new URL to ensure 404s and broken links are not created.



The pattern below is used to specify the URL of *Organism* content pages. This allows you to present more friendly, informative URLs to your user.

You must choose a combination of tokens that results in a unique path for each page!

URL Alias Pattern *

organism/[TripalEntity__entity_id]

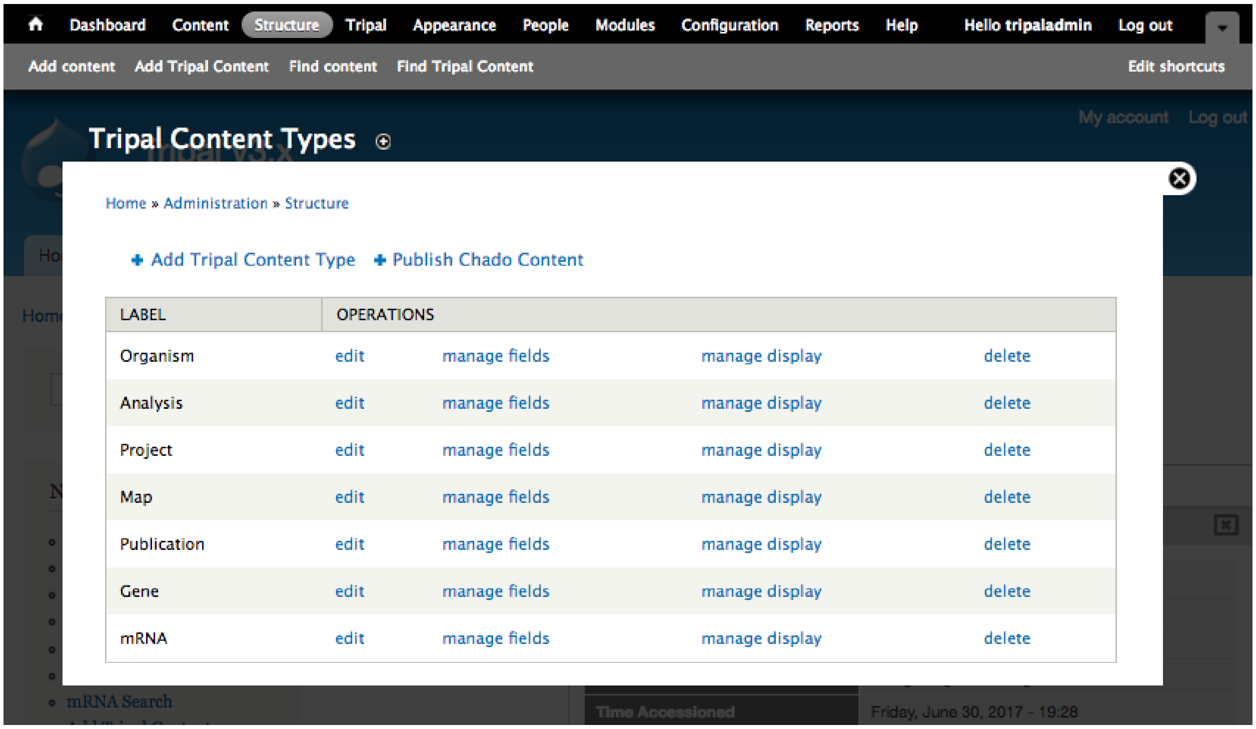
You may rearrange elements in this text box to customize the url alias. The available tokens are listed below. **Make sure the pattern forms a valid, unique URL.** Leave this field blank to use the original path.

Retroactively apply the new url alias pattern to existing content by clicking the button below.

1.5.3 Configuring Page Layout

This is one of the many new exciting features of Tripal v3. Integration with Drupal Fields has gone to a whole new level. Site builders have unprecedented control over the display of each piece of data through the administrative user interface. Previously, site builders were required to edit PHP template files to change the order, grouping or wording of content.

You can configure the display of a given Tripal Content Type by navigating to **Structure** → **Tripal Content Types** and then selecting the **Manage Display** link beside the content type you would like to configure.



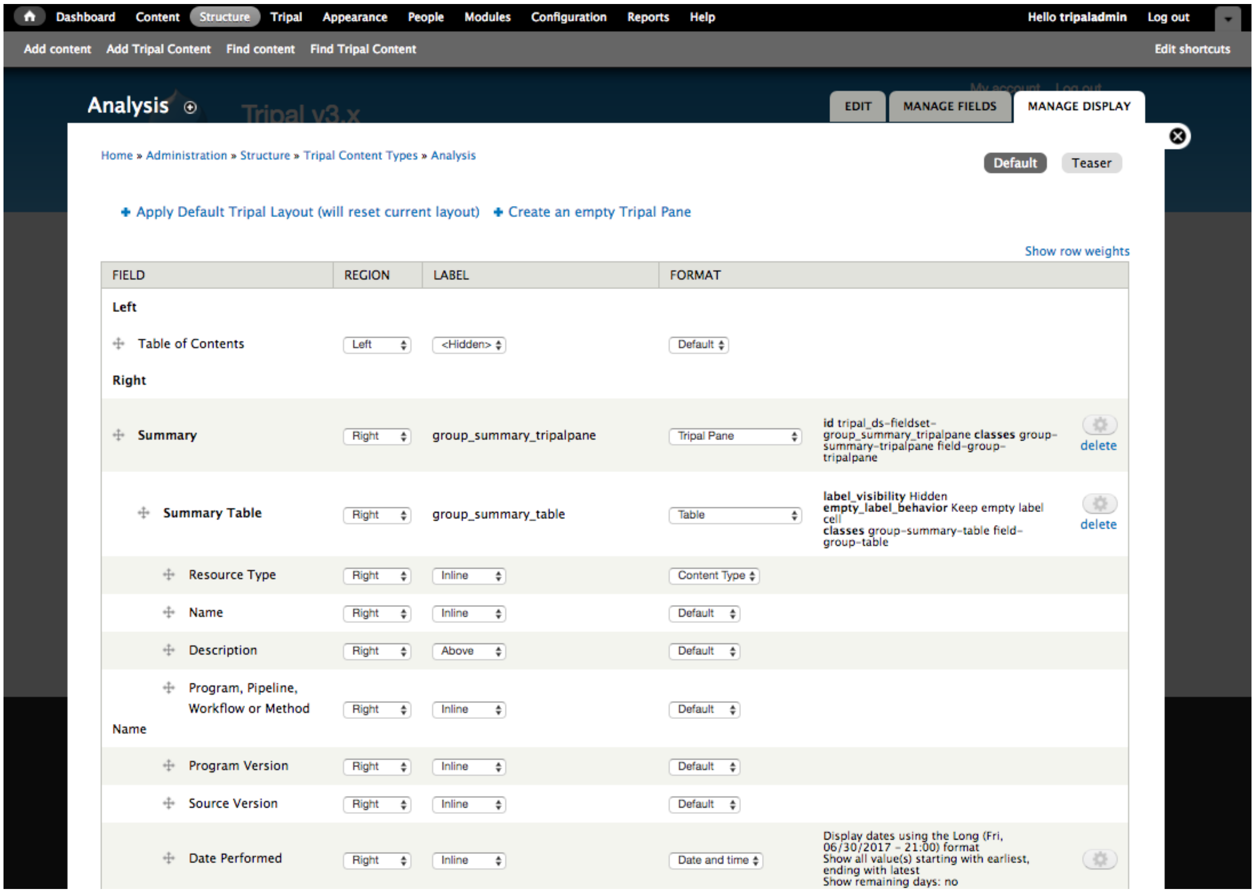
Tripal Content Types

Home » Administration » Structure

✚ Add Tripal Content Type ✚ Publish Chado Content

LABEL	OPERATIONS			
Organism	edit	manage fields	manage display	delete
Analysis	edit	manage fields	manage display	delete
Project	edit	manage fields	manage display	delete
Map	edit	manage fields	manage display	delete
Publication	edit	manage fields	manage display	delete
Gene	edit	manage fields	manage display	delete
mRNA	edit	manage fields	manage display	delete

The Manage Display User Interface lists each Drupal Field in the order they will be displayed on the page. Fields are grouped into Tripal Panes by the Tripal DS module and the page is automatically divided into a right and left column. By default the left column contains the table of contents which lists the Tripal Panes available to the user in the order they are listed in this UI. The following screenshots are using the Analysis Content Type for demonstatration.



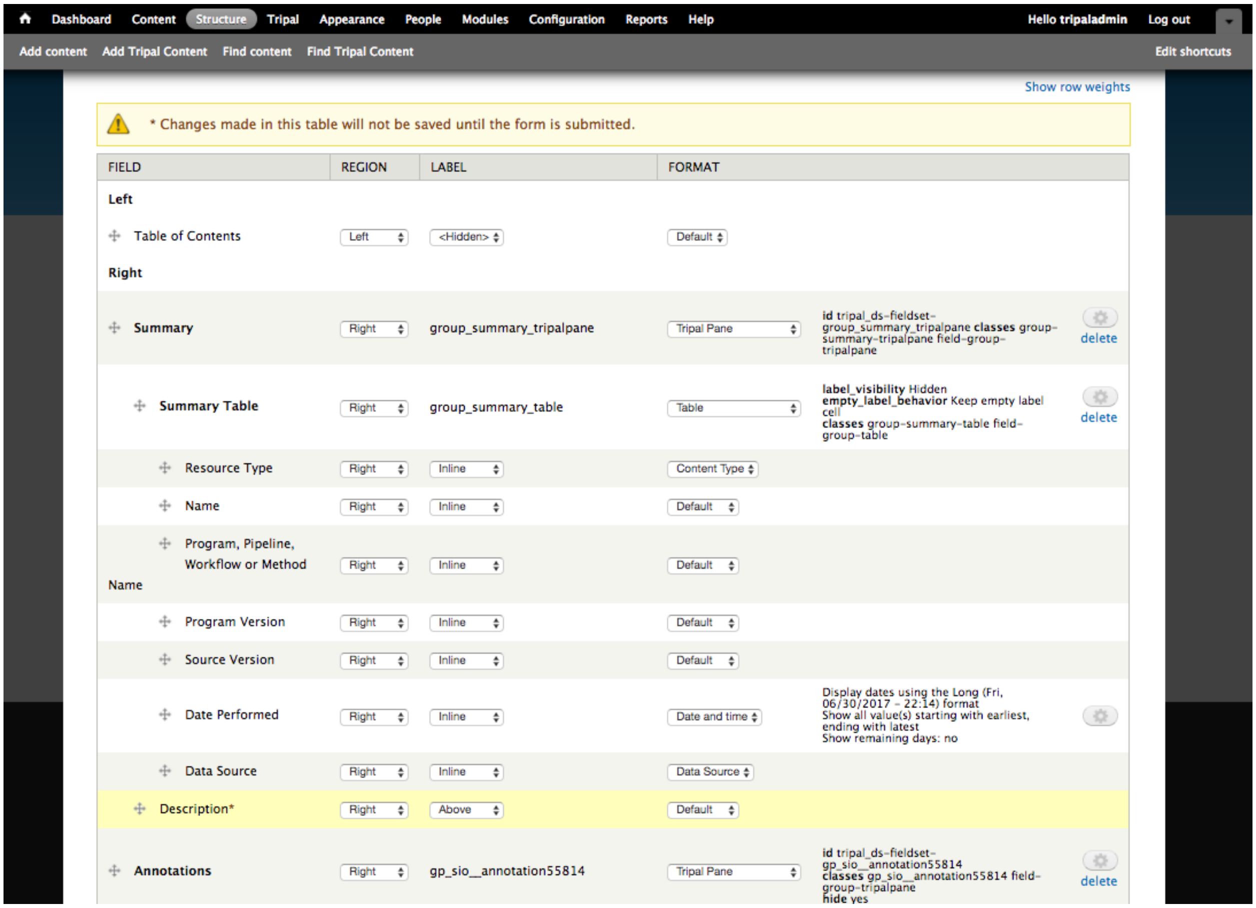
Home » Administration » Structure » Tripal Content Types » Analysis

✚ Apply Default Tripal Layout (will reset current layout) ✚ Create an empty Tripal Pane

FIELD	REGION	LABEL	FORMAT		
Left					
Table of Contents	Left	<Hidden>	Default		
Right					
Summary	Right	group_summary_tripalpane	Tripal Pane	**id** tripal_ds-fieldset-group_summary_tripalpane **classes** group-summary-tripalpane field-group-tripalpane	delete
Summary Table	Right	group_summary_table	Table	**label_visibility** Hidden **empty_label_behavior** Keep empty label cell **classes** group-summary-table field-group-table	delete
Resource Type	Right	Inline	Content Type		
Name	Right	Inline	Default		
Description	Right	Above	Default		
Program, Pipeline, Workflow or Method Name	Right	Inline	Default		
Program Version	Right	Inline	Default		
Source Version	Right	Inline	Default		
Date Performed	Right	Inline	Date and time	Display dates using the Long (Fri, 06/30/2017 - 21:00) format Show all value(s) starting with earliest, ending with latest Show remaining days: no	

1.5.3.1 Rearranging Fields

To rearrange the fields within a Tripal pane, simply drag them into the order you would like them. For example, the description is currently within the Summary table –it makes much more sense for it to be below the table but still within the summary. To do this, simply drag the description field to the bottom of the summary table and then move it in one level as shown in the following screenshot. Then click the **Save** button at the bottom to save the changes.



* Changes made in this table will not be saved until the form is submitted.

FIELD	REGION	LABEL	FORMAT		
Left					
Table of Contents	Left	<Hidden>	Default		
Right					
Summary	Right	group_summary_tripalpane	Tripal Pane	**id** tripal_ds-fieldset-group_summary_tripalpane **classes** group-summary-tripalpane field-group-tripalpane	delete
Summary Table	Right	group_summary_table	Table	**label_visibility** Hidden **empty_label_behavior** Keep empty label cell **classes** group-summary-table field-group-table	delete
Resource Type	Right	Inline	Content Type		
Name	Right	Inline	Default		
Program, Pipeline, Workflow or Method Name	Right	Inline	Default		
Program Version	Right	Inline	Default		
Source Version	Right	Inline	Default		
Date Performed	Right	Inline	Date and time	Display dates using the Long (Fri, 06/30/2017 - 22:14) format Show all value(s) starting with earliest, ending with latest Show remaining days: no	
Data Source	Right	Inline	Data Source		
Description*	Right	Above	Default		
Annotations	Right	gp_sio__annotation55814	Tripal Pane	**id** tripal_ds-fieldset-gp_sio__annotation55814 **classes** gp_sio__annotation55814 field-group-tripalpane **hide** yes	delete

1.5.3.2 Removing Fields and/or Field Labels

Now say we don't want the label "Description" in front of description content since it's pretty self explanatory. We can do that by changing the drop-down beside "Description" which currently says "Above" to "Hidden". This removes the label for the field assuming it's not within a table.

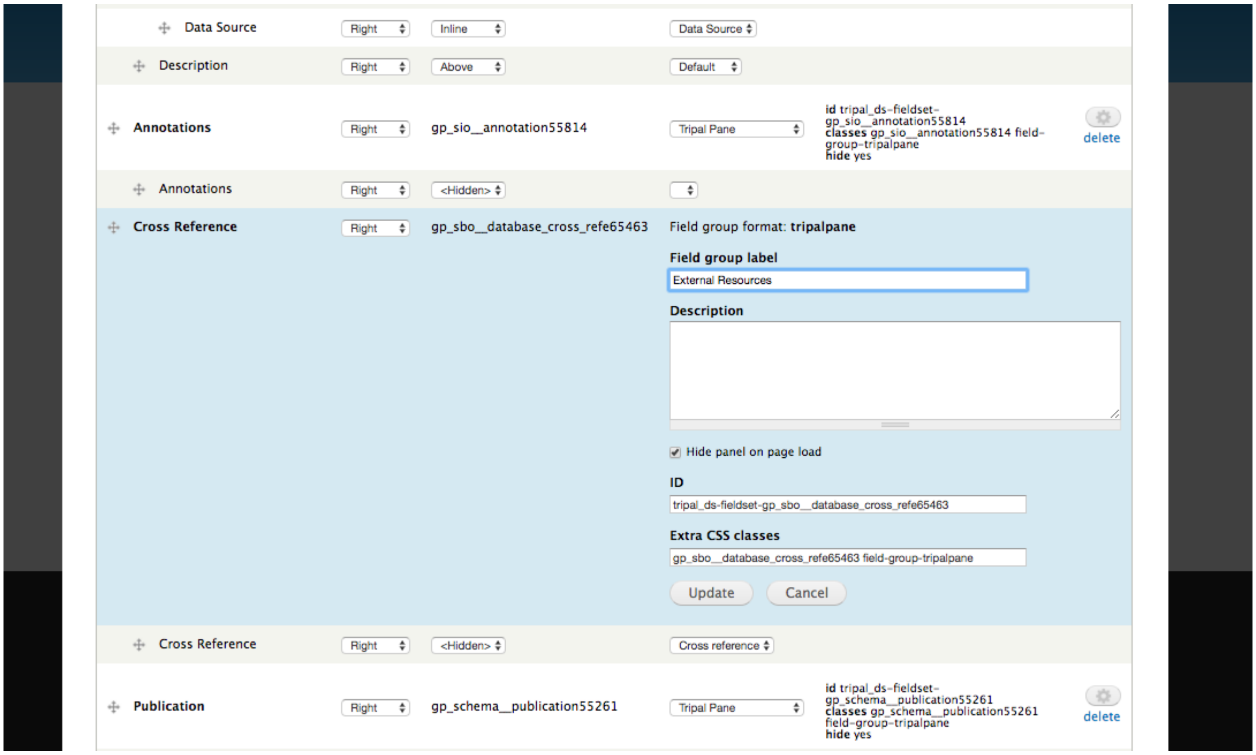
There may also be data you want to collect from your user but don't want to display on the page. This can be achomplished by disabling the field in the Manage Display UI. For example, we might not feel the need to tell users that this is an alaysis page and thus want to hide the Resource Type Field. This is done by changing the drop-down beside the Resource type field from "Right" to "Disabled".

Warning: Don't forget to save the configuration often as you are changing it. You will not see changes to the page unless the **Save** button at the bottom of the Manage Display UI is clicked.

1.5.3.3 Changing Tripal Pane Names

If you enabled the *tripal_ds* module during installation then you will have what is called **Panes** into which fields can be grouped. With the *tripal_ds* module all field come pre-organizd into panes. The name of a pane is displayed both in the header of the pane itself and in the Table of Contents. To change this name, click the gear button to the far right of the Tripal Pane you would like to change. This will bring up a blue pane of settings. Changing the Field Group Label will change the display name of the pane. For example, the following screenshot shows how you would change the

"Cross References" Tripal Pane to be labeled "External Resources" instead if that it what you prefer. Then just click the Update button to see your changes take effect.



Data Source	Right	Inline	Data Source		
Description	Right	Above	Default		
Annotations	Right	gp_sio__annotation55814	Tripal Pane	**id** tripal_ds-fieldset-gp_sio__annotation55814 **classes** gp_sio__annotation55814 field-group-tripalpane **hide** yes	delete
Annotations	Right	<Hidden>			
Cross Reference	Right	gp_sbo__database_cross_refe65463	Field group format: **tripalpane** **Field group label**: External Resources **Description**: ☑ Hide panel on page load **ID**: tripal_ds-fieldset-gp_sbo__database_cross_refe65463 **Extra CSS classes**: gp_sbo__database_cross_refe65463 field-group-tripalpane Update Cancel		
Cross Reference	Right	<Hidden>	Cross reference		
Publication	Right	gp_schema__publication55261	Tripal Pane	**id** tripal_ds-fieldset-gp_schema__publication55261 **classes** gp_schema__publication55261 field-group-tripalpane **hide** yes	delete

1.5.3.4 Display/Hide Tripal Panes on Page Load

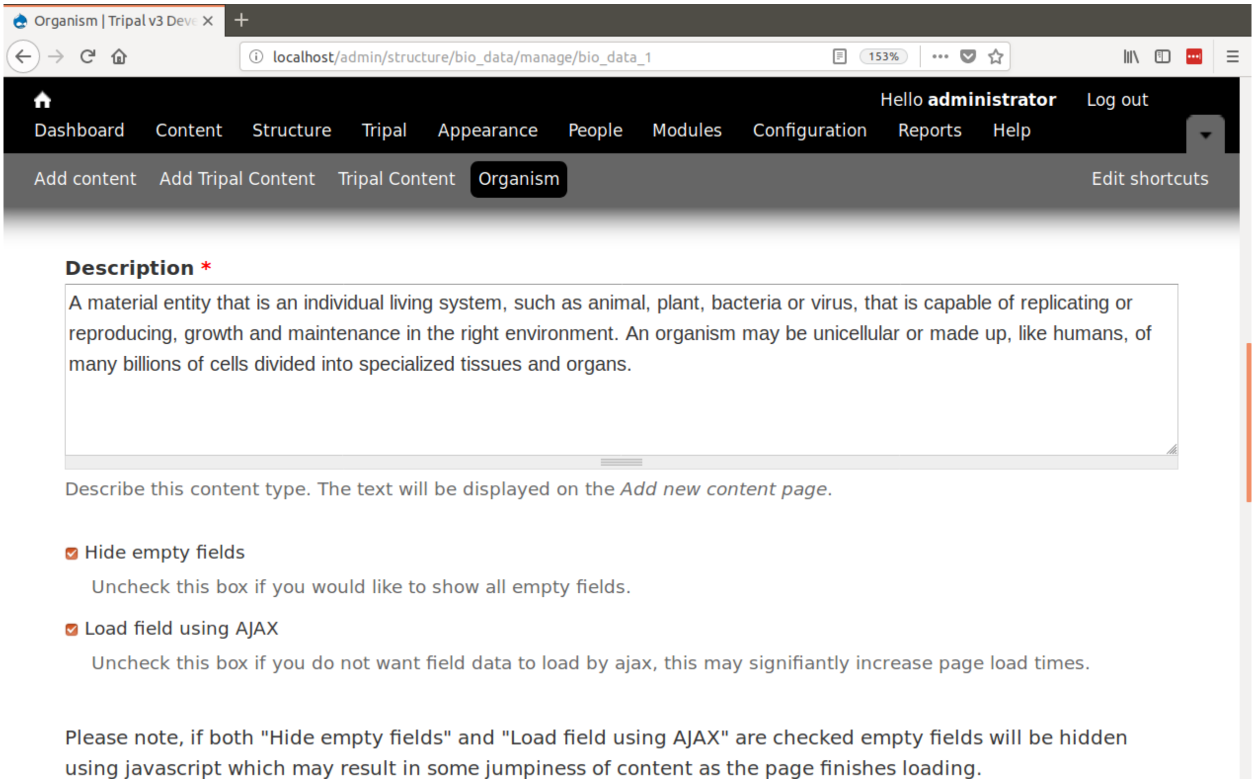
You can also easily control which Tripal Panes you would like displayed to the user on initial page load. By default the Summary Pane is the only one configured to show by default. However, if you would prefer for all panes or even a specific subset of panes to show by default, you can simply click the gear button to the far right of each Tripal Pane you want displayed by default and uncheck the "Hide panel on page load" checkbox. This gives you complete control over which panes you want your user to see first. If more then one pane is displayed by default then they will be shown in the order they are listed on the Manage Display UI.

1.5.4 Hide Empty Fields and AJAX loading

Tripal provides two additional controls for display of fields on a page:

- Hiding fields with no data.
- Loading fields using AJAX.

You will find two check boxes when editing any content page that gives you these controls. Navigate to `Structure → Tripal Content Types` and then click on any Tripal Content Type. You will see options similar to the following:



Description *

A material entity that is an individual living system, such as animal, plant, bacteria or virus, that is capable of replicating or reproducing, growth and maintenance in the right environment. An organism may be unicellular or made up, like humans, of many billions of cells divided into specialized tissues and organs.

Describe this content type. The text will be displayed on the *Add new content page*.

☑ Hide empty fields

Uncheck this box if you would like to show all empty fields.

☑ Load field using AJAX

Uncheck this box if you do not want field data to load by ajax, this may signifiantly increase page load times.

Please note, if both "Hide empty fields" and "Load field using AJAX" are checked empty fields will be hidden using javascript which may result in some jumpiness of content as the page finishes loading.

1.5.4.1 Hiding Empty Fields

The previous sections of this guide instructed how to rearrange fields on a page, hide their titles, and organize them into panes. However, while there are many fields many of them may not have any data. All of these fields are present because the data store (e.g. Chado) has the capacity to house the type of data the fields represent, but if you did not load data appropriate for those fields then they will have no data.

By default Tripal v3 hides all empty fields from the user. However if you would prefer to show all fields to the user regardless of whether there is content for that particular page edit the content type and click the box labeled *Hide Empty Fields* and click the *Save* button at the bottom. The next time anyone loads any page for the given content type all fields will be shown regardless if they have data.

1.5.4.2 Using AJAX to Load Fields

Depending on the number of fields for your content type and the amount of data that those fields contain you may notice that page loads can take a few seconds to load. AJAX is a method to help decrease load times by allowing the page to load quickly with minimal data and allowing fields with larger amounts of data to load after the initial page load. Tripal has this setting enabled by default. but you can disable this feature. Similar to the check box for hiding fields, there is a check box on the content type edit page labeled *Load field using AJAX*. Remove the check for box to disable all AJAX loading of fields and save the content type settings.

Note: You can control AJAX loading and hiding of empty fields differently for each content type.

1.5.5 Field Specific Permissions

1.5.5.1 Why Field Permissions?

Not all Tripal Fields are created equal. You may have some fields that you don't want all users to be able to view, or even to be able to edit. This might be the case for a variety of reasons. Some Chado base tables may have **type** fields that you don't utilize: for example, the contact table. Some of your content types may be configured with a lot of property fields, with only a subset of them being relevant to an end user. Some fields require prior insertion of data elsewhere: for example, the Cross-Reference field. Perhaps you have some Chado property fields that are for internal use only.

Note: If you're following this guide because you want users to submit data into Chado, consider using Tripal HeadQuarters (HQ). Tripal HQ provides a user-contributed content control center and administrative toolbox for your Tripal site. This means that users are able to create whatever Chado content you'd like them, but withhold inserting it into the database until someone has approved it. Find out more here: https://tripal-hq.readthedocs.io/en/latest/index.html

Simply disabling the display of the formatter won't prevent the widget from showing up on the submission page, and besides, you might want site admins to still have access to those fields! Deleting the field will cause them to re-appear when you press the "Check for New Fields" button! Field Permissions allows you to configure field-specific permissions so that users contributing content via Chado only see the fields they need to see.

Warning: Because all Tripal Entities are the same base entity type (`TripalEntity`), configuring field permissions on one bundle will configure it for **all** bundles. This means, for example, hiding a field on Organism will also hide it on Analysis.

1.5.5.2 Installing the Drupal Field Permissions module

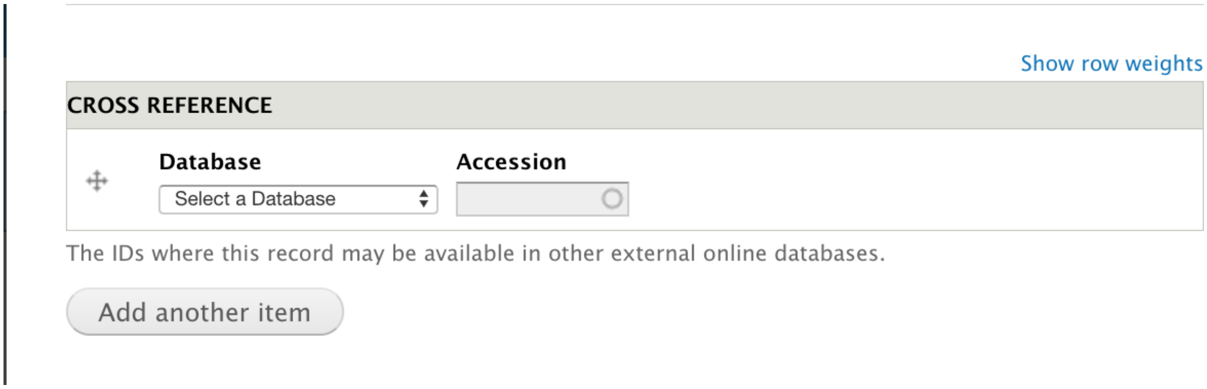
The module can be enabled directly from Drush with the below command.

```
drush pm-enable -y field_permissions
```

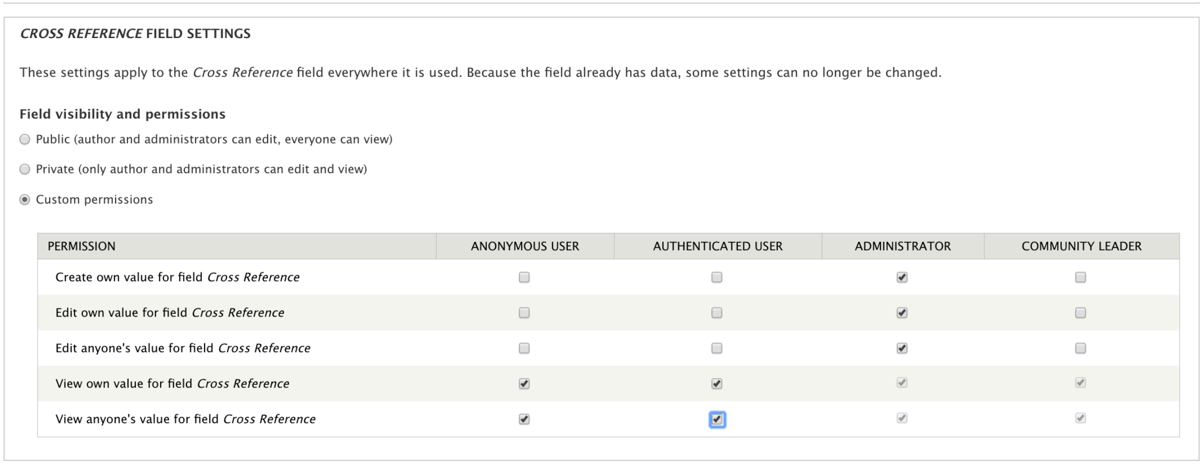
You can find the Field Permission module page here: https://www.drupal.org/project/field_permissions and a more in-depth user guide here: https://www.drupal.org/node/2802067

1.5.5.3 Setting Field-specific Permissions

Let's assume I want to hide the Cross-Reference field from my users submitting Genome Assembly data, but still want it available for my administrators.



First, navigate to the content type field configuration page via **Admin –> Structure –> Tripal Content –> Genome Assembly**. For each field we want to hide, we must configure the field instance settings individually. Click **Edit** for the Cross Reference field, and scroll down to **CROSS REFERENCE FIELD SETTINGS**. Select **Custom Permissions** and ensure that the user role you set up for submitters can view, but cannot edit, this field.



CROSS REFERENCE FIELD SETTINGS

These settings apply to the *Cross Reference* field everywhere it is used. Because the field already has data, some settings can no longer be changed.

Field visibility and permissions

○ Public (author and administrators can edit, everyone can view)

○ Private (only author and administrators can edit and view)

● Custom permissions

PERMISSION	ANONYMOUS USER	AUTHENTICATED USER	ADMINISTRATOR	COMMUNITY LEADER
Create own value for field *Cross Reference*	□	□	☑	□
Edit own value for field *Cross Reference*	□	□	☑	□
Edit anyone's value for field *Cross Reference*	□	□	☑	□
View own value for field *Cross Reference*	☑	☑	☑	☑
View anyone's value for field *Cross Reference*	☑	☑	☑	☑

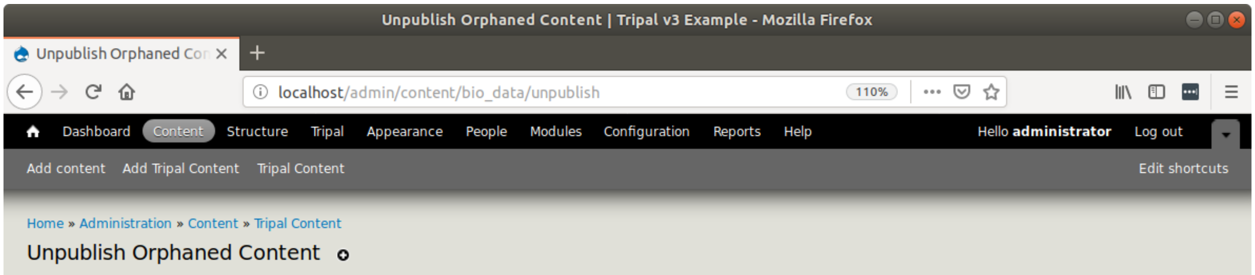
Once permissions are configured to your liking, click **Save Settings**.

Warning: Some fields are **Required**. Do not disable required fields that can't be null. If you do, users won't be able to submit content!

Now, if you submit content as a user with that role, the field will not display on the widgets, but will still appear on normal content.

1.5.6 Handling Orphaned Entities

It is common for site developers to work directly with Chado, especially as they become more familiar with it. And sometimes, they may purposefully or accidentally remove data from Chado that is utilized by published Tripal entities. This results in an entity being "orphaned". The entity will still have a page on the website but the data about that entity is no longer available. If this occurs you can easily remove published entities that are "orphaned" by navigating to **Administer > Tripal Content Types** and clicking the **Unpublish Orphaned Content** link. The following page appears.



Unpublish Orphaned Content

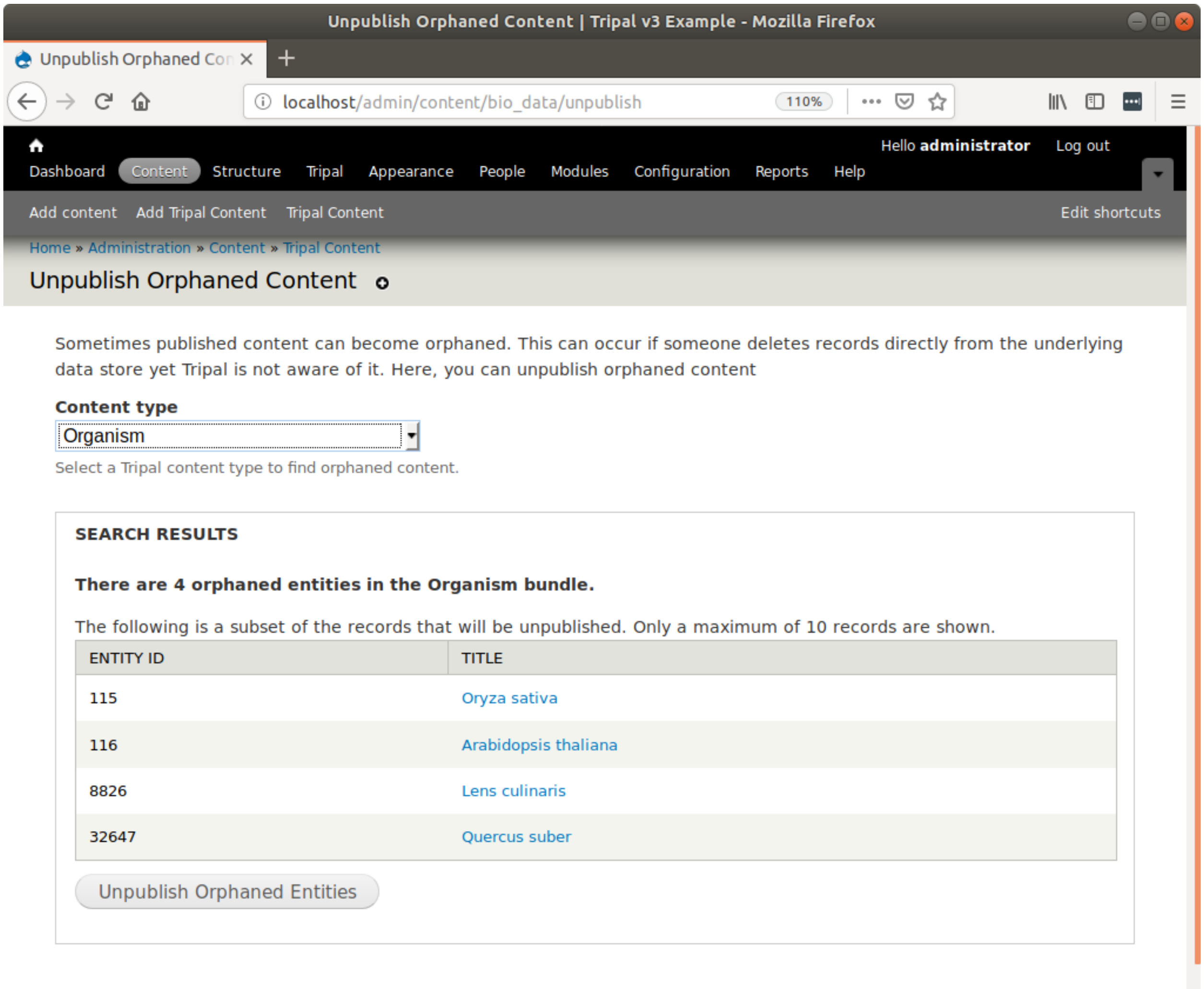
Sometimes published content can become orphaned. This can occur if someone deletes records directly from the underlying data store yet Tripal is not aware of it. Here, you can unpublish orphaned content

Content type

-- Select a Content Type --

Select a Tripal content type to find orphaned content.

Next, select the content type that has missing data in the **Content Type** select box. For example, suppose an organism was removed:



Unpublish Orphaned Content

Sometimes published content can become orphaned. This can occur if someone deletes records directly from the underlying data store yet Tripal is not aware of it. Here, you can unpublish orphaned content

Content type

Organism

Select a Tripal content type to find orphaned content.

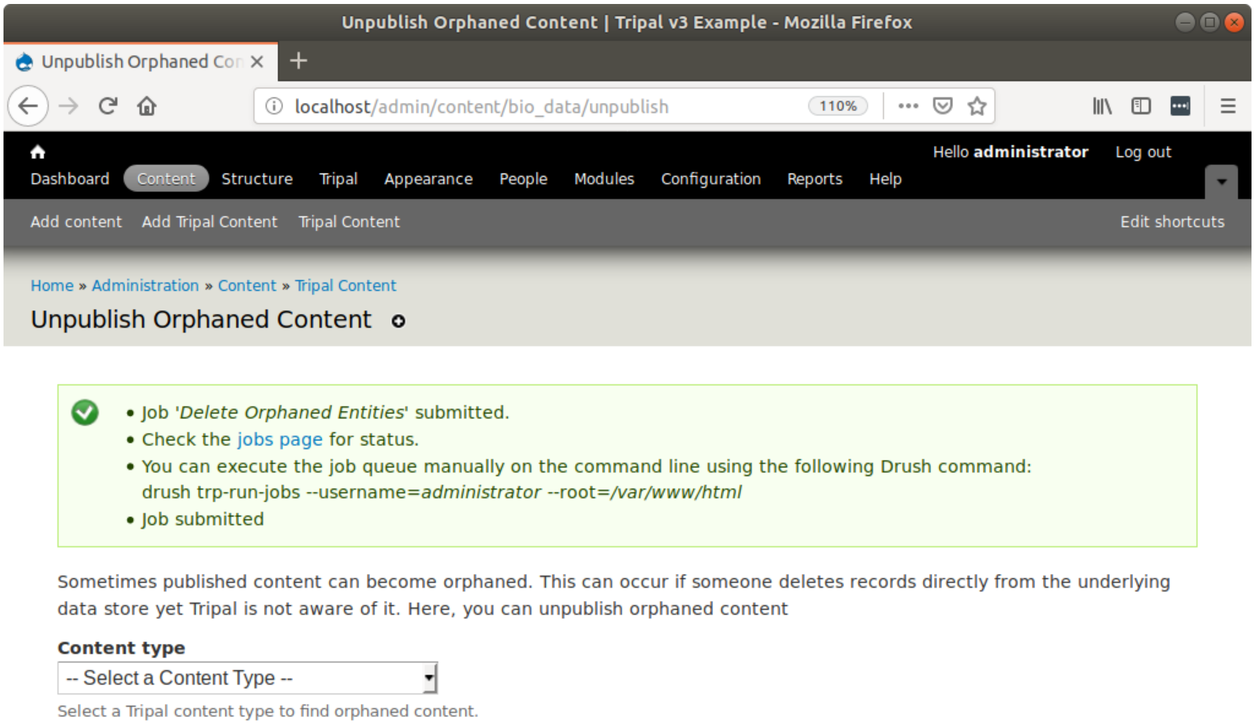
SEARCH RESULTS

There are 4 orphaned entities in the Organism bundle.

The following is a subset of the records that will be unpublished. Only a maximum of 10 records are shown.

ENTITY ID	TITLE
115	Oryza sativa
116	Arabidopsis thaliana
8826	Lens culinaris
32647	Quercus suber

A list of at most 10 entities that are orphaned are shown. This list is just for convenience and does not show all of the orphaned entities that will be removed. If the content type has no orphaned entities then nothing is shown. You can submit a job to clean the orphaned entities by clicking the **Unpublish Orphaned Entities** button.



Unpublish Orphaned Content

- Job '*Delete Orphaned Entities*' submitted.
- Check the jobs page for status.
- You can execute the job queue manually on the command line using the following Drush command: drush trp-run-jobs --username=*administrator* --root=*/var/www/html*
- Job submitted

Sometimes published content can become orphaned. This can occur if someone deletes records directly from the underlying data store yet Tripal is not aware of it. Here, you can unpublish orphaned content

Content type

-- Select a Content Type --

Select a Tripal content type to find orphaned content.

If you have automatic job execution enabled then the cleanup will occur automatically. Otherwise you should manually execute the job on the command-line using Drush as instructed.

1.6 Setup of an Example Site

The following tutorial will walk you through creating content and loading genomic data. This is a good introduction to Tripal v3.x Content Types and the new Administrative User Interface regardless of whether you intend to store genomic data in your particular Tripal v3 site. This demonstration uses three extension modules: Tripal Analysis Blast, Tripal Analysis KEGG, and Tripal Analysis InterPro.

Note: As a rule, extension modules are not documented in the Tripal User's Guide. You can find documentation for extension modules that enhance Tripal functionality on the *Extension Modules* page. However, to provide a functioning demonstration the three analysis extension modules are described here.

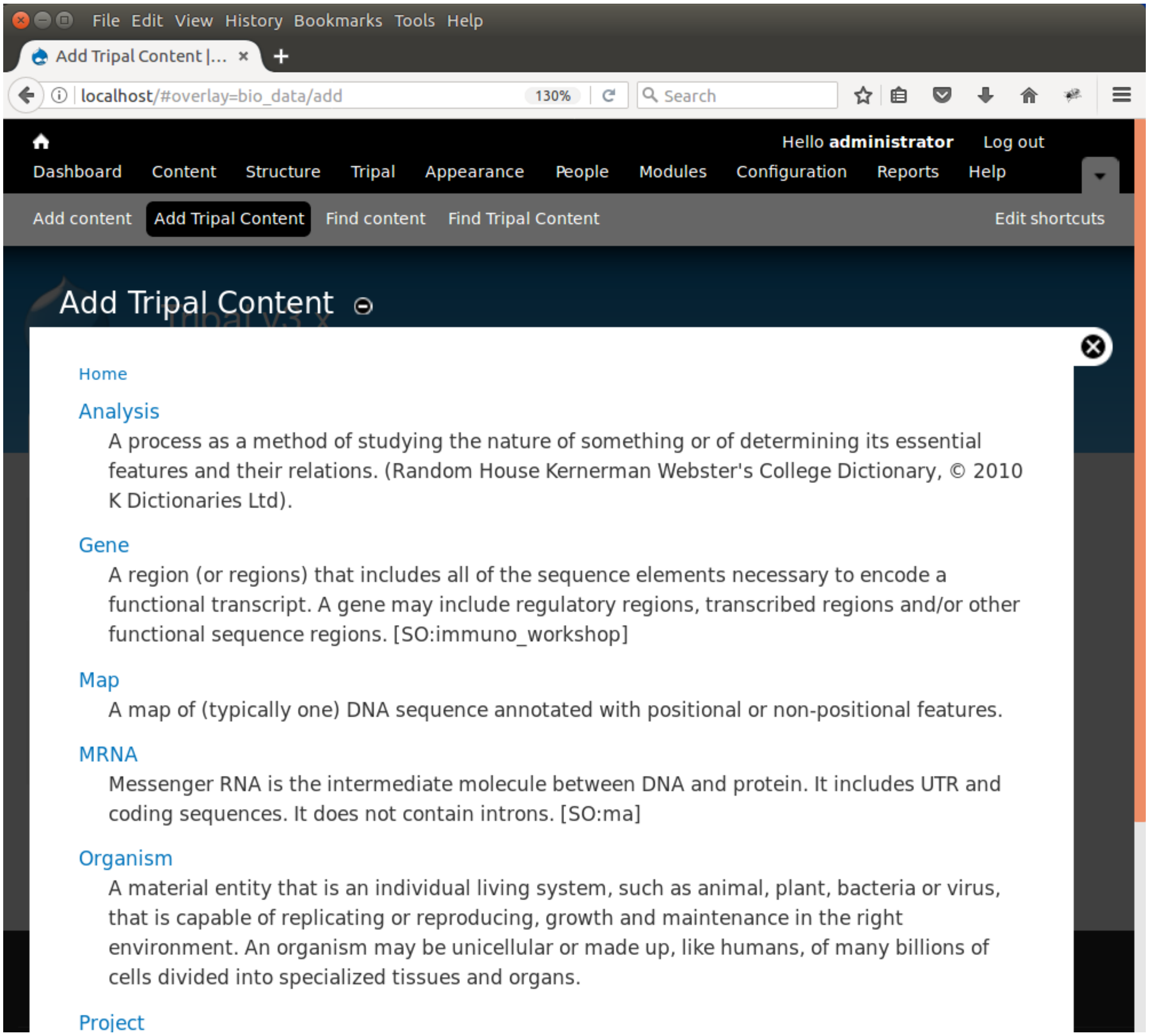
1.6.1 Organisms

Before we can load our data we must first have an organism to which the data will be associated. Chado v1.3 does not come preloaded with any organisms (although previous version of Chado do). For this tutorial we will import genomic data for Citrus sinesis (sweet orange), so we must first create the organism.

1.6.1.1 Creating an Organism Page

We can add the organism using the **Add Tripal Content** link in the top administrative menu. The **Add Tripal Content** page has several content types already available, including the **Organism** content type.

Note: Drupal provides it's own content types such as **Article** and **Basic Page**. These content types are referred to as **nodes** in Drupal speak. You can add these content types via the **Add Content** page. Tripal v3 derived content types are separated from these Drupal content types.



Add Tripal Content

Home

Analysis
A process as a method of studying the nature of something or of determining its essential features and their relations. (Random House Kernerman Webster's College Dictionary, © 2010 K Dictionaries Ltd).

Gene
A region (or regions) that includes all of the sequence elements necessary to encode a functional transcript. A gene may include regulatory regions, transcribed regions and/or other functional sequence regions. [SO:immuno_workshop]

Map
A map of (typically one) DNA sequence annotated with positional or non-positional features.

MRNA
Messenger RNA is the intermediate molecule between DNA and protein. It includes UTR and coding sequences. It does not contain introns. [SO:ma]

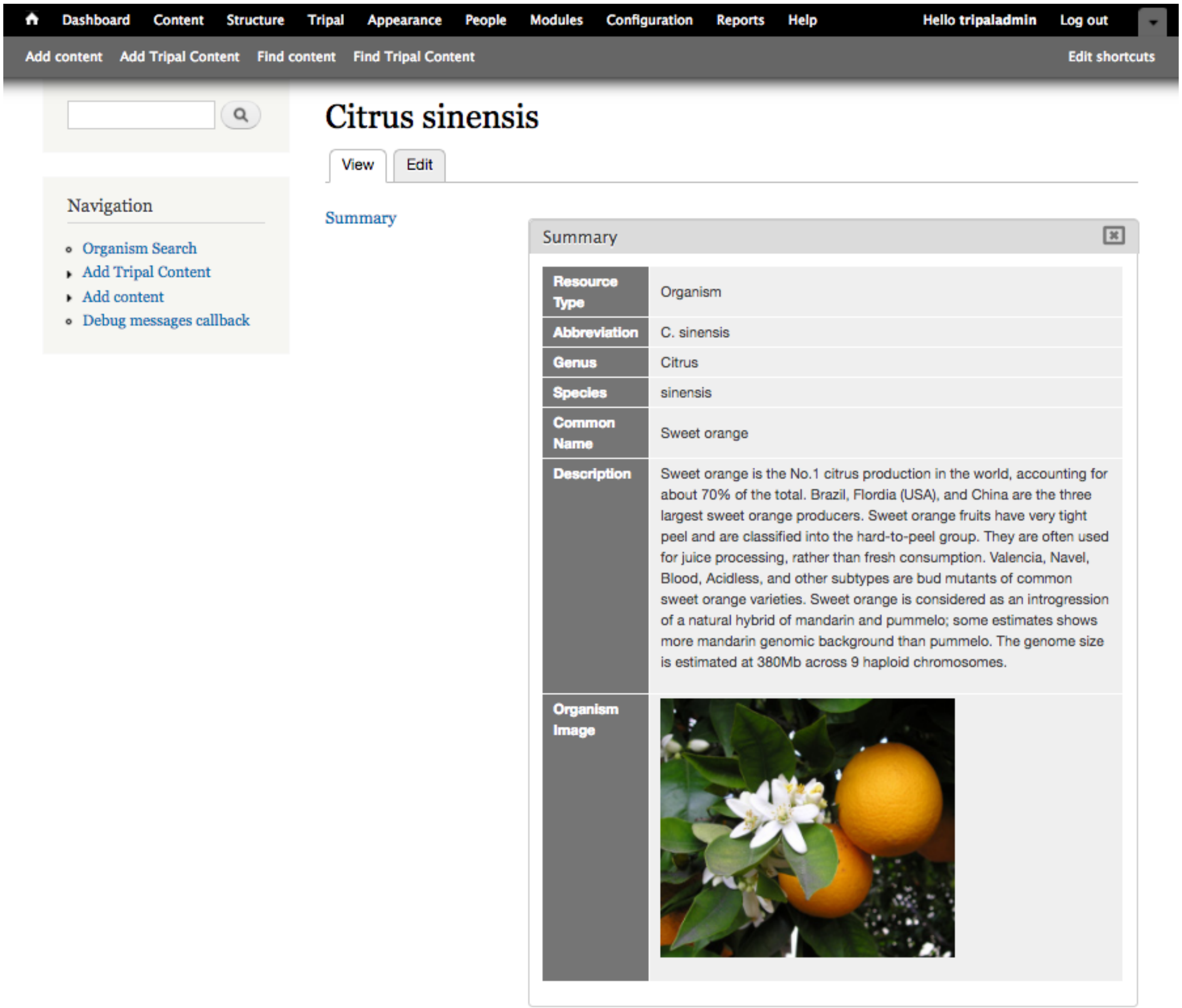
Organism
A material entity that is an individual living system, such as animal, plant, bacteria or virus, that is capable of replicating or reproducing, growth and maintenance in the right environment. An organism may be unicellular or made up, like humans, of many billions of cells divided into specialized tissues and organs.

Project

To add a new organism click the **Organism** link and a form will appear with multiple fields. Fill in the fields with these values:

Field Name	Value
Abbreviation	C. sinensis
Genus	Citrus
Species	sinensis
Common name	Sweet orange
Description	Sweet orange is the No.1 citrus production in the world, accounting for about 70% of the total. Brazil, Flordia (USA), and China are the three largest sweet orange producers. Sweet orange fruits have very tight peel and are classified into the hard-to-peel group. They are often used for juice processing, rather than fresh consumption. Valencia, Navel, Blood, Acidless, and other subtypes are bud mutants of common sweet orange varieties. Sweet orange is considered as an introgression of a natural hybrid of mandarin and pummelo; some estimates shows more mandarin genomic background than pummelo. The genome size is estimated at 380Mb across 9 haploid chromosomes.
Image	

Leave all remaining fields empty and save the page. You should now have an organism page that appears as follows:



Citrus sinensis

Summary

Resource Type	Organism
Abbreviation	C. sinensis
Genus	Citrus
Species	sinensis
Common Name	Sweet orange
Description	Sweet orange is the No.1 citrus production in the world, accounting for about 70% of the total. Brazil, Flordia (USA), and China are the three largest sweet orange producers. Sweet orange fruits have very tight peel and are classified into the hard-to-peel group. They are often used for juice processing, rather than fresh consumption. Valencia, Navel, Blood, Acidless, and other subtypes are bud mutants of common sweet orange varieties. Sweet orange is considered as an introgression of a natural hybrid of mandarin and pummelo; some estimates shows more mandarin genomic background than pummelo. The genome size is estimated at 380Mb across 9 haploid chromosomes.
Organism Image	

Note: The layout of the organism page is provided by the **tripal_ds** module that was enabled during Tripal installation. If you decided not to enable that module then your page will look quite different.

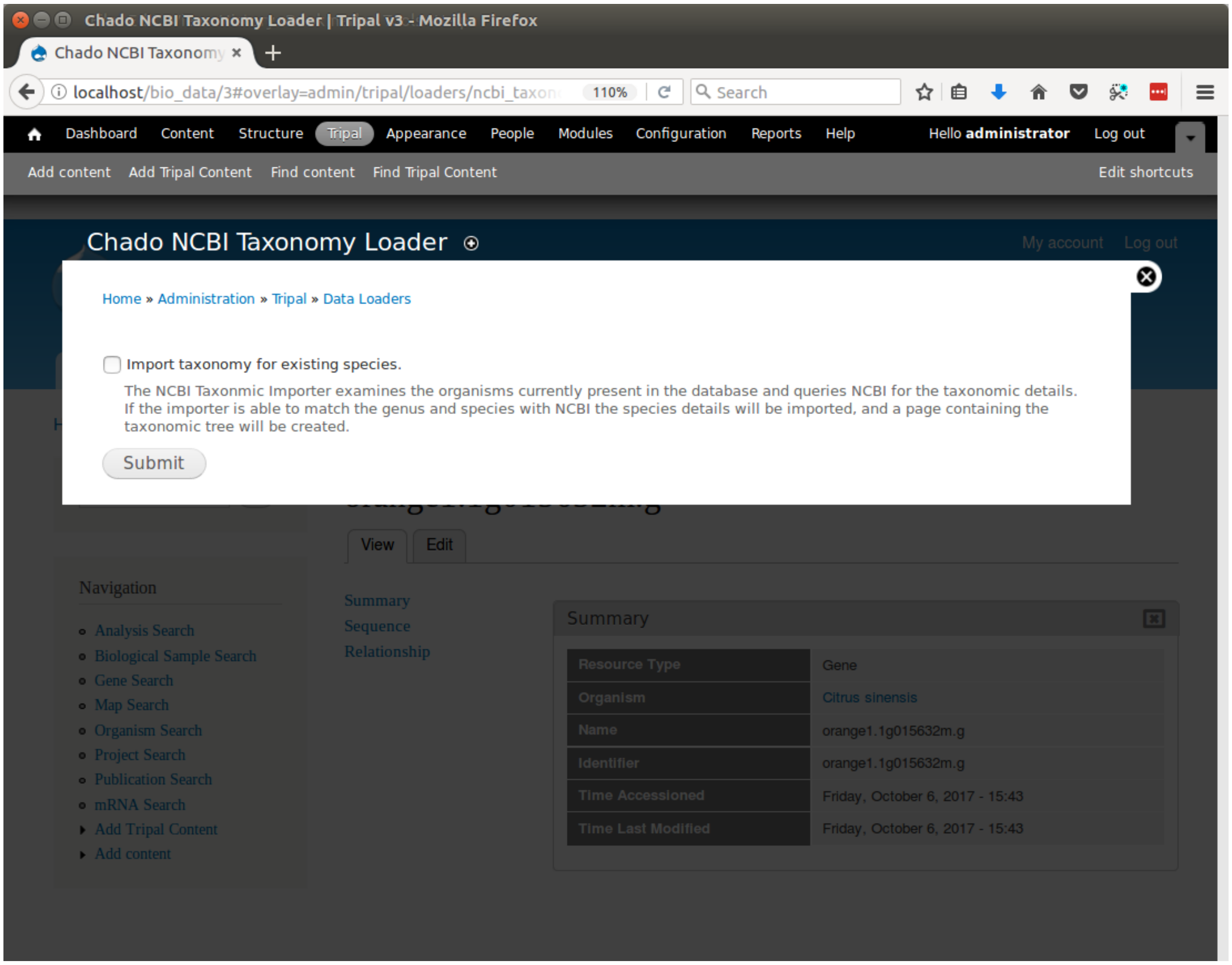
This page has three primary sections. A left sidebar that contains a search box and a **block** titled **Navigation**. To the right of the sidebar is the content section. Here the content is divided into two columns. In the first column is a table of contents listing the "Table of Contents" of the page. Currently this list simply contains the **Summary**. The second column contains all of the content about the organism. Clicking the links in the table of contents causes each section to slide to the top of the page for viewing while all other content slides downward. Users need not scroll the entire page to see all content. The close link (X) at the top right of each section allows the user to remove sections they may not want to see and which clutter their view.

There is more information about an organism that does not appear on this page. By default, Tripal will not show fields that have no data. However, if you prefer, you can change this behavior by configuring the content type to not hide empty fields. You can find this option by navigating to **Structure > Tripal Content Types**, and click on the **edit** link for the content type. You will find the option to toggle this behavior there.

If you do not like this layout you can change it! One of the benefits for using Drupal is the ability to customize the entire look and feel of your site. Tripal v3 along with Drupal allow you to reorganize this (and any) page however you like. This default layout is provided by the **tripal_ds** module to help simplify the process of laying out a page. If you decided not to enable the **tripal_ds** module then your page will require manual layout. Later in this tutorial instructions to reorganize and re-theme the Tripal content pages are provided. No programming is required to do this.

1.6.1.2 Load data from NCBI Taxonomy

Tripal makes it easy to import additional information about any organisms within a Tripal site from the NCBI Taxonomy database. The importer will only import data for species that you currently have in the Tripal database. The taxonomic names must match those in the NCBI Taxonomy database. Currently, we only have a single organism (Citrus sinensis) and we will import additional properties for this organism from NCBI but we can return later to import data for new organisms we may add later. To import additional organism details, navigate to **Tripal** → **Data Loaders** → **Chado NCBI Taxonomy Loader**. The following page appears:



Chado NCBI Taxonomy Loader

Home » Administration » Tripal » Data Loaders

□ Import taxonomy for existing species.

The NCBI Taxonmic Importer examines the organisms currently present in the database and queries NCBI for the taxonomic details. If the importer is able to match the genus and species with NCBI the species details will be imported, and a page containing the taxonomic tree will be created.

Submit

Resource Type	Gene
Organism	Citrus sinensis
Name	orange1.1g015632m.g
Identifier	orange1.1g015632m.g
Time Accessioned	Friday, October 6, 2017 - 15:43
Time Last Modified	Friday, October 6, 2017 - 15:43

Click the checkbox beside the 'Import taxonomy for existing species' and click Submit. Now run the submitted job:

```
drush trp-run-jobs --username=administrator --root=$DRUPAL_HOME
```

You will see the following output:

```
Tripal Job Launcher (in parallel)
Running as user 'administrator'
--------------------
```

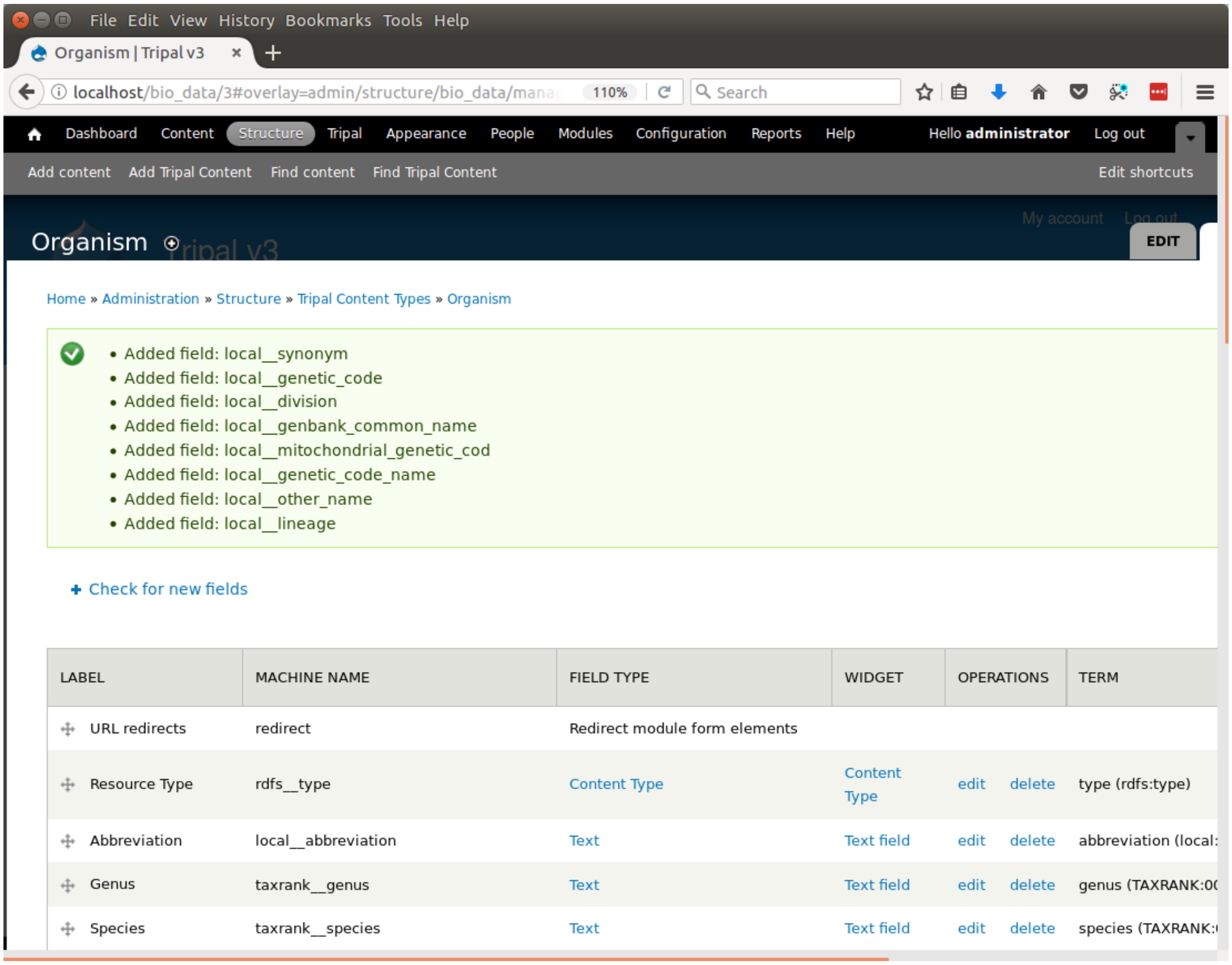
(continues on next page)

(continued from previous page)

```
2017-10-06 15:45:47: There are 1 jobs queued.
2017-10-06 15:45:47: Calling: tripal_chado_ncbi_taxonomy_import()

NOTE: Importing of NCBI taxonomy data is performed using a database transaction.
If the load fails or is terminated prematurely then the entire set of
insertions/updates is rolled back and will not be found in the database

2711    Citrus sinensis
```

1.6.1.3 Adding New Fields

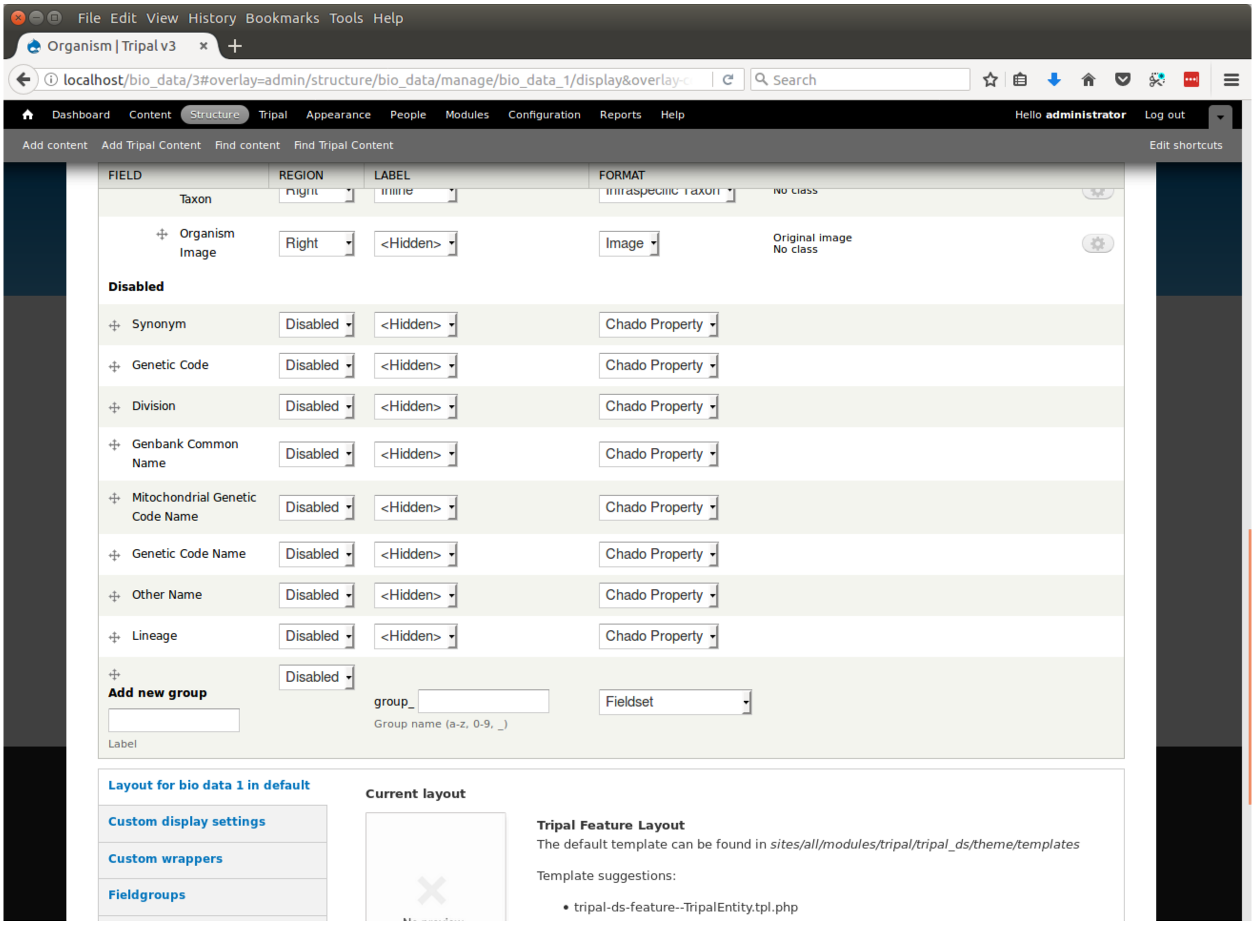
We have now imported many new properties about the Citrus sinensis organism from NCBI Taxonomy. However, these properties won't show up on the page automatically. We need to tell Drupal that our organism pages now have new property fields for display. To do this, navigate to **Structure** → **Tripal Content Types** and in the row for the Organism content type, click the link titled managed fields. Here we see a list the fields that associated with an Organism content type. Click the link at the top of the page **Check for new fields**. You will see that several new fields have been added.


File Edit View History Bookmarks Tools Help
Organism | Tripal v3
localhost/bio_data/3#overlay=admin/structure/bio_data/mana
110%
Search
Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help
Hello administrator Log out
Add content Add Tripal Content Find content Find Tripal Content
Edit shortcuts
My account Log out
EDIT
Organism
Tripal v3
Home » Administration » Structure » Tripal Content Types » Organism
Check for new fields

- Added field: local__synonym
- Added field: local__genetic_code
- Added field: local__division
- Added field: local__genbank_common_name
- Added field: local__mitochondrial_genetic_cod
- Added field: local__genetic_code_name
- Added field: local__other_name
- Added field: local__lineage

LABEL	MACHINE NAME	FIELD TYPE	WIDGET	OPERATIONS		TERM
URL redirects	redirect	Redirect module form elements				
Resource Type	rdfs__type	Content Type	Content Type	edit	delete	type (rdfs:type)
Abbreviation	local__abbreviation	Text	Text field	edit	delete	abbreviation (local:
Genus	taxrank__genus	Text	Text field	edit	delete	genus (TAXRANK:0
Species	taxrank__species	Text	Text field	edit	delete	species (TAXRANK:

Drupal now knows about these new fields! But if we were to look at the Citrus sinensis page we would see that the new properties do not appear. Despite that Drupal knows about the fields it has disabled their display. To enable display of these fields click the **Manage Display** tab at the top right of the page. Here all of the fields are organized into the structure that they will be displayed on the page. Later in this tutorial a more formal description is provided about how you use this interface to change the way the page appears. For now, we simply need to get the new fields to be shown. Scroll to the bottom of the page and the new fields can be seen in the **Disabled** section.


File Edit View History Bookmarks Tools Help
Organism | Tripal v3
localhost/bio_data/3#overlay=admin/structure/bio_data/manage/bio_data_1/display&overlay-c
Search
Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help
Hello administrator Log out
Add content Add Tripal Content Find content Find Tripal Content
Edit shortcuts

FIELD	REGION	LABEL	FORMAT	
Taxon	Right	Inline	Infraspecific Taxon	No class
Organism Image	Right	<Hidden>	Image	Original image No class
Disabled				
Synonym	Disabled	<Hidden>	Chado Property	
Genetic Code	Disabled	<Hidden>	Chado Property	
Division	Disabled	<Hidden>	Chado Property	
Genbank Common Name	Disabled	<Hidden>	Chado Property	
Mitochondrial Genetic Code Name	Disabled	<Hidden>	Chado Property	
Genetic Code Name	Disabled	<Hidden>	Chado Property	
Other Name	Disabled	<Hidden>	Chado Property	
Lineage	Disabled	<Hidden>	Chado Property	
Add new group Label	Disabled	group_ Group name (a-z, 0-9, _)	Fieldset	

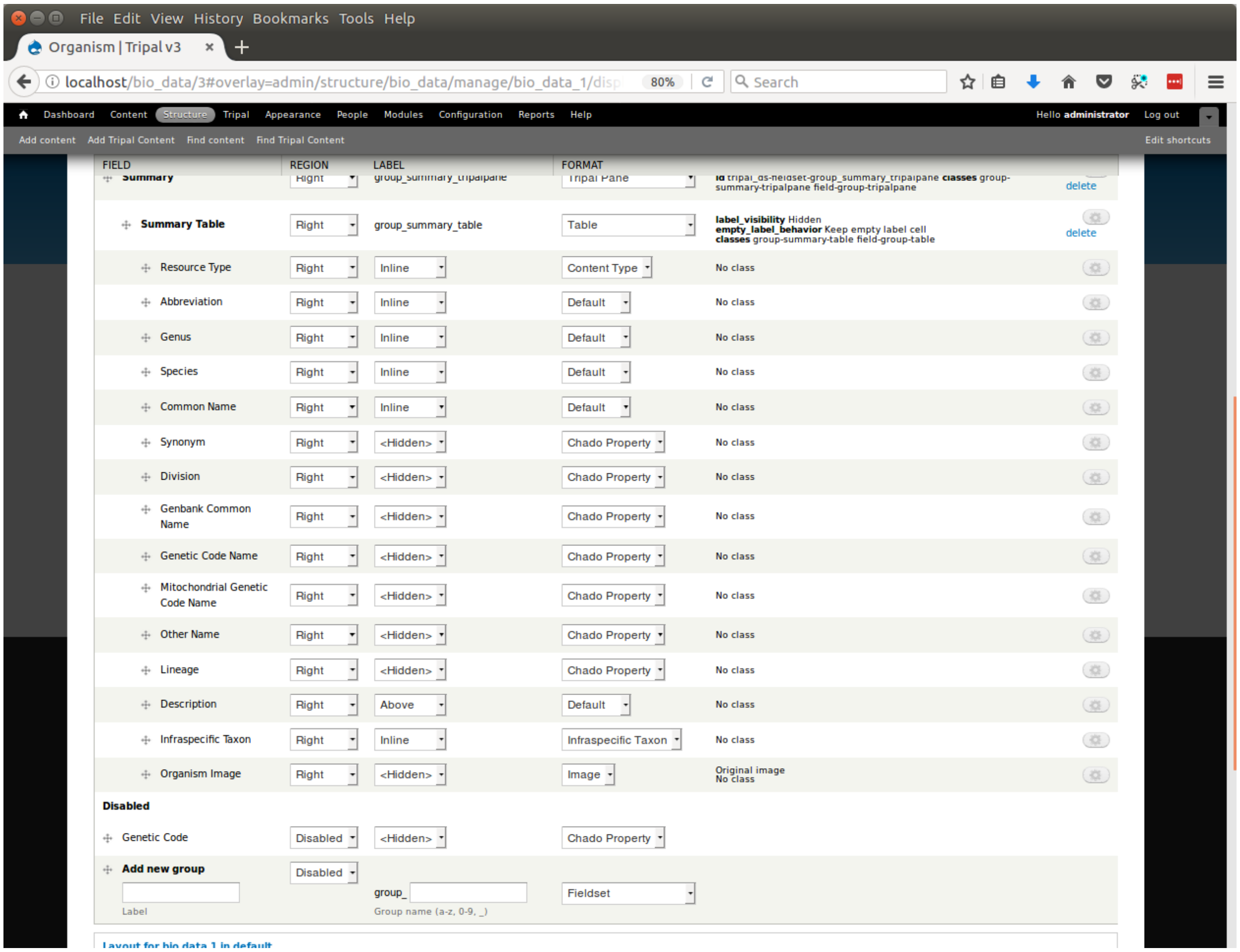
Tripal Feature Layout

The default template can be found in *sites/all/modules/tripal/tripal_ds/theme/templates*

Template suggestions:

- tripal-ds-feature--TripalEntity.tpl.php

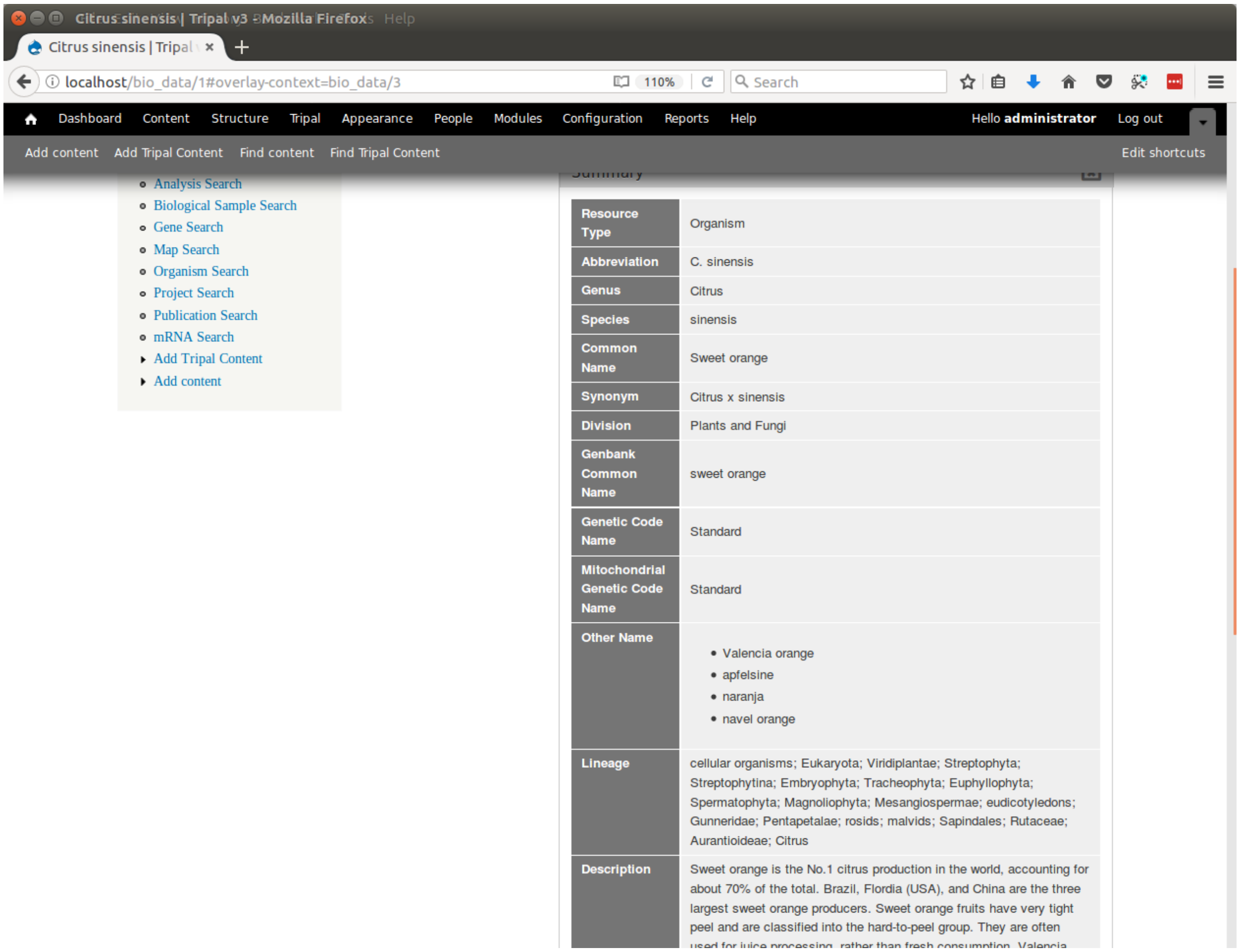
We can move these new fields out of the Disabled section by clicking on the cross-hair icons to the left of the name and dragging the field into a section above. Drag these fields into the **Summary** section underneath the **Summary Table**. Notice in the screenshot below that the fields that were once in the **Disabled** section are now in the **Summary Table** section. Click the **Save** button at the bottom to make all changes final.



FIELD	REGION	LABEL	FORMAT		
Summary	Right	group_summary_tripalpane	Tripal Pane	**id** tripal_ds-fieldset-group_summary_tripalpane **classes** group-summary-tripalpane field-group-tripalpane	delete
Summary Table	Right	group_summary_table	Table	**label_visibility** Hidden **empty_label_behavior** Keep empty label cell **classes** group-summary-table field-group-table	delete
Resource Type	Right	Inline	Content Type	No class	
Abbreviation	Right	Inline	Default	No class	
Genus	Right	Inline	Default	No class	
Species	Right	Inline	Default	No class	
Common Name	Right	Inline	Default	No class	
Synonym	Right	<Hidden>	Chado Property	No class	
Division	Right	<Hidden>	Chado Property	No class	
Genbank Common Name	Right	<Hidden>	Chado Property	No class	
Genetic Code Name	Right	<Hidden>	Chado Property	No class	
Mitochondrial Genetic Code Name	Right	<Hidden>	Chado Property	No class	
Other Name	Right	<Hidden>	Chado Property	No class	
Lineage	Right	<Hidden>	Chado Property	No class	
Description	Right	Above	Default	No class	
Infraspecific Taxon	Right	Inline	Infraspecific Taxon	No class	
Organism Image	Right	<Hidden>	Image	Original image No class	
Disabled					
Genetic Code	Disabled	<Hidden>	Chado Property		
Add new group Label	Disabled	group_ Group name (a-z, 0-9, _)	Fieldset		

Layout for bio data 1 in default

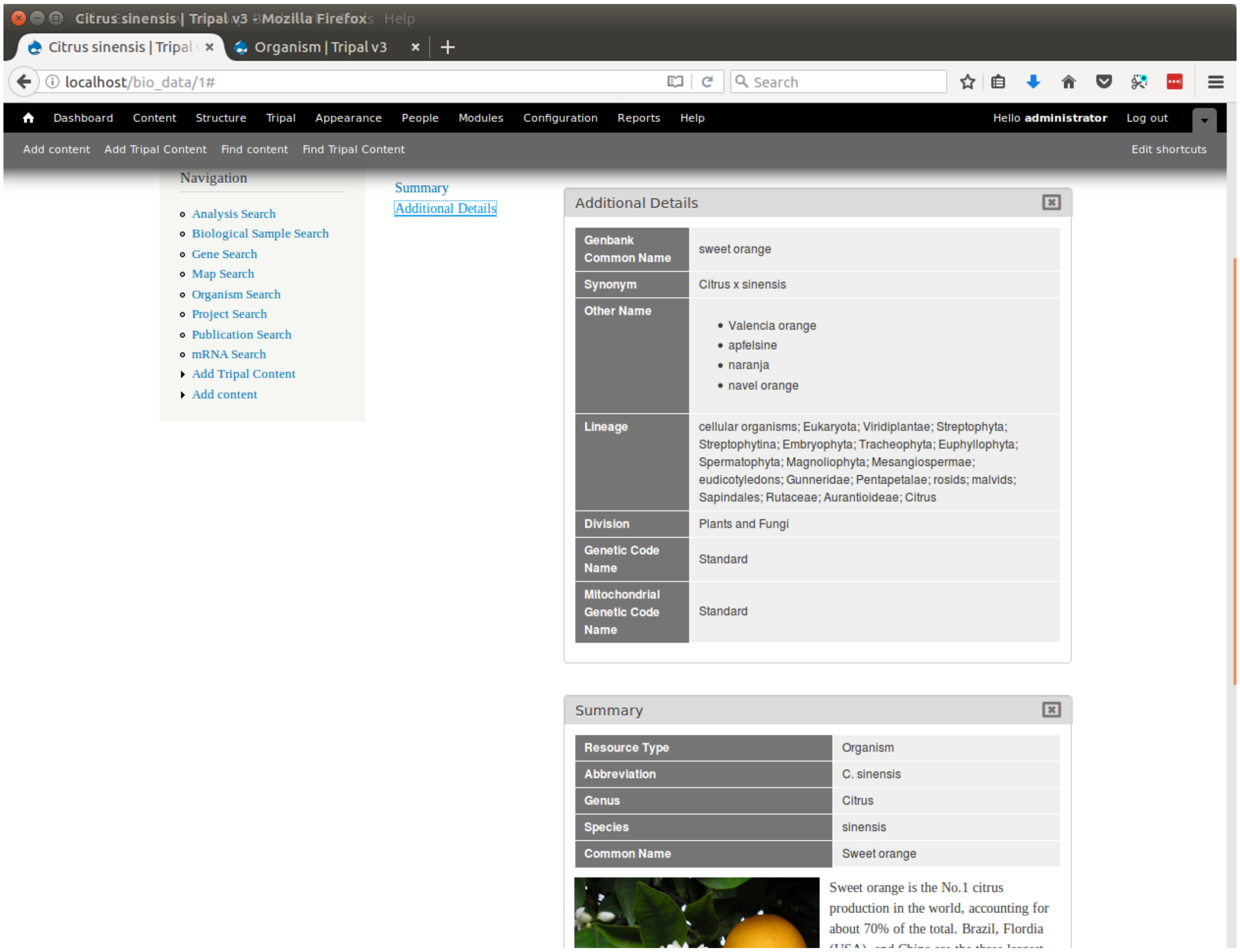
Now, if we return to the organism page we will see these new properties were added to the page inside of the Summary Table.



Resource Type	Organism
Abbreviation	C. sinensis
Genus	Citrus
Species	sinensis
Common Name	Sweet orange
Synonym	Citrus x sinensis
Division	Plants and Fungi
Genbank Common Name	sweet orange
Genetic Code Name	Standard
Mitochondrial Genetic Code Name	Standard
Other Name	• Valencia orange • apfelsine • naranja • navel orange
Lineage	cellular organisms; Eukaryota; Viridiplantae; Streptophyta; Streptophytina; Embryophyta; Tracheophyta; Euphyllophyta; Spermatophyta; Magnoliophyta; Mesangiospermae; eudicotyledons; Gunneridae; Pentapetalae; rosids; malvids; Sapindales; Rutaceae; Aurantioideae; Citrus
Description	Sweet orange is the No.1 citrus production in the world, accounting for about 70% of the total. Brazil, Flordia (USA), and China are the three largest sweet orange producers. Sweet orange fruits have very tight peel and are classified into the hard-to-peel group. They are often used for juice processing, rather than fresh consumption. Valencia

1.6.1.4 Further Customizations

You may not like this arrangement of fields. You may prefer to place these extra fields inside of a new **pane** rather than inside of the **Summary pane**. Perhaps a pane named Additional Details. You can rearrange the order of these fields and create new panes, as desired by following the more details instructions on the Configure Page Display page of this tutorial. For example, the following shows these fields organized into a new pane named **Additional Details** which is separate from the **Summary** Pane. Note the table of contents sidebar now lists the **Summary** and **Additional Details** links. When clicked, the pane selected by the user migrates to the top of the page



Additional Details

Genbank Common Name	sweet orange
Synonym	Citrus x sinensis
Other Name	• Valencia orange • apfelsine • naranja • navel orange
Lineage	cellular organisms; Eukaryota; Viridiplantae; Streptophyta; Streptophytina; Embryophyta; Tracheophyta; Euphyllophyta; Spermatophyta; Magnoliophyta; Mesangiospermae; eudicotyledons; Gunneridae; Pentapetalae; rosids; malvids; Sapindales; Rutaceae; Aurantioideae; Citrus
Division	Plants and Fungi
Genetic Code Name	Standard
Mitochondrial Genetic Code Name	Standard

Summary

Resource Type	Organism
Abbreviation	C. sinensis
Genus	Citrus
Species	sinensis
Common Name	Sweet orange

Sweet orange is the No.1 citrus production in the world, accounting for about 70% of the total. Brazil, Flordia

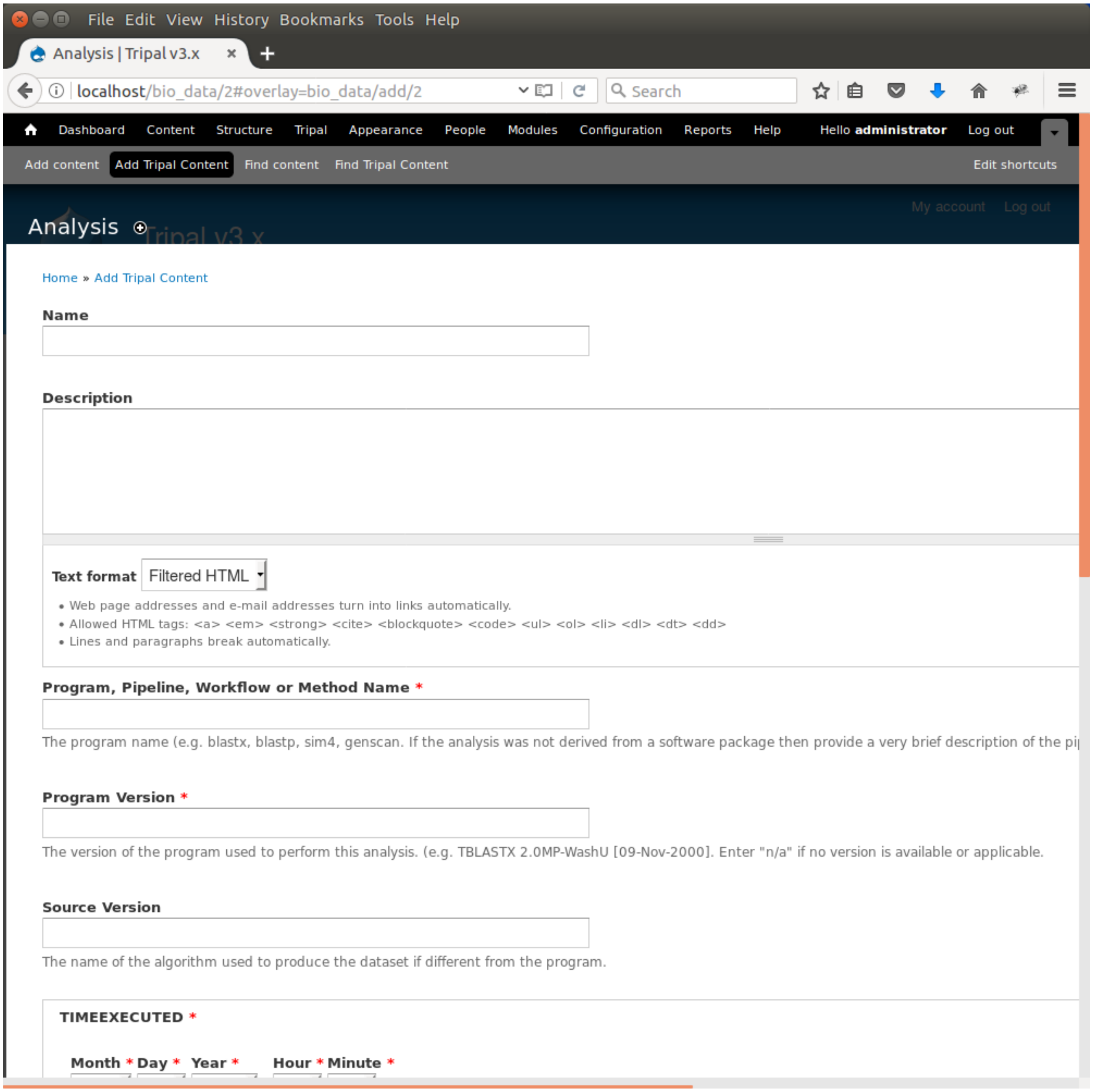
1.6.2 Analyses

For this tutorial we will later import a set of genes, and their associated mRNA, CDS, UTRs, etc. Tripal's Chado loader for importing genomic data requires that an analysis be associated with all imported features. This has several advantages, including:

- The source of features (sequences) can be traced. Even for features simply downloaded from a database, someone else can see where the features came from.
- Provenance describing how the features were created can be provided (e.g. whole genome structural and functional annotation description).
- The analysis associates all of the features together.

To create an analysis for loading our genomic data, navigate to the Add Tripal Content and click on the link: **Analysis**

The analysis creation page will appear:


File Edit View History Bookmarks Tools Help
Analysis | Tripal v3.x
localhost/bio_data/2#overlay=bio_data/add/2
Search
Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help
Hello administrator Log out
Add content Add Tripal Content Find content Find Tripal Content
Edit shortcuts
My account Log out
Tripal v3.x

Analysis

Home » Add Tripal Content

Name

Description

Text format Filtered HTML

- Web page addresses and e-mail addresses turn into links automatically.
- Allowed HTML tags: <a> <em> <strong> <cite> <blockquote> <code> <ul> <ol> <li> <dl> <dt> <dd>
- Lines and paragraphs break automatically.

Program, Pipeline, Workflow or Method Name *

The program name (e.g. blastx, blastp, sim4, genscan. If the analysis was not derived from a software package then provide a very brief description of the pi

Program Version *

The version of the program used to perform this analysis. (e.g. TBLASTX 2.0MP-WashU [09-Nov-2000]. Enter "n/a" if no version is available or applicable.

Source Version

The name of the algorithm used to produce the dataset if different from the program.

TIMEEXECUTED *

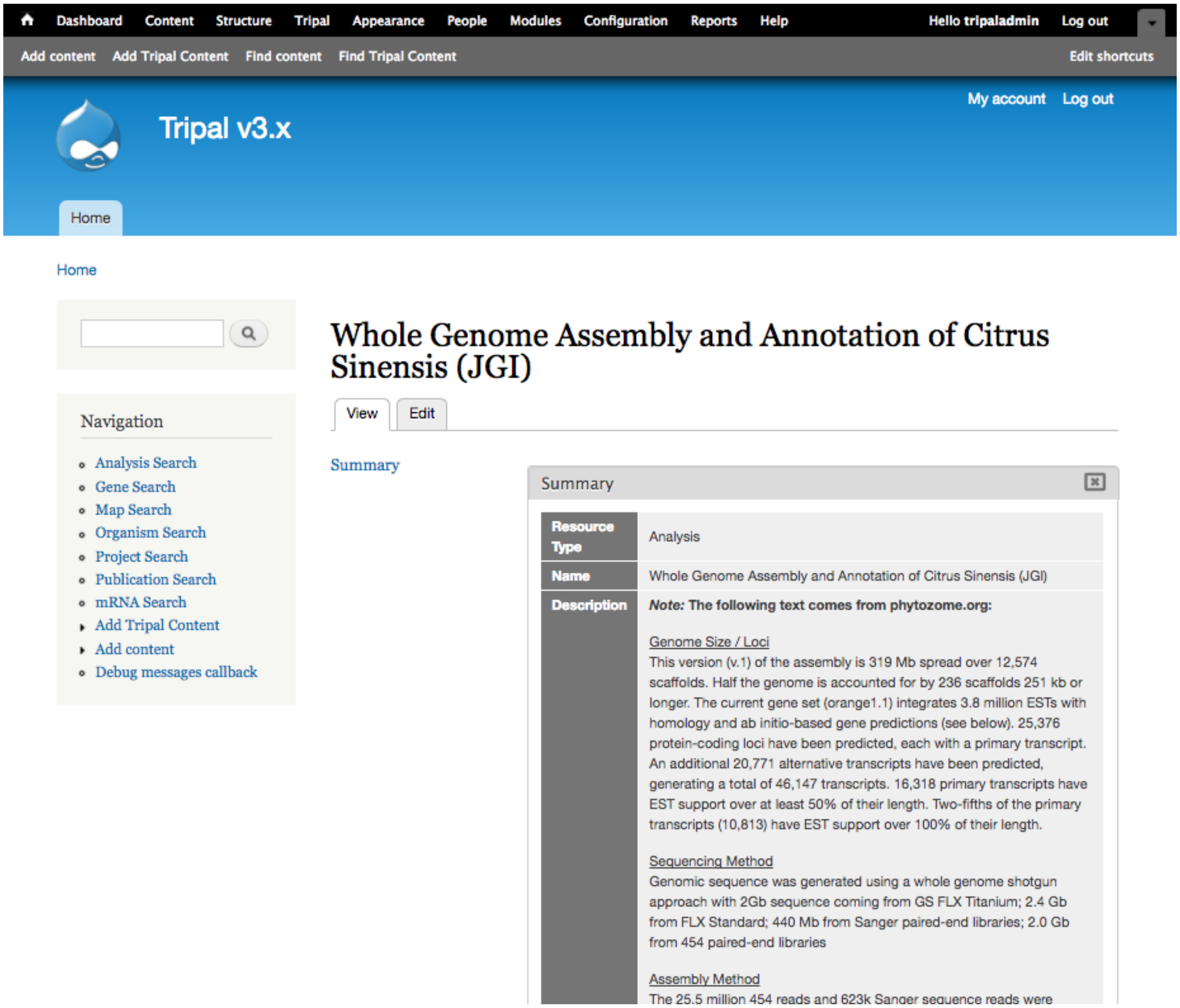
Month * Day * Year * Hour * Minute *

Here you can provide the necessary details to help others understand the source of your data. For this tutorial, enter the following:

Form Element	Value
Name	Whole Genome Assembly and Annotation of Citrus Sinensis (JGI)
Program, Pipeline Name or Method Name	Assembly and Annotation Performed by JGI
Program Version	Phytozome v9
Time Executed	For this tutorial just select any date.
Data Source Name	JGI Citrus sinensis assembly/annotation v1.0 (154)
Data Source URI	http://www.phytozome.net/citrus.php
Description (Set to Full HTML):	<p> <strong><em>Note: </em>The following text comes from phytozome.org:</strong></p> <p> <u>Genome Size / Loci</u><br /> This version (v.1) of the assembly is 319 Mb spread over 12,574 scaffolds. Half the genome is accounted for by 236 scaffolds 251 kb or longer. The current gene set (orange1.1) integrates 3.8 million ESTs with homology and ab initio-based gene predictions (see below). 25,376 protein-coding loci have been predicted, each with a primary transcript. An additional 20,771 alternative transcripts have been predicted, generating a total of 46,147 transcripts. 16,318 primary transcripts have EST support over at least 50% of their length. Two-fifths of the primary transcripts (10,813) have EST support over 100% of their length.</p> <p> <u>Sequencing Method</u><br /> Genomic sequence was generated using a whole genome shotgun approach with 2Gb sequence coming from GS FLX Titanium; 2.4 Gb from FLX Standard; 440 Mb from Sanger paired-end libraries; 2.0 Gb from 454 paired-end libraries</p> <p> <u>Assembly Method</u><br /> The 25.5 million 454 reads and 623k Sanger sequence reads were generated by a collaborative effort by 454 Life Sciences, University of Florida and JGI. The assembly was generated by Brian Desany at 454 Life Sciences using the Newbler assembler.</p> <p> <u>Identification of Repeats</u><br /> A de novo repeat library was made by running RepeatModeler (Arian Smit, Robert Hubley) on the genome to produce a library of repeat sequences. Sequences with Pfam domains associated with non-TE functions were removed from the library of repeat sequences and the library was then used to mask 31% of the genome with RepeatMasker.</p> <p> <u>EST Alignments</u><br /> We aligned the sweet orange EST sequences using Brian Haas's PASA pipeline which aligns ESTs to the best place in the genome via gmap, then filters hits to ensure proper splice boundaries.</p>

Note: Above, the description is provided as HTML code. However if you enabled the **ckeditor** module (as instructed in the Tripal Prerequisites section), you should click the link **Switch to plain-text editor** found below the Description field before cut-and-pasting the code above. Normally, you would enter the text free-hand but for this tutorial it is fastest to cut-and-paste the HTML.

After saving, you should have the following analysis page:

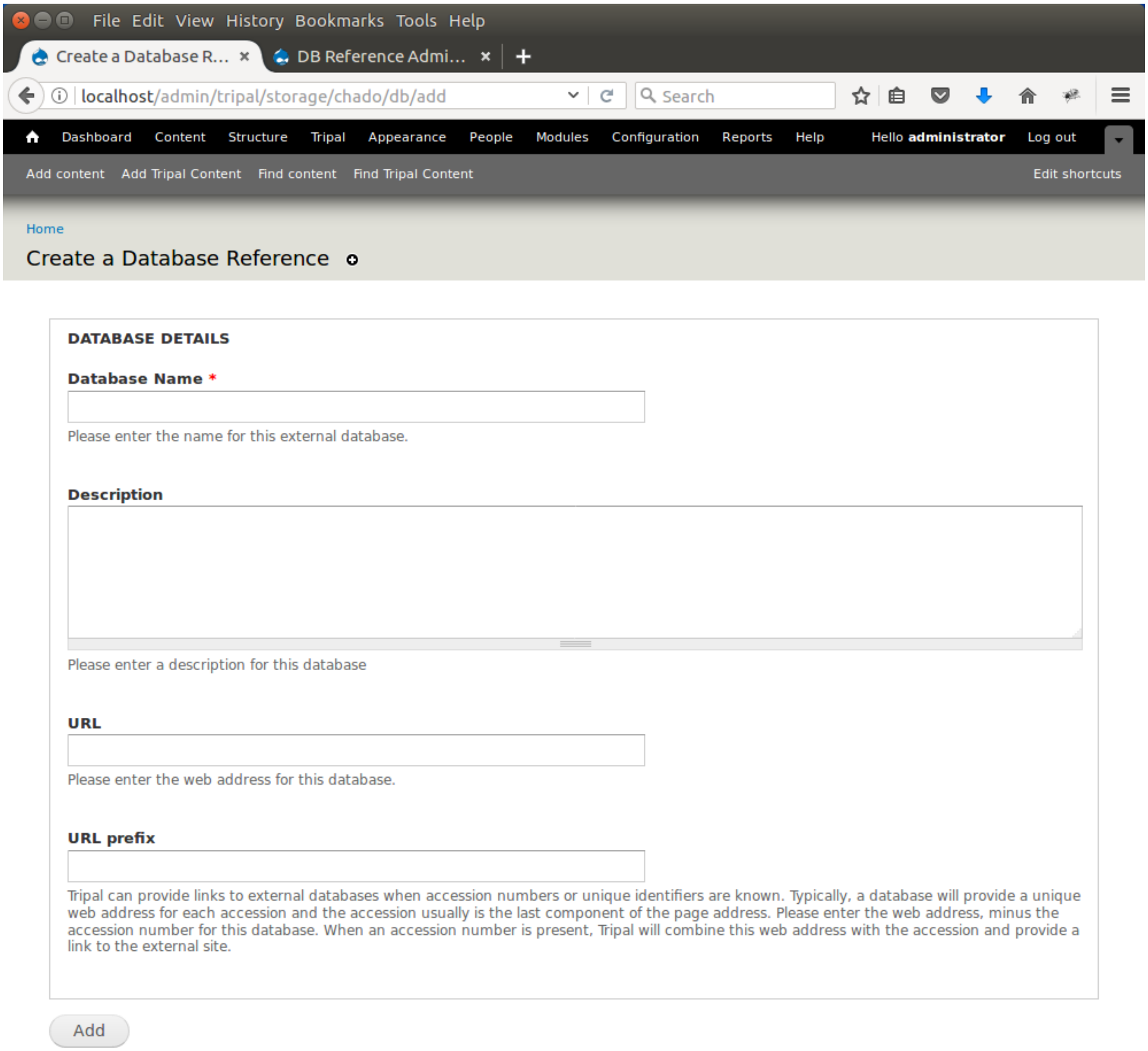


Whole Genome Assembly and Annotation of Citrus Sinensis (JGI)

Resource Type	Analysis
Name	Whole Genome Assembly and Annotation of Citrus Sinensis (JGI)
Description	***Note:*** **The following text comes from phytozome.org:** Genome Size / Loci This version (v.1) of the assembly is 319 Mb spread over 12,574 scaffolds. Half the genome is accounted for by 236 scaffolds 251 kb or longer. The current gene set (orange1.1) integrates 3.8 million ESTs with homology and ab initio-based gene predictions (see below). 25,376 protein-coding loci have been predicted, each with a primary transcript. An additional 20,771 alternative transcripts have been predicted, generating a total of 46,147 transcripts. 16,318 primary transcripts have EST support over at least 50% of their length. Two-fifths of the primary transcripts (10,813) have EST support over 100% of their length. Sequencing Method Genomic sequence was generated using a whole genome shotgun approach with 2Gb sequence coming from GS FLX Titanium; 2.4 Gb from FLX Standard; 440 Mb from Sanger paired-end libraries; 2.0 Gb from 454 paired-end libraries Assembly Method The 25.5 million 454 reads and 623k Sanger sequence reads were

1.6.3 Cross References

For our gene pages and mRNA pages we want to link back to JGI where we obtained the genes. Therefore, we want to add a database reference for JGI. To add a new external databases, navigate to **Tripal** → **Data Loaders** → **Chado Databases** and click the link titled **Add a Database**. The resulting page provides fields for adding a new database:


File Edit View History Bookmarks Tools Help
Create a Database R...
DB Reference Admi...
localhost/admin/tripal/storage/chado/db/add
Search
Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help
Hello administrator Log out
Add content Add Tripal Content Find content Find Tripal Content
Edit shortcuts
Home

Create a Database Reference

DATABASE DETAILS

Database Name *

Please enter the name for this external database.

Description

Please enter a description for this database

URL

Please enter the web address for this database.

URL prefix

Tripal can provide links to external databases when accession numbers or unique identifiers are known. Typically, a database will provide a unique web address for each accession and the accession usually is the last component of the page address. Please enter the web address, minus the accession number for this database. When an accession number is present, Tripal will combine this web address with the accession and provide a link to the external site.

Enter the following values for the fields:

Field Name	Value
Database Name	Phytozome
Description	Phytozome is a joint project of the Department of Energy's Joint Genome Institute and the Center for Integrative Genomics to facilitate comparative genomic studies amongst green plants
URL	http://www.phytozome.net/
URL prefix	https://phytozome.jgi.doe.gov/phytomine/portal.do?externalid=PAC:{accession}

The URL prefix is important as it will be used to create the links on our gene pages. When an object (e.g. gene) is present in another database, typically those database have a unique identifier (or accession) for the resource. If we want to link records in our database to records in the remote database we need to provide a URL prefix that Tripal will use to create the URL. Typically a remote database has a standard URL schema by which someone can specify a unique resource. Often the resource accession is the last word in the URL to allow others to easily build the URL for any resource. Tripal can take advantage of these type URL schemas via the URL Prefix field.

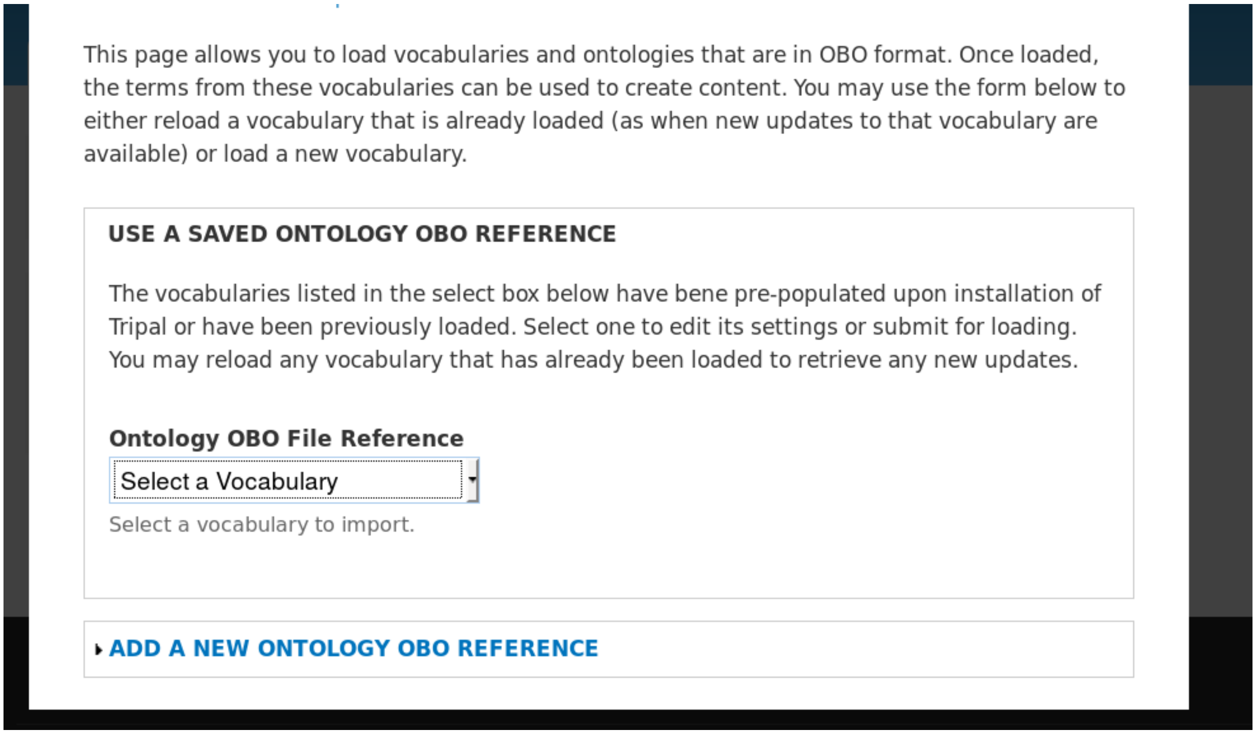
The URL prefix should be the URL used to identify a resource. Two tokens, {db} and {accession}, can be used in place of where the database name and accession might be needed to create the URL. If no {db} or {accession} are provided in the URL prefix then Tripal will append the database name and the accession to the URL prefix to form the final URL. In this example, the Phytozome URL only requires the accession. The position where that accession will be placed is indicated with the {accession} token. The {db} token is not needed.

Click **Add**.

We now have added a new database!

1.6.4 Controlled Vocabularies

Before we proceed with setup of our example genomics site we will want to load the Gene Ontology. This is because we will be loading a whole genome, genes and transcripts with annotations. These annotations include Gene Ontology terms. To load the Gene Ontolgoy, navigate to **Tripal** → **Data Loaders** → **Chado Vocabularies** → **OBO Vocabulary Loader**. You will see the following page:



This page allows you to load vocabularies and ontologies that are in OBO format. Once loaded, the terms from these vocabularies can be used to create content. You may use the form below to either reload a vocabulary that is already loaded (as when new updates to that vocabulary are available) or load a new vocabulary.

USE A SAVED ONTOLOGY OBO REFERENCE

The vocabularies listed in the select box below have bene pre-populated upon installation of Tripal or have been previously loaded. Select one to edit its settings or submit for loading. You may reload any vocabulary that has already been loaded to retrieve any new updates.

Ontology OBO File Reference

Select a Vocabulary

Select a vocabulary to import.

The Ontology loader allows you to select a pre-defined vocabulary for loading or allow you to provide your own. If you provide your own, you give the remote URL of the OBO file or provide the full path on the local web server where the OBO file is located. In the case of a remote URL, Tripal first downloads and then parses the OBO file for loading. If you do provide your own OBO file it will appear in the saved drop down list for loading of future updates to the ontology.

During the Tripal installation portion of this tutorial, several vocabularies were pre-installed for you. The Gene Ontology, however, was not. To import the Gene Ontology, select it from the drop-down and click the Import Vocabulary button. You will notice a job is added to the jobs system. Now manually launch the jobs

```
drush trp-run-jobs --username=administrator --root=/var/www/html
```

Note: Loading the Gene Ontology will take several hours.

1.6.5 Genomes and Genes

1.6.5.1 Loading Feature Data

Now that we have our organism and whole genome analysis ready, we can begin loading genomic data. For this tutorial only a single gene from sweet orange will be loaded into the databsae. This is to ensure we can move through the tutorial rather quickly. The following datasets will be used for this tutorial:

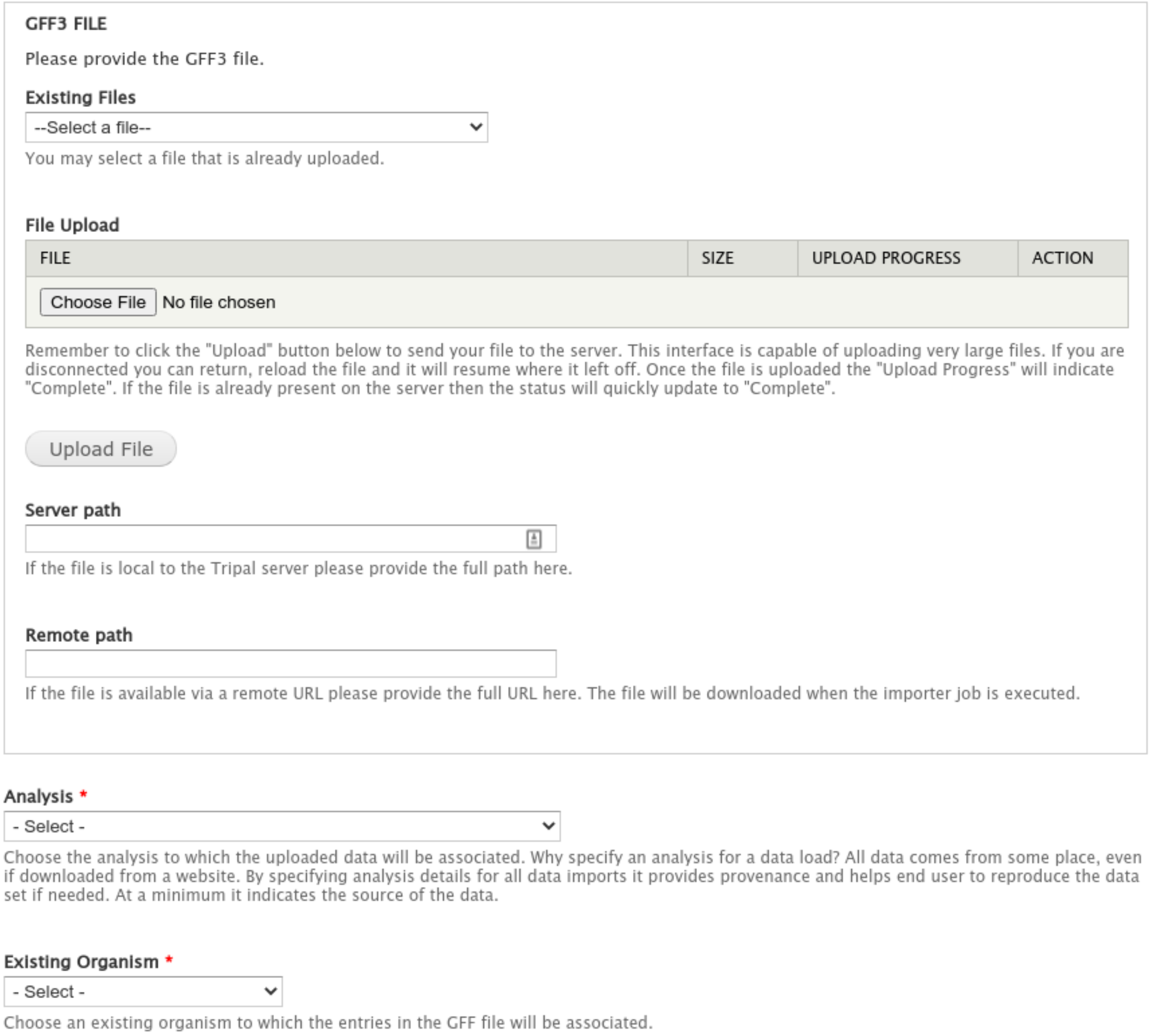
- Citrus sinensis-orange1.1g015632m.g.gff3
- Citrus sinensis-scaffold00001.fasta
- Citrus sinensis-orange1.1g015632m.g.fasta

One of the new features available in many of the Tripal v3 data loaders is an HTML5 file upload element which allows administrators and users to upload large files reliably. This removes the requirement in previous versions of this tutorial to download these files directly on the server and provide a path to the file. Instead, if you have the file on your current local machine you can now simply upload it for loading.

Another new option in Tripal v3 Data Loaders is the ability to provide a remote path of a file to be loaded. This completely alleviates the need to transfer large files multiple times and eases the loading process.

1.6.5.2 Loading a GFF3 File

The gene features (e.g. gene, mRNA, 5_prime_UTRs, CDS 3_prime_UTRS) are stored in the GFF3 file downloaded in the previous step. We will load this GFF3 file and consequently load our gene features into the database. Navigate to **Tripal** → **Data Loaders** → **Chado GFF3 Loader**.



GFF3 FILE

Please provide the GFF3 file.

Existing Files

--Select a file--

You may select a file that is already uploaded.

File Upload

FILE	SIZE	UPLOAD PROGRESS	ACTION
Choose File No file chosen			

Remember to click the "Upload" button below to send your file to the server. This interface is capable of uploading very large files. If you are disconnected you can return, reload the file and it will resume where it left off. Once the file is uploaded the "Upload Progress" will indicate "Complete". If the file is already present on the server then the status will quickly update to "Complete".

Upload File

Server path

If the file is local to the Tripal server please provide the full path here.

Remote path

If the file is available via a remote URL please provide the full URL here. The file will be downloaded when the importer job is executed.

Analysis *

- Select -

Choose the analysis to which the uploaded data will be associated. Why specify an analysis for a data load? All data comes from some place, even if downloaded from a website. By specifying analysis details for all data imports it provides provenance and helps end user to reproduce the data set if needed. At a minimum it indicates the source of the data.

Existing Organism *

- Select -

Choose an existing organism to which the entries in the GFF file will be associated.

Enter the following:

Field Name	Value
File	Upload the file name Citrus_sinensis-orange1.1g015632m.g.gff3
Analysis	Whole Genome Assembly and Annotation of Citrus sinensis
Existing Organism	Citrus sinensis
Landmark Type	supercontig
All other options	leave as default

Note: The Landmark Type is provided for this demo GFF3 file because the chromosome is not defined in the file, only the genomic features on the chromosomes. The landmark type is not needed if the GFF3 file has the chromosomes (scaffolds or contigs) defined in the GFF3 file.

Finally, click the Import GFF3 file button. You'll notice a job was submitted to the jobs subsystem. Now, to complete the process we need the job to run. We'll do this manually:

```
drush trp-run-jobs --username=administrator --root=/var/www/html
```

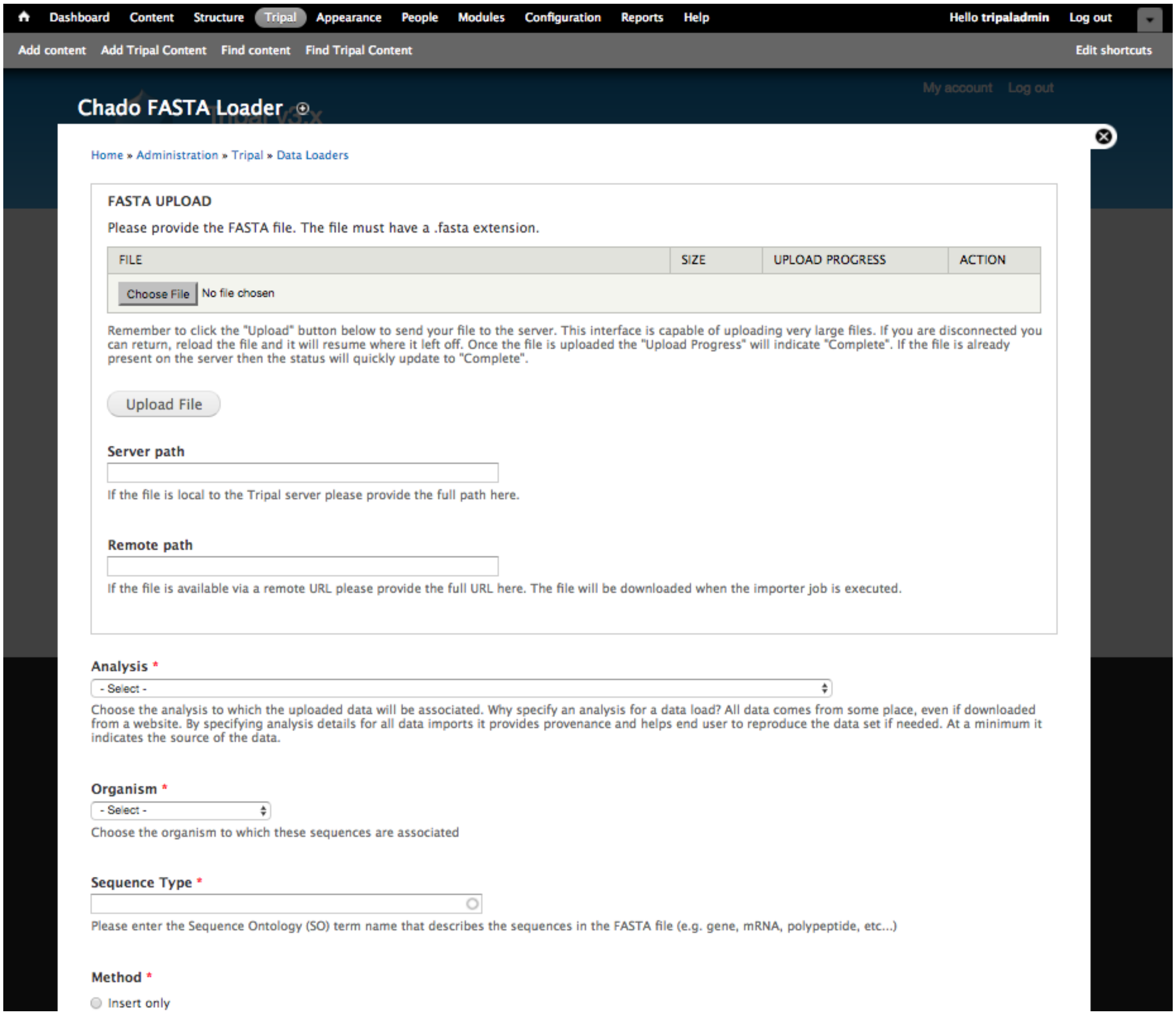
You should see output similar to the following:

```
2020-10-02 21:53:18
Tripal Job Launcher
Running as user 'admin'
-------------------
2020-10-02 21:53:18: There are 1 jobs queued.
2020-10-02 21:53:18: Job ID 1310.
2020-10-02 21:53:18: Calling: tripal_run_importer(123)

Running 'Chado GFF3 File Loader' importer
NOTE: Loading of file is performed using a database transaction.
If it fails or is terminated prematurely then all insertions and
updates are rolled back and will not be found in the database

Opening /var/www/html/sites/default/files/tripal/users/1/Citrus_sinensis-orange1.
↪1g015632m.g.gff3
Opening temporary cache file: /tmp/TripalGFF3Import_aUgoru
Step  1 of 26: Caching GFF3 file...
Step  2 of 26: Find existing landmarks...
Step  3 of 26: Insert new landmarks (if needed)...
Step  4 of 26: Find missing proteins...
Step  5 of 26: Add missing proteins to list of features...
Step  6 of 26: Find existing features...
Step  7 of 26: Clear attributes of existing features...
Step  8 of 26: Processing 135 features...
Step  9 of 26: Get new feature IDs...
Step 10 of 26: Insert locations...
Step 11 of 26: Associate parents and children...
Step 12 of 26: Calculate child ranks...
Step 13 of 26: Add child-parent relationships...
Step 14 of 26: Insert properties...
Step 15 of 26: Find synonyms (aliases)...
Step 16 of 26: Insert new synonyms (aliases)...
Step 17 of 26: Insert feature synonyms (aliases)...
Step 18 of 26: Find cross references...
Step 19 of 26: Insert new cross references...
Step 20 of 26: Get new cross references IDs...
Step 21 of 26: Insert feature cross references...
Step 22 of 26: Insert feature ontology terms...
Step 23 of 26: Insert 'derives_from' relationships...
Step 24 of 26: Insert Targets...
Step 25 of 26: Associate features with analysis....
Step 26 of 26: Adding sequences data (Skipped: none available)...

Done.
Committing Transaction...

Remapping Chado Controlled vocabularies to Tripal Terms...
Done.
```

Note: For very large GFF3 files the loader can take quite a while to complete.

1.6.5.3 Loading FASTA files

Using the Tripal GFF3 loader we were able to populate the database with the genomic features for our organism. However, those features now need nucleotide sequence data. To do this, we will load the nucleotide sequences for the mRNA features and the scaffold sequence. Navigate to the **Tripal** → **Data Loaders** → **Chado FASTA Loader**.



Chado FASTA Loader

Home » Administration » Tripal » Data Loaders

FASTA UPLOAD

Please provide the FASTA file. The file must have a .fasta extension.

FILE	SIZE	UPLOAD PROGRESS	ACTION
Choose File No file chosen			

Remember to click the "Upload" button below to send your file to the server. This interface is capable of uploading very large files. If you are disconnected you can return, reload the file and it will resume where it left off. Once the file is uploaded the "Upload Progress" will indicate "Complete". If the file is already present on the server then the status will quickly update to "Complete".

Server path

If the file is local to the Tripal server please provide the full path here.

Remote path

If the file is available via a remote URL please provide the full URL here. The file will be downloaded when the importer job is executed.

Analysis *

- Select -

Choose the analysis to which the uploaded data will be associated. Why specify an analysis for a data load? All data comes from some place, even if downloaded from a website. By specifying analysis details for all data imports it provides provenance and helps end user to reproduce the data set if needed. At a minimum it indicates the source of the data.

Organism *

- Select -

Choose the organism to which these sequences are associated

Sequence Type *

Please enter the Sequence Ontology (SO) term name that describes the sequences in the FASTA file (e.g. gene, mRNA, polypeptide, etc...)

Method *

○ Insert only

Before loading the FASTA file we must first know the Sequence Ontology (SO) term that describes the sequences we are about to upload. We can find the appropriate SO terms from our GFF file. In the GFF file we see the SO terms that correspond to our FASTA files are 'scaffold' and 'mRNA'.

Note: It is important to ensure prior to importing, that the FASTA loader will be able to appropriately match the sequence in the FASTA file with existing sequences in the database. Before loading FASTA files, take special care to ensure the definition line of your FASTA file can uniquely identify the feature for the specific organism and sequence type.

For example, in our GFF file an mRNA feature appears as follows:

```
scaffold00001   phytozome6      mRNA    4058460 4062210 .       +       .       ␣
↪ID=PAC:18136217;Name=orange1.1g015632m;PACid=18136217;Parent=orange1.1g015632m.g
```

Note that for this mRNA feature the ID is **PAC:18136217** and the name is **orange1.1g015632m**. In Chado, features always have a human readable name which does not need to be unique, and also a unique name which must be unique for the organism and SO type. In the GFF file, the ID becomes the unique name and the Name becomes the human readable name.

In our FASTA file the definition line for this mRNA is:

```
>orange1.1g015632m PAC:18136217 (mRNA) Citrus sinensis
```

By default Tripal will match the sequence in a FASTA file with the feature that matches the first word in the definition line. In this case the first word is **orange1.1g015632m**. As defined in the GFF file, the name and unique name are different for this mRNA. However, we can see that the first word in the definition line of the FASTA file is the name and the second is the unique name. Therefore, when we load the FASTA file we should specify that we are matching by the name because it appears first in the definition line.

If however, we cannot guarantee the that feature name is unique then we can use a regular expressions in the **Advanced Options** to tell Tripal where to find the name or unique name in the definition line of your FASTA file.

Note: When loading FASTA files for features that have already been loaded via a GFF file, always choose “Update only” as the import method. Otherwise, Tripal may add the features in the FASTA file as new features if it cannot properly match them to existing features.

Now, enter the following values in the fields on the web form:

Field Name	Value
FASTA file	Upload the file named Citrus_sinensis-scaffold00001.fasta
Analysis	Whole Genome Assembly and Annotation of Citrus sinensis
Organism	Citrus sinensis (Sweet orange)
Sequence type	supercontig (scaffold is an alias for supercontig in the sequence ontology)
Method	Update only (we do not want to insert these are they should already be there)
Name Match Type	Name

Click the Import Fasta File, and a job will be added to the jobs system. Run the job:

```
drush trp-run-jobs --username=administrator --root=/var/www/html
```

Notice that the loader reports the it “Found 1 sequences(s).”. Next fill out the same form for the mRNA (transcripts) FASTA file:

Field Name	Value
FASTA file	Upload the file named Citrus_sinensis-orange1.1g015632m.g.fasta
Analysis	Whole Genome Assembly and Annotation of Citrus sinensis
Organism	Citrus sinensis (Sweet orange)
Sequence type	mRNA
Method	Update only
Name Match	Name

The FASTA loader has some advanced options. The advanced options allow you to create relationships between features and associate them with external databases. For example, the definition line for the mRNA in our FASTA file is:

```
>orange1.1g015632m PAC:18136217 (mRNA) Citrus sinensis
```

Here we have more information than just the feature name. We have a unique Phytozome accession number (e.g. PAC:18136217) for the mRNA. Using the **External Database Reference** section under **Additional Options** we can import this information to associate the Phytozome accession with the features. A regular expression is required to uniquely capture that ID. In the example above the unique accession is 18136217. Because Tripal is a PHP application, the syntax for regular expressions follows the PHP method. Documentation for regular expressions used in PHP can be found here. Enter the following value to make the associate between the mRNA and it's corresponding accession at Phytozome:

Field Name	Value
External Database	Phytozome
Regular expression for the accession	^.*PAC:(d+).*$

Remember, we have the name **Phytozome** in our **External Database** drop down because we manually added it as a database cross reference earlier in the turorial. After adding the values above, click the **Import FASTA file** button, and manually run the submitted job:

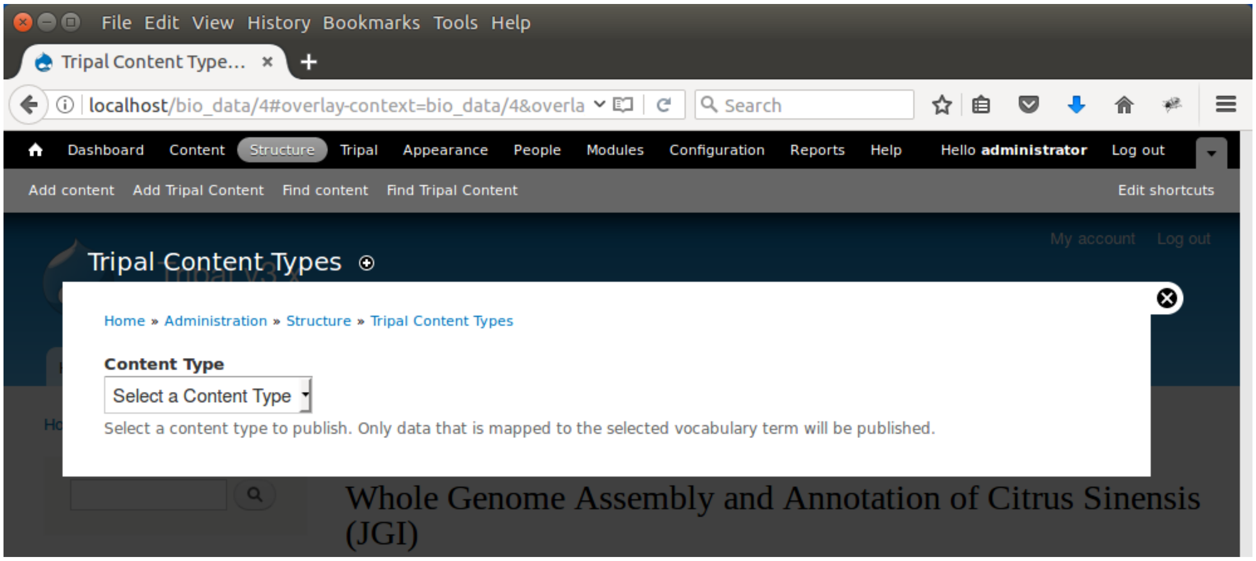
```
drush trp-run-jobs --username=administrator --root=/var/www/html
```

Now the scaffold sequence and mRNA sequences are loaded!

Note: It is not required to load the mRNA sequences as those can be derived from their alignments with the scaffold sequence. However, in Chado the **feature** table has a **residues** column. Therefore, it is best practice to load the sequence when possible.

1.6.5.4 Creating Gene Pages

Now that we've loaded our feature data, we must publish them. This is different than when we manually created our Organism and Analysis pages. Using the GFF and FASTA loaders we imported our data into Chado, but currently there are no published pages for this data that we loaded. To publish these genomic features, navigating to Structure → Tripal Content Types and click the link titled Publish Chado Content. The following page appears:


File Edit View History Bookmarks Tools Help
Tripal Content Type...
localhost/bio_data/4#overlay-context=bio_data/4&overla
Search
Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help Hello administrator Log out
Add content Add Tripal Content Find content Find Tripal Content Edit shortcuts
My account Log out
Tripal Content Types
Home » Administration » Structure » Tripal Content Types

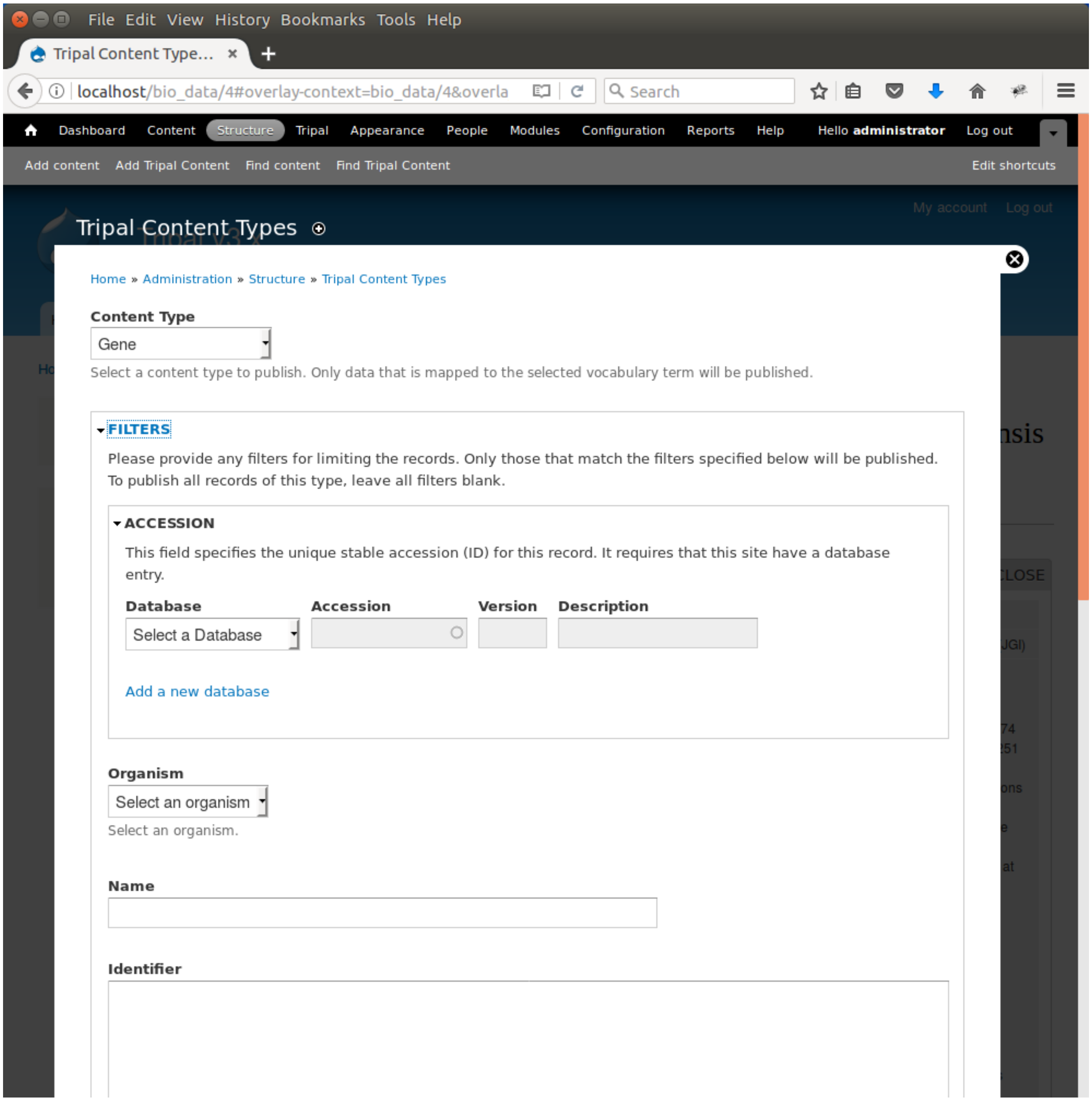
Content Type

Select a Content Type

Select a content type to publish. Only data that is mapped to the selected vocabulary term will be published.

Whole Genome Assembly and Annotation of Citrus Sinensis (JGI)

Here we can specify the types of content to publish. For our site we want to offer both gene and mRNA pages (these types were present in our GFF file). First, to create pages for genes select 'Gene' from the dropdown. A new Filter section is present and when opened appears as follows.



File Edit View History Bookmarks Tools Help
Tripal Content Type...
localhost/bio_data/4#overlay-context=bio_data/4&overla
Search
Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help Hello administrator Log out
Add content Add Tripal Content Find content Find Tripal Content Edit shortcuts
My account Log out
Tripal Content Types

Home » Administration » Structure » Tripal Content Types

Content Type

Gene

Select a content type to publish. Only data that is mapped to the selected vocabulary term will be published.

FILTERS

Please provide any filters for limiting the records. Only those that match the filters specified below will be published. To publish all records of this type, leave all filters blank.

ACCESSION

This field specifies the unique stable accession (ID) for this record. It requires that this site have a database entry.

Database Select a Database | **Accession** | **Version** | **Description**

Add a new database

Organism

Select an organism

Select an organism.

Name

Identifier

The **Filters** section allows you to provide filters to limit what you want to publish. For example, if you only want to publish genes for a single organism you can select that organism in the Organism drop down list. We only have one organism in our site, but for the sake of experience, add a filter to publish only genes for Citrus sinesis by selecting it from the Organism drop down. Scroll to the bottom a click the Publish button. A new job is added to the job queue. Manually run the job:

```
drush trp-run-jobs --username=administrator --root=/var/www/html
```

You should see output similar to the following:

```
Tripal Job Launcher
Running as user 'administrator'
-------------------
Calling: tripal_chado_publish_records(Array, 12)
```

(continues on next page)

(continued from previous page)

```

NOTE: publishing records is performed using a database transaction.
If the load fails or is terminated prematurely then the entire set of
is rolled back with no changes to the database

Succesfully published 1 Gene record(s).
```

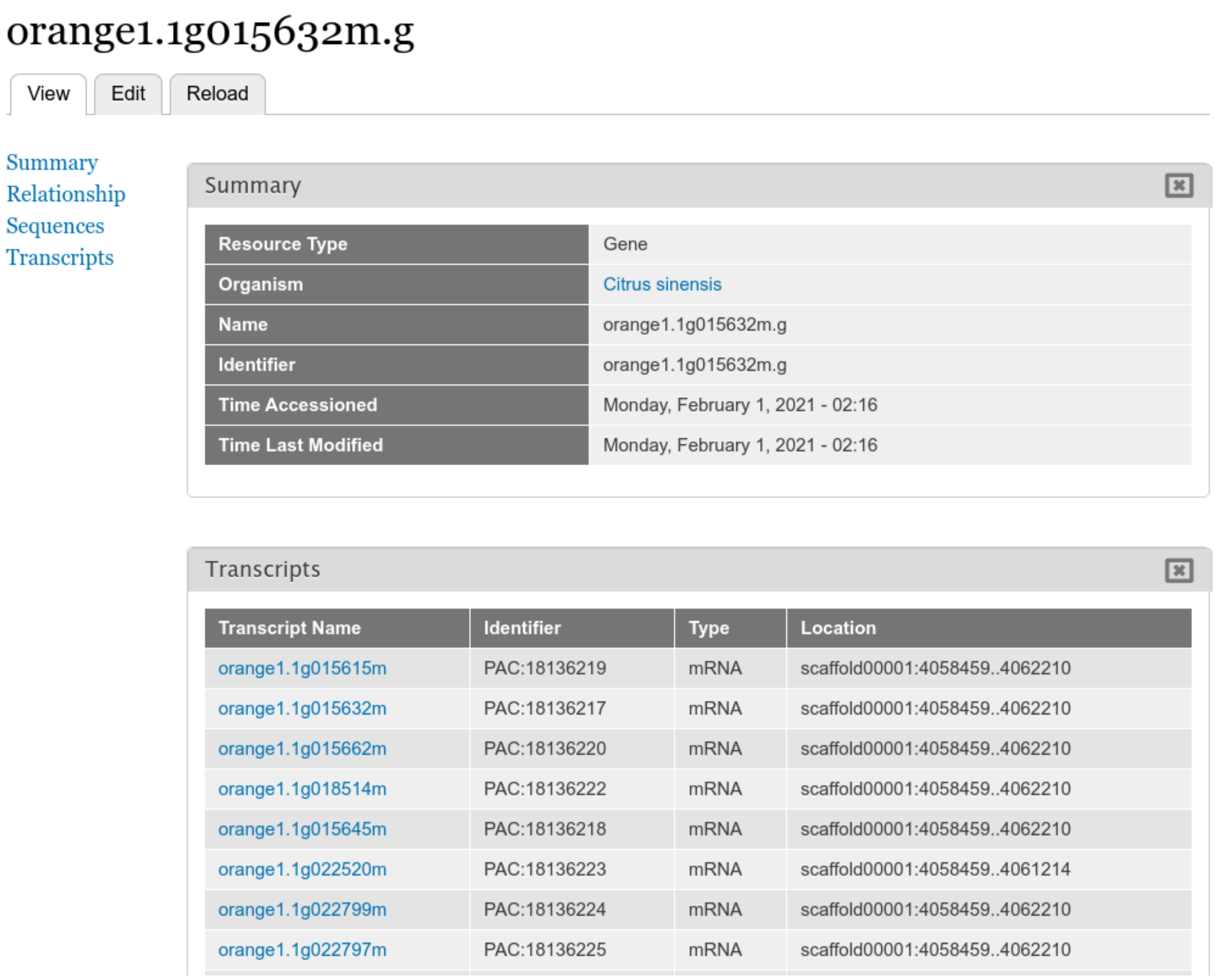
Here we see that 1 gene was successfully published. This is because the GFF file we used previously to import the genes only had one gene present.

Now, repeat the steps above to publish the mRNA content type. You should see that 9 mRNA records were published:

```
Tripal Job Launcher
Running as user 'administrator'
-------------------
Calling: tripal_chado_publish_records(Array, 13)

NOTE: publishing records is performed using a database transaction.
If the load fails or is terminated prematurely then the entire set of
is rolled back with no changes to the database

Succesfully published 9 mRNA record(s).
```

Note: It is not necessary to publish all types of features in the GFF file. For example, we do not want to publish features of type **scaffold**. The feature is large and would have many relationships to other features, as well as a very long nucleotide sequence. These can greatly slow down page loading, and in general would be overwhelming to the user to view on one page. As another example, each **mRNA** is composed of several **CDS** features. These **CDS** features do not need their own page and therefore do not need to be published.

Now, we can view our gene and mRNA pages. Click the Find Tripal Content link. Find and click the new page titled **orange1.1g015632m.g**. Here we can see the gene feature we added and its corresponding mRNA's.



orange1.1g015632m.g

Summary

Resource Type	Gene
Organism	Citrus sinensis
Name	orange1.1g015632m.g
Identifier	orange1.1g015632m.g
Time Accessioned	Monday, February 1, 2021 - 02:16
Time Last Modified	Monday, February 1, 2021 - 02:16

Transcripts

Transcript Name	Identifier	Type	Location
orange1.1g015615m	PAC:18136219	mRNA	scaffold00001:4058459..4062210
orange1.1g015632m	PAC:18136217	mRNA	scaffold00001:4058459..4062210
orange1.1g015662m	PAC:18136220	mRNA	scaffold00001:4058459..4062210
orange1.1g018514m	PAC:18136222	mRNA	scaffold00001:4058459..4062210
orange1.1g015645m	PAC:18136218	mRNA	scaffold00001:4058459..4062210
orange1.1g022520m	PAC:18136223	mRNA	scaffold00001:4058459..4061214
orange1.1g022799m	PAC:18136224	mRNA	scaffold00001:4058459..4062210
orange1.1g022797m	PAC:18136225	mRNA	scaffold00001:4058459..4062210

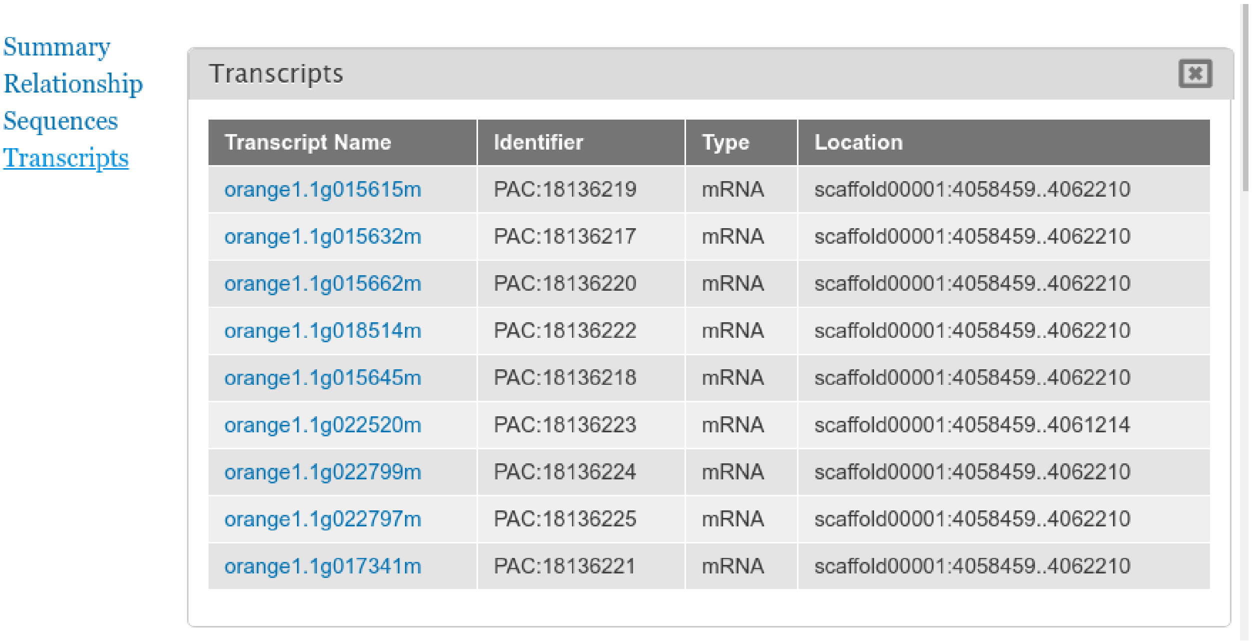
Next find an mRNA page to view. Remember when we loaded our FASTA file for mRNA that we associated the record with Phytozome. On these mRNA pages you will see a link in the left side bar titled **Database Cross Reference**. Clicking that will open a panel with a link to Phytozome. This link appears because:

- We added a Database Cross Reference for Phytozome in a previous step
- We associated the Phytozome accession with the features using a regular expression when importing the FASTA file.

All data that appears on the page is derived from the GFF file and the FASTA files we loaded.

1.6.5.5 Customizing Transcripts on Gene Pages

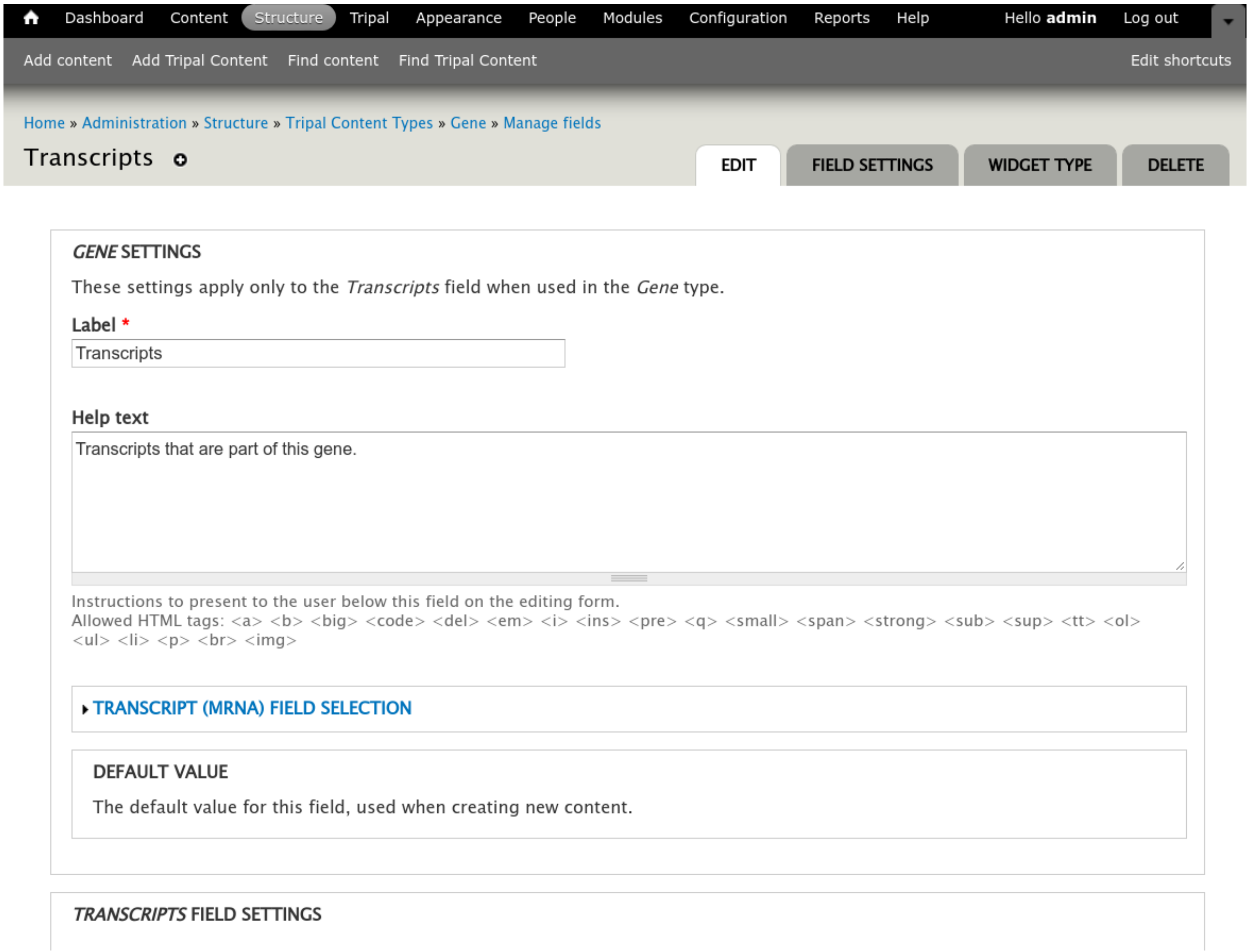
By default the gene pages provided by Tripal will have a link in the sidebar table of contents named **Transcripts** and when clicked a table appears that lists all of the transcripts (or mRNA) that belong to the gene. The user can click to view more information about each published transcript.



Transcripts

Transcript Name	Identifier	Type	Location
orange1.1g015615m	PAC:18136219	mRNA	scaffold00001:4058459..4062210
orange1.1g015632m	PAC:18136217	mRNA	scaffold00001:4058459..4062210
orange1.1g015662m	PAC:18136220	mRNA	scaffold00001:4058459..4062210
orange1.1g018514m	PAC:18136222	mRNA	scaffold00001:4058459..4062210
orange1.1g015645m	PAC:18136218	mRNA	scaffold00001:4058459..4062210
orange1.1g022520m	PAC:18136223	mRNA	scaffold00001:4058459..4061214
orange1.1g022799m	PAC:18136224	mRNA	scaffold00001:4058459..4062210
orange1.1g022797m	PAC:18136225	mRNA	scaffold00001:4058459..4062210
orange1.1g017341m	PAC:18136221	mRNA	scaffold00001:4058459..4062210

Sometimes however, more than just a listing of transcripts is desired on a gene page. You can customize the information that is presented about each transcript by navigating to the gene content type at **Structure** → **Tripal Content Types** and clicking **mange fields** in the **Gene** row. This page allows you to customize the way fields are displayed on the gene page. Scroll down the page to the **Transcript** row and click the **edit** button. The following page should appear.



Transcripts

GENE SETTINGS

These settings apply only to the *Transcripts* field when used in the *Gene* type.

Label *

Transcripts

Help text

Transcripts that are part of this gene.

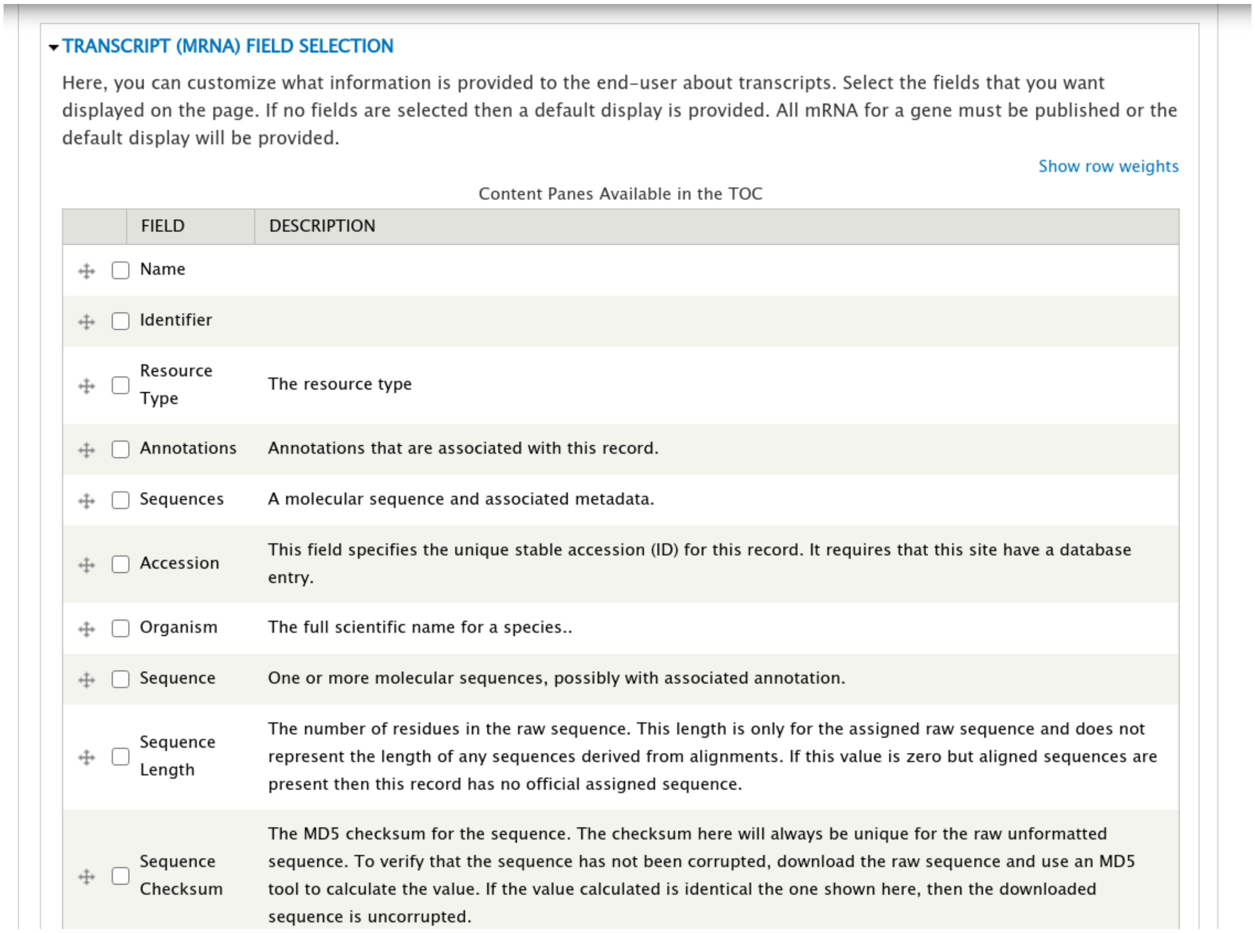
Instructions to present to the user below this field on the editing form.
Allowed HTML tags: <a> <b> <big> <code> <del> <em> <i> <ins> <pre> <q> <small> <span> <strong> <sub> <sup> <tt> <ol> <ul> <li> <p>
 <img>

DEFAULT VALUE

The default value for this field, used when creating new content.

TRANSCRIPTS FIELD SETTINGS

Open the field set titled **Transcript (mRNA) Field Selection** to view a table that lists all of the available fields for a transcript.



▾ TRANSCRIPT (MRNA) FIELD SELECTION

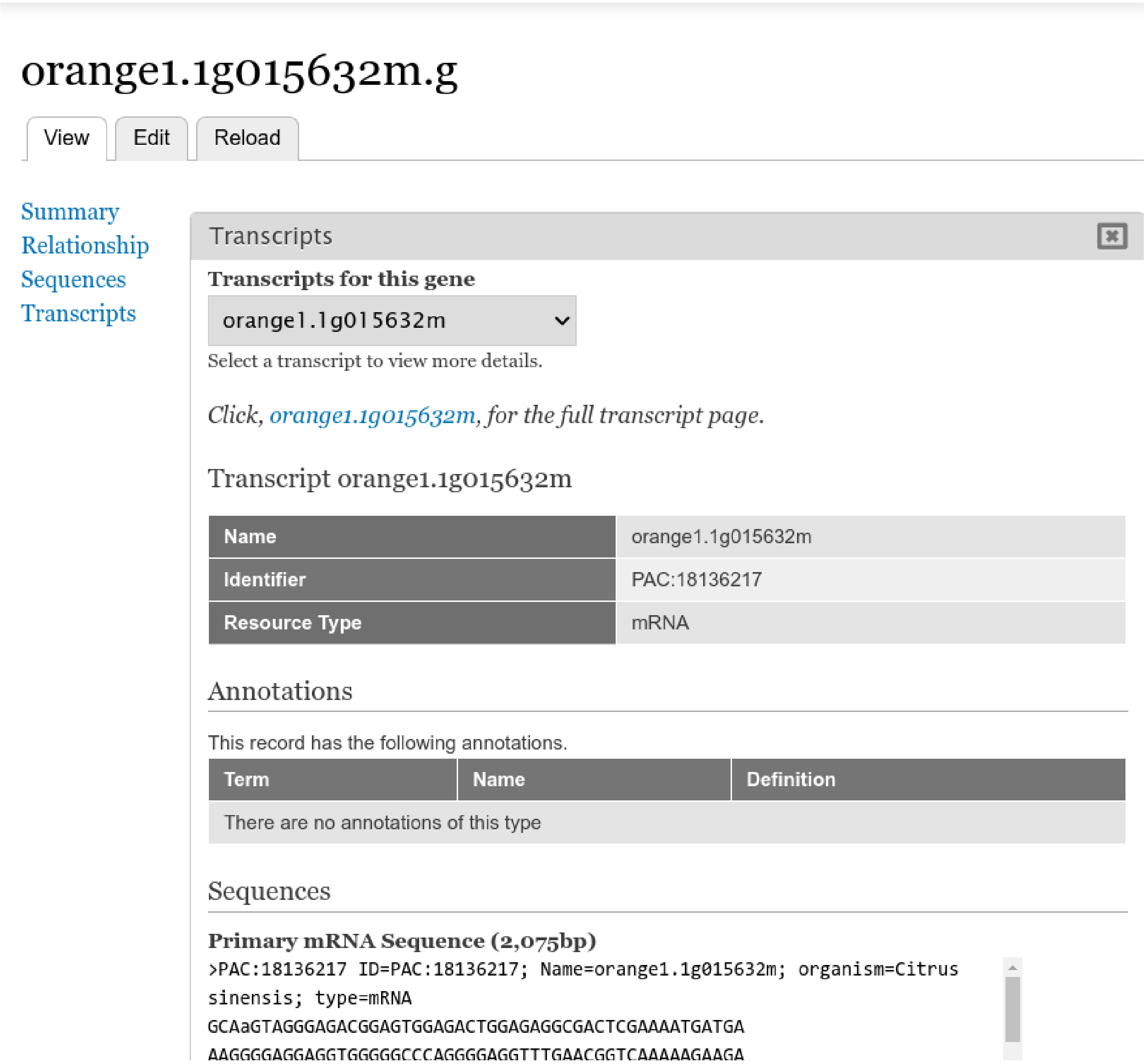
Here, you can customize what information is provided to the end-user about transcripts. Select the fields that you want displayed on the page. If no fields are selected then a default display is provided. All mRNA for a gene must be published or the default display will be provided.

Content Panes Available in the TOC

	FIELD	DESCRIPTION
✥ □	Name	
✥ □	Identifier	
✥ □	Resource Type	The resource type
✥ □	Annotations	Annotations that are associated with this record.
✥ □	Sequences	A molecular sequence and associated metadata.
✥ □	Accession	This field specifies the unique stable accession (ID) for this record. It requires that this site have a database entry.
✥ □	Organism	The full scientific name for a species..
✥ □	Sequence	One or more molecular sequences, possibly with associated annotation.
✥ □	Sequence Length	The number of residues in the raw sequence. This length is only for the assigned raw sequence and does not represent the length of any sequences derived from alignments. If this value is zero but aligned sequences are present then this record has no official assigned sequence.
✥ □	Sequence Checksum	The MD5 checksum for the sequence. The checksum here will always be unique for the raw unformatted sequence. To verify that the sequence has not been corrupted, download the raw sequence and use an MD5 tool to calculate the value. If the value calculated is identical the one shown here, then the downloaded sequence is uncorrupted.

On this page you can check the boxes next to the field that you want to show for a transcript on the gene page. For this example, we will select the fields **Name**, **Identifier**, **Resource Type**, **Anotations**, and **Sequences** (they may not be in this order on your own site). You can control the order in which fields will be shown by dragging them using the crosshairs icon next to each one. Scroll to the bottom of the page and click the **Save Settings** button.

Next return to the gene page, reload it, and click on the **Transcripts** link. Now you are provided a select box with the transcript names. When a transcript is selected, the pane below will populate with the fields that you selected when editing in the Transcript field.



orange1.1g015632m.g

Transcripts

Transcripts for this gene

orange1.1g015632m

Select a transcript to view more details.

Click, orange1.1g015632m, for the full transcript page.

Transcript orange1.1g015632m

Name	orange1.1g015632m
Identifier	PAC:18136217
Resource Type	mRNA

Annotations

This record has the following annotations.

Term	Name	Definition
There are no annotations of this type		

Sequences

Primary mRNA Sequence (2,075bp)

```
>PAC:18136217 ID=PAC:18136217; Name=orange1.1g015632m; organism=Citrus
sinensis; type=mRNA
GCAaGTAGGGAGACGGAGTGGAGACTGGAGAGGCGACTCGAAAATGATGA
AAGGGGAGGAGGTGGGGGCCCAGGGGAGGTTTGAACGGTCAAAAAGAAGA
```

You can return to the Transcript field edit page under the Gene content type at any time to add, remove or change the order of fields that appear for the transcript.

Note: Transcripts on a gene page can only be customized if all of them are published. If not, the default table listing is shown.

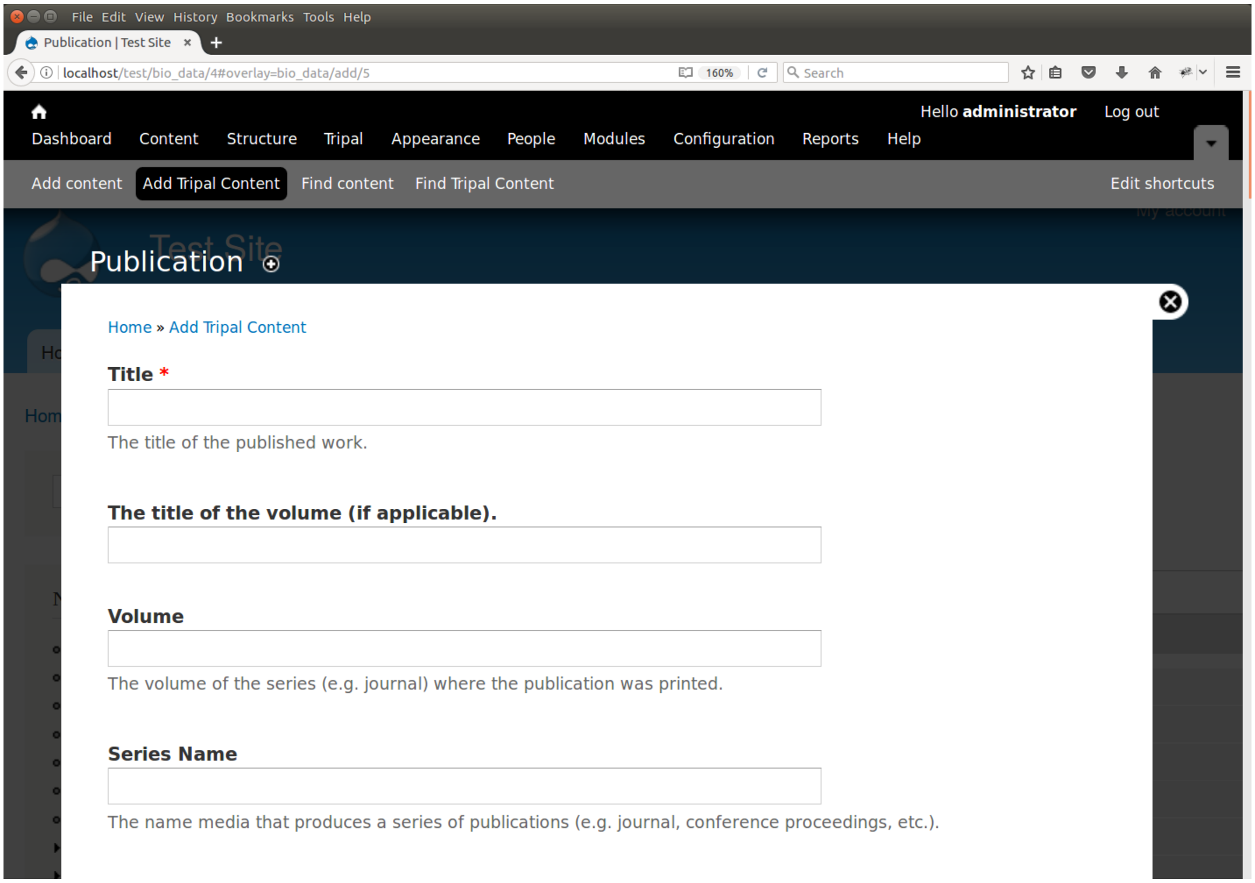
1.6.6 Importing Publications

Note: Remember you must set the `$DRUPAL_HOME` environment variable if you want to cut-and-paste the commands below. See *DRUPAL_HOME Variable*

Tripal provides an interface for automatically and manually adding publications.

1.6.6.1 Manually Adding a Publication

First, we will manually add a publication. Click the Add Tripal Content link in the administrative menu and then Publication.



Publication

Home » Add Tripal Content

Title *

The title of the published work.

The title of the volume (if applicable).

Volume

The volume of the series (e.g. journal) where the publication was printed.

Series Name

The name media that produces a series of publications (e.g. journal, conference proceedings, etc.).

We will add information about the Tripal publication. Enter the following values:

Field Name	Value
Title	Tripal v3: an ontology-based toolkit for construction of FAIR biological community databases.
Volume	baz077
Series Name	Database
Publication Year	2019
Unique Local Identifier	Tripal v3: an ontology-based toolkit for construction of FAIR biological community databases.
Publication Type	Journal
Cross Reference	Database: PMID
Accession	31328773
Publication Date	2019 Jul 22
Citation	Spoor S, Cheng CH, Sanderson LA, Condon B, Almsaeed A, Chen M, Bretaudeau A, Rasche H, Jung S, Main D, Bett K, Staton M, Wegrzyn JL, Feltus FA, Ficklin SP. Tripal v3: an ontology-based toolkit for construction of FAIR biological community databases. Database. July 2019, 2019: baz077
Authors	Spoor S, Cheng CH, Sanderson LA, Condon B, Almsaeed A, Chen M, Bretaudeau A, Rasche H, Jung S, Main D, Bett K, Staton M, Wegrzyn JL, Feltus FA, Ficklin SP
Abstract	Community biological databases provide an important online resource for both public and private data, analysis tools and community engagement. These sites house genomic, transcriptomic, genetic, breeding and ancillary data for specific species, families or clades. Due to the complexity and increasing quantities of these data, construction of online resources is increasingly difficult especially with limited funding and access to technical expertise. Furthermore, online repositories are expected to promote FAIR data principles (findable, accessible, interoperable and reusable) that presents additional challenges. The open-source Tripal database toolkit seeks to mitigate these challenges by creating both the software and an interactive community of developers for construction of online community databases. Additionally, through coordinated, distributed co-development, Tripal sites encourage community-wide sustainability. Here, we report the release of Tripal version 3 that improves data accessibility and data sharing through systematic use of controlled vocabularies (CVs). Tripal uses the community-developed Chado database as a default data store, but now provides tools to support other data stores, while ensuring that CVs remain the central organizational structure for the data. A new site developer can use Tripal to develop a basic site with little to no programming, with the ability to integrate other data types using extension modules and the Tripal application programming interface. A thorough online User's Guide and Developer's Handbook are available at http://tripal.info, providing download, installation and step-by-step setup instructions.

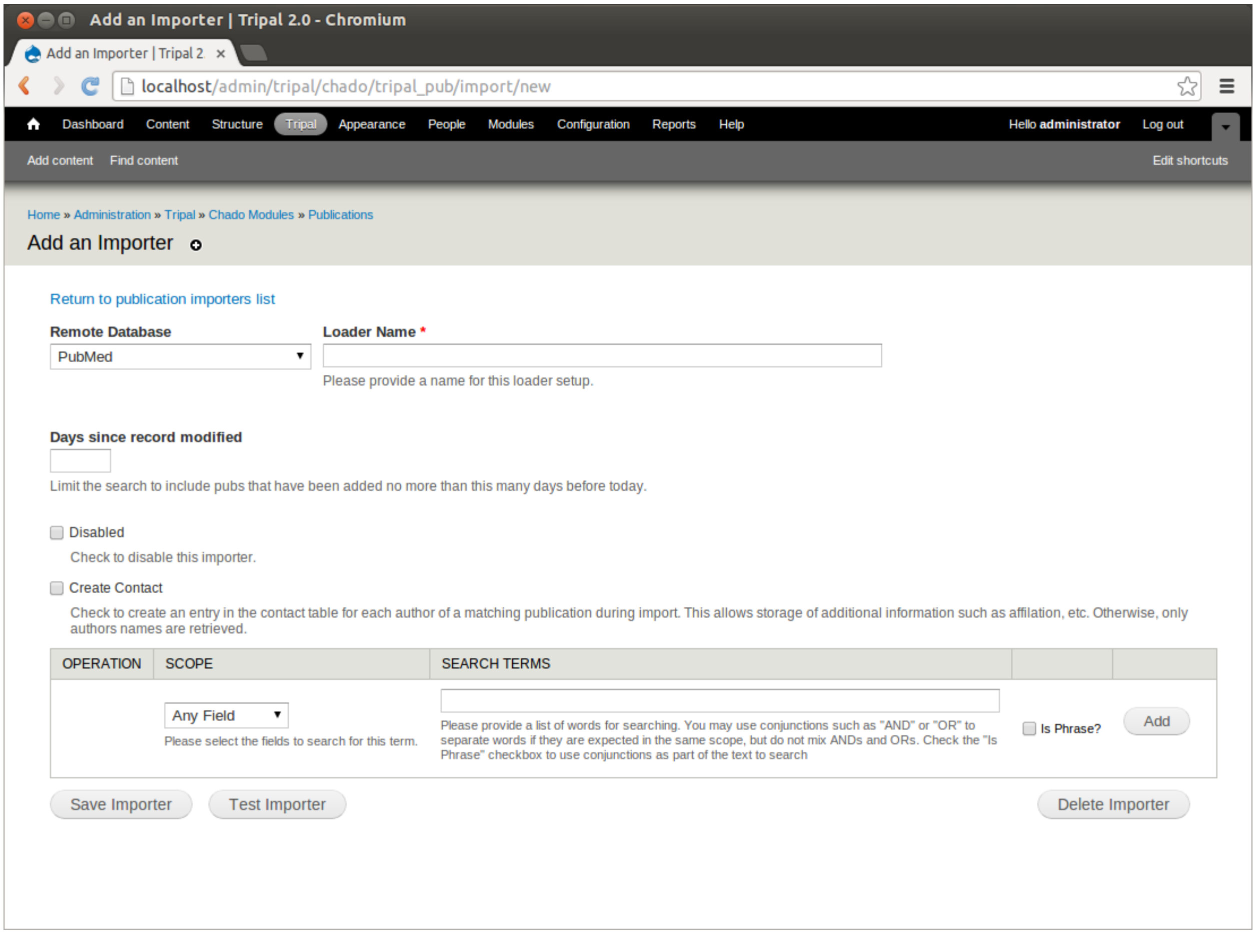
To complete the page click the **Save** button at the bottom

1.6.6.2 Import of Publications

Tripal supports importing of publications from remote databases such as NCBI PubMed.

Creation of an importer is an administrative function. A publication importer is created by the site administrator and consists of a set of search criteria for finding multiple publications at one time. When the importer is run, it will query the remote database, retrieve the publications that match the criteria and add them to the database. Because we loaded genomic data for Citrus sinensis we will create an importer that will find all publications related to this species.

First, navigate to **Tripal** → **Data Loaders** → **Chado Bulk Publication Importers** and click the link New Importer. You will see the following page:



Remote Database
PubMed

Loader Name *
Please provide a name for this loader setup.

Days since record modified
Limit the search to include pubs that have been added no more than this many days before today.

□ Disabled
Check to disable this importer.

□ Create Contact
Check to create an entry in the contact table for each author of a matching publication during import. This allows storage of additional information such as affilation, etc. Otherwise, only authors names are retrieved.

Any Field
Please select the fields to search for this term.

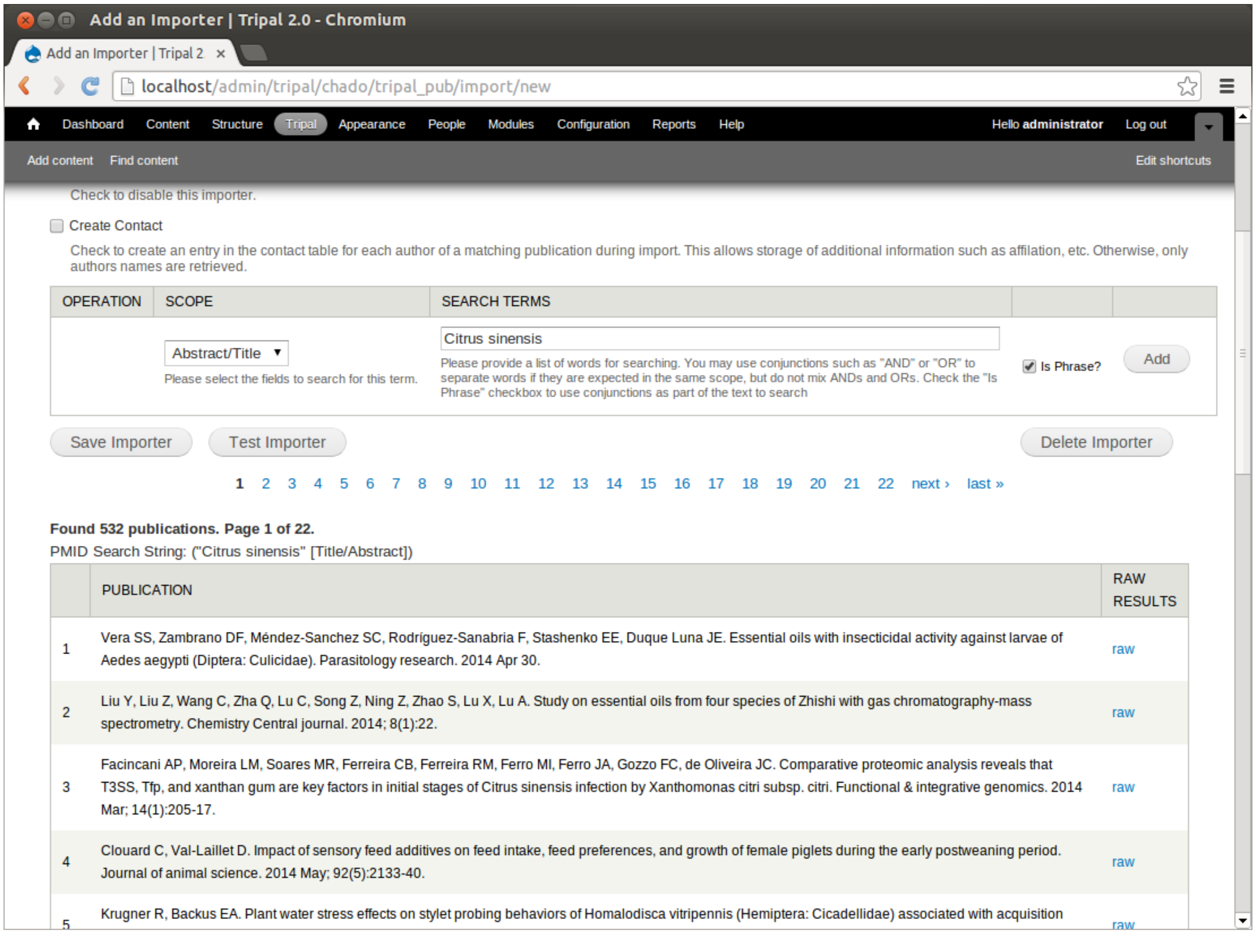
Please provide a list of words for searching. You may use conjunctions such as "AND" or "OR" to separate words if they are expected in the same scope, but do not mix ANDs and ORs. Check the "Is Phrase" checkbox to use conjunctions as part of the text to search

□ Is Phrase?

Enter the following values in the fields:

Field Name	Value
Remote Database	PubMed
Loader Name	Pubs for Citrus sinensis
Criteria #1	• Scope: Abstract/Title • Search Terms: Citrus sinensis • is Phrase?: checked

Now, click the 'Test Importer' button. This will connect to PubMed and search for all publications that match our provided criteria. it may take a few minutes to complete. On the date this portion of the tutorial was written, over 800 publications were found:


Add an Importer | Tripal 2.0 - Chromium
Add an Importer | Tripal 2
localhost/admin/tripal/chado/tripal_pub/import/new
Dashboard
Content
Structure
Tripal
Appearance
People
Modules
Configuration
Reports
Help
Hello administrator
Log out
Add content
Find content
Edit shortcuts
Add
Save Importer
Test Importer
Delete Importer
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 next › last »

Check to disable this importer.

□ Create Contact

Check to create an entry in the contact table for each author of a matching publication during import. This allows storage of additional information such as affilation, etc. Otherwise, only authors names are retrieved.

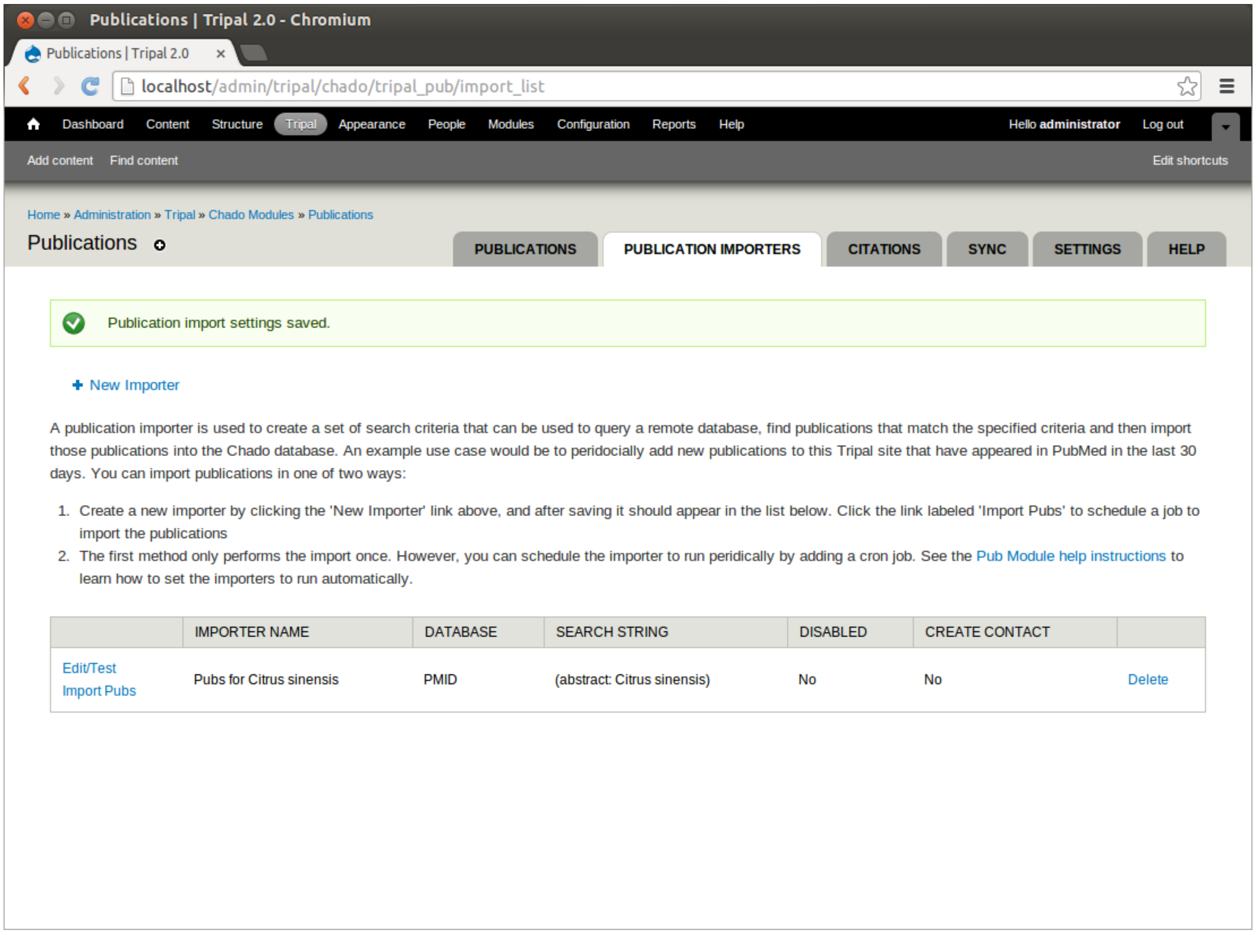
OPERATION	SCOPE	SEARCH TERMS		
	Abstract/Title ▾ Please select the fields to search for this term.	Citrus sinensis Please provide a list of words for searching. You may use conjunctions such as "AND" or "OR" to separate words if they are expected in the same scope, but do not mix ANDs and ORs. Check the "Is Phrase" checkbox to use conjunctions as part of the text to search	☑ Is Phrase?	Add

Found 532 publications. Page 1 of 22.

PMID Search String: ("Citrus sinensis" [Title/Abstract])

	PUBLICATION	RAW RESULTS
1	Vera SS, Zambrano DF, Méndez-Sanchez SC, Rodríguez-Sanabria F, Stashenko EE, Duque Luna JE. Essential oils with insecticidal activity against larvae of Aedes aegypti (Diptera: Culicidae). Parasitology research. 2014 Apr 30.	raw
2	Liu Y, Liu Z, Wang C, Zha Q, Lu C, Song Z, Ning Z, Zhao S, Lu X, Lu A. Study on essential oils from four species of Zhishi with gas chromatography-mass spectrometry. Chemistry Central journal. 2014; 8(1):22.	raw
3	Facincani AP, Moreira LM, Soares MR, Ferreira CB, Ferreira RM, Ferro MI, Ferro JA, Gozzo FC, de Oliveira JC. Comparative proteomic analysis reveals that T3SS, Tfp, and xanthan gum are key factors in initial stages of Citrus sinensis infection by Xanthomonas citri subsp. citri. Functional & integrative genomics. 2014 Mar; 14(1):205-17.	raw
4	Clouard C, Val-Laillet D. Impact of sensory feed additives on feed intake, feed preferences, and growth of female piglets during the early postweaning period. Journal of animal science. 2014 May; 92(5):2133-40.	raw
5	Krugner R, Backus EA. Plant water stress effects on stylet probing behaviors of Homalodisca vitripennis (Hemiptera: Cicadellidae) associated with acquisition	raw

Now, save this importer. You should see that we have one importer in the list:



Publication import settings saved.

+ New Importer

A publication importer is used to create a set of search criteria that can be used to query a remote database, find publications that match the specified criteria and then import those publications into the Chado database. An example use case would be to peridocially add new publications to this Tripal site that have appeared in PubMed in the last 30 days. You can import publications in one of two ways:

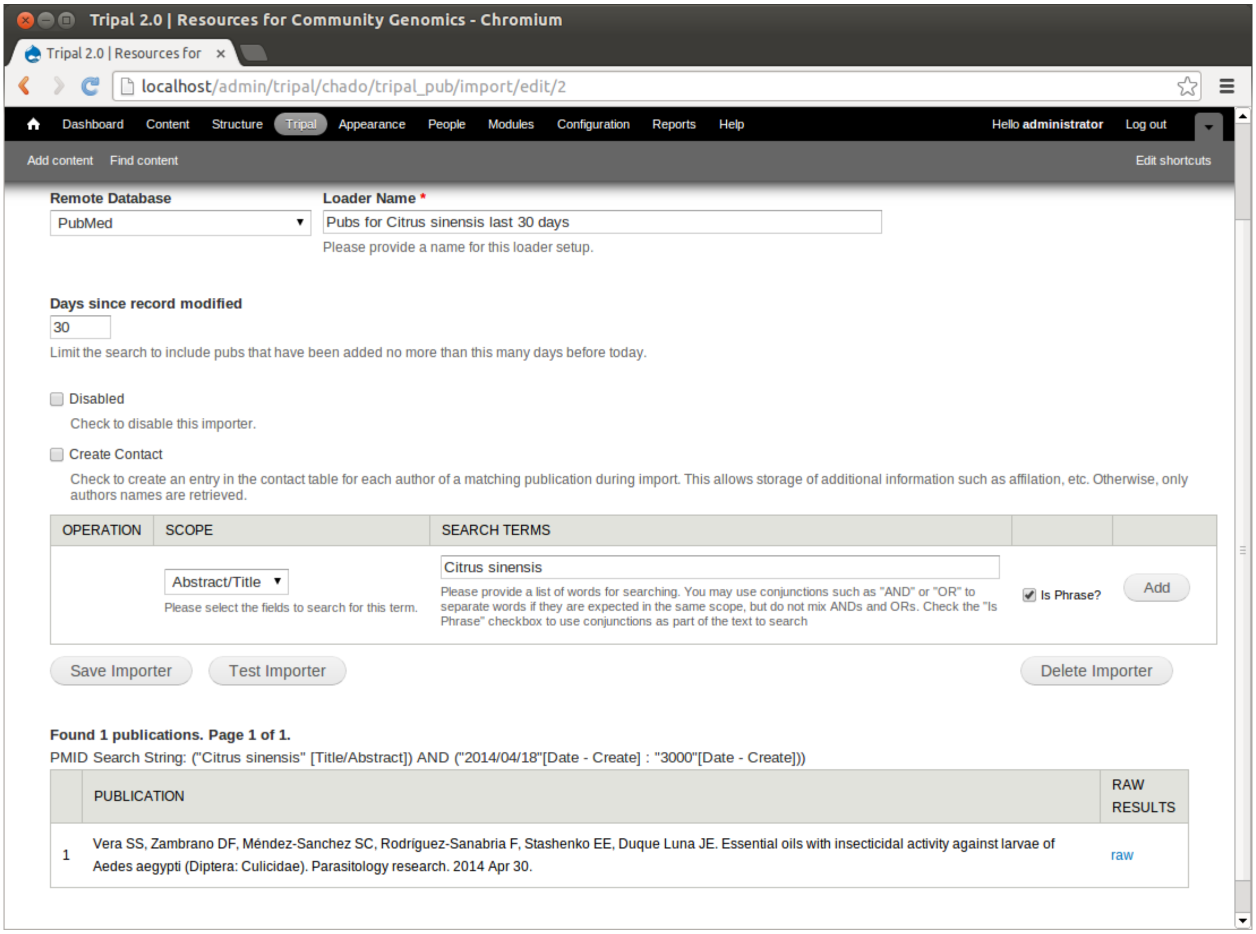
1. Create a new importer by clicking the 'New Importer' link above, and after saving it should appear in the list below. Click the link labeled 'Import Pubs' to schedule a job to import the publications
2. The first method only performs the import once. However, you can schedule the importer to run peridically by adding a cron job. See the Pub Module help instructions to learn how to set the importers to run automatically.

	IMPORTER NAME	DATABASE	SEARCH STRING	DISABLED	CREATE CONTACT	
Edit/Test Import Pubs	Pubs for Citrus sinensis	PMID	(abstract: Citrus sinensis)	No	No	Delete

We can use this importer to load all publications related to <i>Citrus sinensis</i> from PubMed into our database (how to load these will be shown later). However, what if new publications are added? We would like this importer to be run monthly so that we can automatically add new publications as they become available. But we do not need to try to reload the same publications every time the loader runs each month. We will create a new importer that only finds publications within the last 30 days. To do this, click the link New Importer. Now, add the following criteria:

Field Name	Value
Remote Database	PubMed
Loader Name	Pubs for Citrus sinensis last 30 days
Days since record modified	30
Criteria #1	• Scope: Abstract/Title • Search Terms: Citrus sinensis • is Phrase?: checked

Now, when we test the importer we find only 1 publications that has been added (created) to PubMed in the last 30 days:



Remote Database
PubMed

Loader Name *
Pubs for Citrus sinensis last 30 days
Please provide a name for this loader setup.

Days since record modified
30
Limit the search to include pubs that have been added no more than this many days before today.

□ Disabled
Check to disable this importer.

□ Create Contact
Check to create an entry in the contact table for each author of a matching publication during import. This allows storage of additional information such as affilation, etc. Otherwise, only authors names are retrieved.

OPERATION	SCOPE	SEARCH TERMS		
	Abstract/Title Please select the fields to search for this term.	Citrus sinensis Please provide a list of words for searching. You may use conjunctions such as "AND" or "OR" to separate words if they are expected in the same scope, but do not mix ANDs and ORs. Check the "Is Phrase" checkbox to use conjunctions as part of the text to search	☑ Is Phrase?	Add

Save Importer   Test Importer   Delete Importer

Found 1 publications. Page 1 of 1.
PMID Search String: ("Citrus sinensis" [Title/Abstract]) AND ("2014/04/18"[Date - Create] : "3000"[Date - Create]))

	PUBLICATION	RAW RESULTS
1	Vera SS, Zambrano DF, Méndez-Sanchez SC, Rodríguez-Sanabria F, Stashenko EE, Duque Luna JE. Essential oils with insecticidal activity against larvae of Aedes aegypti (Diptera: Culicidae). Parasitology research. 2014 Apr 30.	raw

Save this importer.

Next, there are two ways to import these publications. The first it to manually import them. There is a Drush command that is used for importing publications. Return to the terminal and run the following command:

```
cd $DRUPAL_HOME
drush trp-import-pubs --username=administrator
```

You should see output to the terminal that begins like this:

```
NOTE: Loading of publications is performed using a database transaction.
If the load fails or is terminated prematurely then the entire set of
insertions/updates is rolled back and will not be found in the database

Importing: Pubs for Citrus sinensis
```

The importer will import 100 publications at a time and pause between each set of 100 as it requests more.

Some things to know about the publication importer:

1. The importer keeps track of publications from the remote database using the publication accession (e.g. PubMed ID).
2. If a publication with an accession (e.g. PubMed ID) already exists in the local database, the record will be updated.
3. If a publication in the local database matches by title, journal and year with one that is to be imported, then the record will be updated. Y

4. Run the newly created Tripal Job to finish:

```
cd $DRUPAL_HOME
drush trp-run-jobs --user=administrator
```

The second way to import publications is to add an entry to the UNIX cron. We did this previously for the Tripal Jobs management system when we first installed Tripal. We will add another entry for importing publications. But first, now that we have imported all of the relevant pubs, we need to return to the importers list at **Tripal** → **Data Loaders** → **Chado Publication Importers** and disable the first importer we created. We do not want to run that importer again, as we've already imported all historical publications on record at PubMed. Click the edit button next to the importer named Pubs for Citrus sinensis, click the disable checkbox and then save the template. The template should now be disabled.

Now we have the importer titled **Pubs for Citrus sinensis last 30 days** enabled. This is the importer we want to run on a monthly basis. The cron entry will do this for us. On the terminal open the crontab with the following command:

```
sudo crontab -e
```

Now add the following line to the bottom of the crontab:

```
30 8 1,15 * *  su - www-data -c '/usr/local/drush/drush -r [DRUPAL_HOME] -l http://
↪[site url] trp-import-pubs --report=[your email] > /dev/null'
```

Where

- [site url] is the full URL of your site
- [your email] is the email address of the user that should receive an email containing a list of publications that were imported. You can separate multiple email addresses with a comma.
- [DRUPAL_HOME] is the directory where Drupal is installed

The cron entry above will launch the importer at 8:30am on the first and fifteenth days of the month. We will run this importer twice a month in the event it fails to run (e.g. server is down) at least one time during the month.

1.6.6.3 Import from the USDA National Agricultural Library

The instructions for the Tripal publication importer described previously use the the NCBI PubMed database. However, you can also import publications from the USDA National Agriculture Library (AGRICOLA). However, to use this repository a few software dependences are required. These include:

- The YAZ library
- PHP support for YAZ

The following instructions are to install the necessary dependencies on an Ubuntu 18.04 LTS.

First install yaz, the yaz development library and the php development library:

```
sudo apt-get install yaz libyaz5-dev php-dev
```

Next update the PECL tool and install the PHP yaz library:

```
sudo pecl channel-update pecl.php.net
sudo pecl install yaz
```

Next, edit the *php.ini* files. On Ubuntu 18.04 there are two PHP files:

- */etc/php/7.2/cli/php.ini*

- */etc/php/7.2/apache2/php.ini*

Add the following line to each file:

```
extension=yaz.so
```

Finally, restart the web server so that it picks up the changes to the *php.ini* file.

```
sudo service apache2 restart
```

You can now import publications from Agricola using the same interface as with PubMed.

1.6.7 Functional Annotations

1.6.7.1 Module Setup

Note: Remember you must set the `$DRUPAL_HOME` environment variable if you want to cut-and-paste the commands below. See *DRUPAL_HOME Variable*

For this example we will be load functional data for our gene. To do this we will use the Blast, KEGG, and InterPro extension modules. However, these extension modules are not part of the "core" Tripal package but are available as separate extensions. Anyone may create extensions for Tripal. These extensions are useful for genomic data and therefore are included in this tutorial.

To download these modules:

```
cd $DRUPAL_HOME
drush pm-download tripal_analysis_blast
drush pm-download tripal_analysis_interpro
```

Now, enable these extension modules:

```
drush pm-enable tripal_analysis_blast
drush pm-enable tripal_analysis_interpro
```

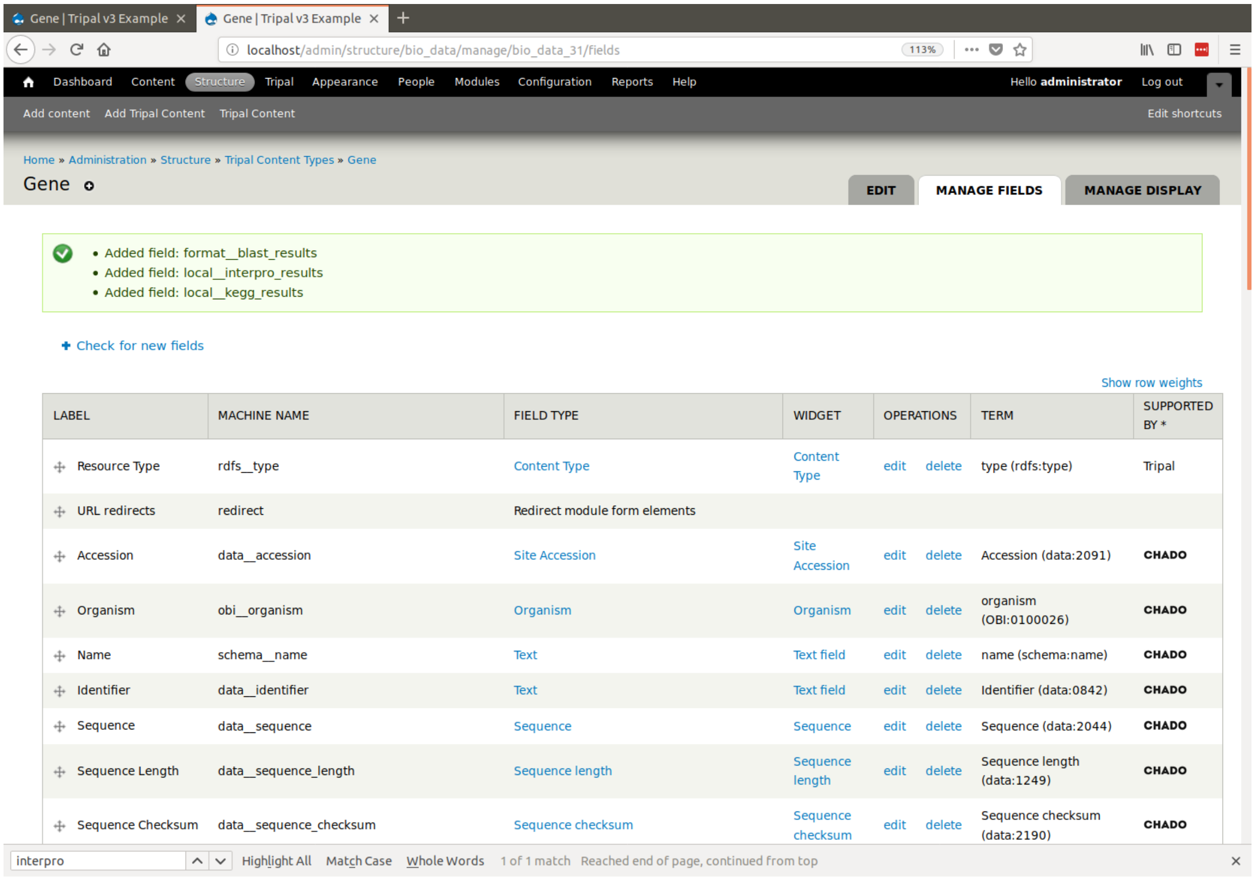
For this example, we will use the following files which are available for downloading:

- Citrus_sinensis-orange1.1g015632m.g.iprscan.xml
- Blastx_citrus_sinensis-orange1.1g015632m.g.fasta.0_vs_uniprot_sprot.fasta.out
- Blastx_citrus_sinensis-orange1.1g015632m.g.fasta.0_vs_nr.out

Download these files to the `` `$DRUPAL_HOME/sites/default/files` `` directory. To do so quickly run these commands:

```
cd $DRUPAL_HOME/sites/default/files
wget http://www.gmod.org/mediawiki/images/0/0c/Citrus_sinensis-orange1.
↪1g015632m.g.iprscan.xml
wget http://www.gmod.org/mediawiki/images/e/e8/Blastx_citrus_sinensis-
↪orange1.1g015632m.g.fasta.0_vs_uniprot_sprot.fasta.out
wget http://www.gmod.org/mediawiki/images/2/24/Blastx_citrus_sinensis-
↪orange1.1g015632m.g.fasta.0_vs_nr.out
```

Each of these modules provides new fields for both the **gene** and **mRNA** content types. To add these new field to those content types, navigate to **Structure > Tripal Content Types** and click the **manage fields** link in the row for the **mRNA** content type. Click the link titled **Check for new fields**. After a few moments the page will refresh and you will be notified that new fields have been added.



- Added field: format__blast_results
- Added field: local__interpro_results
- Added field: local__kegg_results

✚ Check for new fields

Show row weights

LABEL	MACHINE NAME	FIELD TYPE	WIDGET	OPERATIONS		TERM	SUPPORTED BY *
Resource Type	rdfs__type	Content Type	Content Type	edit	delete	type (rdfs:type)	Tripal
URL redirects	redirect	Redirect module form elements					
Accession	data__accession	Site Accession	Site Accession	edit	delete	Accession (data:2091)	CHADO
Organism	obi__organism	Organism	Organism	edit	delete	organism (OBI:0100026)	CHADO
Name	schema__name	Text	Text field	edit	delete	name (schema:name)	CHADO
Identifier	data__identifier	Text	Text field	edit	delete	Identifier (data:0842)	CHADO
Sequence	data__sequence	Sequence	Sequence	edit	delete	Sequence (data:2044)	CHADO
Sequence Length	data__sequence_length	Sequence length	Sequence length	edit	delete	Sequence length (data:1249)	CHADO
Sequence Checksum	data__sequence_checksum	Sequence checksum	Sequence checksum	edit	delete	Sequence checksum (data:2190)	CHADO

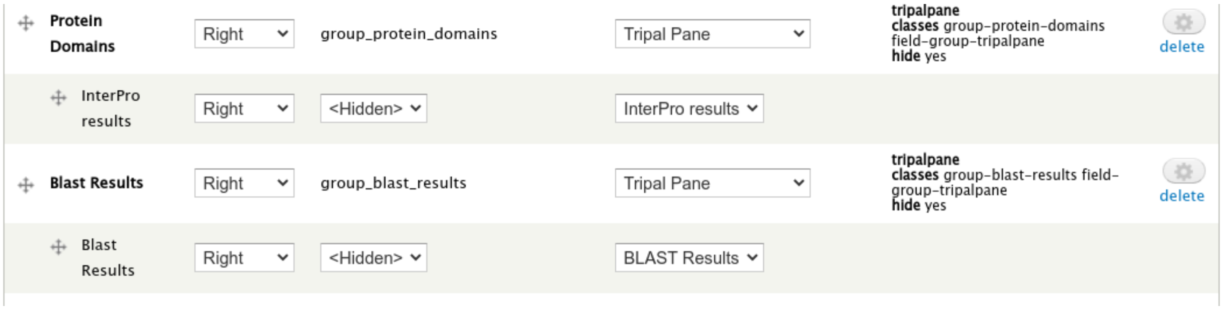
Next, we need to position the new field. Using the skills you learned in the *Configuring Page Layout* Create two new **Tripal Panes** named:

- Blast Results
- Protein Domains

Be sure to:

- Place the new fields into each pane respectively
- Move the Panes out of the **disabled** section.
- Set the label for each field to **Hidden**.

The following shows an example of this layout:



Protein Domains	Right	group_protein_domains	Tripal Pane	**tripalpane** **classes** group-protein-domains field-group-tripalpane **hide** yes	delete
InterPro results	Right	<Hidden>	InterPro results		
Blast Results	Right	group_blast_results	Tripal Pane	**tripalpane** **classes** group-blast-results field-group-tripalpane **hide** yes	delete
Blast Results	Right	<Hidden>	BLAST Results		

The fields are now ready for display once data is added!

Note: If you want both the **Gene** and **mRNA** content type to have BLAST, InterPro and KEGG result fields you must repeat the steps above for both.

Note: Anytime you install a Tripal v3 extension module you should check for new fields, and then place those fields in the layout. Extension modules often will not do this for you because they do not assume you want these new fields.

1.6.7.2 Adding BLAST Results

Note: Remember you must set the `$DRUPAL_HOME` environment variable if you want to cut-and-paste the commands below. See *DRUPAL_HOME Variable*

Adding BLAST Databases

Before we load our BLAST results we want to add some external databases. For this tutorial we have protein BLAST results against NCBI nr and ExPASy SwissProt. We would like the BLAST hits to be clickable such that they link back to their respective databases. To do this, we must add some additional databases. Navigate to **Tripal** → **Data Loaders** → **Chado Databases** and click the link titled **Add a Database**. The resulting page provides fields for adding a new database. Add two new databases, one for NCBI nr and the other for ExPASy SwissProt.

Use these values for adding the NCBI nr database:

Field Name	Value
Name	NCBI nr
Description	NCBI's non-redundant protein database
URL	http://www.ncbi.nlm.nih.gov/
URL Prefix	http://www.ncbi.nlm.nih.gov/protein/

Use these values for adding the SwssProt database:

Field Name	Value
Name	ExPASy Swiss-Prot
Description	A curated protein sequence database which strives to provide a high level of annotation, a minimal level of redundancy and high level of integration with other databases
URL	http://expasy.org/sprot/
URL prefix	http://www.uniprot.org/uniprot/

Configure Parsing of BLAST Results

First, we need to ensure that the BLAST module can properly parse the BLAST hits. To do this, navigate to **Tripal** → **Extensions** → **Tripal Blast Analyses**. On this page are configuration settings for the Tripal BLAST Analysis extension module.

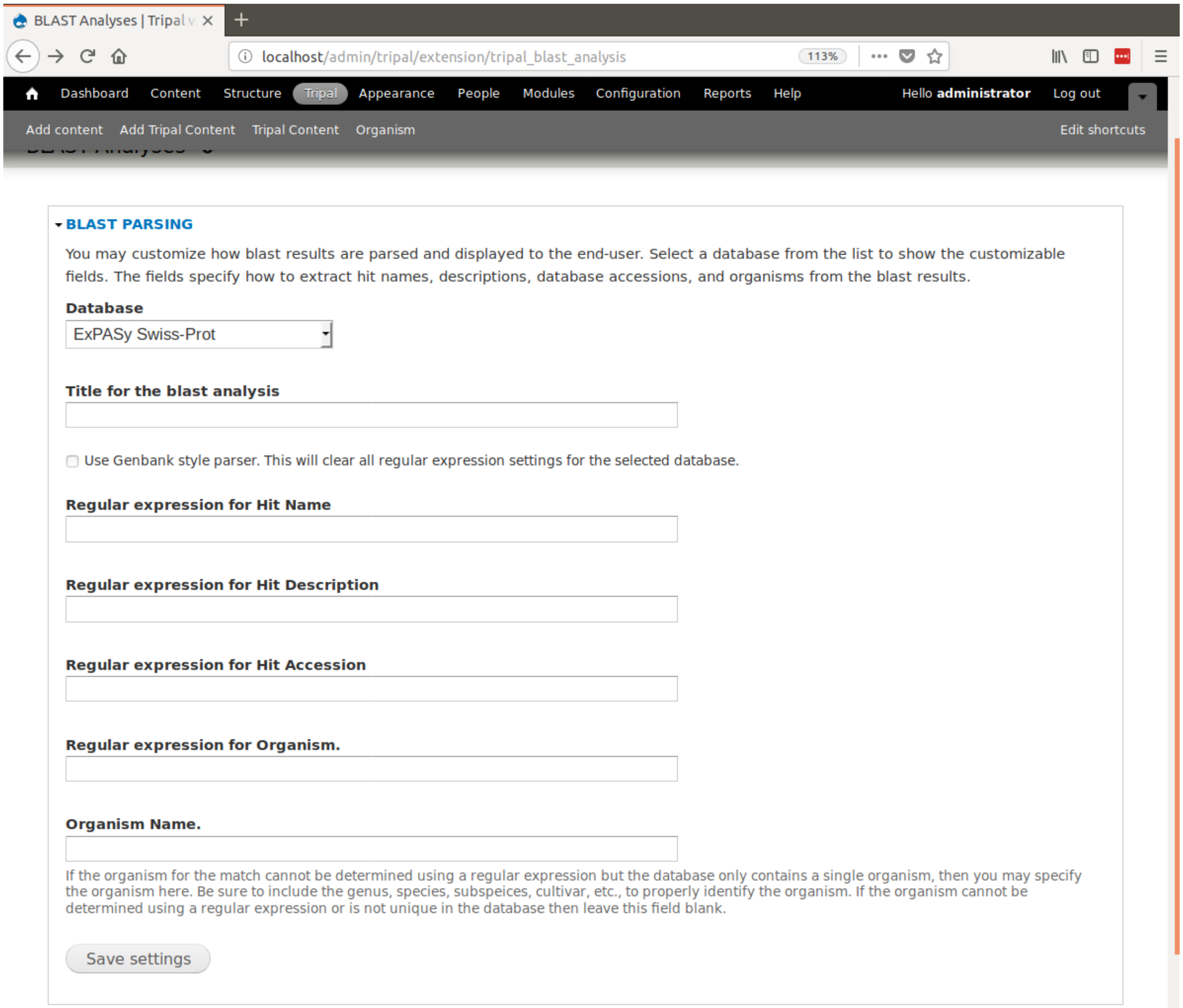
Within the section titled BLAST Parsing, you can specify a different, more meaningful name for the sequence library file (a.k.a. database) used for BLASTing. This name will be displayed with BLAST results. You can also provide regular expressions for parsing BLAST hits. For example, the following is a line for a match from SwissProt:

```
sp|P43288|KSG1_ARATH Shaggy-related protein kinase alpha OS=Arabidopsis thaliana GN=ASK1 PE=2 SV=3
```

Here the hit name is KSG1_ARATH, the accession is P43288, the hit description is Shaggy-related protein kinase alpha OS=Arabidopsis thaliana and the organism is Arabidopsis thaliana. We need regular expressions to tell Tripal how to extract these unique parts from the match text. Because Tripal is a PHP application, the syntax for regular expressions follows the PHP method. Documentation for regular expressions used in PHP can be found here. The following regular expressions can be used to extract the hit name, the accession, hit description and organism for the example SwissProt line above:

Element	Regular Expression
Hit Name	^sp|.*?|(.*?)\s.*?$
Hit Description	^sp|.*?|.*?\s(.*)$
Hit Accession	^sp|(.*?)|.*?\s.*?$
Hit Organism	^.*?OS=(.*?)\s\w\w=.*$

In this tutorial, we will be adding BLAST results for the two databases we just created: ExPASy SwissProt and NCBI nr. First, select ExPASy SwissProt from the drop-down menu. A form will appear:



▾ BLAST PARSING

You may customize how blast results are parsed and displayed to the end-user. Select a database from the list to show the customizable fields. The fields specify how to extract hit names, descriptions, database accessions, and organisms from the blast results.

Database

ExPASy Swiss-Prot

Title for the blast analysis

□ Use Genbank style parser. This will clear all regular expression settings for the selected database.

Regular expression for Hit Name

Regular expression for Hit Description

Regular expression for Hit Accession

Regular expression for Organism.

Organism Name.

If the organism for the match cannot be determined using a regular expression but the database only contains a single organism, then you may specify the organism here. Be sure to include the genus, species, subspeices, cultivar, etc., to properly identify the organism. If the organism cannot be determined using a regular expression or is not unique in the database then leave this field blank.

In the form fields, add the following values:

<table>
<tr><th>Field</th><th>Value</th></tr>
<tr><td>Title for the BLAST analysis</td><td>(leave blank)</td></tr>
<tr><td>Regular expression for Hit Name</td><td><code>^sp\|.*?\|(.*?)\s.*?$</code></td></tr>
<tr><td>Regular expression for Hit Description</td><td><code>^sp\|.*?\|.*?\s(.*)$</code></td></tr>
<tr><td>Regular expression for Hit Accession:</td><td><code>^sp\|(.*?)\|.*?\s.*?$</code></td></tr>
<tr><td>Regular expression for Organism</td><td><code>^.*?OS=(.*?)\s\w\w=.*$</code></td></tr>
<tr><td>Organism Name</td><td>(leave blank)</td></tr>
</table>

Click **Save Settings**.

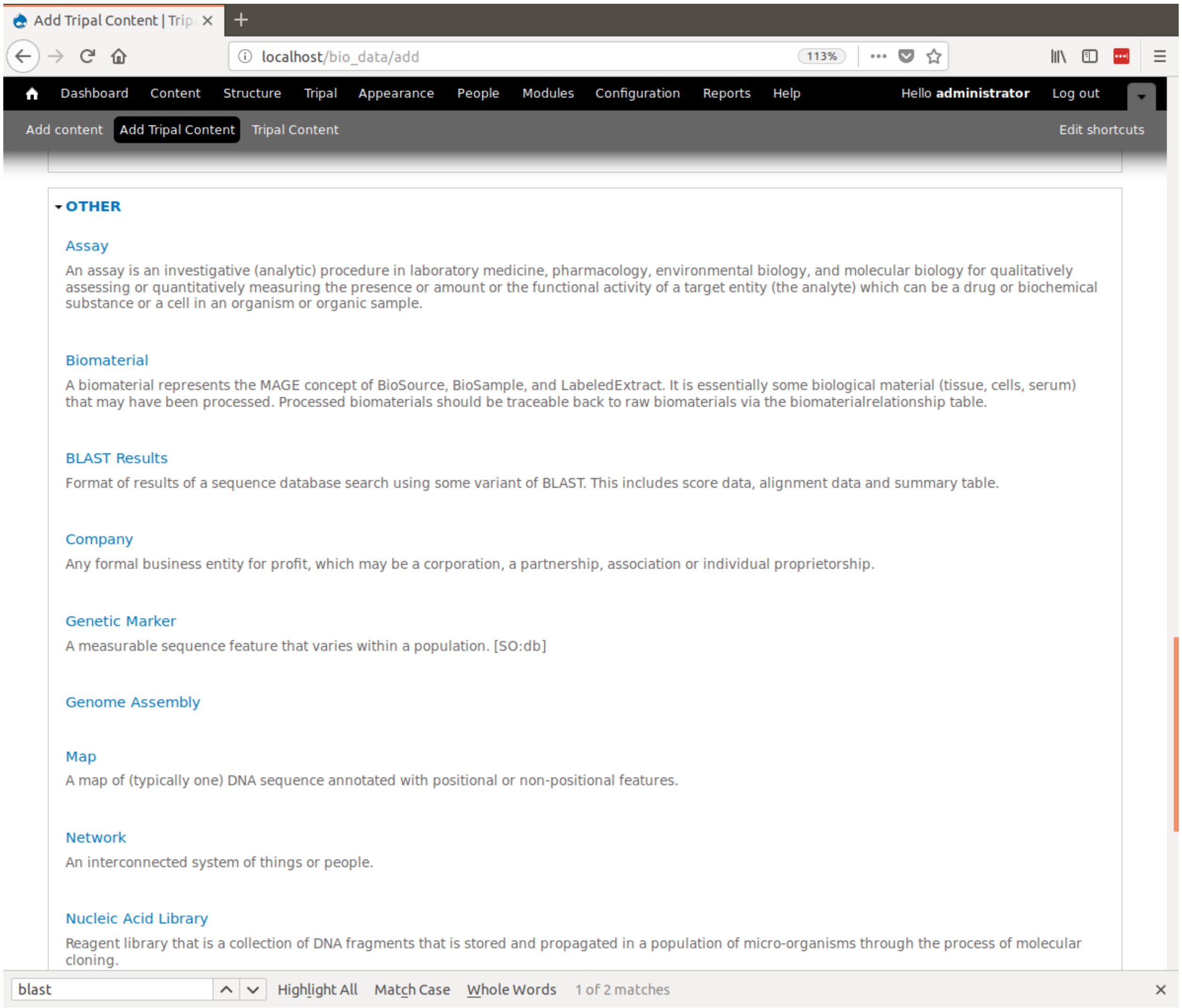
The match accession will be used for building web links to the external database. The accession will be appended to the URL Prefix set earlier when the database record was first created.

Now select the **NCBI nr** database from the drop-down. NCBI databases use a format that is compatible with BLAST. Therefore, the hit name, accession and description are handled differently in the BLAST XML results. To correctly parse results from an NCBI database click the **Use Genbank style parser** checkbox. This should disable all other fields and is all we need for this database. Clikc the Save Settings button.

Create the Analysis Page

Note: It is always recommended to create an analysis page anytime you import data. The purpose of the analysis page is to describe how the data being added was derived or collected.

Now we can create our analysis page. Navigate to **Content** → **Tripal Content** and click the **Add Tripal Content** link. This page contains a variety of content types that the site supports. Scroll to the **Other** section and find the content type named **Blast Results**:



▾ OTHER

Assay

An assay is an investigative (analytic) procedure in laboratory medicine, pharmacology, environmental biology, and molecular biology for qualitatively assessing or quantitatively measuring the presence or amount or the functional activity of a target entity (the analyte) which can be a drug or biochemical substance or a cell in an organism or organic sample.

Biomaterial

A biomaterial represents the MAGE concept of BioSource, BioSample, and LabeledExtract. It is essentially some biological material (tissue, cells, serum) that may have been processed. Processed biomaterials should be traceable back to raw biomaterials via the biomaterialrelationship table.

BLAST Results

Format of results of a sequence database search using some variant of BLAST. This includes score data, alignment data and summary table.

Company

Any formal business entity for profit, which may be a corporation, a partnership, association or individual proprietorship.

Genetic Marker

A measurable sequence feature that varies within a population. [SO:db]

Genome Assembly

Map

A map of (typically one) DNA sequence annotated with positional or non-positional features.

Network

An interconnected system of things or people.

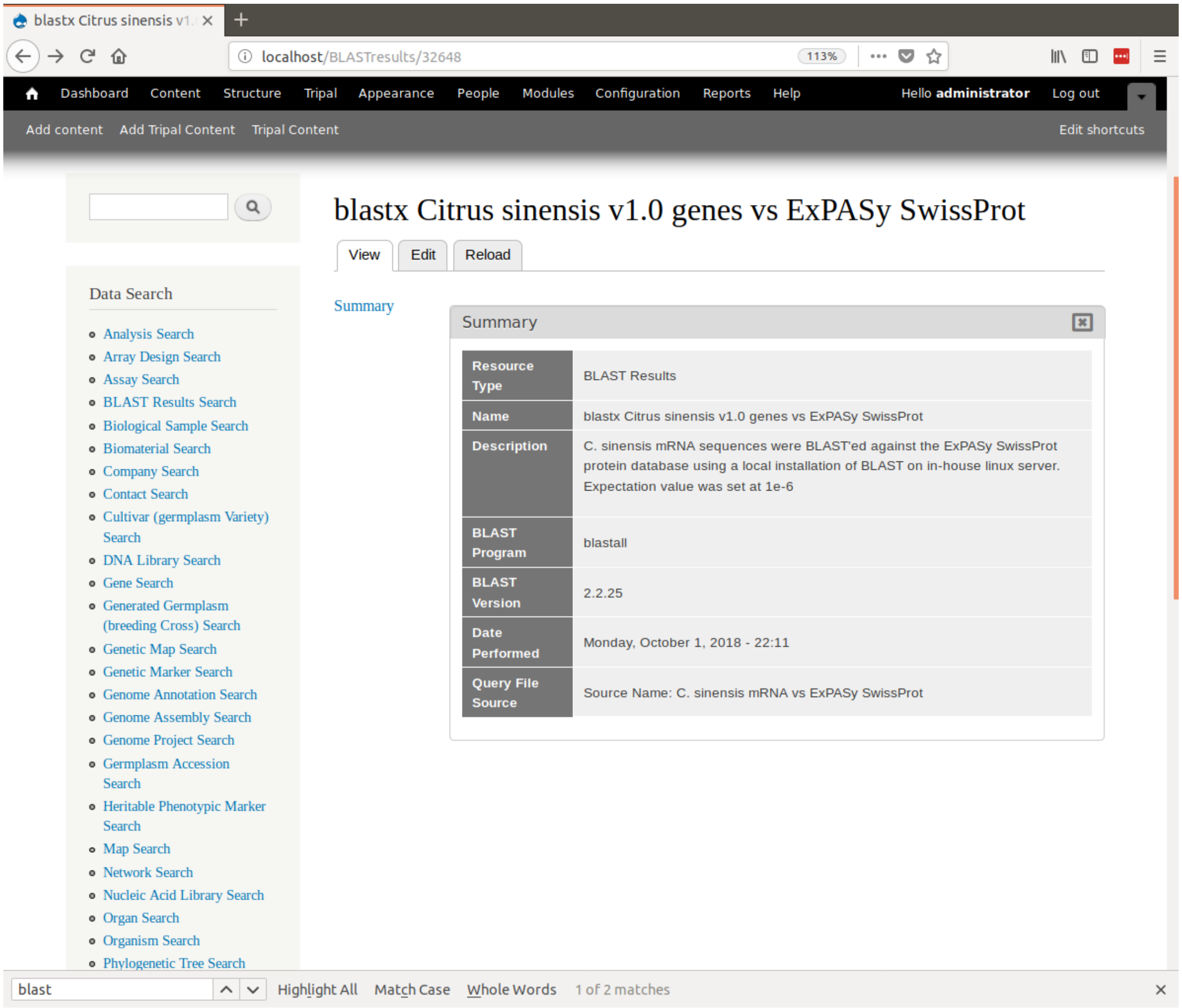
Nucleic Acid Library

Reagent library that is a collection of DNA fragments that is stored and propagated in a population of micro-organisms through the process of molecular cloning.

Here we can save details about the analysis used to create the BLAST results. Enter the following in the fields that appear on the page:

Field	Value
Name	blastx Citrus sinensis v1.0 genes vs ExPASy SwissProt
Description	Citrus sinensis mRNA sequences were BLAST'ed against the ExPASy SwissProt protein database using a local installation of BLAST on in-house linux server. Expectation value was set at 1e-6
BLAST Program	blastx
BLAST Version	2.2.25
Data Source Name	Citrus sinensis mRNA vs ExPASy SwissProt
Date Performed	(today's date)

Click the **Save** button. You can now see our new BLAST analysis page.



blastx Citrus sinensis v1.0 genes vs ExPASy SwissProt

Summary

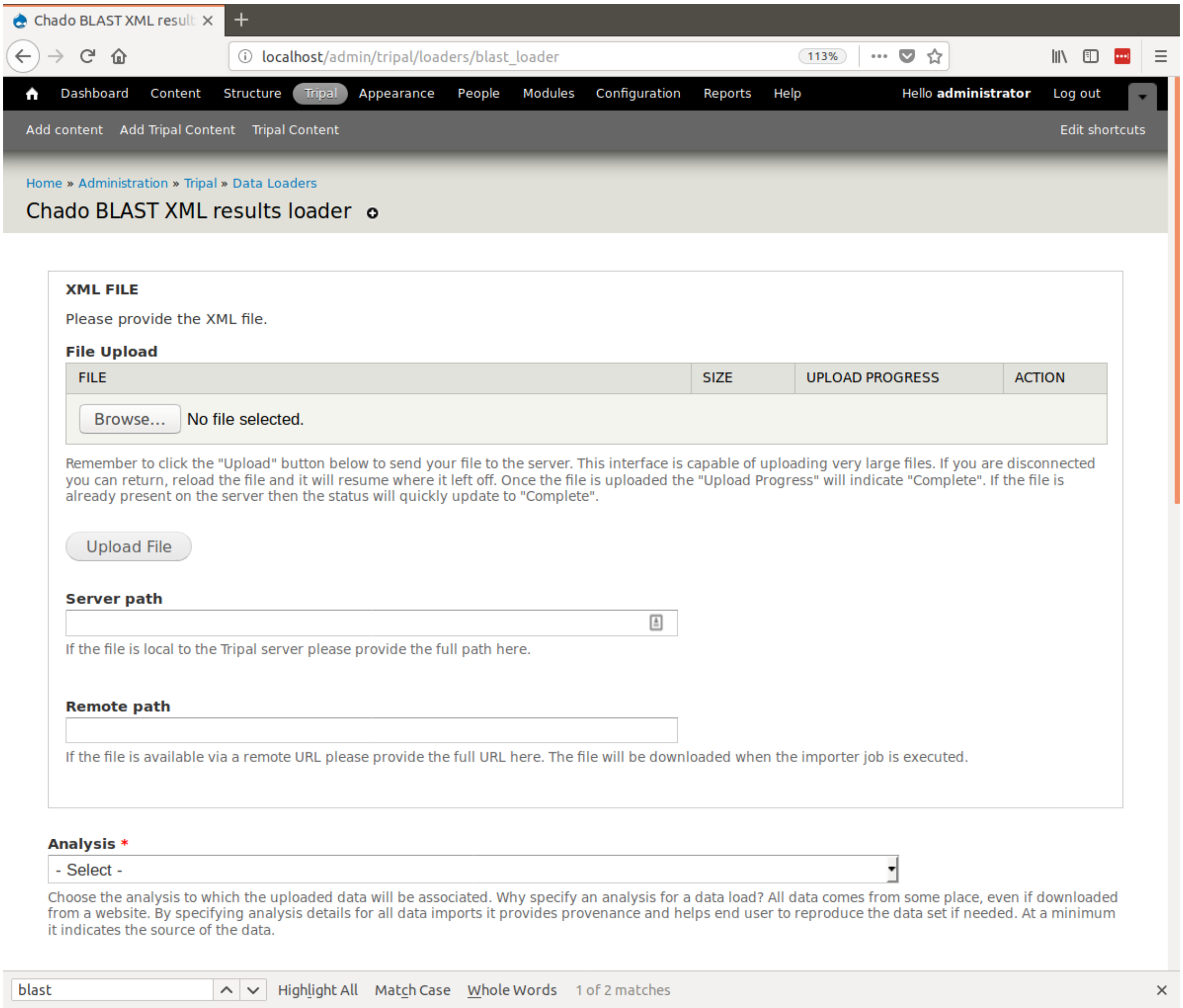
Resource Type	BLAST Results
Name	blastx Citrus sinensis v1.0 genes vs ExPASy SwissProt
Description	C. sinensis mRNA sequences were BLAST'ed against the ExPASy SwissProt protein database using a local installation of BLAST on in-house linux server. Expectation value was set at 1e-6
BLAST Program	blastall
BLAST Version	2.2.25
Date Performed	Monday, October 1, 2018 - 22:11
Query File Source	Source Name: C. sinensis mRNA vs ExPASy SwissProt

Create a second Analysis page for the results of the NCBI nr BLAST analysis. Use the following values:

Field	Value
Name	blastx Citrus sinensis v1.0 genes vs NCBI nr
Description	Citrus sinensis mRNA sequences were BLAST'ed against the NCBI non-redundant protein database using a local installation of BLAST on in-house linux server. Expectation value was set at 1e-6
BLAST Program	blastx
BLAST Version	2.2.25
Data Source Name	Citrus sinensis mRNA vs NCBI nr
Date Performed	(today's date)

Import the BLAST XML results

First, we will load BLAST results for our citrus gene vs ExPASy SwissProt. Now that we have our database records setup and configured and we have our analysis record created, we are ready to import the blast results. To do this, navigate to **Tripal > Data Loaders > Chado BLAST XML results loader**. The following page will be presented:



Chado BLAST XML results loader

XML FILE

Please provide the XML file.

File Upload

FILE	SIZE	UPLOAD PROGRESS	ACTION
Browse… No file selected.			

Remember to click the "Upload" button below to send your file to the server. This interface is capable of uploading very large files. If you are disconnected you can return, reload the file and it will resume where it left off. Once the file is uploaded the "Upload Progress" will indicate "Complete". If the file is already present on the server then the status will quickly update to "Complete".

Upload File

Server path

If the file is local to the Tripal server please provide the full path here.

Remote path

If the file is available via a remote URL please provide the full URL here. The file will be downloaded when the importer job is executed.

Analysis *

- Select -

Choose the analysis to which the uploaded data will be associated. Why specify an analysis for a data load? All data comes from some place, even if downloaded from a website. By specifying analysis details for all data imports it provides provenance and helps end user to reproduce the data set if needed. At a minimum it indicates the source of the data.

The top section of this page provides multiple methods for providing results file: via an upload interface, specifying a remote URL or a file path that is local to the server. Most likely, you will always upload or provide a remote URL. However, we download the file earlier, and stored them here: `$DRUPAL_HOME/sites/default/files`. So, in this case we can use the path on the local server. Provide the following value for this form:

Field	Value
Server path	sites/default/files/Blastx_citrus_sinensis-orange1.1g015632m.g.fasta.0_vs_uniprot_sprot.fasta.out
Analysis	blastx Citrus sinensis v1.0 genes vs ExPASy SwissProt (blastall 2.2.25, Citrus sinensis mRNA vs ExPASy SwissProt)
Database	ExPASy SwissProt
BLAST XML File Extension	out
Query Type	mRNA

Note: For the **Server path** we need not give the full path. Because we downloaded the files into the Drupal directory we can leave off any preceding path and Tripal will resolve the path. Otherwise we could provide the full path.

Note: Specifying **ExPASy SwissProt** as the database will allow the importer to use the database configuration settings we entered earlier.

Clicking the **Import BLAST file** will add a job which we can manually execute with the following command:

```
drush trp-run-jobs --username=administrator --root=$DRUPAL_HOME
```

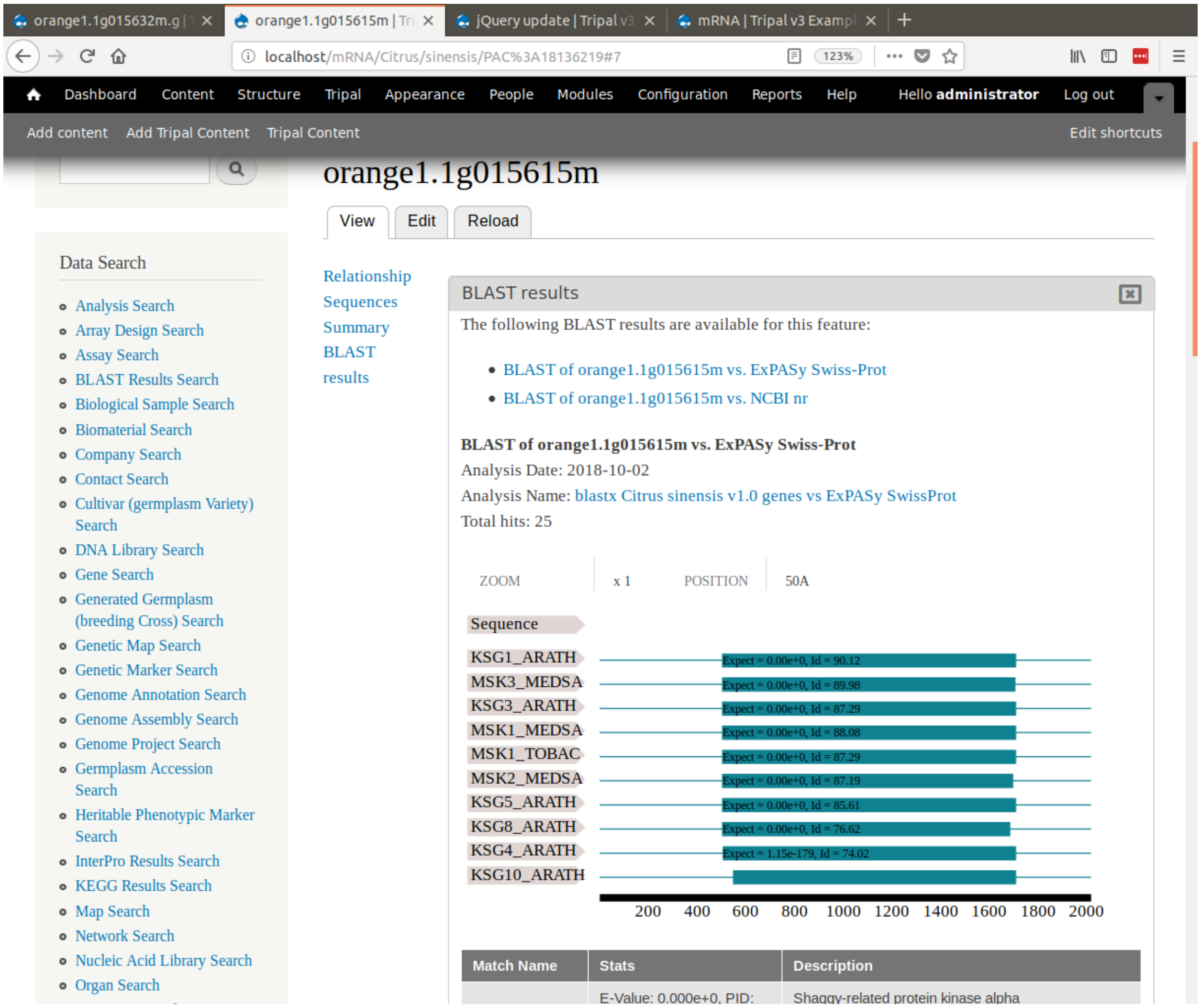
The results should now be loaded. Now we want to add the results for NCBI nr. Repeat the steps above to add a new analysis with the following details:

Field	Value
Server path	sites/default/files/Blastx_citrus_sinensis-orange1.1g015632m.g.fasta.0_vs_nr.out
Analysis	blastx Citrus sinensis v1.0 genes vs ExPASy SwissProt (blastall 2.2.25, Citrus sinensis mRNA vs NCBI nr)
Database	ExPASy SwissProt
BLAST XML File Extension	out
Query Type	mRNA

Click the Save button and manually run the job:

```
drush trp-run-jobs --username=administrator --root=$DRUPAL_HOME
```

To view results we must find the mRNA that has BLAST hits. For this example, click on the **mRNA Search** link in the **Data Search** block. Search for the mRNA named *orange1.1g015615m*. Viewing the page, we should now see BLAST results by clicking the 'BLAST results' link in the left table of contents.



Data Search

- Analysis Search
- Array Design Search
- Assay Search
- BLAST Results Search
- Biological Sample Search
- Biomaterial Search
- Company Search
- Contact Search
- Cultivar (germplasm Variety) Search
- DNA Library Search
- Gene Search
- Generated Germplasm (breeding Cross) Search
- Genetic Map Search
- Genetic Marker Search
- Genome Annotation Search
- Genome Assembly Search
- Genome Project Search
- Germplasm Accession Search
- Heritable Phenotypic Marker Search
- InterPro Results Search
- KEGG Results Search
- Map Search
- Network Search
- Nucleic Acid Library Search
- Organ Search

orange1.1g015615m

Relationship
Sequences
Summary
BLAST results

BLAST results

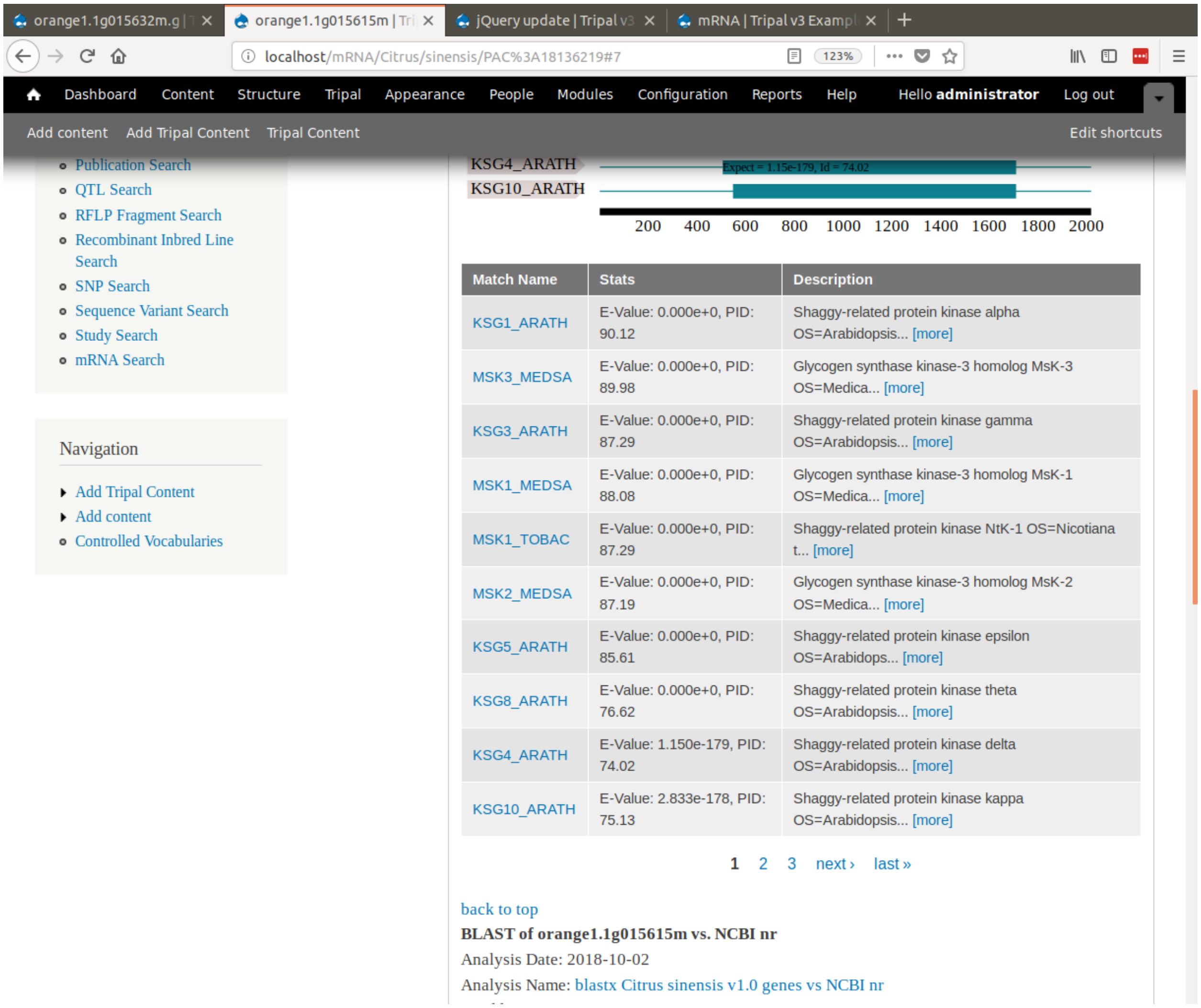
The following BLAST results are available for this feature:

- BLAST of orange1.1g015615m vs. ExPASy Swiss-Prot
- BLAST of orange1.1g015615m vs. NCBI nr

BLAST of orange1.1g015615m vs. ExPASy Swiss-Prot
Analysis Date: 2018-10-02
Analysis Name: blastx Citrus sinensis v1.0 genes vs ExPASy SwissProt
Total hits: 25

Match Name	Stats	Description
	E-Value: 0.000e+0, PID:	Shaggy-related protein kinase alpha

Notice, that when viewing the results, the SwissProt matches are links. When clicked they redirect the user to the SwissProt website where users can find more information about that protein.



Match Name	Stats	Description
KSG1_ARATH	E-Value: 0.000e+0, PID: 90.12	Shaggy-related protein kinase alpha OS=Arabidopsis... [more]
MSK3_MEDSA	E-Value: 0.000e+0, PID: 89.98	Glycogen synthase kinase-3 homolog MsK-3 OS=Medica... [more]
KSG3_ARATH	E-Value: 0.000e+0, PID: 87.29	Shaggy-related protein kinase gamma OS=Arabidopsis... [more]
MSK1_MEDSA	E-Value: 0.000e+0, PID: 88.08	Glycogen synthase kinase-3 homolog MsK-1 OS=Medica... [more]
MSK1_TOBAC	E-Value: 0.000e+0, PID: 87.29	Shaggy-related protein kinase NtK-1 OS=Nicotiana t... [more]
MSK2_MEDSA	E-Value: 0.000e+0, PID: 87.19	Glycogen synthase kinase-3 homolog MsK-2 OS=Medica... [more]
KSG5_ARATH	E-Value: 0.000e+0, PID: 85.61	Shaggy-related protein kinase epsilon OS=Arabidops... [more]
KSG8_ARATH	E-Value: 0.000e+0, PID: 76.62	Shaggy-related protein kinase theta OS=Arabidopsis... [more]
KSG4_ARATH	E-Value: 1.150e-179, PID: 74.02	Shaggy-related protein kinase delta OS=Arabidopsis... [more]
KSG10_ARATH	E-Value: 2.833e-178, PID: 75.13	Shaggy-related protein kinase kappa OS=Arabidopsis... [more]

BLAST of orange1.1g015615m vs. NCBI nr

Analysis Date: 2018-10-02

Analysis Name: blastx Citrus sinensis v1.0 genes vs NCBI nr

Note: The match links are able to link out to SwissProt and NCBI because of the initial setup where we added the database settings and we set regular expressions for parsing the match accessions.

1.6.7.3 Adding InterProScan Results

Note: Remember you must set the `$DRUPAL_HOME` environment variable if you want to cut-and-paste the commands below. See *DRUPAL_HOME Variable*

For this tutorial, these results were obtained by using a local installation of InterProScan installed on a computational cluster. However, you may choose to use Blast2GO or the online InterProScan utility. Results should be saved in `XML` format.

What is InterProScan?

To learn more about InterProScan, please visit https://www.ebi.ac.uk/interpro/

Create the Analysis Page

> **Note:** It is always recommended to create an analysis page anytime you import data. The purpose of the analysis page is to describe how the data being added was derived or collected.

Tripal defines the **InterPro Results** Bundle, which is a specific type of Chado analysis. Create a new record by going to `Content -> Tripal Content -> Add Tripal Content --> InterPro Results`.

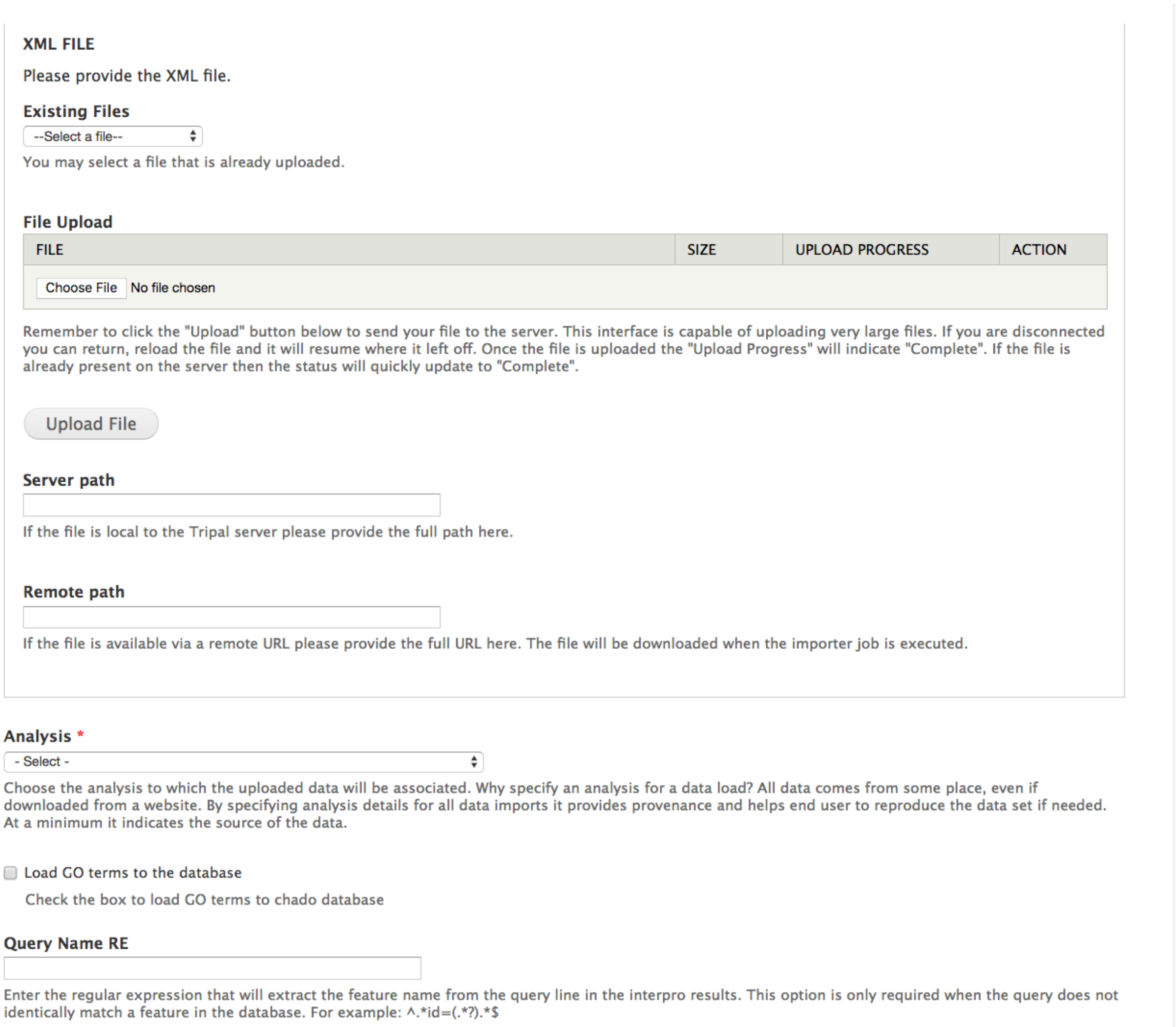
Fill out the fields as described in the table below.

Field	Value
Name	InterPro Annotations of C. sinensis v1.0
InterPro Program	InterProScan
InterPro Version	4.8
Date Performed	Current Date
Data Source Name	C. sinensis v1.0 mRNA
Data Source Version	v1.0
Data Source URI	n/a
Description	Materials & Methods: C. sinensis mRNA sequences were mapped to IPR domains and GO terms using a local installation of InterProScan executed on a computational cluster. InterProScan date files used were MATCH_DATA_v32, DATA_v32.0 and PTHR_DATA v31.0.

Press the **Save** button.

Import the InterProScan XML results

Next, we will load InterProScan results for our citrus gene. To do this, navigate to **Tripal > Data Loaders > Chado InterProScan XML results loader**. The following page will be presented:



XML FILE

Please provide the XML file.

Existing Files

--Select a file--

You may select a file that is already uploaded.

File Upload

FILE	SIZE	UPLOAD PROGRESS	ACTION
Choose File No file chosen			

Remember to click the "Upload" button below to send your file to the server. This interface is capable of uploading very large files. If you are disconnected you can return, reload the file and it will resume where it left off. Once the file is uploaded the "Upload Progress" will indicate "Complete". If the file is already present on the server then the status will quickly update to "Complete".

Server path

If the file is local to the Tripal server please provide the full path here.

Remote path

If the file is available via a remote URL please provide the full URL here. The file will be downloaded when the importer job is executed.

Analysis *

- Select -

Choose the analysis to which the uploaded data will be associated. Why specify an analysis for a data load? All data comes from some place, even if downloaded from a website. By specifying analysis details for all data imports it provides provenance and helps end user to reproduce the data set if needed. At a minimum it indicates the source of the data.

☐ Load GO terms to the database

Check the box to load GO terms to chado database

Query Name RE

Enter the regular expression that will extract the feature name from the query line in the interpro results. This option is only required when the query does not identically match a feature in the database. For example: ^.*id=(.*?).*$

The top section of this page provides multiple methods for providing results file: via an upload interface, specifying a remote URL or a file path that is local to the server. Most likely, you will always upload or provide a remote URL. However, we downloaded the files earlier, and stored them here: `$DRUPAL_HOME/sites/default/files`. So, in this case we can use the path on the local server. Provide the following value for this form:

Field	Value
Server path	sites/default/files/Citrus_sinensis-orange1.1g015632m.g.iprscan.xml
Analysis	InterPro Annotations of C. sinensis v1.0
'Load GO terms to the database'	'unchecked'
Query Name RE	
Use Unique Name	unchecked
Query Type	mRNA

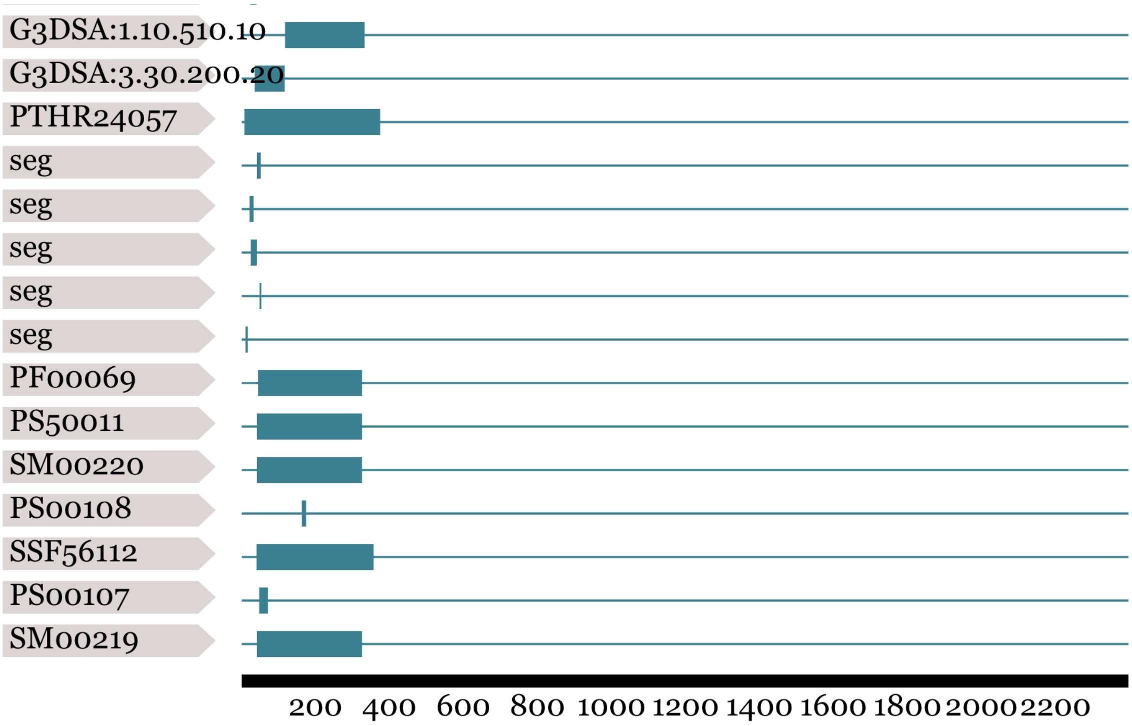
In order for GO terms to be imported, the Gene Ontology must be loaded on your site: for this tutorial, we leave the box unchecked.

Note: For the **Server path** we need not give the full path. Because we downloaded the files into the Drupal directory we can leave off any preceding path and Tripal will resolve the path. Otherwise we could provide the full path.

Clicking the **Import InterProScan file** will add a job which we can manually execute with the following command:

```
drush trp-run-jobs --username=administrator --root=$DRUPAL_HOME
```

After the job is run, our InterPro field will be populated on the mRNA page with an annotation diagram:



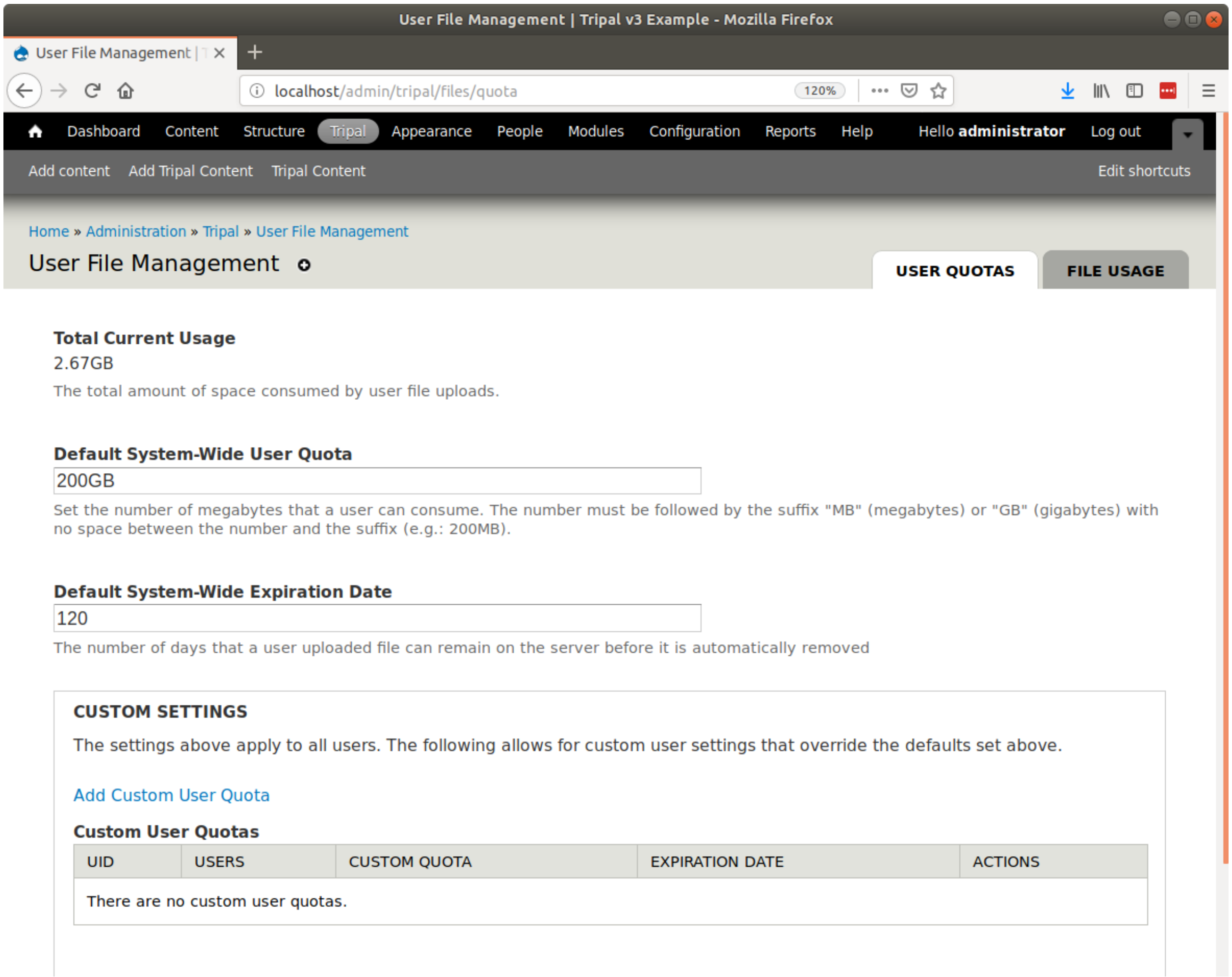
1.6.7.4 Adding KEGG Results

1.6.7.5 Managing Gene Ontology Annotations

1.7 File Management

1.7.1 User Quotas

Data importers that use the Tripal API and Tripal supported widgets automatically associate uploaded files with users. If you are allowing end-users to upload files you may want to consider adding quotas to prevent the server storage from filling. To ensure that users do not exceed the limits of the server a quota system is available. Navigate to **Administer > Tripal > User File Management** and click the **User Quotas** tab to reveal the following page:



Total Current Usage

2.67GB

The total amount of space consumed by user file uploads.

Default System-Wide User Quota

200GB

Set the number of megabytes that a user can consume. The number must be followed by the suffix "MB" (megabytes) or "GB" (gigabytes) with no space between the number and the suffix (e.g.: 200MB).

Default System-Wide Expiration Date

120

The number of days that a user uploaded file can remain on the server before it is automatically removed

CUSTOM SETTINGS

The settings above apply to all users. The following allows for custom user settings that override the defaults set above.

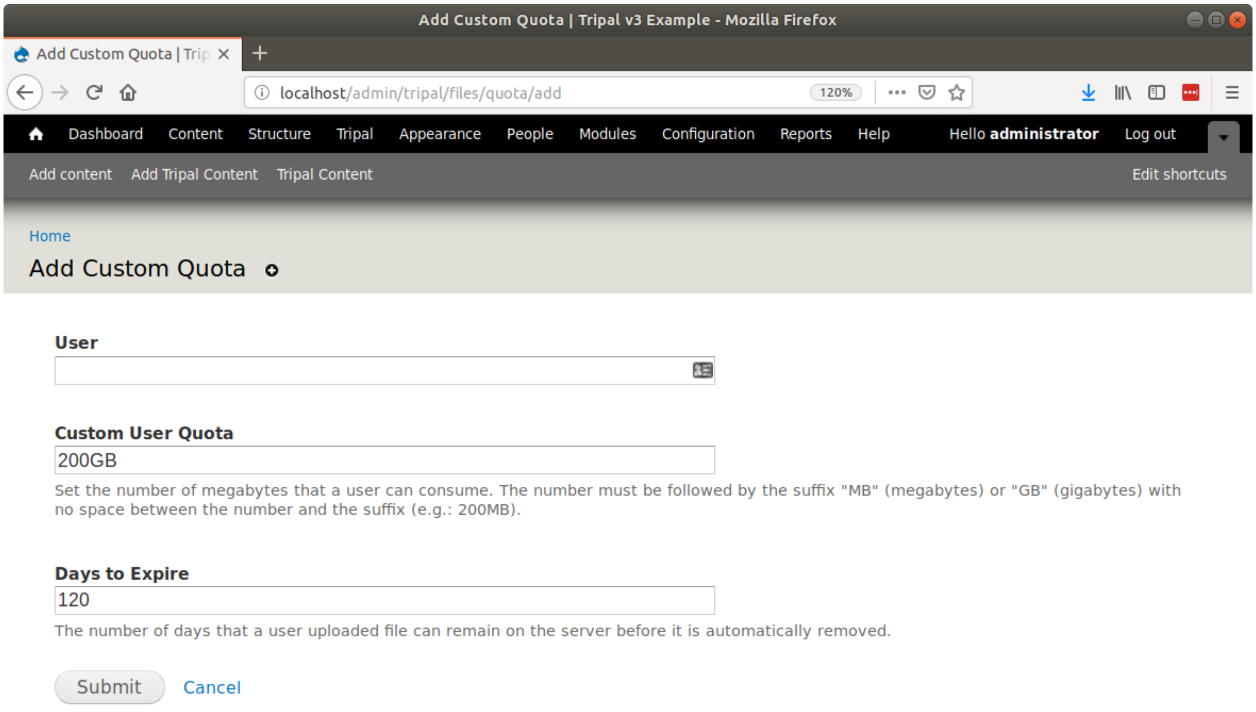
Add Custom User Quota

Custom User Quotas

UID	USERS	CUSTOM QUOTA	EXPIRATION DATE	ACTIONS
There are no custom user quotas.				

First, the total amount of space consumed by all uploaded files is shown at the top of the page. Initially this will indicate 0 B (for zero bytes); as users upload files this value will change. You may return to this page in the future to check how much space is currently used by user uploads. Here you can also specify the default system-wide quota that all users receive. By default this is set to 64 Megabytes and an expiration of 60 days. Once a file has existed on the site for 60 days the file is marked for deletion and will be removed when the Drupal cron is executed. The default of 64MB per user is most likely too small for your site. Adjust this setting and the days to expire as appropriate for your site's expected number of users and storage limitations and click the **Save** button to preserve any changes you have made.

In addition to the default settings for all users, you may want to allow specific users to have a larger (or perhaps smaller) quota. You can set user-specific quotas by clicking the **Add Custom User Quota** link near the bottom of the page. The following page appears:



Add Custom Quota

User

Custom User Quota

200GB

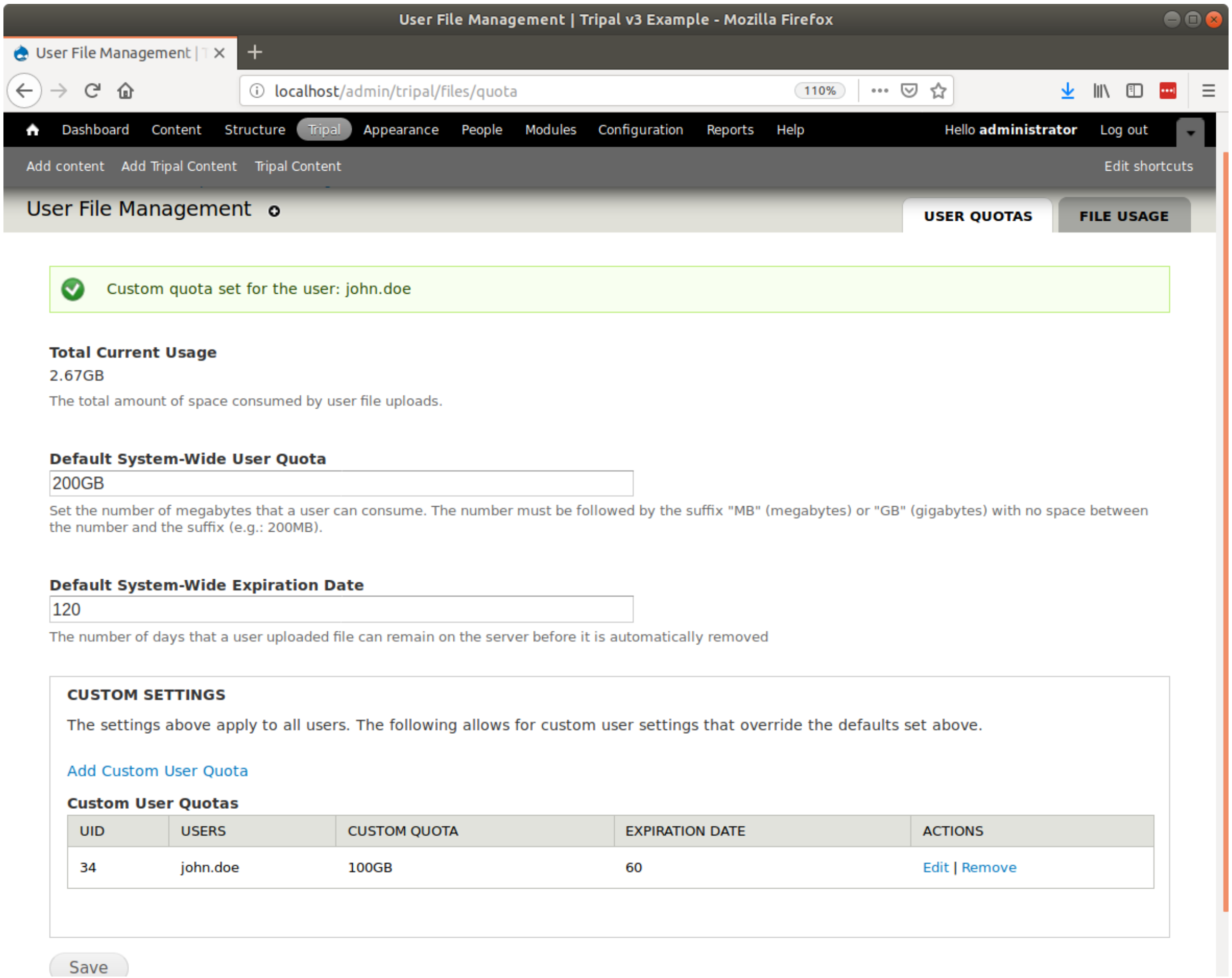
Set the number of megabytes that a user can consume. The number must be followed by the suffix "MB" (megabytes) or "GB" (gigabytes) with no space between the number and the suffix (e.g.: 200MB).

Days to Expire

120

The number of days that a user uploaded file can remain on the server before it is automatically removed.

Here you must specify the Drupal user name of the user who should be granted a custom quota. This field will auto populate suggestions as you type to help you find the correct username. Enter the desired quota size and expiration days and click the **Submit** button. you will then see the user-specific quota listed in the table at the bottom of the page:



User File Management

Custom quota set for the user: john.doe

Total Current Usage

2.67GB

The total amount of space consumed by user file uploads.

Default System-Wide User Quota

200GB

Set the number of megabytes that a user can consume. The number must be followed by the suffix "MB" (megabytes) or "GB" (gigabytes) with no space between the number and the suffix (e.g.: 200MB).

Default System-Wide Expiration Date

120

The number of days that a user uploaded file can remain on the server before it is automatically removed

CUSTOM SETTINGS

The settings above apply to all users. The following allows for custom user settings that override the defaults set above.

Add Custom User Quota

Custom User Quotas

UID	USERS	CUSTOM QUOTA	EXPIRATION DATE	ACTIONS
34	john.doe	100GB	60	Edit | Remove

1.7.2 User's Files

User's with permission to upload files are able to use the Tripal file uploader to add files to the server. The core Tripal Data Importers use the Tripal file uploader and extension modules may use it as well. You can enable this functionality for users by Navigating to **Admin > People** and click the **Permissions** Tab. next scrol to the **Tripal** section and set the **Upload Files** permissions as desired for your site. The following screenshot shows the permission on a default Drupal site.

user_guide/../file_upload_permission.png.png

User's who have the ability to upload files can manage files on their own Account pages.

As described in the previous section, the site administrator can set a system-wide or user-specific default expiration number of days for a file. This means files will be removed automatically from the server once their expiration data is set.

Note: Automatic removal of files can only occur if the Drupal cron is setup to run automatically.

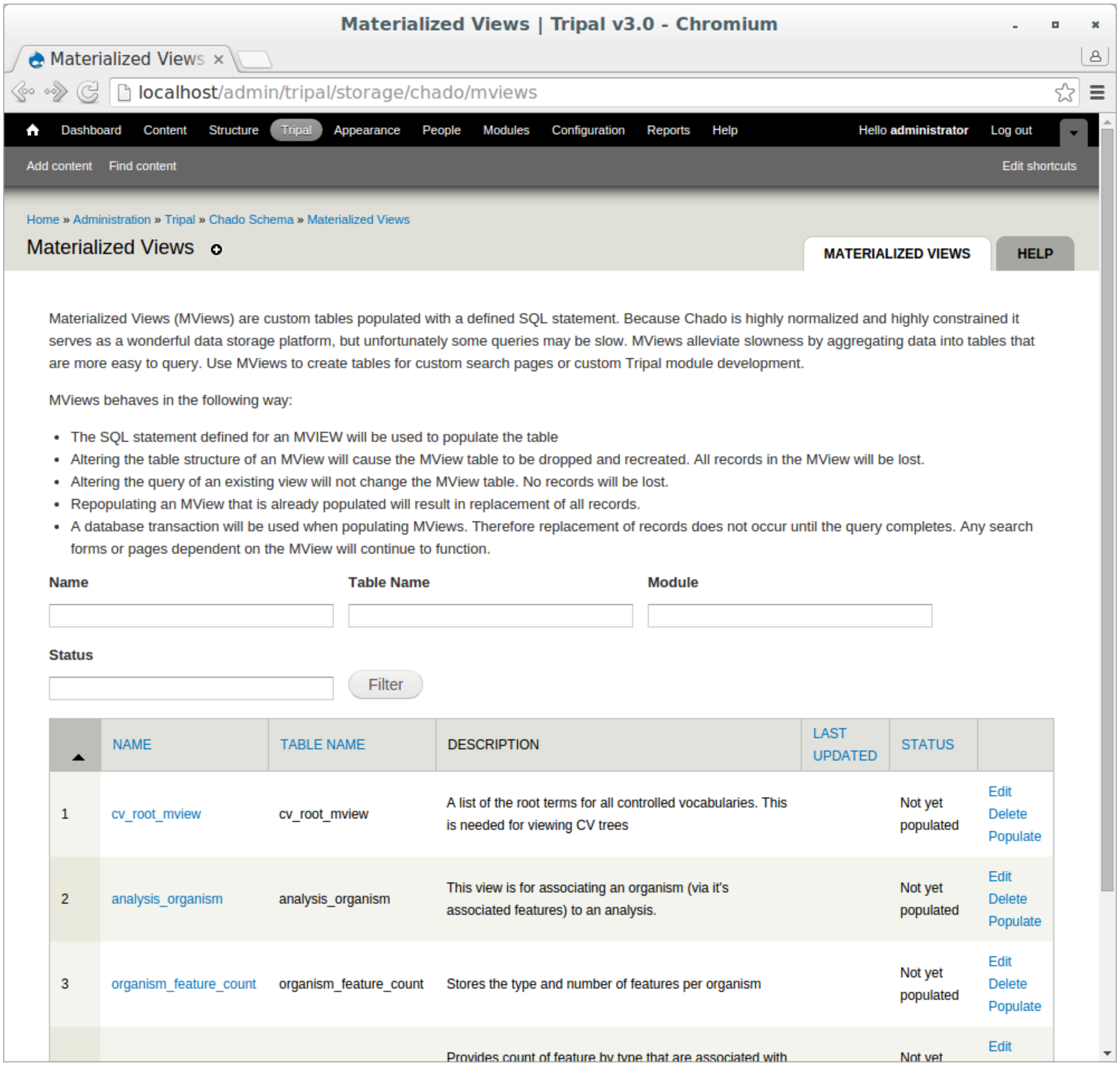
Each

1.8 Materialized Views

Note: Remember you must set the `$DRUPAL_HOME` environment variable if you want to cut-and-paste the commands below. See *DRUPAL_HOME Variable*

Chado is efficient as a data warehouse but queries can become slow depending on the type of query. To help simplify and speed up these queries, materialized views can be employed. For a materialized view, a new database table is created and then populated with the results of a pre-defined SQL query. This allows you to execute a much simpler and faster query on the materialized view when producing user pages. A side effect, however is redundant data, with the materialized view becoming stale if not updated regularly.

Tripal provides a mechanism for populating and updating these materialized views. These can be found on the **Tripal** → **Data Storage** → **Chado -> Materialized Views** page.


Materialized Views | Tripal v3.0 - Chromium
Materialized Views
localhost/admin/tripal/storage/chado/mviews
Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help
Hello administrator Log out
Add content Find content
Edit shortcuts
Home » Administration » Tripal » Chado Schema » Materialized Views
MATERIALIZED VIEWS
HELP
Filter

Materialized Views

Materialized Views (MViews) are custom tables populated with a defined SQL statement. Because Chado is highly normalized and highly constrained it serves as a wonderful data storage platform, but unfortunately some queries may be slow. MViews alleviate slowness by aggregating data into tables that are more easy to query. Use MViews to create tables for custom search pages or custom Tripal module development.

MViews behaves in the following way:

- The SQL statement defined for an MVIEW will be used to populate the table
- Altering the table structure of an MView will cause the MView table to be dropped and recreated. All records in the MView will be lost.
- Altering the query of an existing view will not change the MView table. No records will be lost.
- Repopulating an MView that is already populated will result in replacement of all records.
- A database transaction will be used when populating MViews. Therefore replacement of records does not occur until the query completes. Any search forms or pages dependent on the MView will continue to function.

Name **Table Name** **Module**

Status

▲	NAME	TABLE NAME	DESCRIPTION	LAST UPDATED	STATUS	
1	cv_root_mview	cv_root_mview	A list of the root terms for all controlled vocabularies. This is needed for viewing CV trees		Not yet populated	Edit Delete Populate
2	analysis_organism	analysis_organism	This view is for associating an organism (via it's associated features) to an analysis.		Not yet populated	Edit Delete Populate
3	organism_feature_count	organism_feature_count	Stores the type and number of features per organism		Not yet populated	Edit Delete Populate
			Provides count of feature by type that are associated with		Not yet	Edit

Here we see several materialized views. These were installed automatically by the Tripal Chado module. To update these views, click the **Populate** link for each one.

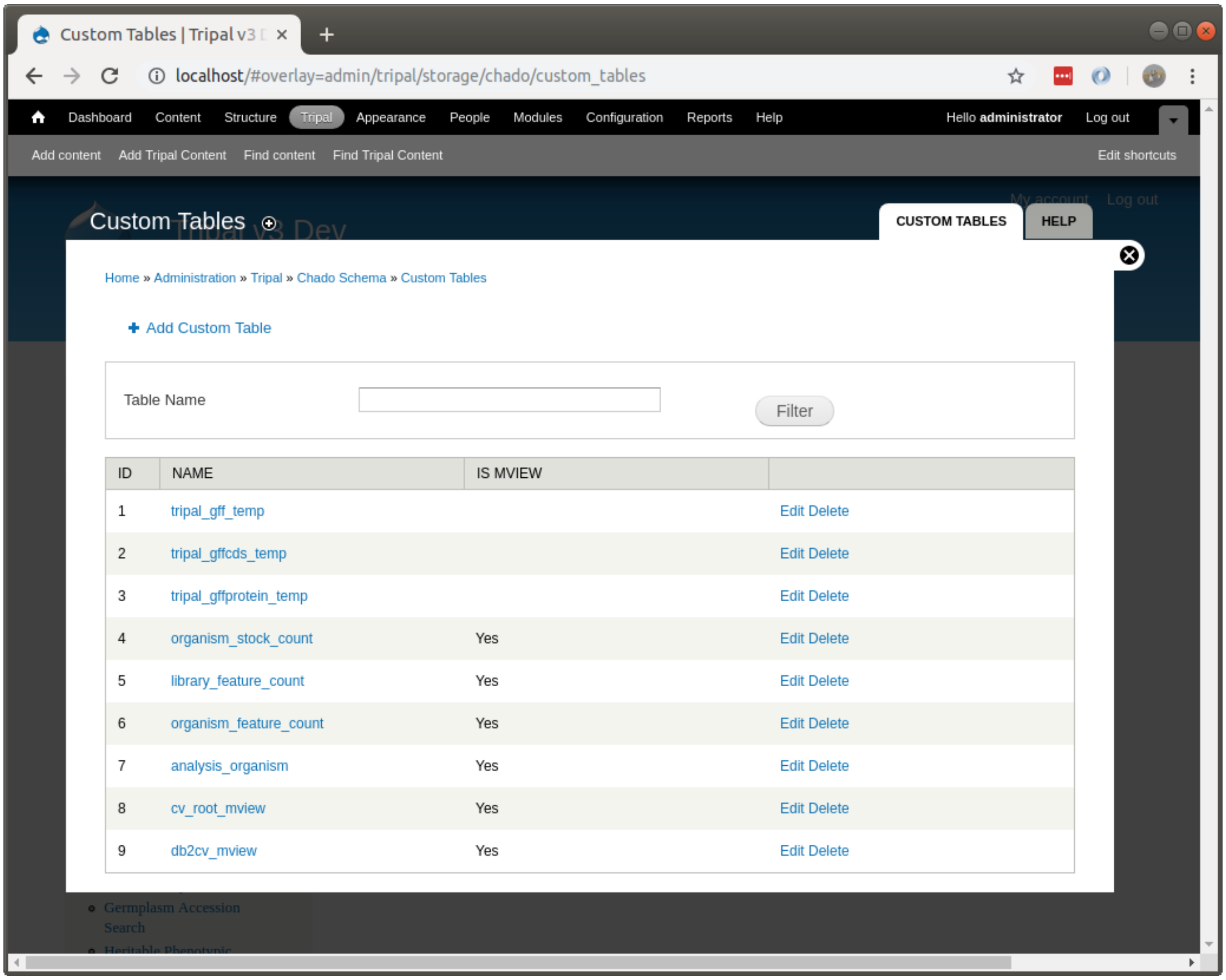
This will submit jobs to populate the views with data. Now, run the jobs:

```
cd $DRUPAL_HOME
drush trp-run-jobs --user=administrator
```

You can now see that all views are up-to-date on the **Materialized Views Page**. The number of rows in the view table is shown.

1.9 Custom Tables

Chado has a large number of tables that can be used for storing a variety of biological and ancillary data. All effort should be made to use the existing tables for storing data. However, there are times when the current set of tables is not adequate for storing some data. Tripal allows you to create custom tables in Chado using the Tripal web interface. The interface for managing custom tables can be found on the **Tripal** → **Data Storage** → **Chado -> Custom Tables** page.



ID	NAME	IS MVIEW	
1	tripal_gff_temp		Edit Delete
2	tripal_gffcds_temp		Edit Delete
3	tripal_gffprotein_temp		Edit Delete
4	organism_stock_count	Yes	Edit Delete
5	library_feature_count	Yes	Edit Delete
6	organism_feature_count	Yes	Edit Delete
7	analysis_organism	Yes	Edit Delete
8	cv_root_mview	Yes	Edit Delete
9	db2cv_mview	Yes	Edit Delete

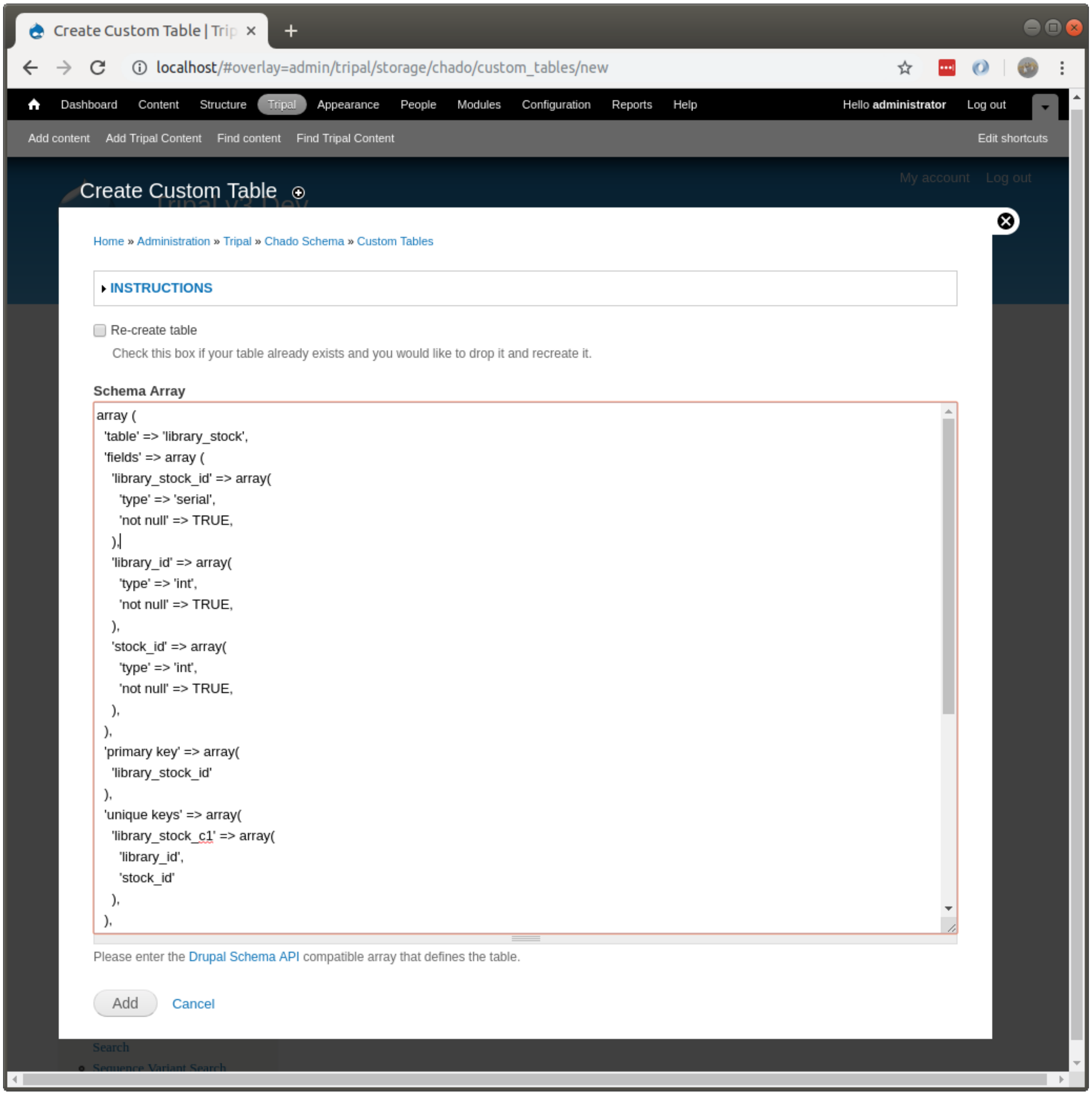
This page contains a link to add a new custom table (at the top), a search interface to find tables by name, and a listing of existing tables. You can use this interface to add, edit and remove custom tables. By default, several custom tables have been created automatically by Tripal modules. These tables hold data managed by the modules. Also, tables used as materialized views appear in this list as well. They are identified with the *Is Mview* column.

> **Warning:** You should not remove or edit custom tables that were created programmatically by tripal modules as it may cause problems with the proper function of the modules that created them.

If you need to create a new table you should follow the following procedure.

1. Plan and design your table. It is best if you follow Chado design style so that your table would easily fit into Chado.
2. Reach out to the Chado community on the Chado mailing list and ask for advice on your design.
3. If you think your new table would benefit others, consider requesting it be added to a future version of Chado by adding an issue in the Chado GitHub issue queue.
4. Write your table in the Drupal Schema API array format.

5. Navigate to the **Tripal** → **Data Storage** → **Chado -> Custom Tables** page on your site. Click the *Add Custom Table* link at the top and cut-and-paste the schema array and click the *Add* button.



□ Re-create table

Check this box if your table already exists and you would like to drop it and recreate it.

Schema Array

```
array (
  'table' => 'library_stock',
  'fields' => array (
    'library_stock_id' => array(
      'type' => 'serial',
      'not null' => TRUE,
    ),
    'library_id' => array(
      'type' => 'int',
      'not null' => TRUE,
    ),
    'stock_id' => array(
      'type' => 'int',
      'not null' => TRUE,
    ),
  ),
  'primary key' => array(
    'library_stock_id'
  ),
  'unique keys' => array(
    'library_stock_c1' => array(
      'library_id',
      'stock_id'
    ),
  ),
```

Please enter the Drupal Schema API compatible array that defines the table.

One your table is created, you will want to add data to it. You can do so without any programming by using the *Bulk Loader*. The bulk loader expects that your data is housed in a tab-delimited text file and allows you to map columns from your file to columns in your new custom table. Each row of your file becomes a new record in your table.

Alternatively, if you can write custom loaders using the Tripal API as described in the *Developer's Guide*

1.10 Searching

Drupal and Tripal offer a variety of methods for searching biological content on your site. Each has it's own advantages and meets different needs. This section provides a description of several different ways to add searching. The two primary categories of search tools are content-specific and site-wide searching. The site-wide search tools typically provide a single text box that allow a user to provide a set of key words. The results of the search will span multiple content types. Often, site-wide searches allow users to quickly find content regardless of the content type. But, they sometimes lack fine-grained control for filtering. The content-specific search tools provide more fine-grained filtering for a single content type. Therefore, it is often necessary to provide multiple types of search tools for different content types.

There are several options for the addition of both site-wide and content-specific search tools which include:

For site-wide searching you can:

- Use the Default Drupal Search
- Use the Search API Module
- Use an independent search tool. Two popular tools that integrate with Drupal include: - ElasticSearch - Apache Solr

For content-specific searching you can:

- Use the search tools that Tripal provides
- Develop your own search tools using Drupal Views
- Write your own custom search tools using PHP and Tripal's API functions.

You may not want to consider using multiple search tools, such as a site-wide tool and content-specific tools. The following sections provide a description for use and setup of some of these different options.

1.10.1 Tripal Content-Specific Search Tools

By default, Tripal will provide a search tool for every Tripal content type. When a new content type is created, a new search tool is automatically created for that tool.

These search tools will be available at `/data_search/[content label]`, such as `/data_search/mrna`.

If you have the `view_ui` module enabled, you can find a list of all search views under **Structure –> Views**.

1.10.2 Search API Module: Site-Wide Searching

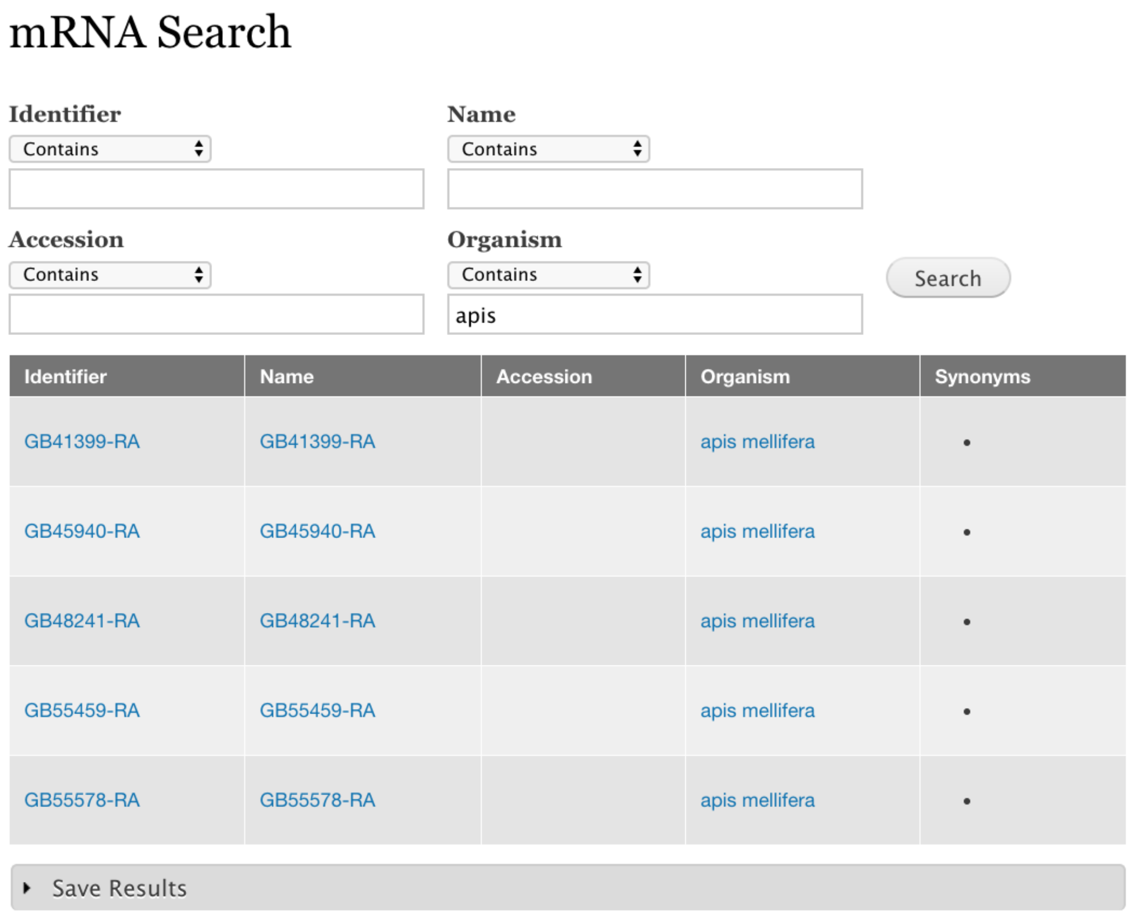
1.10.2.1 Installing Drupal Search API

- Search API: This module provides an interface for much more powerful, efficient searching than the Drupal core search module. Specifically, it allows you to use more powerful engines such as Elastic Search and Apache Solr, as well as, advanced features such as facets (for narrowing down search results based on fields or entity type), fuzzy search, etc.
- Search API Database Service: This module provides a Search Backend/Server defining how your search index should be stored. Specifically, it just stores the index in your current drupal database.

Install the **Search API** and **Database search** modules as you would any other Drupal module. This can be done using Drupal's module installation page as shown in the screenshot below. For installation instructions reference the Drupal.org Tutorial.

Alternatively, installation can be accomplished on the command-line by executing the following drush commands inside of the Drupal directory:

```
drush pm-enable search_api
drush pm-enable search_api_db
drush pm-enable search_api_views
```



mRNA Search

Identifier: Contains

Name: Contains

Accession: Contains

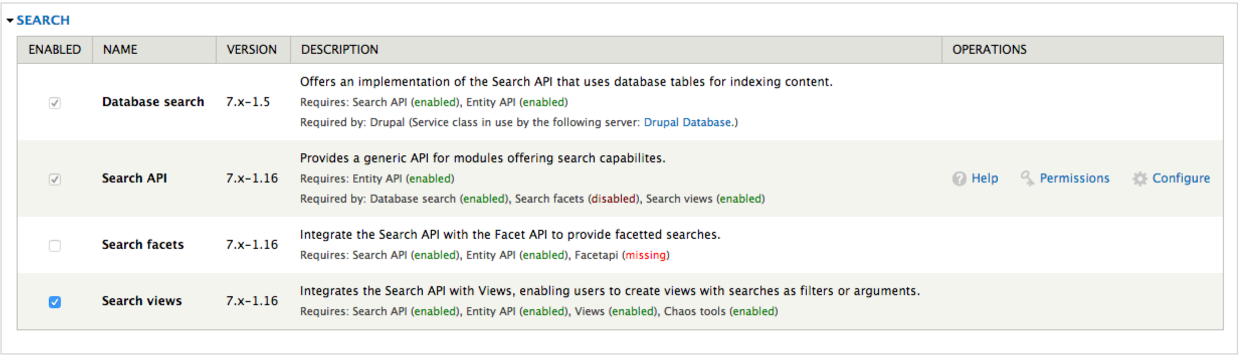
Organism: Contains
apis

Search

Identifier	Name	Accession	Organism	Synonyms
GB41399-RA	GB41399-RA		apis mellifera	•
GB45940-RA	GB45940-RA		apis mellifera	•
GB48241-RA	GB48241-RA		apis mellifera	•
GB55459-RA	GB55459-RA		apis mellifera	•
GB55578-RA	GB55578-RA		apis mellifera	•

▸ Save Results

Fig. 1: An example search with the default mRNA content type. Results are sortable by clicking the column name.



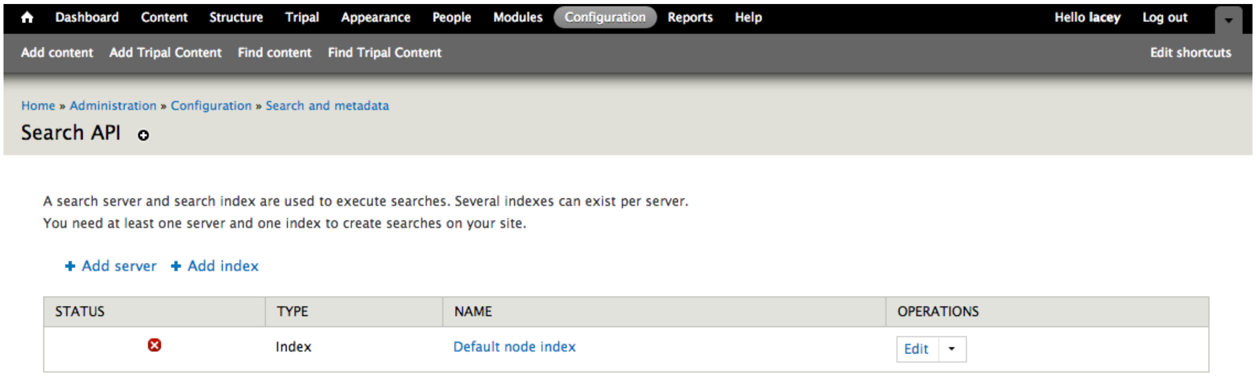
▾ SEARCH

ENABLED	NAME	VERSION	DESCRIPTION	OPERATIONS
☑	Database search	7.x-1.5	Offers an implementation of the Search API that uses database tables for indexing content. Requires: Search API (enabled), Entity API (enabled) Required by: Drupal (Service class in use by the following server: Drupal Database.)	
☑	Search API	7.x-1.16	Provides a generic API for modules offering search capabilites. Requires: Entity API (enabled) Required by: Database search (enabled), Search facets (disabled), Search views (enabled)	Help Permissions Configure
□	Search facets	7.x-1.16	Integrate the Search API with the Facet API to provide facetted searches. Requires: Search API (enabled), Entity API (enabled), Facetapi (missing)	
☑	Search views	7.x-1.16	Integrates the Search API with Views, enabling users to create views with searches as filters or arguments. Requires: Search API (enabled), Entity API (enabled), Views (enabled), Chaos tools (enabled)	

Fig. 2: Screenshot Modules enable page with Database Search, Search API and Search views enabled.

1.10.2.2 Define your Search Backend/Server

This tutorial covers using a basic Drupal database storage backend for your search. For large sites, it is recommended to use Elastic Search or Apache Solr. First, we need to tell the Search API where we want our index stored. Navigate, to the configuration page for the Search API. You can either click on the **Configure** link shown in the above screenshot or navigate to `Configuration > Search API` through the administrative toolbar. You should see the following screen:



A search server and search index are used to execute searches. Several indexes can exist per server.
You need at least one server and one index to create searches on your site.

STATUS	TYPE	NAME	OPERATIONS
⊗	Index	Default node index	Edit ▾

Before proceeding, consider deleting the "Default node index". We don't need it. Next, click the **Add Server link**. We are configuring a basic drupal database search server we do not need to install any third-part software or set-up an external server. Instead, fill out the configuration form to tell the Search API to use its own database to store the search index. Give this server the name "Drupal Database" and select "Database service" from the **Service Class** drop down. In the **Database Service** section, select "Search on parts of a word." If the search is slow due to this feature, then it is an indicator that your site should use a different service class (ie: Elastic Search or Apache Solr). Click "Create Server" to finish configuring the Search backend.

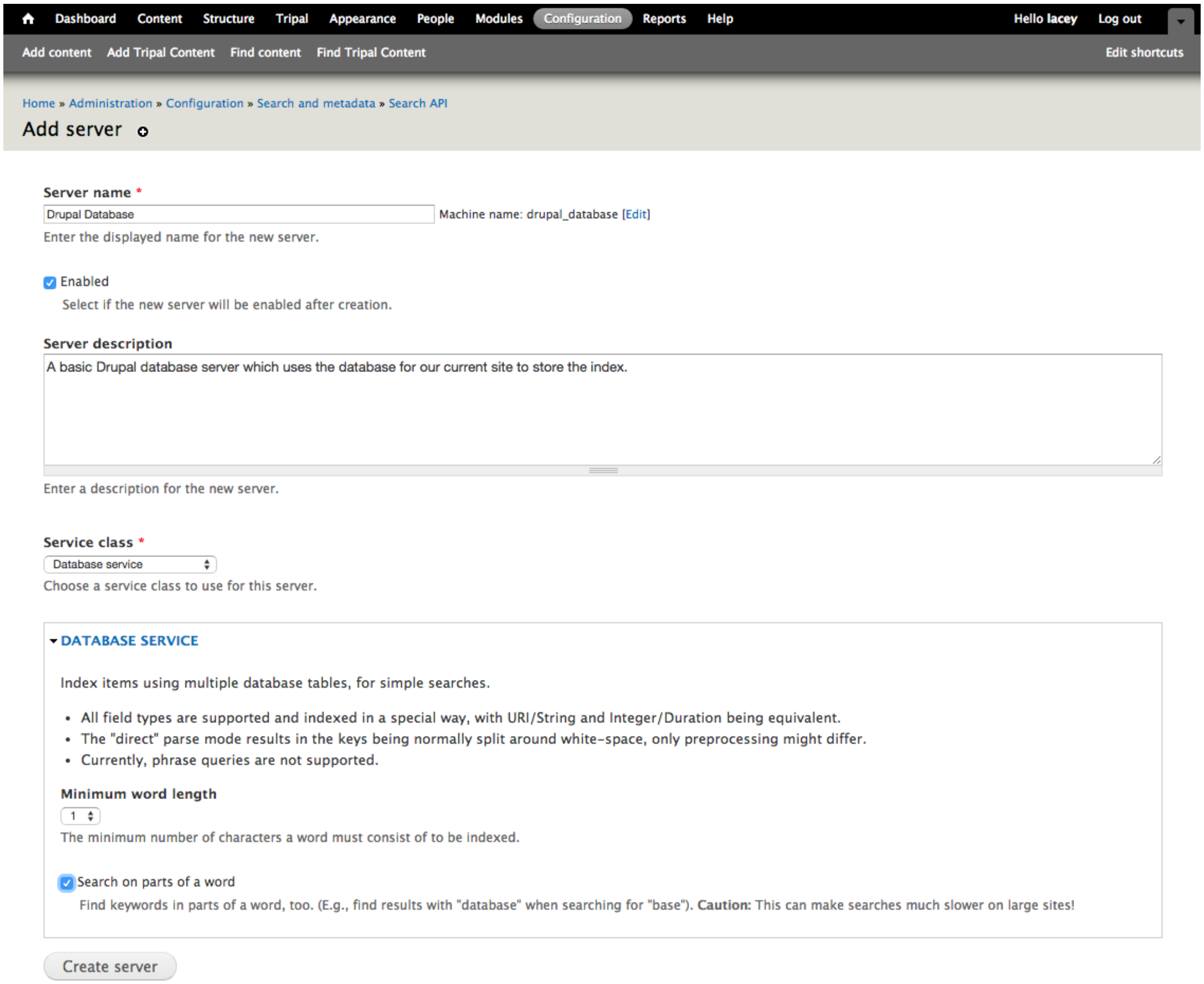
You should see the following screen–assuming all went well. Click on Search API link at the top of the screen (circled in the screenshot below) to return to the **Search API** configuration screen.

1.10.2.3 Define a Search Index

Now that we have created a server where the Search API will store our index, we have to define the index itself. On the Search API Configuration page click on the **Add index** link. The resulting page appears in the following screenshot. Name your index something descriptive. Consider including the word "search" somewhere in the name as this name will be used when setting up the search form/listing (view). For example, enter the name "Tripal Content Search." Next, select "Tripal Content" as the **Item Type**. The item type defines what content types should be indexed. One thing to keep in mind, is that the Search API currently does not support multi-entity (ie: Both Tripal and Node content) search in the same index without the Search API Multi-index Search extension module. Notice that we didn't check any of the **Bundles**. By not selecting bundles, this ensures that all Tripal Content will be indexed by the search. Finally, select from the Server dropdown the server created in the previous step and click the Create Index button.

Next we need to configure which fields should be indexed. You will be presented with a very long list of fields (the length is dependent on how many Tripal Content types you have). First, scroll to the bottom of the list and expand the **Add Related Fields** fieldset. If you are interested, add any additional fields first before checking boxes of fields above. Otherwise it may cause you to lose selection you've already made. Next, check the box beside each field you would like to be searched. Unfortunately, the interface does not indicate which fields are used per content type. Save your selection.

The first few fields will be present for all Tripal Content Types (ie: Content Id, Type, Bundle, etc, as shown in the blue box of the screenshot below), Notice, that for some checked fields there is a boost drop-down. The **boost** drop-down influences the "relevance" that a search result will have. By increasing the boost for the title indicates "if the user's



Home » Administration » Configuration » Search and metadata » Search API

Add server

Server name *

Drupal Database — Machine name: drupal_database [Edit]

Enter the displayed name for the new server.

☑ Enabled

Select if the new server will be enabled after creation.

Server description

A basic Drupal database server which uses the database for our current site to store the index.

Enter a description for the new server.

Service class *

Database service

Choose a service class to use for this server.

DATABASE SERVICE

Index items using multiple database tables, for simple searches.

- All field types are supported and indexed in a special way, with URI/String and Integer/Duration being equivalent.
- The "direct" parse mode results in the keys being normally split around white-space, only preprocessing might differ.
- Currently, phrase queries are not supported.

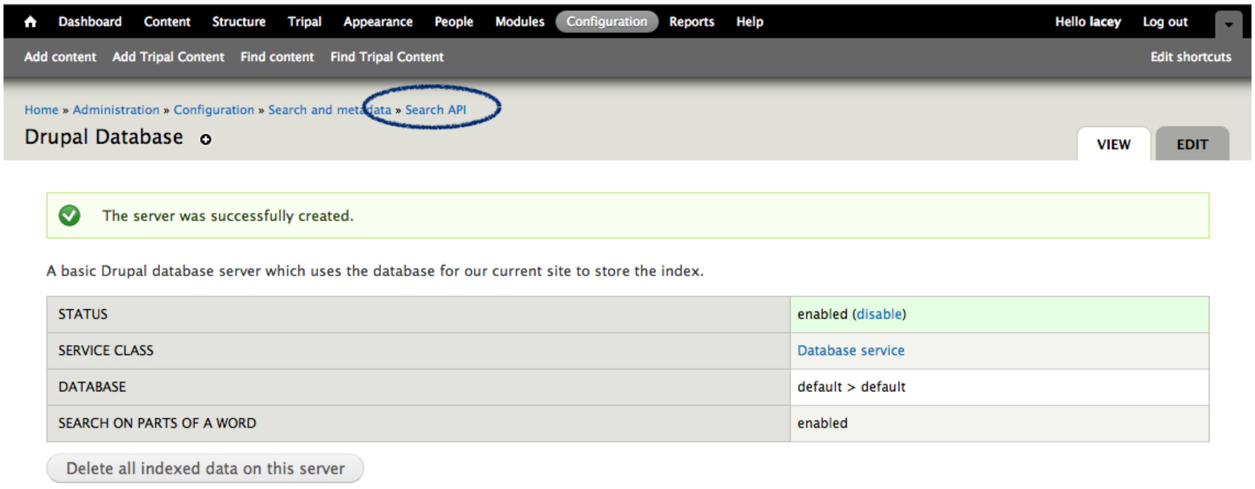
Minimum word length

1

The minimum number of characters a word must consist of to be indexed.

☑ Search on parts of a word

Find keywords in parts of a word, too. (E.g., find results with "database" when searching for "base"). **Caution:** This can make searches much slower on large sites!



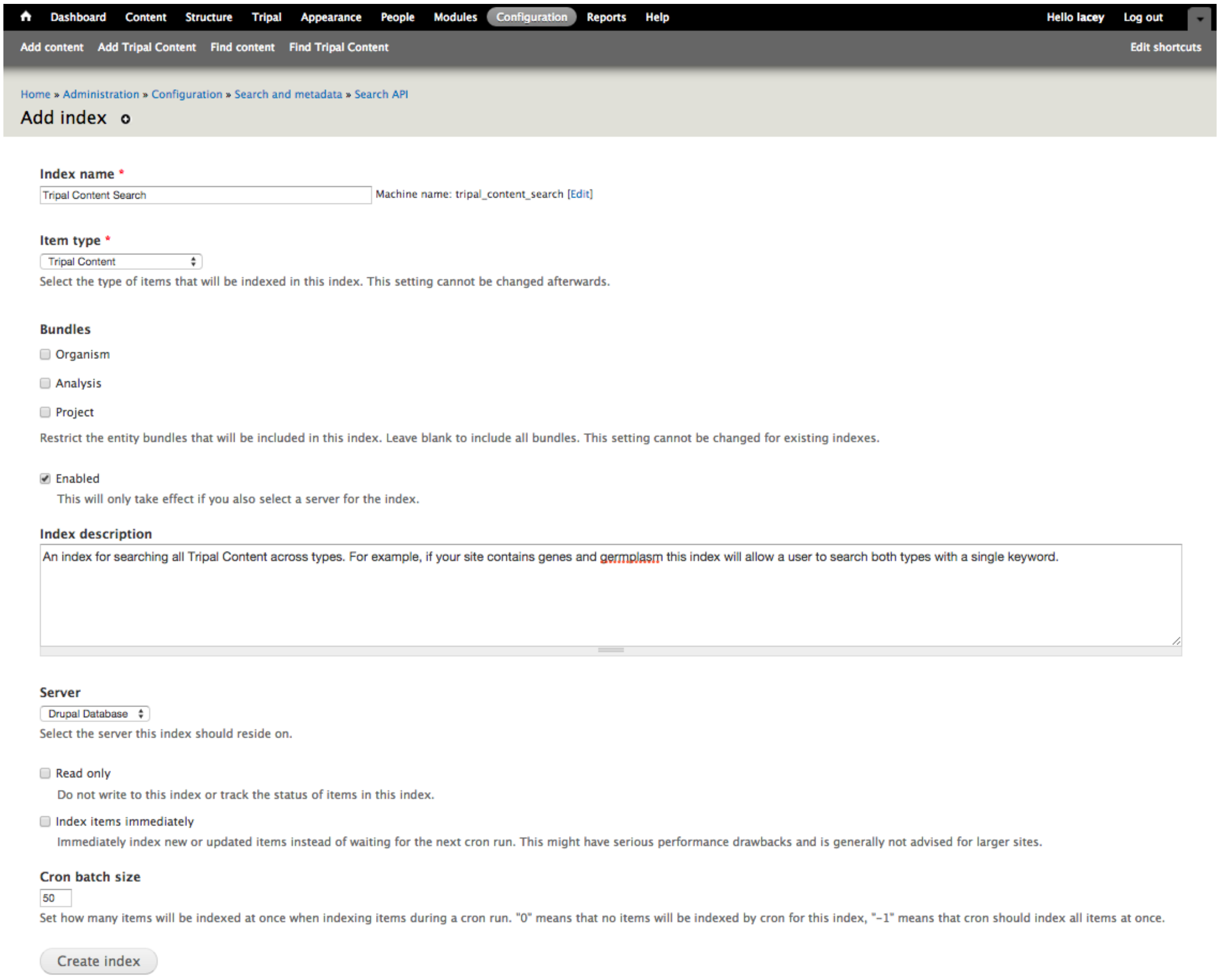
Home » Administration » Configuration » Search and metadata » Search API

Drupal Database

The server was successfully created.

A basic Drupal database server which uses the database for our current site to store the index.

STATUS	enabled (disable)
SERVICE CLASS	Database service
DATABASE	default > default
SEARCH ON PARTS OF A WORD	enabled



Home » Administration » Configuration » Search and metadata » Search API

Add index

Index name *

Tripal Content Search

Machine name: tripal_content_search [Edit]

Item type *

Tripal Content

Select the type of items that will be indexed in this index. This setting cannot be changed afterwards.

Bundles

☐ Organism

☐ Analysis

☐ Project

Restrict the entity bundles that will be included in this index. Leave blank to include all bundles. This setting cannot be changed for existing indexes.

☑ Enabled

This will only take effect if you also select a server for the index.

Index description

An index for searching all Tripal Content across types. For example, if your site contains genes and germplasm this index will allow a user to search both types with a single keyword.

Server

Drupal Database

Select the server this index should reside on.

☐ Read only

Do not write to this index or track the status of items in this index.

☐ Index items immediately

Immediately index new or updated items instead of waiting for the next cron run. This might have serious performance drawbacks and is generally not advised for larger sites.

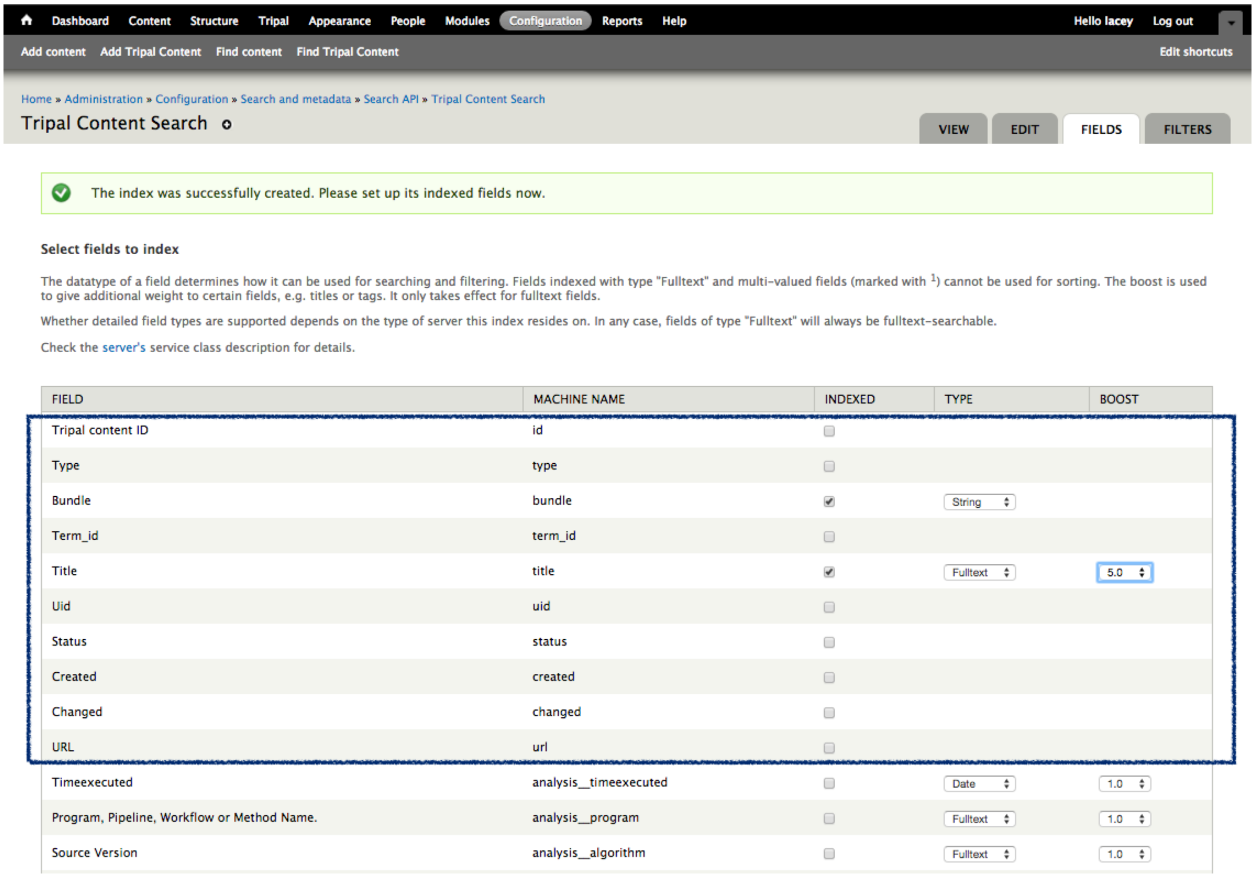
Cron batch size

50

Set how many items will be indexed at once when indexing items during a cron run. "0" means that no items will be indexed by cron for this index, "-1" means that cron should index all items at once.

Create index

keywords are in the title it is more likely this content is the one they're looking for". Because we want titles and contnet types (i.e. bundles) highly searchable, set the boost 5 for these (see screenshot below).



Tripal Content Search

The index was successfully created. Please set up its indexed fields now.

Select fields to index

The datatype of a field determines how it can be used for searching and filtering. Fields indexed with type "Fulltext" and multi-valued fields (marked with ^{1}) cannot be used for sorting. The boost is used to give additional weight to certain fields, e.g. titles or tags. It only takes effect for fulltext fields.

Whether detailed field types are supported depends on the type of server this index resides on. In any case, fields of type "Fulltext" will always be fulltext-searchable.

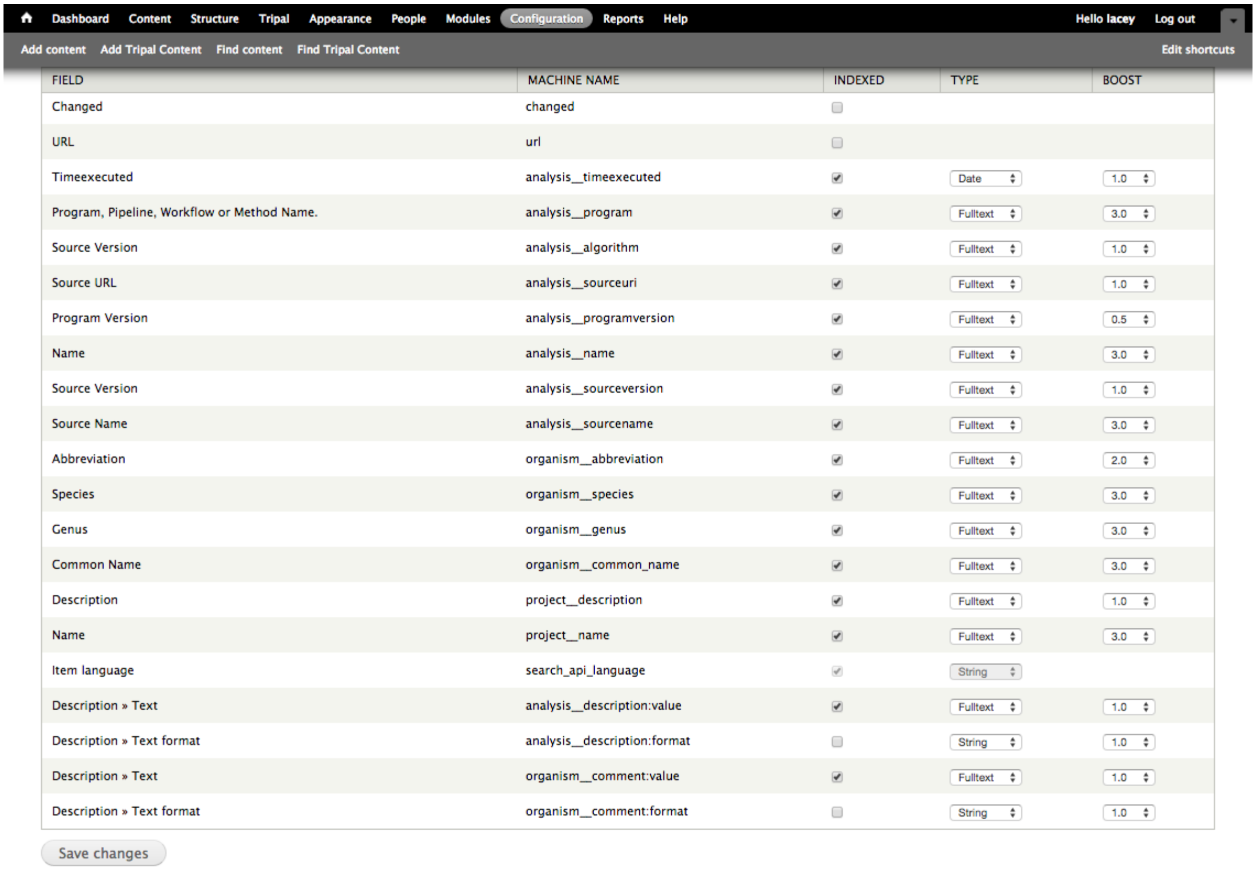
Check the server's service class description for details.

FIELD	MACHINE NAME	INDEXED	TYPE	BOOST
Tripal content ID	id	□		
Type	type	□		
Bundle	bundle	☑	String	
Term_id	term_id	□		
Title	title	☑	Fulltext	5.0
Uid	uid	□		
Status	status	□		
Created	created	□		
Changed	changed	□		
URL	url	□		
Timeexecuted	analysis__timeexecuted	□	Date	1.0
Program, Pipeline, Workflow or Method Name.	analysis__program	□	Fulltext	1.0
Source Version	analysis__algorithm	□	Fulltext	1.0

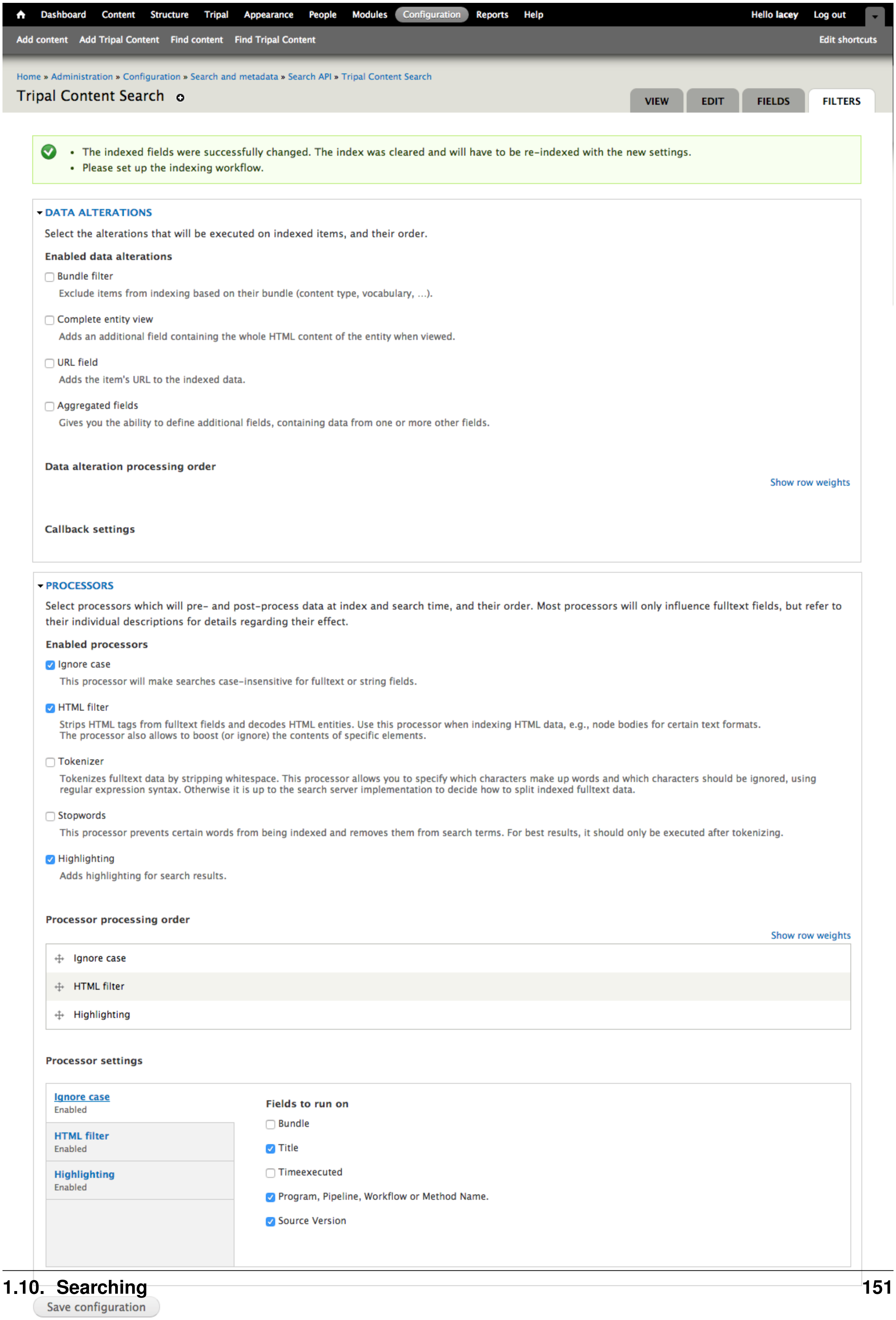
After the first set of general fields, we see the list of content type specific fields. Select fields that are appropriate for your own site and content. For full searchability, select most (if not all) of these fields. Keep in mind the number of fields selected affects the size of your index. If you know there is no useful information in a given field then do not select it. You can return later and edit the selected fields at a later date (although it does require re-indexing your site). The most important consideration at this point is what boost to apply to the various fields. As a rule of thumb, give a modest boost (but not as high as the title; e.g. set a boost of 3) for name fields and a default boost otherwise. You may want to apply a negative boost to fields users are extremely unlikely to search (but that you may want to use in facets) or that are likely to produce false positives (e.g.: analysis program version). Once you are done, click on "Save Changes".

After saving our fields we are redirected to the **Filters** tab. Finally, (last step for creating the index!), pick the extra features you would like supported. For now we will ignore the **Data Alterations** section. However, we will set some items in the **Processors** section. Keep in mind that the order you select processors is important (i.e.: if you have html filter after highlighting then it will remove your highlighting). We would like to provide case-insensitive searching with searched words highlighted. To do this, select **Ignore case**, **HTML Filter** and **Highlighting** in that order. You may want to add **Tokenizer** if you are indexing any long text fields because errors can occur if the default tokenize can sometimes fail with long words. Click "Save Configuration".

Your index is now scheduled for building! Depending upon the amount of content you have, this could take awhile as it will only index 50 pages of Tripal content per Drupal Cron run. If you click the view tab you can see the progress on the indexing process. You can return to this screen in the future from the main Search API configuration page and clicking on the name of the index.



FIELD	MACHINE NAME	INDEXED	TYPE	BOOST
Changed	changed	☐		
URL	url	☐		
Timeexecuted	analysis__timeexecuted	☑	Date	1.0
Program, Pipeline, Workflow or Method Name.	analysis__program	☑	Fulltext	3.0
Source Version	analysis__algorithm	☑	Fulltext	1.0
Source URL	analysis__sourceuri	☑	Fulltext	1.0
Program Version	analysis__programversion	☑	Fulltext	0.5
Name	analysis__name	☑	Fulltext	3.0
Source Version	analysis__sourceversion	☑	Fulltext	1.0
Source Name	analysis__sourcename	☑	Fulltext	3.0
Abbreviation	organism__abbreviation	☑	Fulltext	2.0
Species	organism__species	☑	Fulltext	3.0
Genus	organism__genus	☑	Fulltext	3.0
Common Name	organism__common_name	☑	Fulltext	3.0
Description	project__description	☑	Fulltext	1.0
Name	project__name	☑	Fulltext	3.0
Item language	search_api_language	☑	String	
Description » Text	analysis__description:value	☑	Fulltext	1.0
Description » Text format	analysis__description:format	☐	String	1.0
Description » Text	organism__comment:value	☑	Fulltext	1.0
Description » Text format	organism__comment:format	☐	String	1.0



Tripal Content Search

- The indexed fields were successfully changed. The index was cleared and will have to be re-indexed with the new settings.
- Please set up the indexing workflow.

▾ DATA ALTERATIONS

Select the alterations that will be executed on indexed items, and their order.

Enabled data alterations

☐ Bundle filter
Exclude items from indexing based on their bundle (content type, vocabulary, ...).

☐ Complete entity view
Adds an additional field containing the whole HTML content of the entity when viewed.

☐ URL field
Adds the item's URL to the indexed data.

☐ Aggregated fields
Gives you the ability to define additional fields, containing data from one or more other fields.

Data alteration processing order

Callback settings

▾ PROCESSORS

Select processors which will pre- and post-process data at index and search time, and their order. Most processors will only influence fulltext fields, but refer to their individual descriptions for details regarding their effect.

Enabled processors

☑ Ignore case
This processor will make searches case-insensitive for fulltext or string fields.

☑ HTML filter
Strips HTML tags from fulltext fields and decodes HTML entities. Use this processor when indexing HTML data, e.g., node bodies for certain text formats. The processor also allows to boost (or ignore) the contents of specific elements.

☐ Tokenizer
Tokenizes fulltext data by stripping whitespace. This processor allows you to specify which characters make up words and which characters should be ignored, using regular expression syntax. Otherwise it is up to the search server implementation to decide how to split indexed fulltext data.

☐ Stopwords
This processor prevents certain words from being indexed and removes them from search terms. For best results, it should only be executed after tokenizing.

☑ Highlighting
Adds highlighting for search results.

Processor processing order

- Ignore case
- HTML filter
- Highlighting

Processor settings

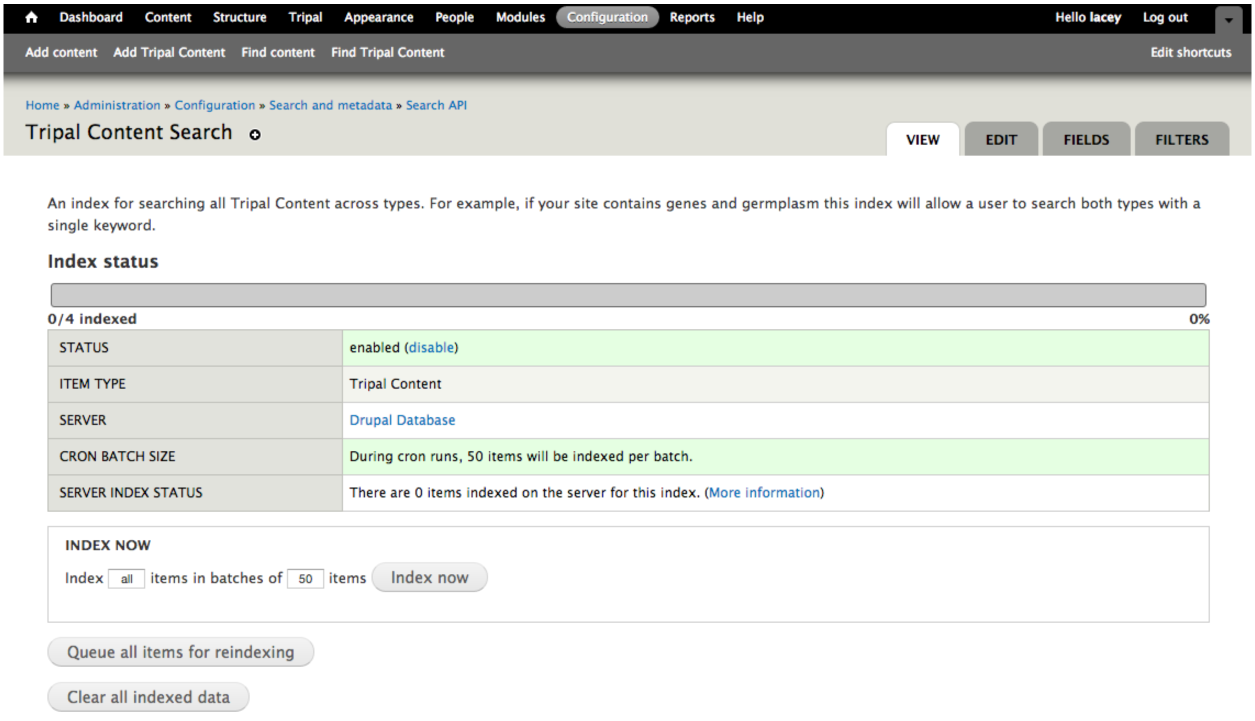
Ignore case
Enabled

HTML filter
Enabled

Highlighting
Enabled

Fields to run on

☐ Bundle
☑ Title
☐ Timeexecuted
☑ Program, Pipeline, Workflow or Method Name.
☑ Source Version



Tripal Content Search

An index for searching all Tripal Content across types. For example, if your site contains genes and germplasm this index will allow a user to search both types with a single keyword.

Index status

0/4 indexed 0%

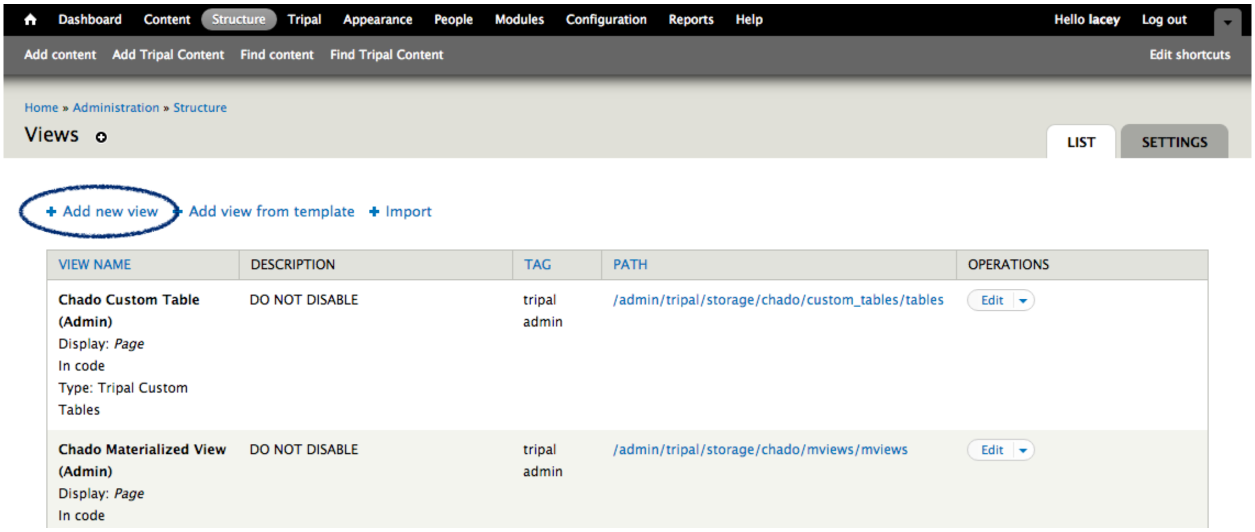
STATUS	enabled (disable)
ITEM TYPE	Tripal Content
SERVER	Drupal Database
CRON BATCH SIZE	During cron runs, 50 items will be indexed per batch.
SERVER INDEX STATUS	There are 0 items indexed on the server for this index. (More information)

INDEX NOW

Index all items in batches of 50 items

1.10.2.4 Creating a Search Interface for your users

At this point you should have an index for your Tripal Content. However, you still have not created any functionality for end users—the data might be indexed, but they can't search it, yet. To create the Search page we are going to use views. Start by going to the Views Administration UI (**Structure > Views**) and click on **Add new view.**

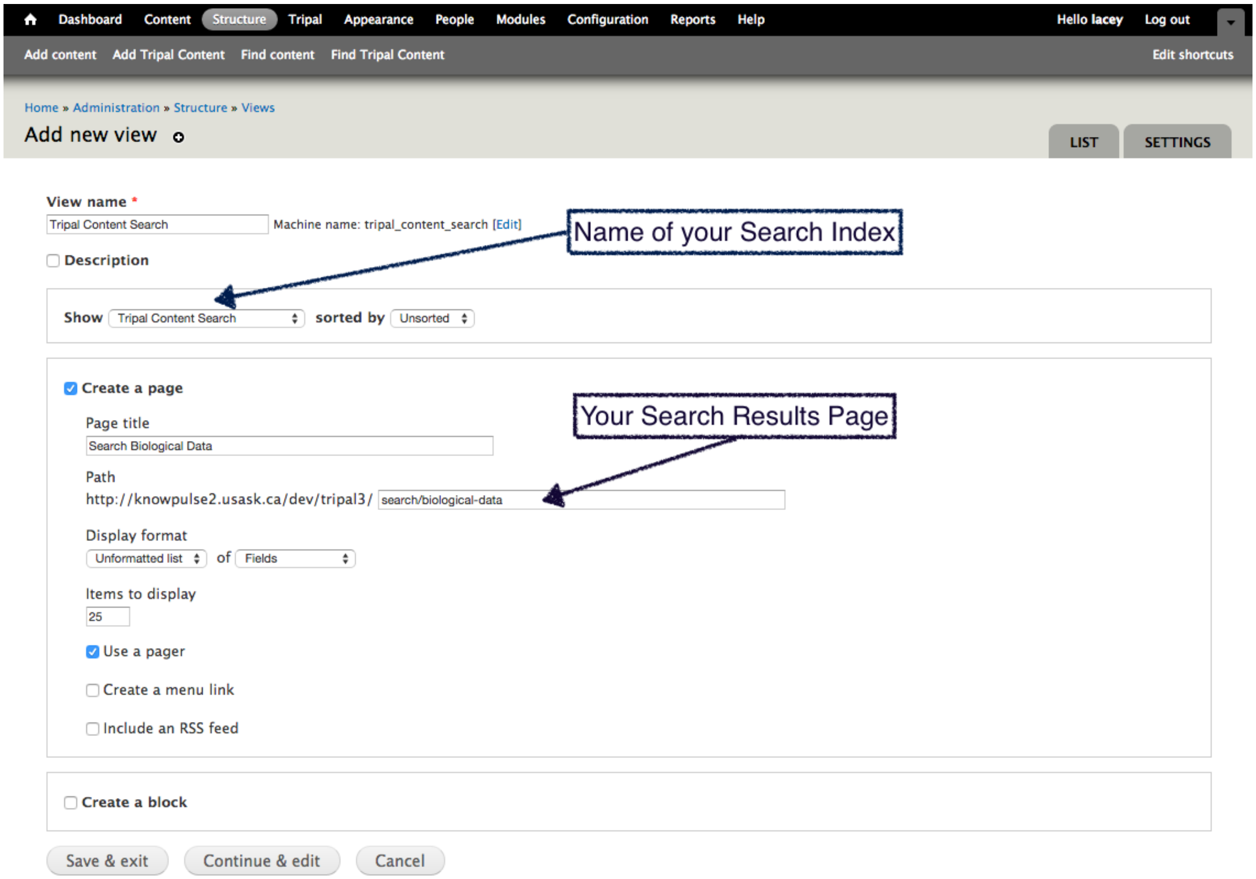


Views

VIEW NAME	DESCRIPTION	TAG	PATH	OPERATIONS
Chado Custom Table (Admin) Display: *Page* In code Type: Tripal Custom Tables	DO NOT DISABLE	tripal admin	/admin/tripal/storage/chado/custom_tables/tables	Edit
Chado Materialized View (Admin) Display: *Page* In code	DO NOT DISABLE	tripal admin	/admin/tripal/storage/chado/mviews/mviews	Edit

Name it something descriptive (e.g. Search Biological Data) as this will show up in the administrative listing. For the view type (the drop-down beside **Show**) select the name of the index you created in the last step (e.g.: Tripal Content Search). Name the page something helpful to the user (avoid the word Tripal and describe the data instead; e/g.: "Search Biological Data") and then change the path (e.g. `search/biological-data`). Click **Continue & edit.**

Next, will appear is the Edit Views UI which can be intimidating, even if you've been introduced to it before. With



Home » Administration » Structure » Views

Add new view

View name *
Tripal Content Search
Machine name: tripal_content_search [Edit]

Name of your Search Index

□ Description

Show Tripal Content Search sorted by Unsorted

☑ Create a page

Page title
Search Biological Data

Your Search Results Page

Path
http://knowpulse2.usask.ca/dev/tripal3/ search/biological-data

Display format
Unformatted list of Fields

Items to display
25

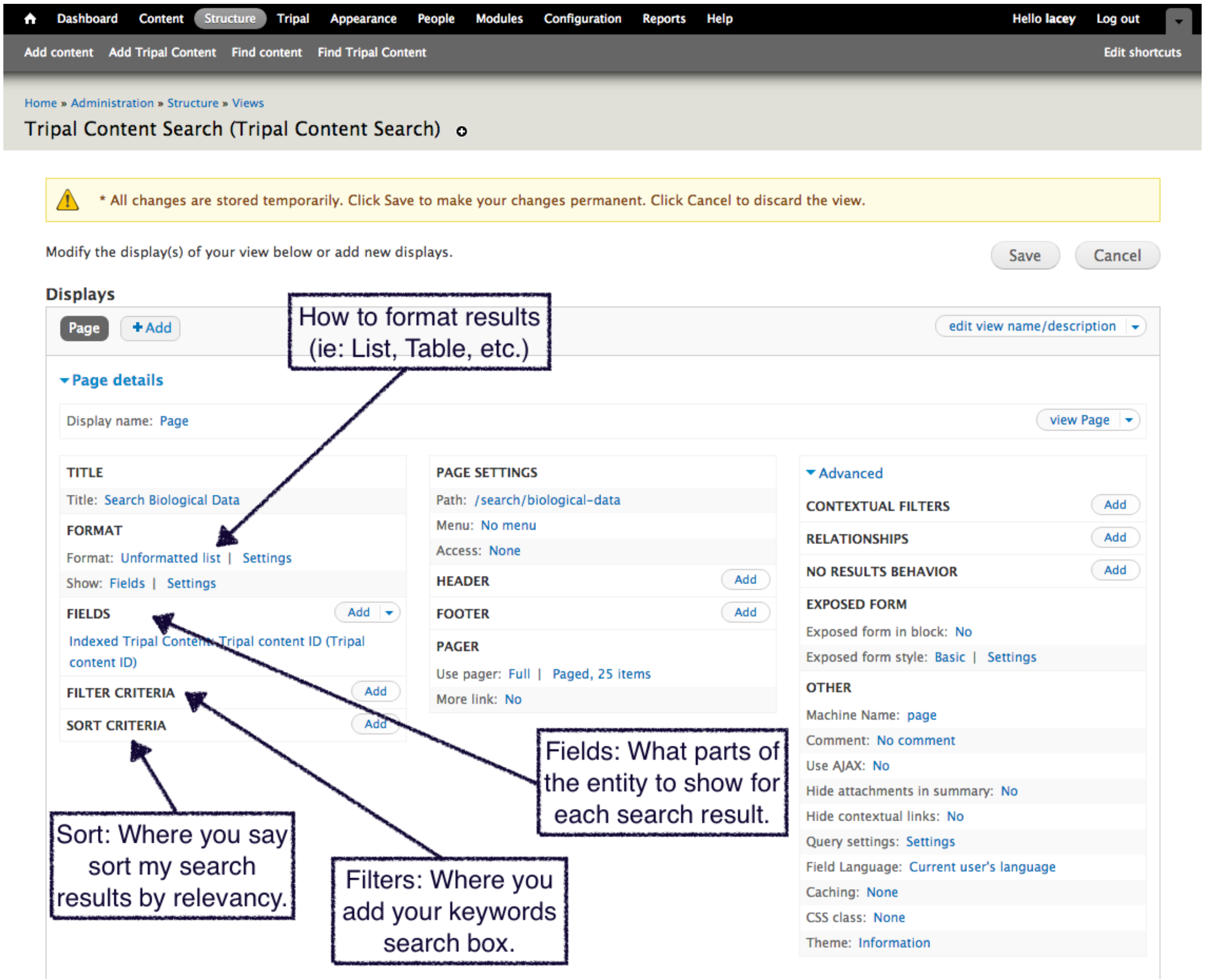
☑ Use a pager

□ Create a menu link

□ Include an RSS feed

□ Create a block

that in mind the following screenshot attempts to orient you to the parts of the UI we will use in reference to a search form/results. This tutorial will address **Fields**, **Filters** and **Sort Criteria**. It is not necessary to understand more except to point out that you should focus on the left side of the UI when looking for the sections discussed below.



Tripal Content Search (Tripal Content Search)

* All changes are stored temporarily. Click Save to make your changes permanent. Click Cancel to discard the view.

Modify the display(s) of your view below or add new displays.

Displays

Page details

Display name: Page

TITLE
Title: Search Biological Data

FORMAT
Format: Unformatted list | Settings
Show: Fields | Settings

FIELDS
Indexed Tripal Content: Tripal content ID (Tripal content ID)

FILTER CRITERIA

SORT CRITERIA

PAGE SETTINGS
Path: /search/biological-data
Menu: No menu
Access: None

HEADER

FOOTER

PAGER
Use pager: Full | Paged, 25 items
More link: No

CONTEXTUAL FILTERS

RELATIONSHIPS

NO RESULTS BEHAVIOR

EXPOSED FORM
Exposed form in block: No
Exposed form style: Basic | Settings

OTHER
Machine Name: page
Comment: No comment
Use AJAX: No
Hide attachments in summary: No
Hide contextual links: No
Query settings: Settings
Field Language: Current user's language
Caching: None
CSS class: None
Theme: Information

How to format results (ie: List, Table, etc.)

Fields: What parts of the entity to show for each search result.

Sort: Where you say sort my search results by relevancy.

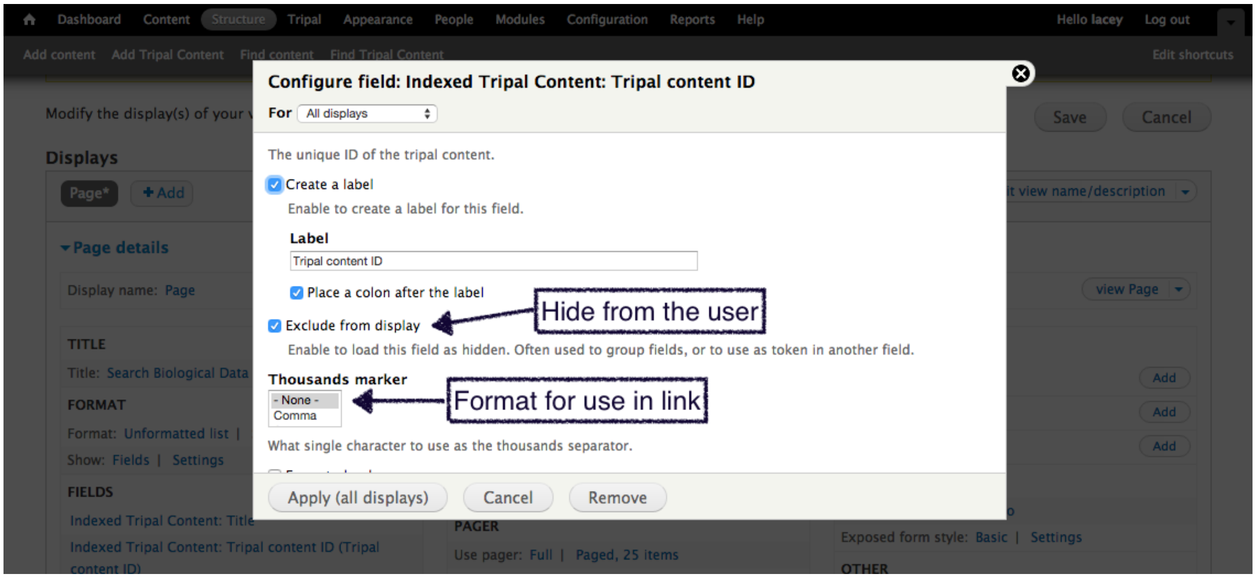
Filters: Where you add your keywords search box.

Note: Make sure to save your view periodically by clicking on the "Save" button at the top of the page.

1.10.2.5 Configuring What is displayed for each Search Result

First, we are going to change what is displayed for each result. By default just the unique identifier is displayed which of course is not useful to the user. We want to hide that field by clicking on its name, **Indexed Tripal Content: Tripal content id** which opens the configuration pop-up and then checking **Exclude from display**. Since we will be using this field to create our link, we also want to change the **Thousands marker** to **-None-**. Click **Apply (all displays)** to save these changes.

Next, click on the **Add** button beside the fields title to open the **Add Fields** pop-up shown in the next screenshot. For this tutorial our search results are going to include the title linked to the content and the highlighted "context" of the search. To add the title, scroll through the fields and click the checkbox beside **Indexed Tripal Content: Title**. Be sure the item description reads **Tripal content "title" property** as there may be other title fields (e.g. for publications). Click **Apply (all displays)** to add this field to the view.



Configure field: Indexed Tripal Content: Tripal content ID

For All displays

The unique ID of the tripal content.

☑ Create a label

Enable to create a label for this field.

Label

Tripal content ID

☑ Place a colon after the label

☑ Exclude from display

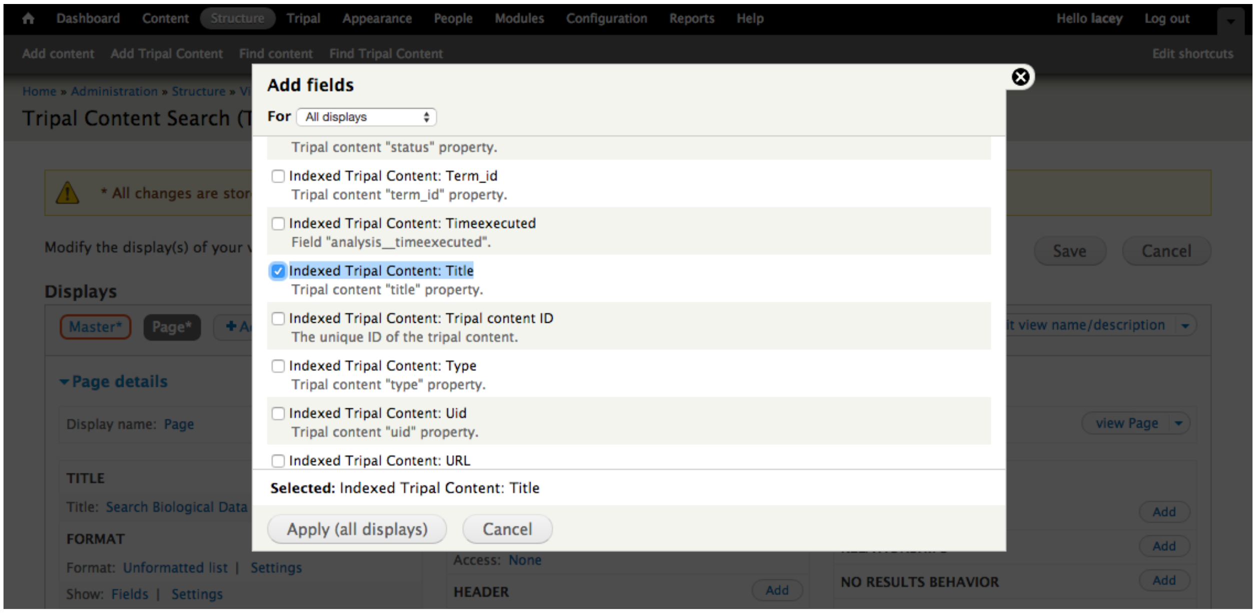
Enable to load this field as hidden. Often used to group fields, or to use as token in another field.

Thousands marker

- None -
Comma

What single character to use as the thousands separator.

Apply (all displays) Cancel Remove



Add fields

For All displays

Tripal content "status" property.

☐ Indexed Tripal Content: Term_id
Tripal content "term_id" property.

☐ Indexed Tripal Content: Timeexecuted
Field "analysis__timeexecuted".

☑ Indexed Tripal Content: Title
Tripal content "title" property.

☐ Indexed Tripal Content: Tripal content ID
The unique ID of the tripal content.

☐ Indexed Tripal Content: Type
Tripal content "type" property.

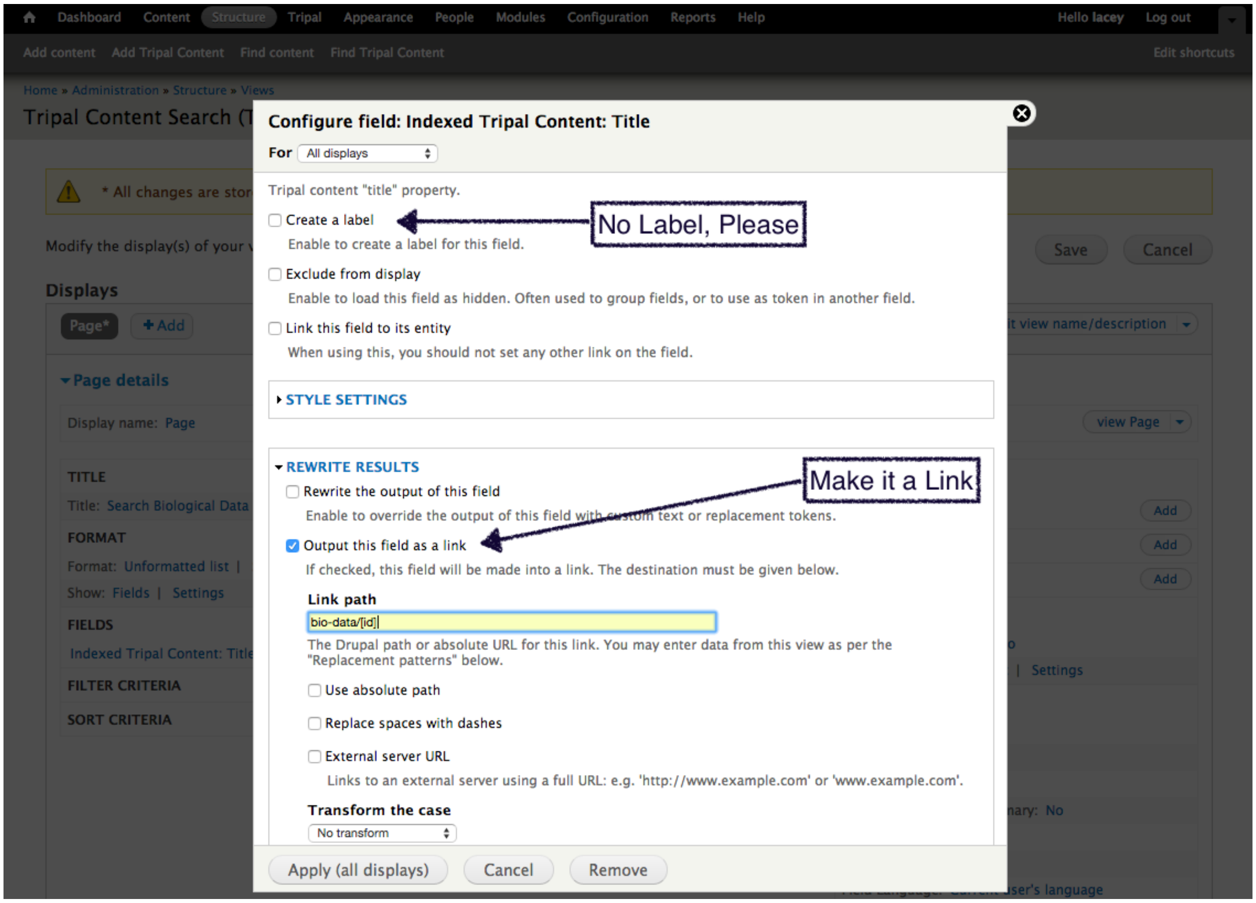
☐ Indexed Tripal Content: Uid
Tripal content "uid" property.

☐ Indexed Tripal Content: URL

Selected: Indexed Tripal Content: Title

Apply (all displays) Cancel

Next, the configuration form for the field is shown. We do not want a label so uncheck the box titled **Create a label**. We want our title to appear as a link to the content, so expand the **Rewrite Results** field set, check **Output this field as a link** and set the link path to `bio_data/[id]`. This uses tokens to fill in the unique identifier and use it to create the path to the entity for each search result. Click the **Apply (all displays)** button to save these settings.



Configure field: Indexed Tripal Content: Title

For: All displays

Tripal content "title" property.

□ Create a label
Enable to create a label for this field.

No Label, Please

□ Exclude from display
Enable to load this field as hidden. Often used to group fields, or to use as token in another field.

□ Link this field to its entity
When using this, you should not set any other link on the field.

▸ STYLE SETTINGS

▾ REWRITE RESULTS

□ Rewrite the output of this field
Enable to override the output of this field with custom text or replacement tokens.

Make it a Link

☑ Output this field as a link
If checked, this field will be made into a link. The destination must be given below.

Link path: bio-data/[id]
The Drupal path or absolute URL for this link. You may enter data from this view as per the "Replacement patterns" below.

□ Use absolute path

□ Replace spaces with dashes

□ External server URL
Links to an external server using a full URL: e.g. 'http://www.example.com' or 'www.example.com'.

Transform the case: No transform

Next, we want to add the highlighted search context. To do this click on the **Add** button again but this time set the **Fields** drop-down to **Search** and check **Search: Excerpt**. Again, click the **Apply (all displays)** button to continue to the configuration pane. On configuration, again, remove the label and apply the settings.

On the resulting page, be sure to uncheck the box **Create** a label just as you did for the Title.

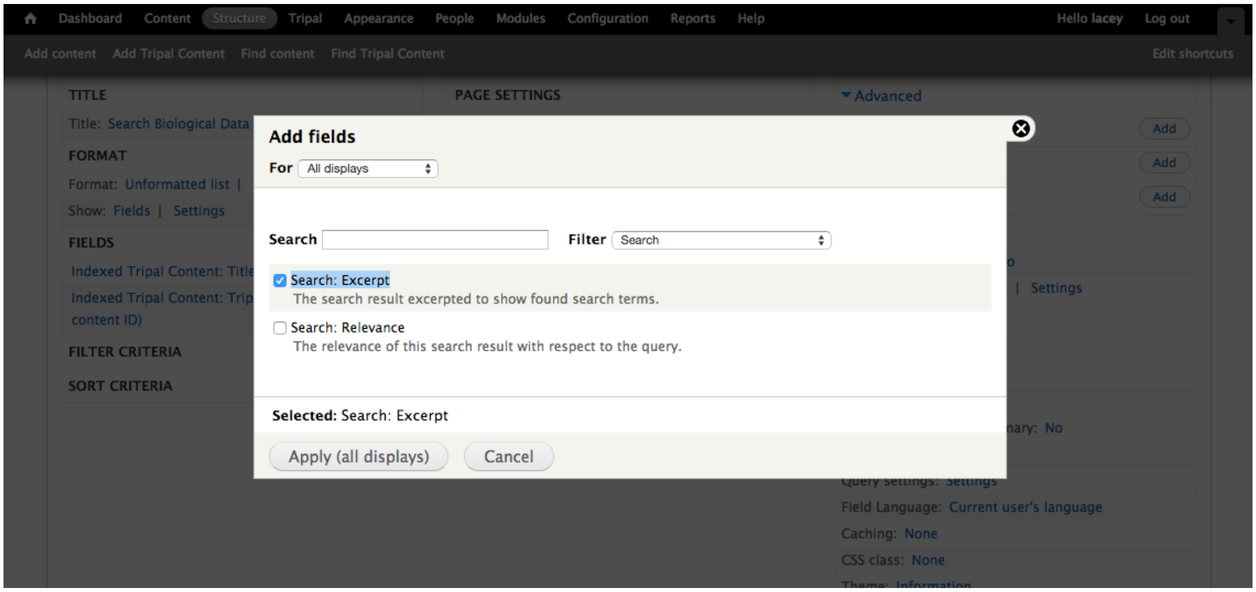
Now that we have a title and excerpt in our Fields section, if you click on the **Update Preview** button you will see a list of titles for your content and then emptiness underneath each title since there was no keyword entered yet so an excerpt could not be generated.

1.10.2.6 Adding the Keywords Search Box

Click on the **Add** button beside **Filter Criteria** and in the resulting pop-up, select **Search** for the filter and then check **Search: Fulltext Search**. Click the **Apply (all displays)** button to add the filter.

In order to let the users see a field for searching, we need to expose this filter. We do that by clicking the checkbox beside **Expose this filter to visitors…** on the filter configuration form. We also want to change the **Label** to **Keywords**. Other then those two changes, the defaults will work so click the **Apply (all displays)**.

Save your view and navigate go to the new search page you created with this new view. The page will be accessible at the URL `http://[your-site-url]/search/biological-data`. You will see a text box titled **Keywords** and if you enter a valid keyword and click **Apply** then filtered results with context highlighting will appear!



Title: Search Biological Data

FORMAT

Format: Unformatted list |

Show: Fields | Settings

FIELDS

Indexed Tripal Content: Title

Indexed Tripal Content: Trip... content ID)

FILTER CRITERIA

SORT CRITERIA

Add fields

For: All displays

Search

Filter: Search

☑ Search: Excerpt

The search result excerpted to show found search terms.

☐ Search: Relevance

The relevance of this search result with respect to the query.

Selected: Search: Excerpt

...nary: No

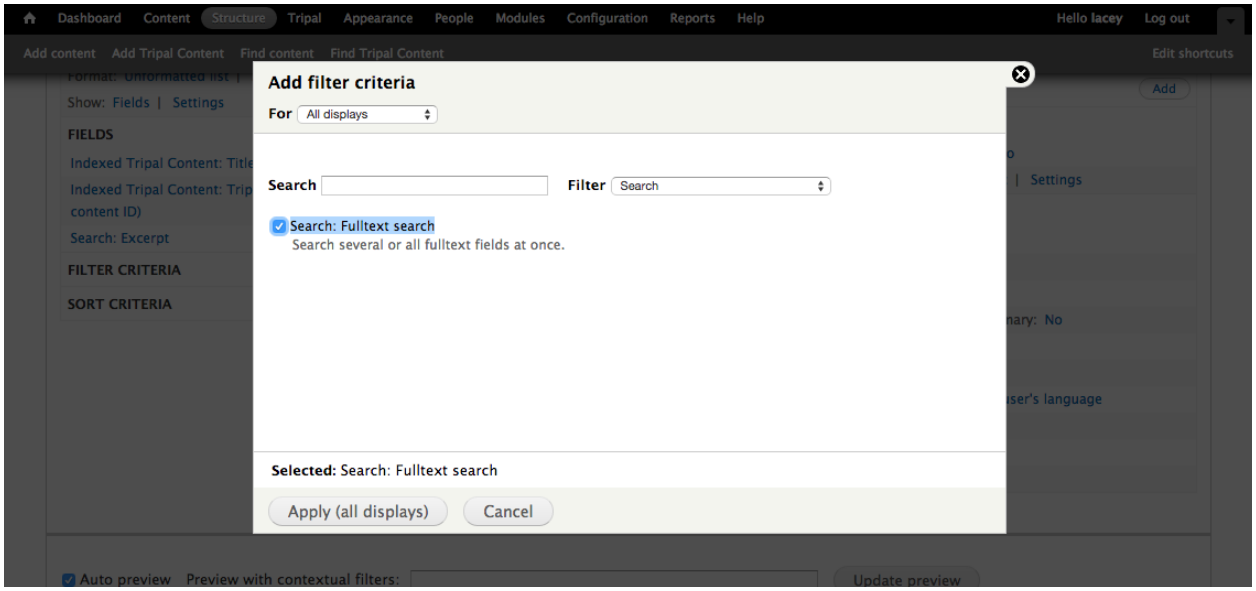
Query settings: Settings

Field Language: Current user's language

Caching: None

CSS class: None

Theme: Information



Format: Unformatted list |

Show: Fields | Settings

FIELDS

Indexed Tripal Content: Title

Indexed Tripal Content: Trip... content ID)

Search: Excerpt

FILTER CRITERIA

SORT CRITERIA

Add filter criteria

For: All displays

Search

Filter: Search

☑ Search: Fulltext search

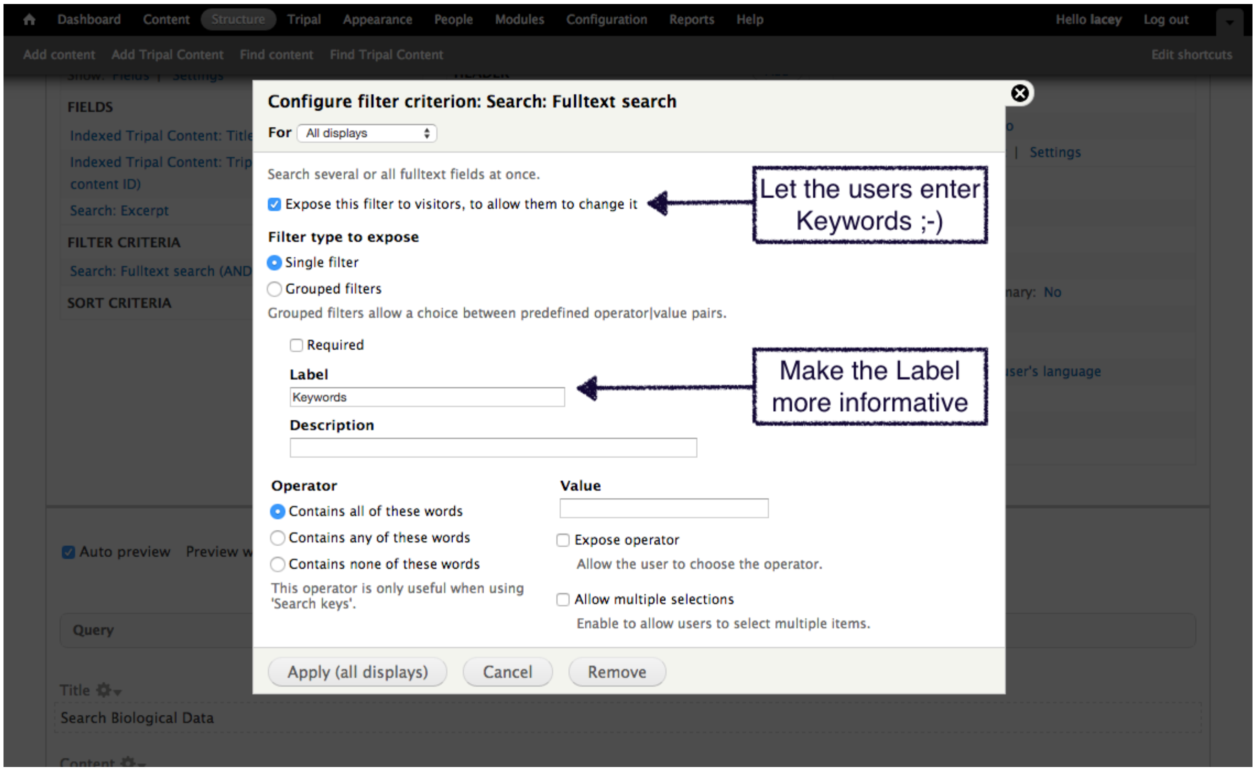
Search several or all fulltext fields at once.

Selected: Search: Fulltext search

...nary: No

...ser's language

☑ Auto preview Preview with contextual filters:



Configure filter criterion: Search: Fulltext search

For: All displays

Search several or all fulltext fields at once.

☑ Expose this filter to visitors, to allow them to change it

Let the users enter Keywords ;-)

Filter type to expose

● Single filter

○ Grouped filters

Grouped filters allow a choice between predefined operator|value pairs.

□ Required

Label

Keywords

Make the Label more informative

Description

Operator

● Contains all of these words

○ Contains any of these words

○ Contains none of these words

This operator is only useful when using 'Search keys'.

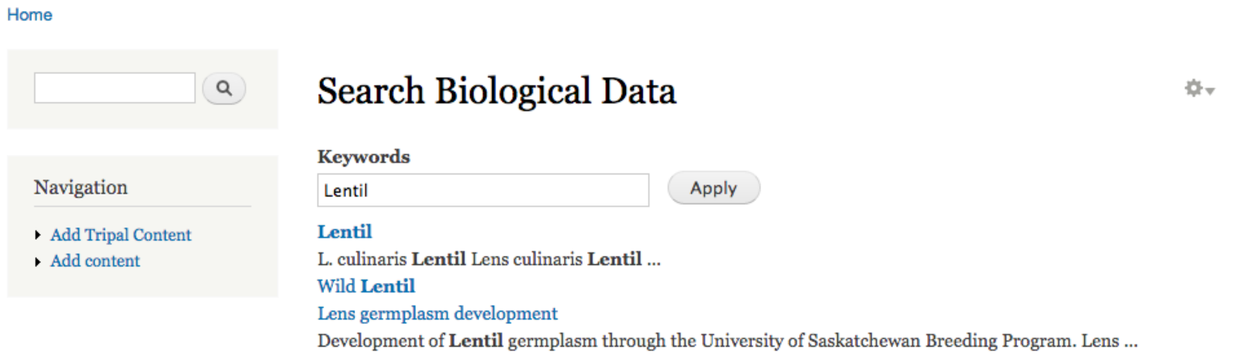
Value

□ Expose operator

Allow the user to choose the operator.

□ Allow multiple selections

Enable to allow users to select multiple items.



Search Biological Data

Keywords

Lentil

Lentil

L. culinaris **Lentil** Lens culinaris **Lentil** ...

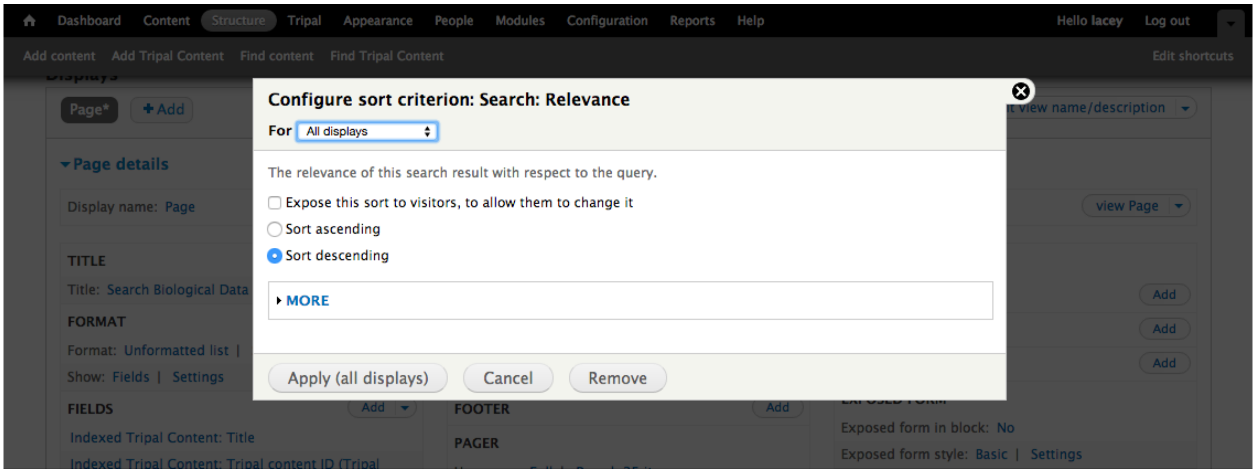
Wild Lentil

Lens germplasm development

Development of **Lentil** germplasm through the University of Saskatchewan Breeding Program. Lens ...

1.10.2.7 Sort by "Relevance"

Next, we want to sort our results. To do this, return to the view configuration page. Click on the **Add** button beside **Sort Criteria** and in the pop-up select **Search** in the **Filter** drop-down. Next, check the **Search: Relevance** field. Apply it and configure it to **Sort descending** so that higher scoring results are shown first. Apply the the configuration settings.



Configure sort criterion: Search: Relevance

For: All displays

The relevance of this search result with respect to the query.

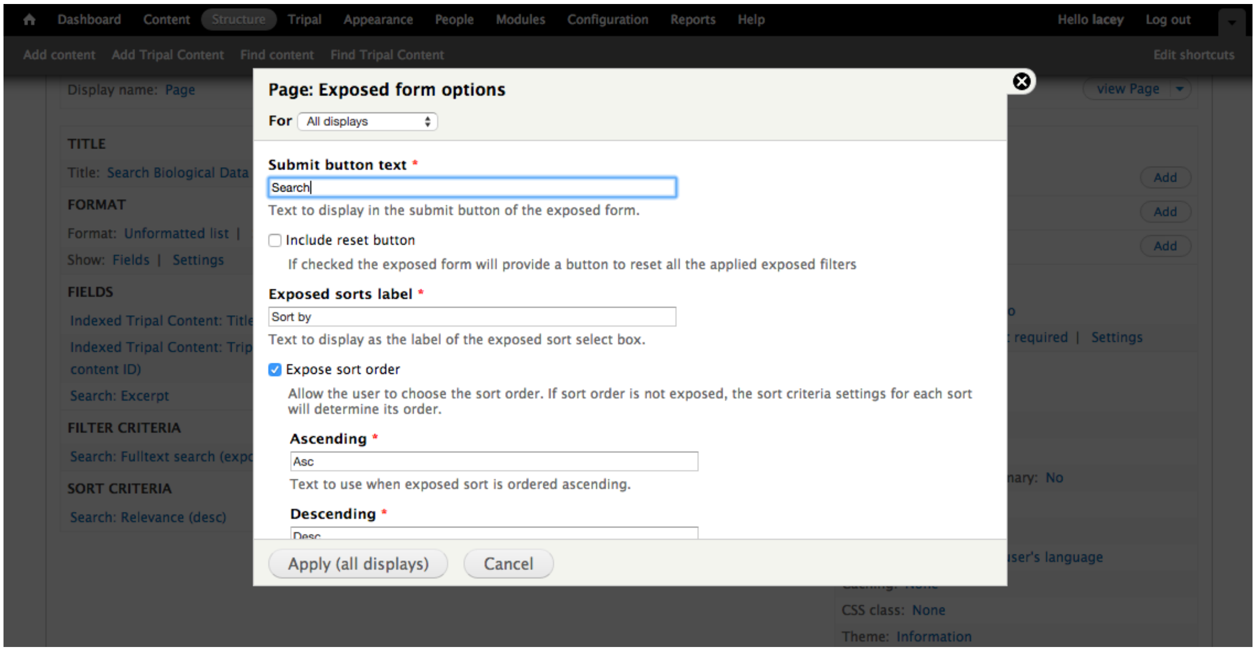
□ Expose this sort to visitors, to allow them to change it

○ Sort ascending

● Sort descending

1.10.2.8 Only Show results when user clicks Search

Finally, we do not want search results to automatically appear. We want the user to click the **Apply** button on the search form first. To do this use the right-side of the Views UI to expand the **Advanced** field set and under **Exposed Form** click on **Exposed form Style: Basic**. Change the setting to **Input Required** and click **Apply**. In the following configuration page change the **Submit button** text to "Search", and uncheck **Expose Sort** order.



Page: Exposed form options

For: All displays

Submit button text *: Search

Text to display in the submit button of the exposed form.

□ Include reset button

If checked the exposed form will provide a button to reset all the applied exposed filters

Exposed sorts label *: Sort by

Text to display as the label of the exposed sort select box.

☑ Expose sort order

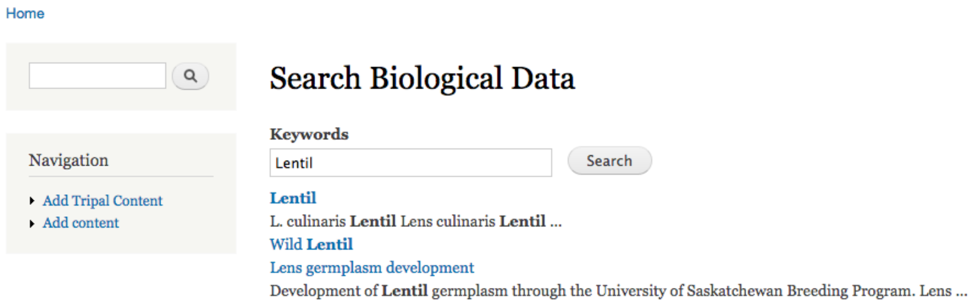
Allow the user to choose the sort order. If sort order is not exposed, the sort criteria settings for each sort will determine its order.

Ascending *: Asc

Text to use when exposed sort is ordered ascending.

Descending *: Desc

Now Save your view -You're Done!



Search Biological Data

Keywords

Lentil

Lentil
L. culinaris **Lentil** Lens culinaris **Lentil** ...
Wild **Lentil**
Lens germplasm development
Development of **Lentil** germplasm through the University of Saskatchewan Breeding Program. Lens ...

1.10.3 Tripal Elasticsearch Module

Tripal Elasticsearch is an extension module created by the Staton Lab at the University of Tennessee that integrates the Elasticsearch search engine with Tripal sites. By enabling the module, you are provided with a set of tools to index and search your Tripal 3 or Tripal 2 site. It provides an easy to set up site wide search and gene search blocks out of the box. Tripal Elasticsearch also provides support for Cross-site querying, which allows users to search multiple websites in a fast and asynchronous manner in one query.

To enable Tripal Elasticsearch on your site, you must install an Elasticsearch instance, enable the Tripal Elasticsearch module, and use it to index your site's data. For documentation on how to install and configure Tripal Elasticsearch, please visit the Github repository.

1.11 Job Management (Tripal Daemon)

Note: Remember you must set the $DRUPAL_HOME environment variable to cut-and-paste the commands below. See see *DRUPAL_HOME Variable*

The Tripal Daemon module is meant to provide a simple means of creating a robust command-line-driven, fully bootstrapped PHP Daemon. It uses the PHP-Daemon (https://github.com/shaneharter/PHP-Daemon) Library to create the Daemon (via the Libraries API) in order to not re-invent the wheel. It allows you to execute Jobs submitted to Tripal without using cron. It provides a faster user experience for running jobs. Prior to Tripal v3, the Tripal Daemon module was an extension module. It was integrated into the core Tripal package.

1.11.1 Features

- Provides a Drush interface to start/stop your Daemon.
- Your daemon starts in the background and is detached from the current terminal.
- Daemon will run all Tripal Jobs submitted within 20 seconds.
- A log including the number of jobs executed, their identifiers and results.
- Lock Files, Automatic restart (8hrs default) and Built-in Signal Handling & Event Logging are only a few of the features provided by the Daemon API making this a fully featured & robust Daemon.

1.11.2 Installation

The Tripal Daemon requires the Libraries API module. You can easily download and install this module using the following drush commands:

```
drush pm-download libraries
drush pm-enable libraries
```

Next, we need the PHP-Daemon Library version 2.0. You must download the PHP-Daemon Library and extract it in your `sites/all/libraries` directory. The folder must be named "PHP-Daemon". The following commands can be used to do this:

```
cd $DRUPAL_HOME/sites/all/libraries
wget https://github.com/shaneharter/PHP-Daemon/archive/v2.0.tar.gz
tar -zxvf v2.0.tar.gz
mv PHP-Daemon-2.0 PHP-Daemon
```

Next, install the Drush Daemon API module.

```
drush pm-download drushd
drush pm-enable drushd
```

Finally, enable the Tripal Daemon module. This module comes with Tripal v3.

```
drush pm-enable tripal_daemon
```

1.11.3 Usage

Start the Daemon

```
drush trpjob-daemon start
```

Stop the Daemon

```
drush trpjob-daemon stop
```

Check the status

```
drush trpjob-daemon status
```

List the last 10 lines of the log file:

```
drush trpjob-daemon show-log
```

List the last N lines of the log file:

```
drush trpjob-daemon show-log --num_lines=N
```

Set N to the number of lines you want to view.

1.12 Web Services

Note: Remember you must set the $DRUPAL_HOME environment variable if you want to cut-and-paste the commands below. See *DRUPAL_HOME Variable*

1.12.1 Overview

New in Tripal v3 are RESTful web services. These web-services are designed to support the following:

1. Allow end-users to access data programmatically using any language of their choice.
2. Allow Tripal sites to share data among themselves.

Tripal v3 now comes with a **tripal_ws** module that provides web services. Once enabled, any Tripal v3 site instantly provides a RESTful web service that can access all publicly available data. Additionally, web services for Tripal are meant to be:

1. Fully discoverable
2. Searchable
3. Provide access to the same data as that which appears on the visible content pages.
4. Adjustable via Drupal's content management interface (i.e. no programming required for existing content)
5. Provide a programmers API for easy addition of new content.
6. Share data the way scientists expect to see it.
7. Use controlled vocabularies to ensure maximal interoperability.

Within the current v3.0 release of Tripal web services are limited to read-only access of already publicly available content. For the future the following is planned for web services:

1. Authenticated data management: creation, deletion and editing of resources via the API.
2. Full implementation of the Hyrda Core Vocabulary specification to provide full discover-ability.

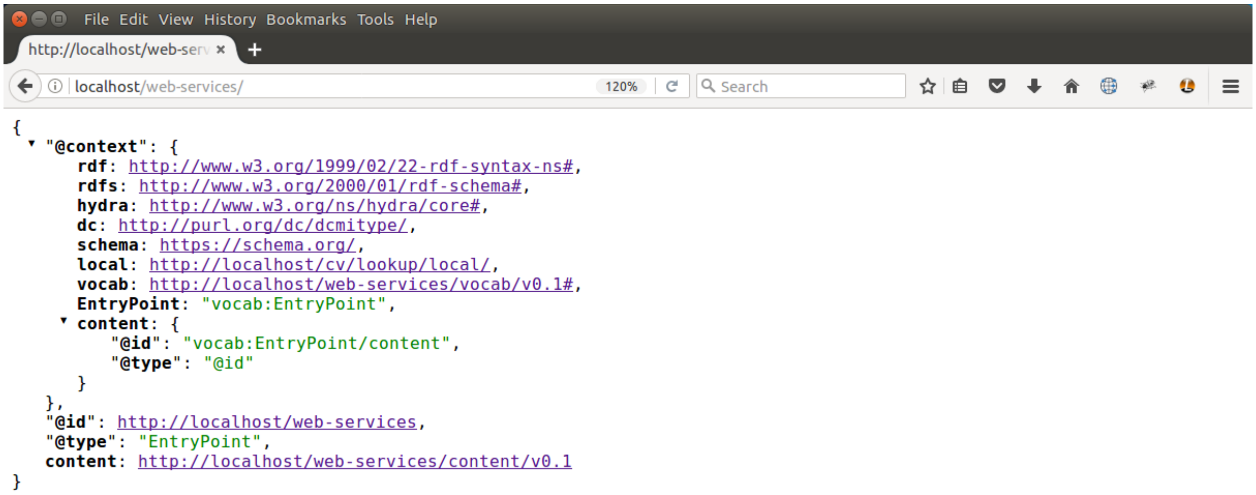
1.12.2 Enable Web Services

To enable web services, simply install the tripal_ws module, either using the module installation interface within Drupal's administrative pages, or on the command-line using Drush:

```
cd $DRUPAL_HOME
drush pm-enable tripal_ws
```

1.12.3 Exploring Web Services

Once enabled, web services are immediately available on your site at the URL http://[your.site.name]/web-services/ (replace [your.site.name] with the address and path of your Tripal site). Web services are meant to be accessed programmatically, but they can be easily explored using a web browser such as with the Firefox browser and JSONView extension enabled. For example, the following screen shot shows an example Tripal site with the data loaded following the Setup of an Example Genomics Site instructions of this User's Guide.


File Edit View History Bookmarks Tools Help
http://localhost/web-serv
localhost/web-services/
120%
Search

```
{
  ▾ "@context": {
      rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#,
      rdfs: http://www.w3.org/2000/01/rdf-schema#,
      hydra: http://www.w3.org/ns/hydra/core#,
      dc: http://purl.org/dc/dcmitype/,
      schema: https://schema.org/,
      local: http://localhost/cv/lookup/local/,
      vocab: http://localhost/web-services/vocab/v0.1#,
      EntryPoint: "vocab:EntryPoint",
    ▾ content: {
          "@id": "vocab:EntryPoint/content",
          "@type": "@id"
      }
  },
  "@id": http://localhost/web-services,
  "@type": "EntryPoint",
  content: http://localhost/web-services/content/v0.1
}
```

This initial resource "home page" of the web services returns results in JSON format. When using the JSONView extension within Firefox you can explore web services by clicking the links that are present in the browser.

1.12.4 Structure of a Web Service Response

The initial response in JSON is in the JSON-LD format where the LD means Linked Data. For example:

```
  {
  "@context": {,
    "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "hydra": "http://www.w3.org/ns/hydra/core#",
    "dc": "http://purl.org/dc/dcmitype/",
    "schema": "https://schema.org/",
    "local": "http://localhost/cv/lookup/local/",
    "vocab": "http://localhost/web-services/vocab/v0.1#",
    "EntryPoint": "vocab:EntryPoint",
    "content": {
      "@id": "vocab:EntryPoint/content",
      "@type": "@id"
    }
  }
  "@id": "http://localhost/web-services",
  "@type": "EntryPoint",
  "content": "http://localhost/web-services/content/v0.1"
}
```

A notable component of JSON-LD is the **@context** sub array. It is within this **@context** section that the "meaning" of the data becomes apparent and where the Linked Data is implemented. Tripal's use of linked data is to take advantage of controlled vocabularies or otologies to unambiguously identify the meaning of each element of the response. By using controlled vocabularies to qualify all data, it becomes possible to exchange data between sites or within a client program while limiting confusion as to the meaning of the data. Therefore, words that are used as the keys key/value pairs will always be defined in the @context section. For example, all of the vocabularies whose terms are used to qualify data on the site are identified in the **@context** section with URLs. Some of these vocabularies include: rdf, rdfs, hydra, schema, etc.

There are two other special keys used in the JSON-LD response. These are the **@id** and **@type** keys. The **@id** indicates the unique URL for this resource and becomes a unique name for the resource. In the example screenshot above, the **@id** of the initial page of web services is `http://localhost/web-services`. This URL will always refer to the initial page for Tripal web services on the given site. The **@type** identifier specifies what type of resource this URL provides. In this example, the type is **EntryPoint**. If a client program is unsure as to what an **EntryPoint** is, then that information is provided in the @context section. The following line indicates that the term **EntryPoint** expands to the vocabulary term: vocab:EntryPoint

```
"EntryPoint": "vocab:EntryPoint",
```

Here we learn tha the term **EntryPoint** belongs to the vocab resource. If we look at the vocab entry in the **@context** array then we can find a link to that resource. A human can follow that link to examine the vocab resource and find the term that it provides. A client program can use that information to uniquely identify the term. By default, Tripal provides the **vocab** vocabulary which uses an implementation of the Hydra Vocabulary.

Finally, all other entries in the JSON-LD response are key/value pairs that provide **properties** about the resource. In the example above, only the **content** property is available. A property can be a scalar value (i.e. number, character string) or a link to another resource.

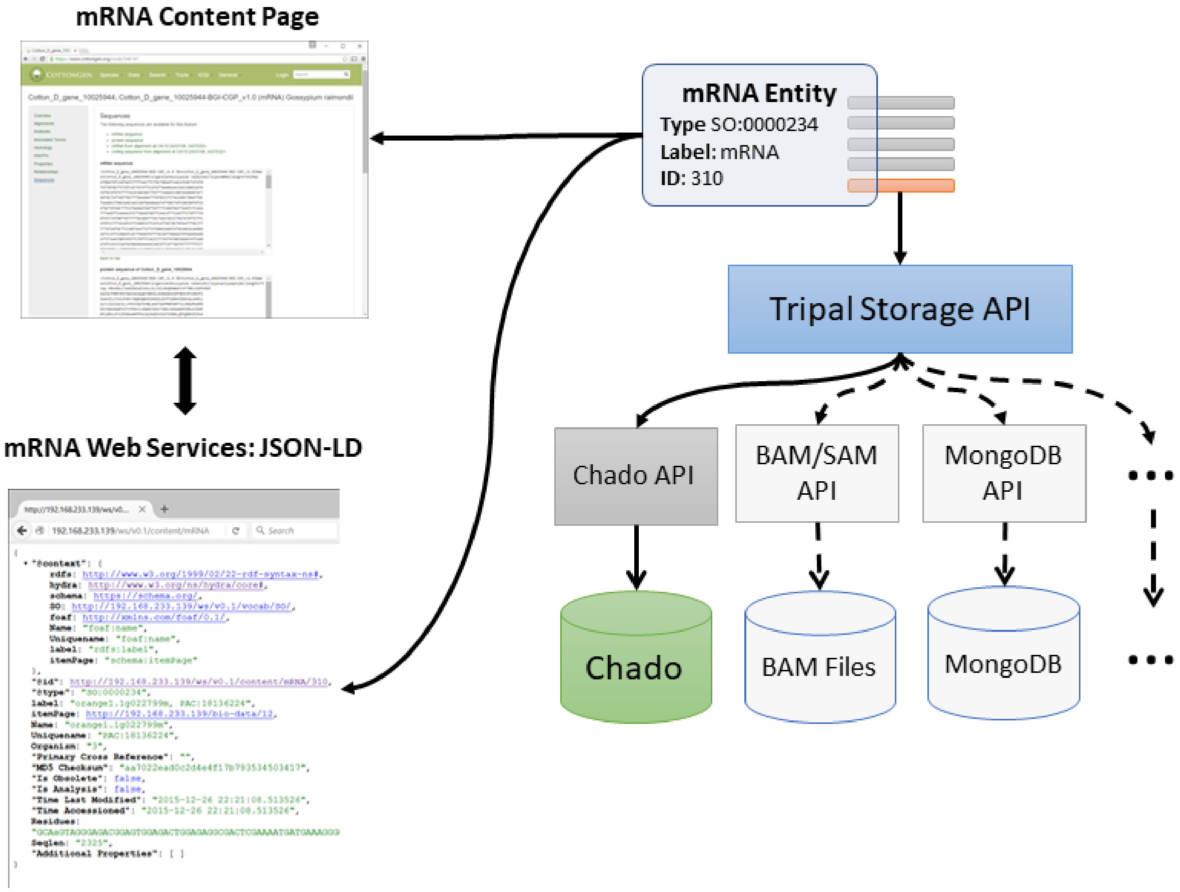
1.12.5 Primary Services

By default, the only resource that Tripal provides at the initial response level is the content resource. Any resource at this level is hereafter referred to as primary service. Tripal is design to allow new web-services to be added to it. These will be more formally described in the Tripal v3 Developer's Handbook. In short, a primary service provides a variety of data and services for related content and function. Each primary resource has a version number to help ensure backwards compatibility as new web services are developed and updated. For example, the default content service currently has a version of v0.1:

```
"content": "http://localhost/web-services/content/v0.1"
```

1.12.6 The Content Service

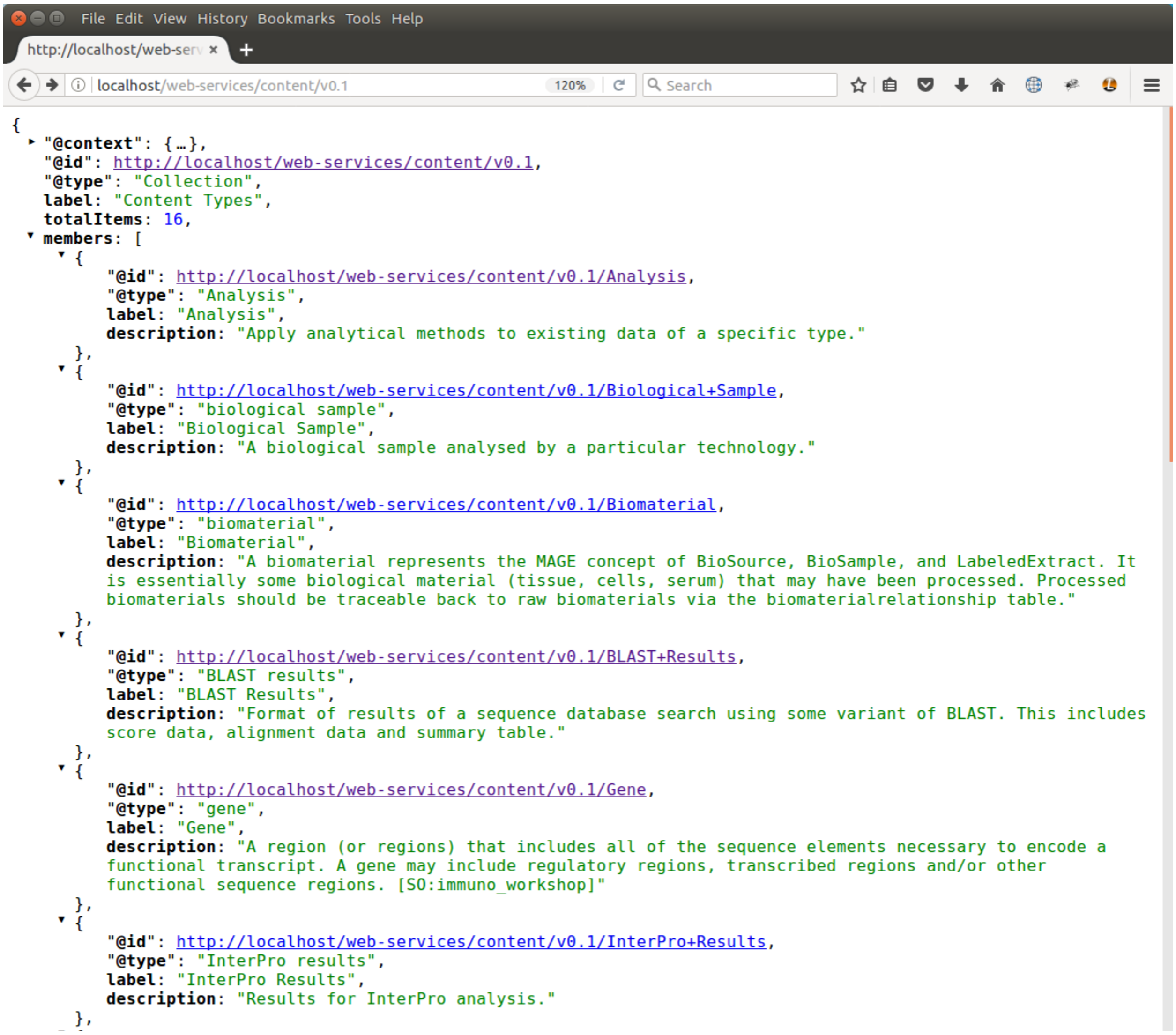
The content service provided by Tripal shares all publicly available content. The content that appears on a page is the same content that appears in web services. A major change in the design of Tripal from v2 to v3 is that all data is organized via controlled vocabularies. The following diagram demonstrates how this is implemented. For example the mRNA term comes from the Sequence Ontology. It's unique term accession is SO:0000234. Every content type in Tripal consists solely of a type (e.g. mRNA or SO:0000234), it's associated label (e.g. mRNA) and a numeric id unique to each Tripal site. The ID is what uniquely identifies every content in Tripal. Each unique content with these three attributes is referred to as an **Entity**. All other data associated with a given entity are called **Fields**. Example fields for an mRNA content type may be the name of the mRNA, a unique name, the coding sequence, the coordinates on the reference genome, etc. In the diagram below, these fields are the rectangular boxes that jut out of the mRNA entity. These fields can be "attached" to an entity by Tripal and data can come from any storage backend. The data that appears on a page and the data in the content service is taken from the same entity and therefore end-users and clients have access to the same data.



```
{
  ▾ "@context": {
        rdfs: http://www.w3.org/1999/02/22-rdf-syntax-ns#,
        hydra: http://www.w3.org/ns/hydra/core#,
        schema: https://schema.org/,
        SO: http://192.168.233.139/ws/v0.1/vocab/SO/,
        foaf: http://xmlns.com/foaf/0.1/,
        Name: "foaf:name",
        Uniquename: "foaf:name",
        label: "rdfs:label",
        itemPage: "schema:itemPage"
    },
    "@id": http://192.168.233.139/ws/v0.1/content/mRNA/310,
    "@type": "SO:0000234",
    label: "orange1.1g022799m, PAC:18136224",
    itemPage: http://192.168.233.139/bio-data/12,
    Name: "orange1.1g022799m",
    Uniquename: "PAC:18136224",
    Organism: "3",
    "Primary Cross Reference": "",
    "MD5 Checksum": "aa7022ead0c2d4e4f17b793534503417",
    "Is Obsolete": false,
    "Is Analysis": false,
    "Time Last Modified": "2015-12-26 22:21:08.513526",
    "Time Accessioned": "2015-12-26 22:21:08.513526",
    Residues:
    "GCAaGTAGGGAGACGGAGTGGAGACTGGAGAGGCGACTCGAAAATGATGAAAGGG
    Seqlen: "2325",
    "Additional Properties": [ ]
}
```

1.12.7 Content Type Listing

When the content service is accessed, the response is always a listing of all available content types on the site. Site administrators can create new content types by following the Create Content Types section of this tutorial. By default, Tripal is installed with several commonly used content types, but new ones can be created as needed for the site. Whenever a new content type is created it immediately is available via the content service, and these content types can be found at the path: `/web-services/content/v0.1`. Below is an example screenshot of the resulting JSON from an example site:

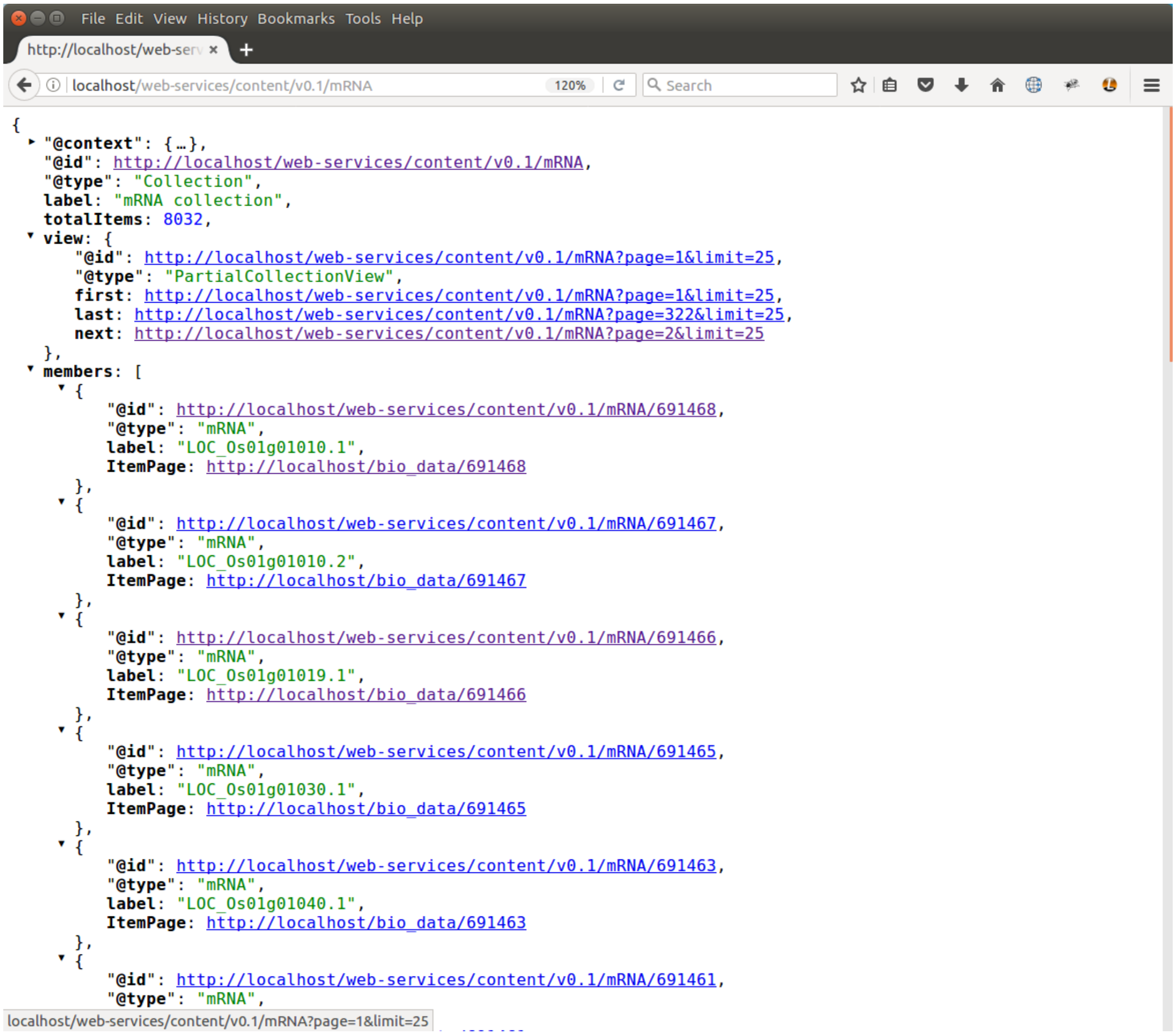

File Edit View History Bookmarks Tools Help
http://localhost/web-serv ×
localhost/web-services/content/v0.1
120%
Search

```
{
 ▸ "@context": {…},
   "@id": http://localhost/web-services/content/v0.1,
   "@type": "Collection",
   label: "Content Types",
   totalItems: 16,
 ▾ members: [
    ▾ {
         "@id": http://localhost/web-services/content/v0.1/Analysis,
         "@type": "Analysis",
         label: "Analysis",
         description: "Apply analytical methods to existing data of a specific type."
      },
    ▾ {
         "@id": http://localhost/web-services/content/v0.1/Biological+Sample,
         "@type": "biological sample",
         label: "Biological Sample",
         description: "A biological sample analysed by a particular technology."
      },
    ▾ {
         "@id": http://localhost/web-services/content/v0.1/Biomaterial,
         "@type": "biomaterial",
         label: "Biomaterial",
         description: "A biomaterial represents the MAGE concept of BioSource, BioSample, and LabeledExtract. It
         is essentially some biological material (tissue, cells, serum) that may have been processed. Processed
         biomaterials should be traceable back to raw biomaterials via the biomaterialrelationship table."
      },
    ▾ {
         "@id": http://localhost/web-services/content/v0.1/BLAST+Results,
         "@type": "BLAST results",
         label: "BLAST Results",
         description: "Format of results of a sequence database search using some variant of BLAST. This includes
         score data, alignment data and summary table."
      },
    ▾ {
         "@id": http://localhost/web-services/content/v0.1/Gene,
         "@type": "gene",
         label: "Gene",
         description: "A region (or regions) that includes all of the sequence elements necessary to encode a
         functional transcript. A gene may include regulatory regions, transcribed regions and/or other
         functional sequence regions. [SO:immuno_workshop]"
      },
    ▾ {
         "@id": http://localhost/web-services/content/v0.1/InterPro+Results,
         "@type": "InterPro results",
         label: "InterPro Results",
         description: "Results for InterPro analysis."
      },
```

Note that the **@type** for this listing is a Collection and the label is **Content Types**. Each content type has a unique **@id**, a **@type** indicating the term that describes it and a brief description. The **@id** serves as a URL to obtain further details about that content type. Also, notice in the above screenshot that the **@context** section is minimized, but as usual, each of the terms used in the key/value pairs are fully qualified in that section. This JSON-LD response also indicates the total number of content types available.

1.12.8 Content Type Members (Entities)

The members or entities that belong to a content type are found at the path: `/web-services/content/v0.1/{name}` where {name} is the name of the content type. The {name} field must be identical to the label field from the content type listing shown previously. For example, the mRNA content type path would be `/web-services/content/v0.1/mRNA`. This resource provides a listing of all members for that content type. The following shows the response for an mRNA listing:


File Edit View History Bookmarks Tools Help
http://localhost/web-serv ×
localhost/web-services/content/v0.1/mRNA
120%
Search

```
{
  ▸ "@context": {…},
    "@id": http://localhost/web-services/content/v0.1/mRNA,
    "@type": "Collection",
    label: "mRNA collection",
    totalItems: 8032,
  ▾ view: {
        "@id": http://localhost/web-services/content/v0.1/mRNA?page=1&limit=25,
        "@type": "PartialCollectionView",
        first: http://localhost/web-services/content/v0.1/mRNA?page=1&limit=25,
        last: http://localhost/web-services/content/v0.1/mRNA?page=322&limit=25,
        next: http://localhost/web-services/content/v0.1/mRNA?page=2&limit=25
    },
  ▾ members: [
      ▾ {
            "@id": http://localhost/web-services/content/v0.1/mRNA/691468,
            "@type": "mRNA",
            label: "LOC_Os01g01010.1",
            ItemPage: http://localhost/bio_data/691468
        },
      ▾ {
            "@id": http://localhost/web-services/content/v0.1/mRNA/691467,
            "@type": "mRNA",
            label: "LOC_Os01g01010.2",
            ItemPage: http://localhost/bio_data/691467
        },
      ▾ {
            "@id": http://localhost/web-services/content/v0.1/mRNA/691466,
            "@type": "mRNA",
            label: "LOC_Os01g01019.1",
            ItemPage: http://localhost/bio_data/691466
        },
      ▾ {
            "@id": http://localhost/web-services/content/v0.1/mRNA/691465,
            "@type": "mRNA",
            label: "LOC_Os01g01030.1",
            ItemPage: http://localhost/bio_data/691465
        },
      ▾ {
            "@id": http://localhost/web-services/content/v0.1/mRNA/691463,
            "@type": "mRNA",
            label: "LOC_Os01g01040.1",
            ItemPage: http://localhost/bio_data/691463
        },
      ▾ {
            "@id": http://localhost/web-services/content/v0.1/mRNA/691461,
            "@type": "mRNA",
```

localhost/web-services/content/v0.1/mRNA?page=1&limit=25

Note that the **@type** is also a Collection byt the label is 'mRNA collection'. To maintain a small response, the results of content member listings is usually paged such that only a subset of members is shown. In this example, there are 8032 mRNA entities available, but only 25 are shown. Notice the view term. It contains several sub elements named first, last and next. These provide navigation links that can be used by a client application to iterate through all entities. The structure of these links is as follows:

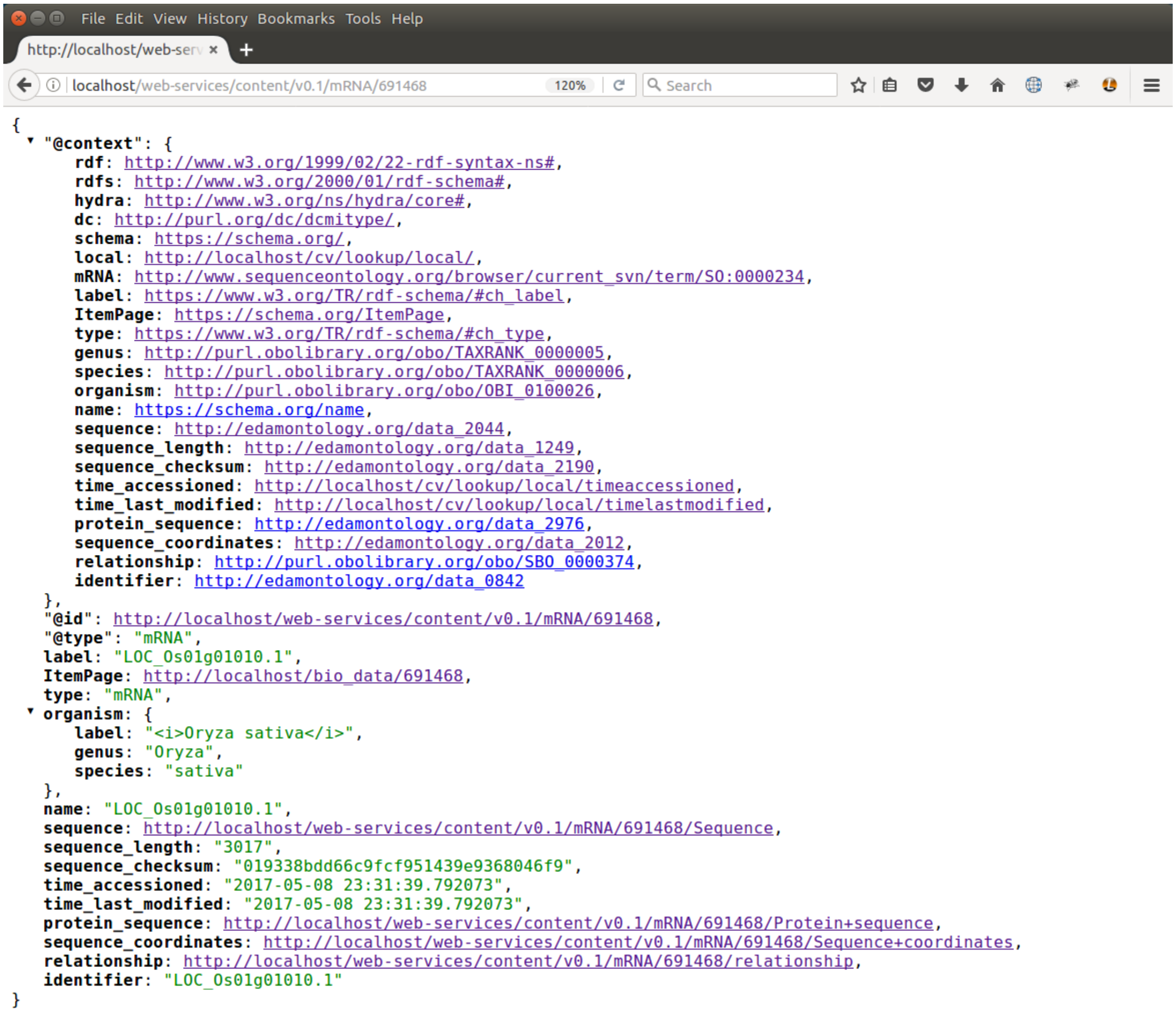
`/web-servies/content/v0.1/{name}?page={page}&limit={limit}`

A client application can therefore navigate through the list of entities by substituting the {name} of the content type, the desired {page} to show (the first page is always 1) and the number of records to show as specified by {limit}. If a client wanted to retrieve the IDs of all 8032 mRNA of this example, then the following path could be used:

`/web-servies/content/v0.1/mRNA?page=1&limit=8032`

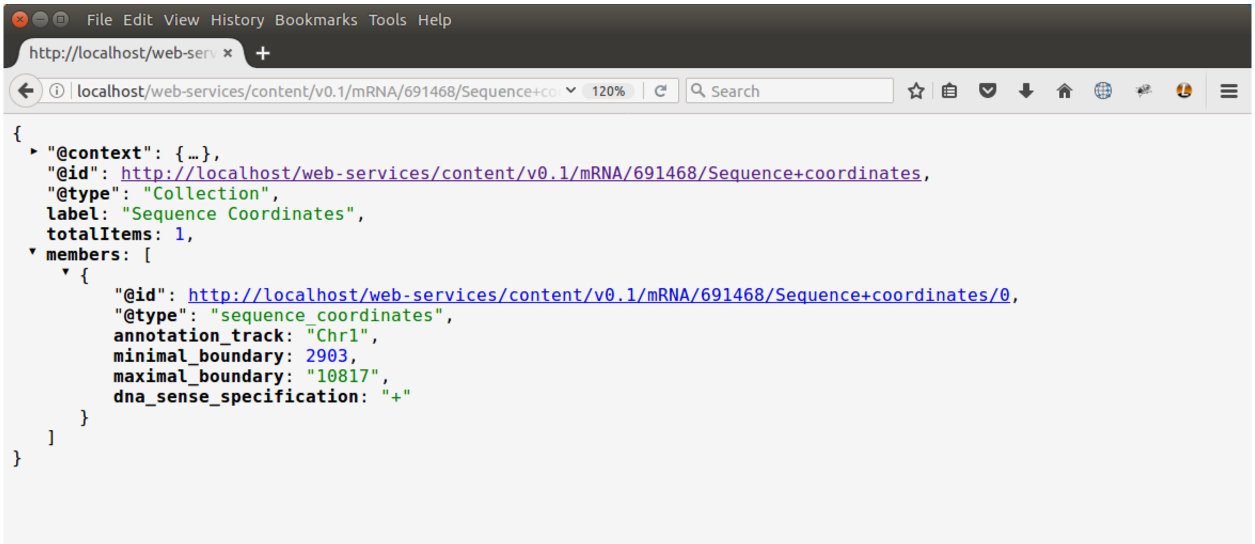
1.12.9 Content (Entity) Resource

Each entity is accessible via the path: `/web-services/content/v0.1/{name}/{id}`. Here {name} continues to refer to the name of the content type (e.g. mRNA) and {id} refers to the unique numeric ID for the entity. In this example an mRNA entity would result in a JSON-LD response like the following:


File Edit View History Bookmarks Tools Help
http://localhost/web-serv ×
localhost/web-services/content/v0.1/mRNA/691468
120%
Search

```
{
 ▾ "@context": {
       rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#,
       rdfs: http://www.w3.org/2000/01/rdf-schema#,
       hydra: http://www.w3.org/ns/hydra/core#,
       dc: http://purl.org/dc/dcmitype/,
       schema: https://schema.org/,
       local: http://localhost/cv/lookup/local/,
       mRNA: http://www.sequenceontology.org/browser/current_svn/term/SO:0000234,
       label: https://www.w3.org/TR/rdf-schema/#ch_label,
       ItemPage: https://schema.org/ItemPage,
       type: https://www.w3.org/TR/rdf-schema/#ch_type,
       genus: http://purl.obolibrary.org/obo/TAXRANK_0000005,
       species: http://purl.obolibrary.org/obo/TAXRANK_0000006,
       organism: http://purl.obolibrary.org/obo/OBI_0100026,
       name: https://schema.org/name,
       sequence: http://edamontology.org/data_2044,
       sequence_length: http://edamontology.org/data_1249,
       sequence_checksum: http://edamontology.org/data_2190,
       time_accessioned: http://localhost/cv/lookup/local/timeaccessioned,
       time_last_modified: http://localhost/cv/lookup/local/timelastmodified,
       protein_sequence: http://edamontology.org/data_2976,
       sequence_coordinates: http://edamontology.org/data_2012,
       relationship: http://purl.obolibrary.org/obo/SBO_0000374,
       identifier: http://edamontology.org/data_0842
   },
   "@id": http://localhost/web-services/content/v0.1/mRNA/691468,
   "@type": "mRNA",
   label: "LOC_Os01g01010.1",
   ItemPage: http://localhost/bio_data/691468,
   type: "mRNA",
 ▾ organism: {
       label: "<i>Oryza sativa</i>",
       genus: "Oryza",
       species: "sativa"
   },
   name: "LOC_Os01g01010.1",
   sequence: http://localhost/web-services/content/v0.1/mRNA/691468/Sequence,
   sequence_length: "3017",
   sequence_checksum: "019338bdd66c9fcf951439e9368046f9",
   time_accessioned: "2017-05-08 23:31:39.792073",
   time_last_modified: "2017-05-08 23:31:39.792073",
   protein_sequence: http://localhost/web-services/content/v0.1/mRNA/691468/Protein+sequence,
   sequence_coordinates: http://localhost/web-services/content/v0.1/mRNA/691468/Sequence+coordinates,
   relationship: http://localhost/web-services/content/v0.1/mRNA/691468/relationship,
   identifier: "LOC_Os01g01010.1"
}
```

In the JSON shown above, note that all of the key/value pairs used are referenced in the **@context** section Also, notice that some key/value pairs contain data while others contain URLs. Tripal is optimized to not load every attribute. For example, sequence data for some content type may be large. By providing a URL for the data, it keeps the response small but allows clients to access that information via the provided URL. For example, if the URL for the **sequence_coordinate** attribute were followed the following response could be seen:


File Edit View History Bookmarks Tools Help
http://localhost/web-serv ×
localhost/web-services/content/v0.1/mRNA/691468/Sequence+co
120%
Search

```
{
  ▸ "@context": {…},
    "@id": http://localhost/web-services/content/v0.1/mRNA/691468/Sequence+coordinates,
    "@type": "Collection",
    label: "Sequence Coordinates",
    totalItems: 1,
  ▾ members: [
      ▾ {
            "@id": http://localhost/web-services/content/v0.1/mRNA/691468/Sequence+coordinates/0,
            "@type": "sequence_coordinates",
            annotation_track: "Chr1",
            minimal_boundary: 2903,
            maximal_boundary: "10817",
            dna_sense_specification: "+"
        }
    ]
}
```

Here the client can obtain the necessary information about the coordinates on the genome for this particular mRNA entity.

1.12.10 Searching for Content

Currently, Tripal provides the ability to search for content via web services by crafting URLs on content type members pages. By default, the path for content type listings is:

`/web-services/content/v0.1/{name}`

Where {name} is the label assigned to the content type (See the Content Type Members section above). Using this path, clients filter content to a specific content type. But further refinement is possible. As a reminder, each member (or entity) on the content type members collection appears similar to the following:

```
{
  "@id": "http://localhost/web-services/content/v0.1/mRNA/691468",
  "@type": "mRNA",
  "label": "LOC_Os01g01010.1",
  "ItemPage": "http://localhost/bio_data/691468"
},
```

When retrieving the data for a specific entity something similar to the following (for our mRNA example) may be seen:

```
"label": "LOC_Os01g01010.1",
"ItemPage": "http://localhost/bio_data/691468",
"type": "mRNA",
"organism": {
    "label": "<i>Oryza sativa</i>",
    "genus": "Oryza",
    "species": "sativa"
},
"name": "LOC_Os01g01010.1",
"sequence": "http://localhost/web-services/content/v0.1/mRNA/691468/Sequence",
"sequence_length": "3017",
"sequence_checksum": "019338bdd66c9fcf951439e9368046f9",
"time_accessioned": "2017-05-08 23:31:39.792073",
"time_last_modified": "2017-05-08 23:31:39.792073",
```

(continues on next page)

(continued from previous page)

```
"protein_sequence": "http://localhost/web-services/content/v0.1/mRNA/691468/
↪Protein+sequence",
"sequence_coordinates": "http://localhost/web-services/content/v0.1/mRNA/691468/
↪Sequence+coordinates",
"relationship": "http://localhost/web-services/content/v0.1/mRNA/691468/relationship",
"identifier": "LOC_Os01g01010.1"
```

As another reminder, when any of these attributes have a URL then further information about that attribute is obtained by following the URL. In the example below, the relationship term yields results similar to the following:

```
{
    "@id": "http://localhost/web-services/content/v0.1/mRNA/691468/relationship/0",
    "@type": "relationship",
    "clause_subject": {
        "type": "mRNA",
        "name": "LOC_Os01g01010.1",
        "identifier": "LOC_Os01g01010.1"
    },
    "relationship_type": "part_of",
    "clause_predicate": {
        "type": "gene",
        "name": "LOC_Os01g01010",
        "identifier": "LOC_Os01g01010"
    },
    "clause": "The mRNA, LOC_Os01g01010.1, is a part of the gene, LOC_Os01g01010."
},
```

Here we see information that describes the relationship of the mRNA with its parent gene. Almost all of the key value pairs shown in the responses above can be used to filter results. But, attention must be paid as to the level that each attribute appears. For example, in the initial entity response above, the organism attribute has several sub terms that include genus, species and label. The organism term appears as a first-level term and genus, species and label appear as a second-level term. For relationships, the relationship is the first-level term but that term has a URL! Tripal does not support filter by URLs. However, we can use the terms from the results of that URL in our filter. Thus, the clause_subject, relationship_type and clause_predicate becomes a second-level terms, and within the clause_subject, the type, name and identifier become third-level terms.

You can easily search for specific entities by knowing these first, second, third, etc. -level terms. The path for searching is as follows:

`/web-services/content/v0.1/{name}?{first-level}[,{second-level},...,{n-th level}]={value}[;{operator}]`

Here, {name} again refers to the content type name (e.g. mRNA). The {first-level} placeholder refers to any term that appears at the first level. Refinement can occur if a term has multiple sublevels by separating those terms with a comma. The {value} placeholder should contain the search word. The {operator} placeholder lets you specify the operator to use (e.g. greater than, less than, starts with, etc.). The {operator} is optional and if not included all searches default to exact matching. As an example, the organism term has sever second-level terms. If we wanted to filter all mRNA to include only those from the genus Oryza we could construct the following URL:

`/web-services/content/v0.1/mRNA?organism,genus=Oryza`

Multiple search criteria can be provided at one time, by repeating the search construct as many times as needed and separating with an ampersand character: &. For example, to filter the mRNA to only those from Oryza sativa the following URL would be constructed:

`/web-services/content/v0.1/mRNA?organism,genus=Oryza&organism,species=sativa`

The examples provided thus far expect that you are searching for exact values. However, you can specify different search operators such as the following:

- **Numeric Values**
  - equals: eq
  - greater than: gt
  - greater than or equal to: gte
  - less than: lt
  - less than or equal to: lte
  - not equal to: ne
- **Text values**
  - equals: eq
  - contains: contains
  - starts with: starts

Following the path format specified above we set the operator. For example, We can use the label as our second-level search term and require that it start with Oryza to find all of the mRNA that belong to the genus Oryza:

```
/web-services/content/v0.1/mRNA?organism,label=Oryza;starts
```

Finally, you can control ordering of the results by constructing a PATH following this format:

```
/web-services/content/{name}?{search filter}&order={term}|{dir}[;{term}|{dir}.
..]
```

Here {search filter} represents the filtering criteria as defined above (keeps the path format from getting extremely long in this document), {term} represents the full term "path" which if it has multi-level terms those levels are separated by a comma (e.g. organism,genus); and {dir} represents the direction of the order and can be **ASC** or **DESC** for ascending and descending sorting respectively. You may order results by as many terms as needed by separating them with a semicolon. The order in which the terms are provided will dictate which term gets sorted first. For example, supposed we want to search for all mRNA within the genus Oryza but order them by the species name. The following URL construct would suffice:

```
/web-services/content/mRNA?organism,genus=Oryza&order=organism,species|ASC
```

To demonstrate multi term sorting, we could try to sort by the Genus as well, although, because we filtered by the genus that would be a useless sort, but it demonstrates use of multiple sort criteria:

```
/web-services/content/mRNA?organism,genus=Oryza&order=organism,genus|ASC;
organism,species|ASC
```

1.12.11 Searching Limitations

The ability to search by any term for any content type creates a powerful mechanism to find almost entity. However there are two primary limitations:

1. Not all fields attached to an entity are conducive for searching. Images can be attached to entities, references to other websites, etc. In these cases the search functionality for those fields has not been implemented. Unfortunately, Tripal v3 does not yet provide a listing of which fields are not searchable. That support will be coming in the future.
2. The format for constructing a search URL is equivalent to an AND operation. For example, you can filter by genus and species but not by genus or species. The addition of search criteria adds additional AND filters.

1.12.12 Hiding or Adding Content

It is relatively easy to hide or add content to web services. The 'Configuring Page Display' tutorial walks the user through the steps for adding fields to a page for display, removing them and organizing the placement of those fields on the entity's page. That same interface is used for indicating which fields are present in web services. When a field is hidden from a page it is likewise hidden from web services. When a new field is added to a page it is added to web services. Folks who develop new fields for custom modules and share those with others should ensure that their fields implementations follow the design specifications. If followed correctly then all fields will behave in this way.

1.13 Bulk Loader

Note: Remember you must set the `$DRUPAL_HOME` environment variable if you want to cut-and-paste the commands below. See *DRUPAL_HOME Variable*

The bulk loader is a tool that Tripal provides for loading of data contained in tab delimited files. Tripal supports loading of files in standard formats (e.g. `FASTA`, `GFF`, `OBO`), but Chado can support a variety of different biological data types and there are often no community standard file formats for loading these data. For example, there is no file format for importing genotype and phenotype data. Those data can be stored in the feature, stock and natural diversity tables of Chado. The Bulk Loader was introduced in Tripal v1.1 and provides a web interface for building custom data loader. In short, the site developer creates the bulk loader "template". This template can then be used and re-used for any tab delimited file that follows the format described by the template. Additionally, bulk loading templates can be exported allowing Tripal sites to share loaders with one another.

The following commands can be executed to install the Tripal Bulk Loader using Drush:

```
cd /var/www/
drush pm-enable tripal_bulk_loader
```

1.13.1 Plan How to Store Data

To demonstrate use of the Bulk Loader, a brief example that imports a list of organisms and associates them with their NCBI taxonomy IDs will be performed. The input tab-delimited file will contains the list of all *Fragaria* (strawberry) species in NCBI at the time of the writing of this document. Click the file link below and download it to `/var/www/html/sites/default/files`.

- Fragaria.txt

```
cd $DRUPAL_HOME/sites/default/files
wget http://tripal.info/sites/default/files/book_pages/Fragaria_0.txt
```

This file has three columns: NCBI taxonomy ID, genus and species:

Table 1: Fragaria sample file

3747 "Fragaria" "x ananassa"
57918 "Fragaria" "vesca"
60188 "Fragaria" "nubicola"
64939 "Fragaria" "iinumae"
64940 "Fragaria" "moschata"
64941 "Fragaria" "nilgerrensis"
64942 "Fragaria" "viridis"

To use the bulk loader you must be familiar with the Chado database schema and have an idea for where data should be stored. It is best practice to consult the GMOD website or consult the Chado community (via the gmod-schema mailing list) when deciding how to store data. For this example, we want to add the species to Chado, and we want to associate the NCBI taxonomy ID with these organisms. The first step, therefore, is to decide where in Chado these data should go. In Chado, organisms are stored in the **organism** table. This table has the following fields:

chado.organism Table Schema

Name	Type	Description
organism_id	serial	PRIMARY KEY
abbreviation	character varying(255)	
genus	character varying(255)	UNIQUE#1 NOT NULL
species	character varying(255)	UNIQUE#1 NOT NULL A type of organism is always uniquely identified by genus and species. When mapping from the NCBI taxonomy names.dmp file, this column must be used where it is present, as the common_name column is not always unique (e.g. environmental samples). If a particular strain or subspecies is to be represented, this is appended onto the species name. Follows standard NCBI taxonomy pattern.
common_name	character varying(255)	
comment	text	

We can therefore store the second and third columns of the tab-delimited input file in the **genus** and **species** columns of the organism table.

In order to store a database external reference (such as for the NCBI Taxonomy ID) we need to use the following tables: **db**, **dbxref**, and **organism_dbxref**. The **db** table will house the entry for the NCBI Taxonomy; the **dbxref** table will house the entry for the taxonomy ID; and the **organism_dbxref** table will link the taxonomy ID stored in the **dbxref** table with the organism housed in the **organism** table. For reference, the fields of these tables are as follows:

chado.db Table Schema

Name	Type	Description
db_id	serial	PRIMARY KEY
name	character varying(255)	UNIQUE NOT NULL
description	character varying(255)	
urlprefix	character varying(255)	
url	character varying(255)	

chado.dbxref Table Schema

Name	Type	Description
dbxref_id	serial	PRIMARY KEY
db_id	integer	Foreign Key db. UNIQUE#1 NOT NULL
accession	character varying(255)	UNIQUE#1 NOT NULL. The local part of the identifier. Guaranteed by the db authority to be unique for that db.
version	character varying(255)	UNIQUE#1 NOT NULL DEFAULT ‘’
description	text	

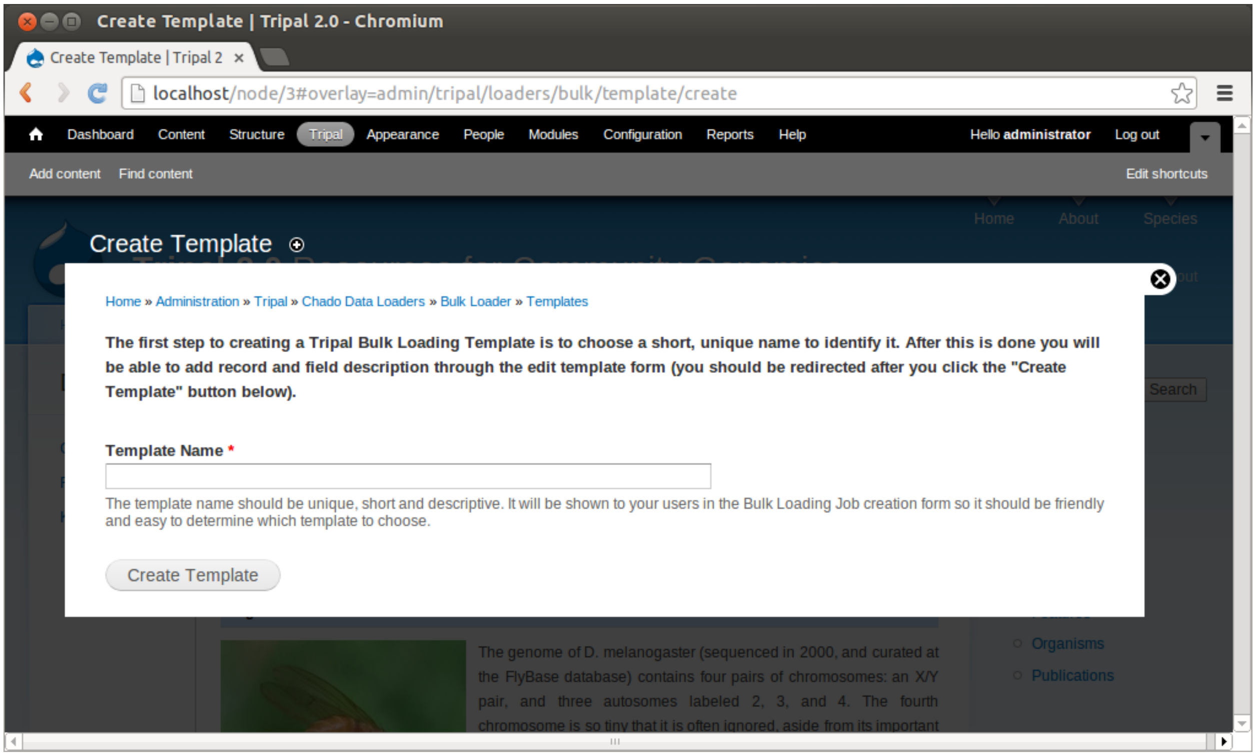
chado.organism_dbxref Table Schema

Name	Type	Description
organism_dbxref_id	serial	PRIMARY KEY
organism_id	integer	Foreign key organism. UNIQUE#1 NOT NULL
dbxref_id	integer	Foreign key dbxref. UNIQUE#1 NOT NULL

For our bulk loader template, we will therefore need to insert values into the **organism**, **db**, **dbxref** and **organism_dbxref** tables. In our input file we have the genus and species and taxonomy ID so we can import these with a bulk loader template. However, we do not have information that will go into the db table (e.g. “NCBI Taxonomy”). This is not a problem as the bulk loader can use existing data to help with import. We simply need to use the “NCBI Taxonomy” database that is currently in the Chado instance of Tripal v3.

1.13.2 Creating a New Bulk Loader Template

Now that we know where all of the data in the input file will go and we have the necessary dependencies in the database (i.e. the NCBI Taxonomy database), we can create a new bulk loader template. Navigate to `Tripal → Data Loaders → Chado Bulk Loader`, click the **Templates** tab in the top right corner, and finally click the link **Add Template**. The following page appears:

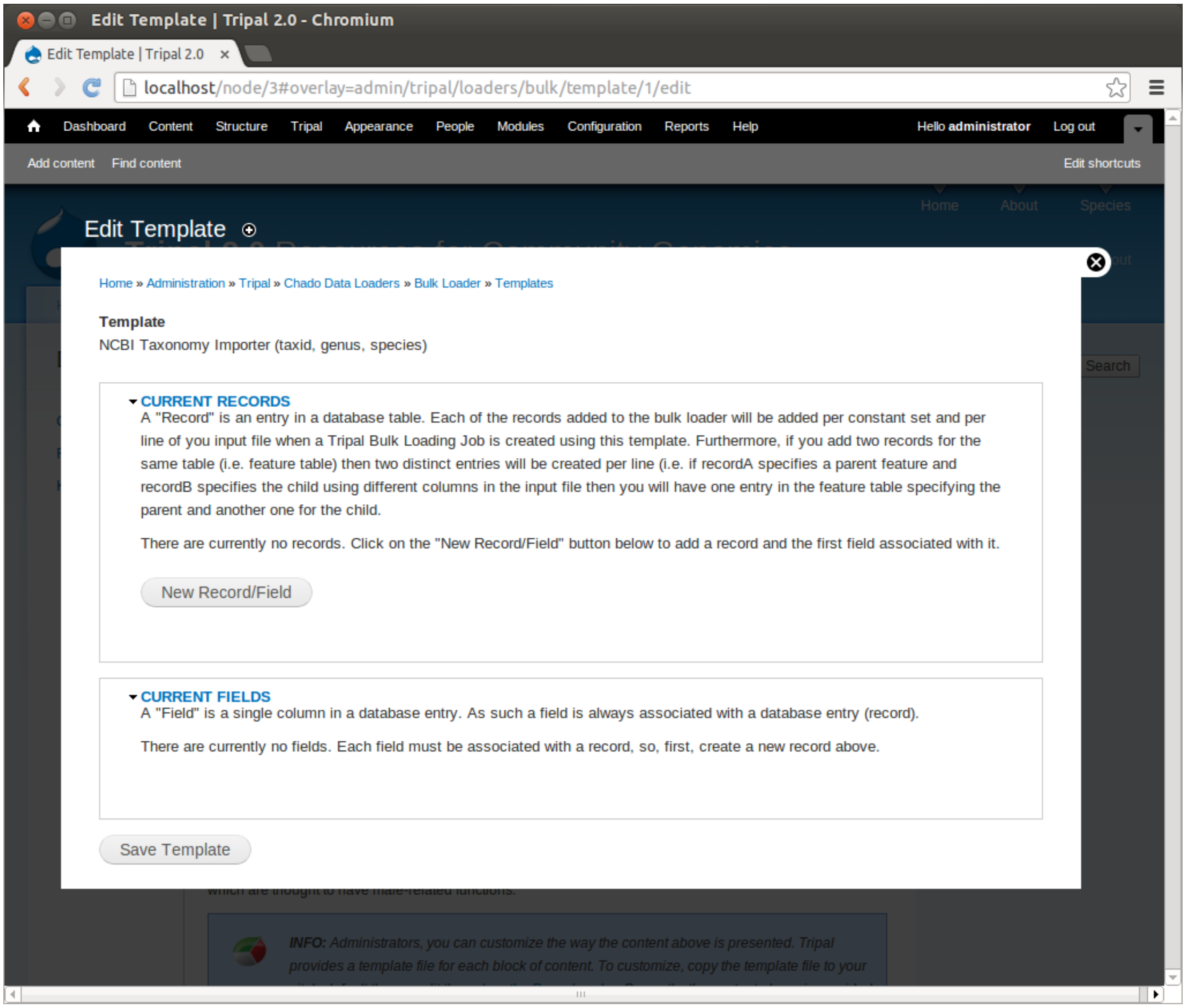

Create Template | Tripal 2.0 - Chromium
Create Template | Tripal 2
localhost/node/3#overlay=admin/tripal/loaders/bulk/template/create
Dashboard
Content
Structure
Tripal
Appearance
People
Modules
Configuration
Reports
Help
Hello administrator
Log out
Add content
Find content
Edit shortcuts
Create Template
Home » Administration » Tripal » Chado Data Loaders » Bulk Loader » Templates

The first step to creating a Tripal Bulk Loading Template is to choose a short, unique name to identify it. After this is done you will be able to add record and field description through the edit template form (you should be redirected after you click the "Create Template" button below).

Template Name *

The template name should be unique, short and descriptive. It will be shown to your users in the Bulk Loading Job creation form so it should be friendly and easy to determine which template to choose.

We need to first provide a name for our template. Try to name templates in a way that are meaningful for others. Currently only site administrators can load files using the bulk loader. But, future versions of Tripal will provide functionality to allow other privileged users the ability to use the bulk loader templates. Thus, it is important to name the templates so that others can easily identify the purpose of the template. For this example, enter the name **NCBI Taxonomy Importer (taxid, genus, species)**. The following page appears:



Edit Template

Home » Administration » Tripal » Chado Data Loaders » Bulk Loader » Templates

Template
NCBI Taxonomy Importer (taxid, genus, species)

▾ CURRENT RECORDS

A "Record" is an entry in a database table. Each of the records added to the bulk loader will be added per constant set and per line of you input file when a Tripal Bulk Loading Job is created using this template. Furthermore, if you add two records for the same table (i.e. feature table) then two distinct entries will be created per line (i.e. if recordA specifies a parent feature and recordB specifies the child using different columns in the input file then you will have one entry in the feature table specifying the parent and another one for the child.

There are currently no records. Click on the "New Record/Field" button below to add a record and the first field associated with it.

New Record/Field

▾ CURRENT FIELDS

A "Field" is a single column in a database entry. As such a field is always associated with a database entry (record).

There are currently no fields. Each field must be associated with a record, so, first, create a new record above.

Save Template

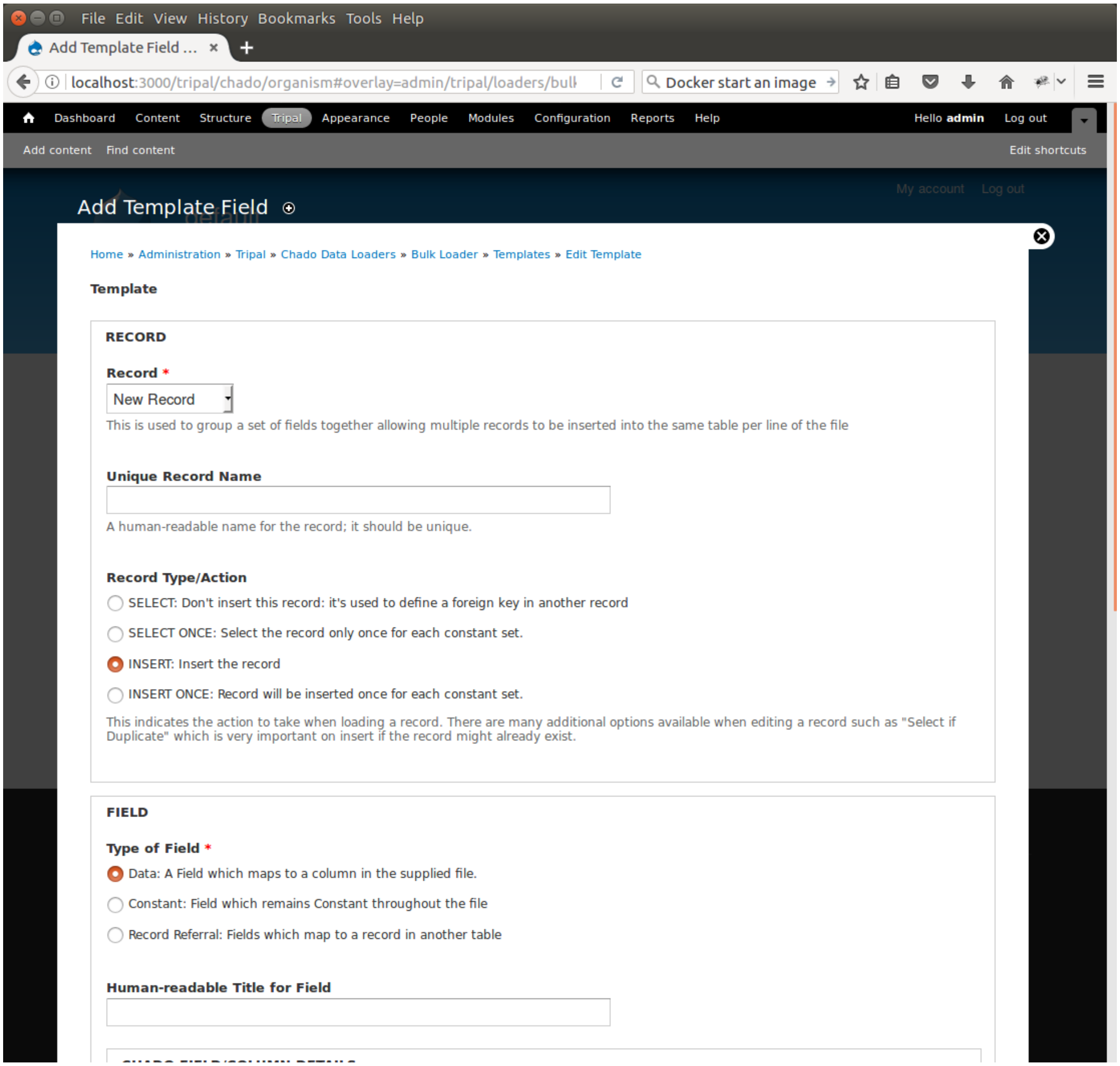
INFO: Administrators, you can customize the way the content above is presented. Tripal provides a template file for each block of content. To customize, copy the template file to your

Notice that the page is divided into two sections: **Current Records** and **Current Fields**. Before we continue with the template we need a bit of explanation as to the terminology used by the bulk loader. A **record** refers to a Chado table and an action on that table. For example, to insert the data from the input file we will need to select the NCBI Taxonomy database from the **db** table and insert entries into the **dbxref**, **organism** and **dbxref_organism** tables. Therefore, we will have four records:

- An insert into the organism table
- A select from the db table (to get the database id (db_id) of the "NCBI Taxonomy" database needed for the insert into the dbxref table)
- An insert into the dbxref table
- An insert into the organism_dbxref table.

Each record contains a set of fields on which the action is performed. Thus, when we insert an entry into the organism table we will insert into two fields: **genus** and **species**.

To create the first record for inserting an organism, click the button **New Record/Field**. The following page appears:


File Edit View History Bookmarks Tools Help
Add Template Field ...
localhost:3000/tripal/chado/organism#overlay=admin/tripal/loaders/bulk
Docker start an image
Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help
Hello admin Log out
Add content Find content
Edit shortcuts
My account Log out

Add Template Field

Home » Administration » Tripal » Chado Data Loaders » Bulk Loader » Templates » Edit Template

Template

RECORD

Record *
New Record
This is used to group a set of fields together allowing multiple records to be inserted into the same table per line of the file

Unique Record Name
A human-readable name for the record; it should be unique.

Record Type/Action
○ SELECT: Don't insert this record: it's used to define a foreign key in another record
○ SELECT ONCE: Select the record only once for each constant set.
● INSERT: Insert the record
○ INSERT ONCE: Record will be inserted once for each constant set.
This indicates the action to take when loading a record. There are many additional options available when editing a record such as "Select if Duplicate" which is very important on insert if the record might already exist.

FIELD

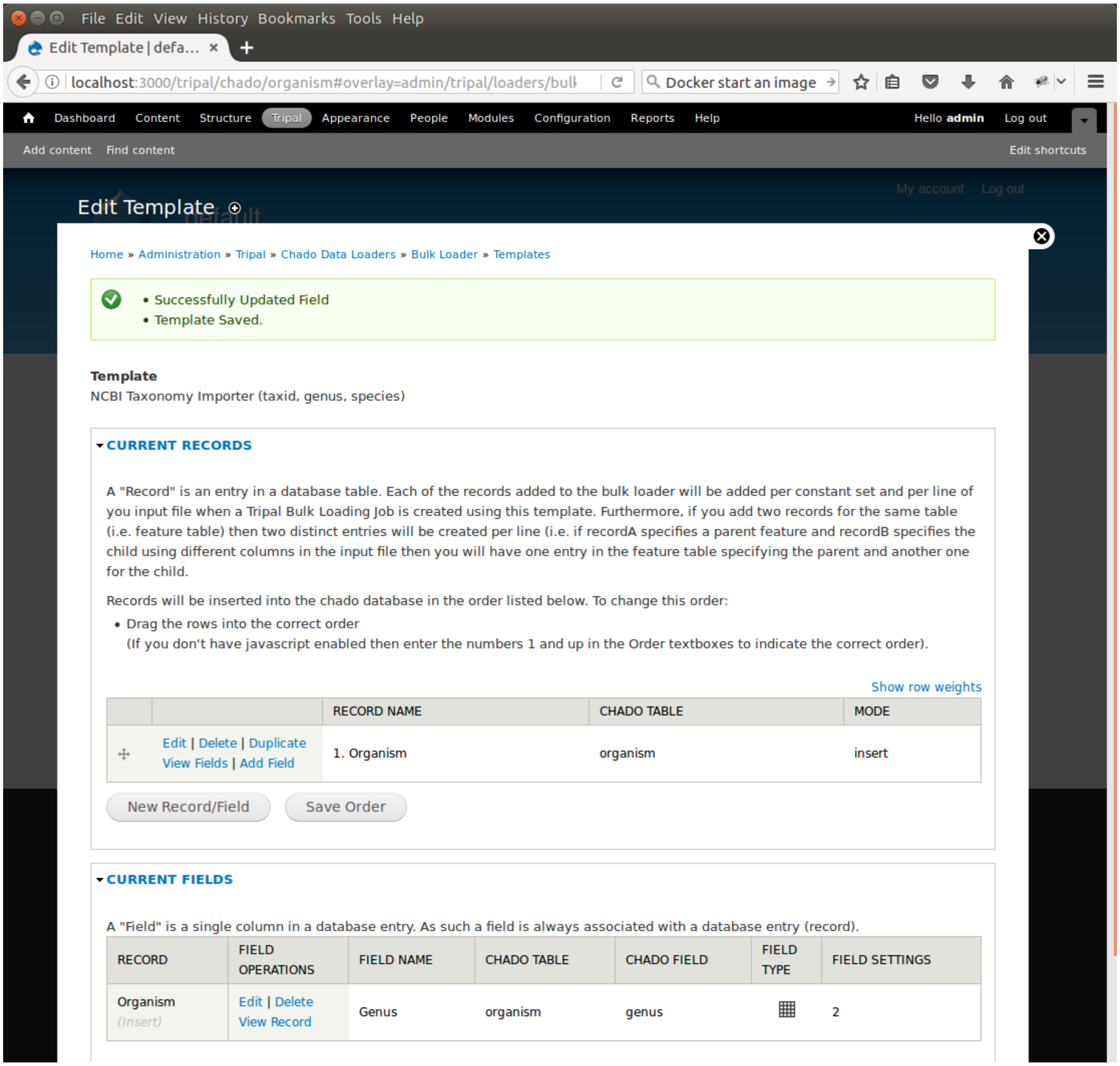
Type of Field *
● Data: A Field which maps to a column in the supplied file.
○ Constant: Field which remains Constant throughout the file
○ Record Referral: Fields which map to a record in another table

Human-readable Title for Field

By default, when adding a new record, the bulk loader also provides the form elements for adding the first field of the record as well. We are adding a new record, so we can leave the **Record** drop-down as **New Record**. Next, give this record a unique record name. Because we are inserting into the organism table, enter the name **Organism** into the **Unique Record Name** box.

We also have the opportunity with this form to add our first field to the record. Because we are adding the organism record we will first add the field for the **genus**. In the **Field** section we specify the source of the field. Because the genus value comes from the input file, select the first radio button titled **Data**. Next we need a human-readable name for the field. This field is the **genus** field so we will enter Genus into the **Human-readable Title for Field** box. Next, we need to specify the **Chado table** for this record. In the Chado table drop down box, choose the **organism** table, and in the **Chado Field/Column** drop down box select **genus**.

In the next section, titled **Data File Column**, we need to indicate the column in the tab-delimited file where the genus is found. For the example file this is column 2 (columns are ordered beginning with number 1). Therefore, enter the number **2** in the **Column** box. There are additional options to expose the field to the user, but for now we can ignore those options. Click the **Save Changes** button at the bottom. We now see that the organism record and the first field have been added to our bulk loader template.


File Edit View History Bookmarks Tools Help
Edit Template | defa...
localhost:3000/tripal/chado/organism#overlay=admin/tripal/loaders/bulk
Docker start an image
Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help
Hello admin Log out
Add content Find content
Edit shortcuts
My account Log out

Edit Template

Home » Administration » Tripal » Chado Data Loaders » Bulk Loader » Templates

- Successfully Updated Field
- Template Saved.

Template
NCBI Taxonomy Importer (taxid, genus, species)

▾ CURRENT RECORDS

A "Record" is an entry in a database table. Each of the records added to the bulk loader will be added per constant set and per line of you input file when a Tripal Bulk Loading Job is created using this template. Furthermore, if you add two records for the same table (i.e. feature table) then two distinct entries will be created per line (i.e. if recordA specifies a parent feature and recordB specifies the child using different columns in the input file then you will have one entry in the feature table specifying the parent and another one for the child.

Records will be inserted into the chado database in the order listed below. To change this order:

- Drag the rows into the correct order
  (If you don't have javascript enabled then enter the numbers 1 and up in the Order textboxes to indicate the correct order).

Show row weights

		RECORD NAME	CHADO TABLE	MODE
✥	Edit | Delete | Duplicate View Fields | Add Field	1. Organism	organism	insert

New Record/Field Save Order

▾ CURRENT FIELDS

A "Field" is a single column in a database entry. As such a field is always associated with a database entry (record).

RECORD	FIELD OPERATIONS	FIELD NAME	CHADO TABLE	CHADO FIELD	FIELD TYPE	FIELD SETTINGS
Organism (Insert)	Edit | Delete View Record	Genus	organism	genus	▦	2

We also see that the **Mode** (or action) for this record has been set to insert by default. Before continuing we should edit the settings for the record so that it is more fault tolerant. Click the **Edit** link to the left of the new organism record. On the resulting page we see the record details we already provided, but now there is a section titled **Action to take when Loading Record**. By default, the **INSERT** option is selected. This is correct. We want to perform an insert. However, notice in the **Additional Insert Options** section, the **SELECT if duplicate (no insert).** Check this box. This is a good option to add because it prevents the bulk loader from failing if the record already exists in the table.

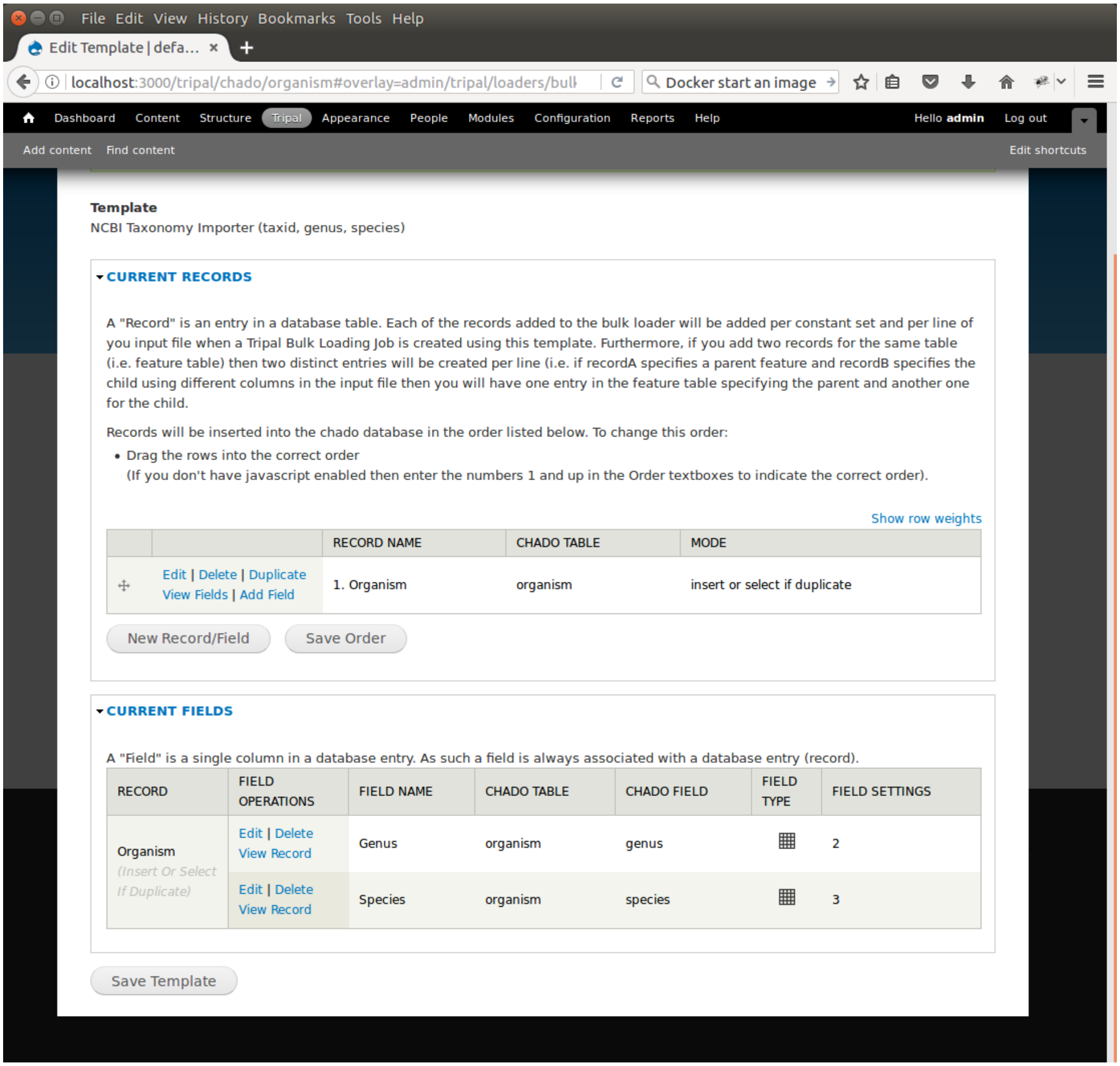
Click the **Save Record** button to save these settings. Now, you will see that the **Mode** is now set to insert or select if duplicate. Previously the **Mode** was just **insert**.

Next, we need to add the **species** field to the record. Click the **Add Field** link to the left of the organism record name. Here we are presented with the same form we used when first adding the organism record. However, this time, the **Record** section is collapsed. If we open that section the drop down already has the **Organism** record as we are not creating a new record. To add the **Species** field, provide the following values and click the **Save Changes button**:

- Type of field: Data

- Human-readable Title for Field: Species
- Chado table: organism (should already be set)
- Chado Field/Column: species
- Column: 3

We now have two fields for our organism record:


File Edit View History Bookmarks Tools Help
Edit Template | defa...
localhost:3000/tripal/chado/organism#overlay=admin/tripal/loaders/bul
Docker start an image
Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help
Hello admin Log out
Add content Find content
Edit shortcuts

Template
NCBI Taxonomy Importer (taxid, genus, species)

CURRENT RECORDS

A "Record" is an entry in a database table. Each of the records added to the bulk loader will be added per constant set and per line of you input file when a Tripal Bulk Loading Job is created using this template. Furthermore, if you add two records for the same table (i.e. feature table) then two distinct entries will be created per line (i.e. if recordA specifies a parent feature and recordB specifies the child using different columns in the input file then you will have one entry in the feature table specifying the parent and another one for the child.

Records will be inserted into the chado database in the order listed below. To change this order:

- Drag the rows into the correct order
(If you don't have javascript enabled then enter the numbers 1 and up in the Order textboxes to indicate the correct order).

Show row weights

		RECORD NAME	CHADO TABLE	MODE
✣	Edit | Delete | Duplicate View Fields | Add Field	1. Organism	organism	insert or select if duplicate

New Record/Field Save Order

CURRENT FIELDS

A "Field" is a single column in a database entry. As such a field is always associated with a database entry (record).

RECORD	FIELD OPERATIONS	FIELD NAME	CHADO TABLE	CHADO FIELD	FIELD TYPE	FIELD SETTINGS
Organism *(Insert Or Select If Duplicate)*	Edit | Delete View Record	Genus	organism	genus	▦	2
	Edit | Delete View Record	Species	organism	species	▦	3

Save Template

At this point our organism record is complete, however there are still a few fields in the organism table of Chado that are not present in our record. These include the **organism_id, abbreviation, common_name** and **comment** fields. We do not have values in our input file for any of these fields. Fortunately, the **organism_id** field is a primary key field and is auto generated when a record is submitted. We do not need to provide a value for that field. The other fields are not part of the unique constraint of the table. Therefore, those fields are optional and we do not need to specify them. Ideally, if we did have values for those non-required fields we would add them as well.

To this point, we have built the loader such that it can load two of the three columns in our input file. We have one remaining column: the NCBI taxonomy ID. In order to associate an organism with the taxonomy ID we must first insert the taxonomy ID into the **dbxref** table. Examining the dbxref table, we see that a **db_id** field is a required value in a foreign key relationship. We must first retrieve the **db_id** from the **db** table of Chado before we can add the entry to the **dbxref** table. Therefore, we will create a second record that will do just that. On the **Edit Template** page click the button **New Record/Field**. Here we see the same form we used for adding the first organism record. Provide the following values:

- **For the record:**
  - Record: New Record
  - Unique Record Name: NCBI Taxonomy DB
  - Record Type/Action: SELECT ONCE: Select the record only once for each constant set.
- **For the field:**
  - Type of field: Constant
  - Human-readable Title for Field: DB name
  - Chado table: db
  - Chado field/column: name
- **Within the Constant section:**
  - Constant Value: NCBITaxon
  - Check "Ensure the value is in the table"

Here we use a field type of **Constant** rather than **Data**. This is because we are providing the value to be used in the record rather than using a value from the input file. The value we are providing is "NCBI Taxonomy" which is the name of the database we added previously. The goal is to match the name "NCBI Taxonomy" with an entry in the **db** table. Click the **Save Changes** button.

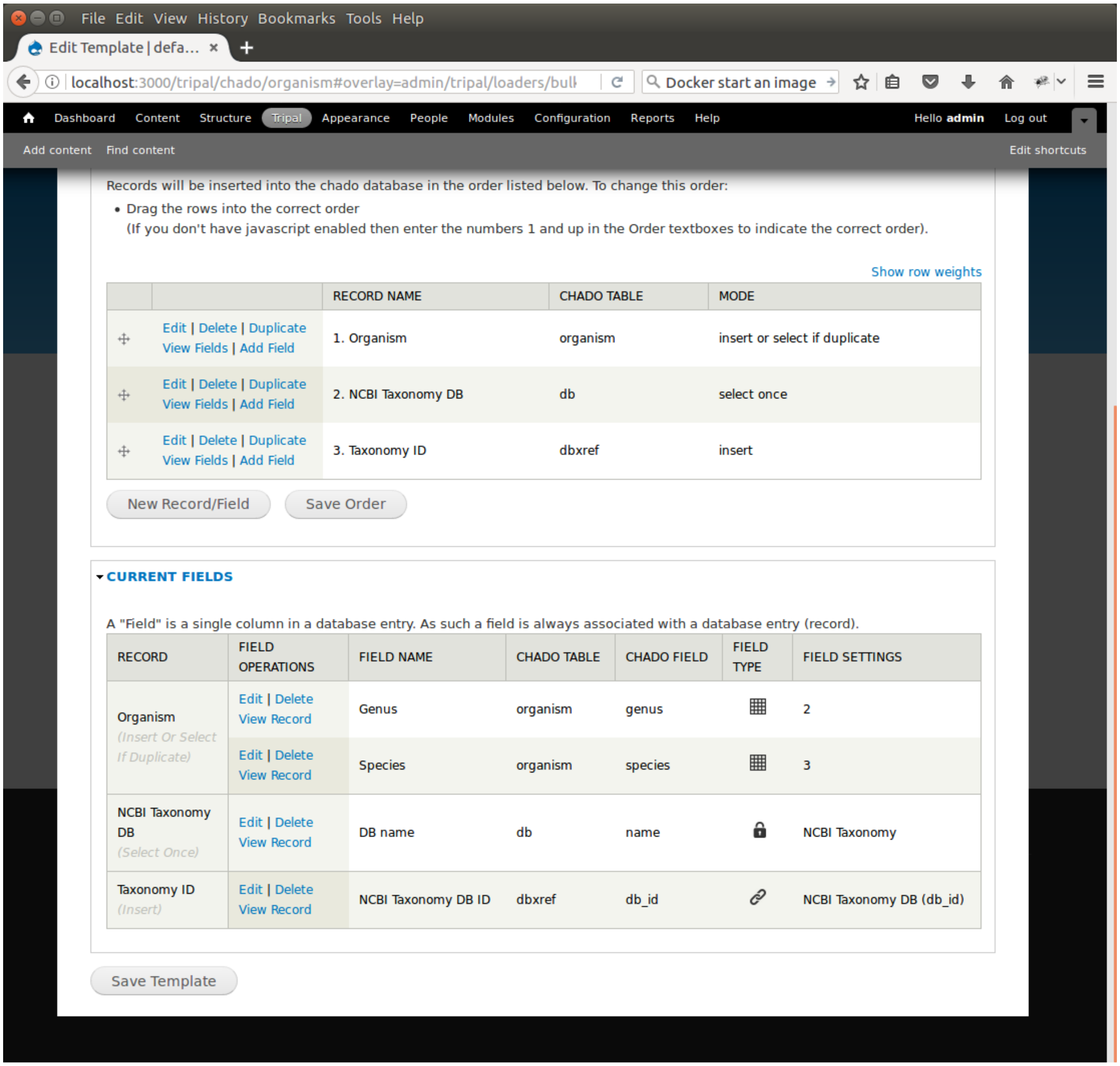
We now see a second record on the **Edit Template** page. However, the mode for this record is insert. We do not want to insert this value into the table, we want to select it because we need the corresponding **db_id** for the **dbxref** record. To change this, click the Edit link to the left of the **NCBI Taxonomy DB** record. Here we want to select only the option **SELECT ONCE**. We choose this option because the database entry that will be returned by the record will apply for the entire input file. Therefore, we only need to select it one time. Otherwise, the select statement would execute for each row in the input file causing excess queries. Finally, click **Save Record**. The **NCBI Taxonomy DB** record now has a mode of **select once**. When we created the record, we selected the option to 'SELECT ONCE'. This means that the bulk loader will perform the action one time for that record for the entire import process. Because the field is a constant the bulk loader need not execute that record for every row it imports from our input file. We simply need to select the record once and the record then becomes available for use through the entire import process.

Now that we have a record that selects the **db_id** we can now create the **dbxref** record. For the **dbxref** record there is a unique constraint that requires the **accession**, **db_id** and **version**. The version record has a default value so we only need to create two fields for this new record: the db_id and the accession. We will use the **db_id** from the **NCBI Taxonomy DB** record and the accession is the first column of the input file. First, we will add the **db_id** record. Click the **New Record/Field** button and set the following:

- **For the record:**
  - Record: New Record
  - Unique Record Name: Taxonomy ID
  - Record Type/Action: INSERT: insert the record
- **For the field:**
  - Type of field: Record referral
  - Human-readable Title for Field: NCBI Taxonomy DB ID

  - Chado table: dbxref
  - Chado Field/Column: db_id
- **In the Record Referral Section:**
  - Record to refer to: NCBI Taxonomy DB
  - Field to refer to: db_id

Click the Save Changes button. The Edit Template page appears.


File Edit View History Bookmarks Tools Help
Edit Template | defa... ×
localhost:3000/tripal/chado/organism#overlay=admin/tripal/loaders/bulk
Docker start an image
Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help
Hello admin Log out
Add content Find content
Edit shortcuts

Records will be inserted into the chado database in the order listed below. To change this order:

- Drag the rows into the correct order
  (If you don't have javascript enabled then enter the numbers 1 and up in the Order textboxes to indicate the correct order).

Show row weights

		RECORD NAME	CHADO TABLE	MODE
✣	Edit | Delete | Duplicate View Fields | Add Field	1. Organism	organism	insert or select if duplicate
✣	Edit | Delete | Duplicate View Fields | Add Field	2. NCBI Taxonomy DB	db	select once
✣	Edit | Delete | Duplicate View Fields | Add Field	3. Taxonomy ID	dbxref	insert

▾ CURRENT FIELDS

A "Field" is a single column in a database entry. As such a field is always associated with a database entry (record).

RECORD	FIELD OPERATIONS	FIELD NAME	CHADO TABLE	CHADO FIELD	FIELD TYPE	FIELD SETTINGS
Organism *(Insert Or Select If Duplicate)*	Edit | Delete View Record	Genus	organism	genus	▦	2
	Edit | Delete View Record	Species	organism	species	▦	3
NCBI Taxonomy DB *(Select Once)*	Edit | Delete View Record	DB name	db	name	🔒	NCBI Taxonomy
Taxonomy ID *(Insert)*	Edit | Delete View Record	NCBI Taxonomy DB ID	dbxref	db_id	🔗	NCBI Taxonomy DB (db_id)

Again, we need to edit the record to make the loader more fault tolerant. Click the Edit link to the left of the Taxonomy ID record. Select the following:

- Insert
- Select if duplicate

To complete this record, we need to add the accession field. Click the Add field link to the left of the Taxonomy ID record name. Provide the following values:

- **For the field:**
  - Type of Field: Data
  - Human-readable Title for Field: Accession
  - Chado table: dbxref
  - Chado field/column: accession
- **In the Data File Column section:**
  - Column: 1

At this state, we should have three records: Organism, NCBI Taxonomy DB, and Taxonomy ID. We can now add the final record that will insert a record into the **organism_dbxref** table. Create this new record with the following details:

- **For the record:**
  - Record: New Record
  - Unique Record Name: Taxonomy/Organism Linker
  - Check: Insert: insert the record
- **For the field:**
  - Type of Field: Record Referral
  - Human-readable Title for Field: Accession Ref
  - Chado table: organism_dbxref
  - Chado field/column: dbxref_id
- **In the Record Referral section:**
  - Record to refer to: Taxonomy ID
  - Field to refer to: dbxref_id

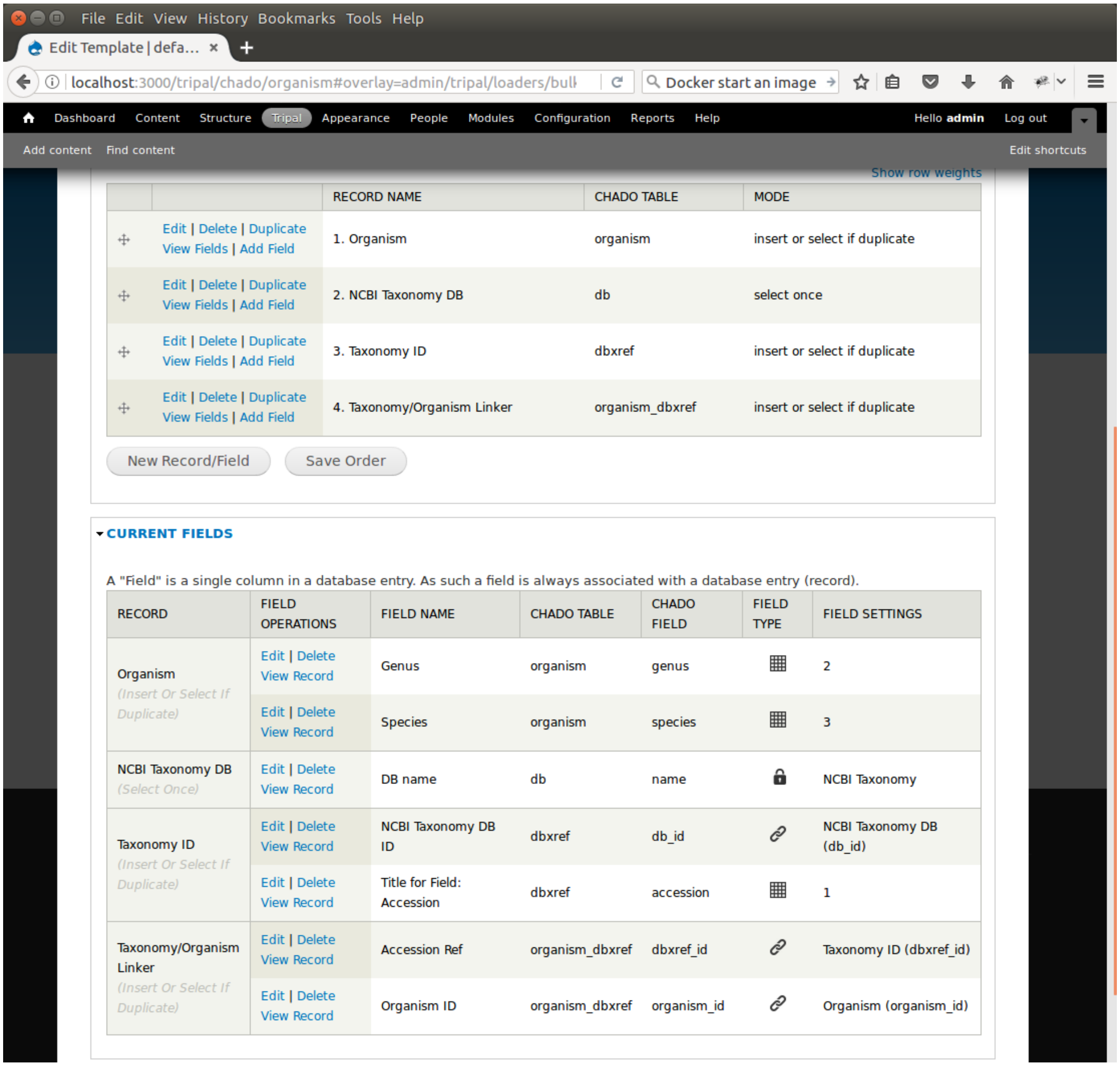
Create the second field:

- **For the field:**
  - Type of Field: Record Referral
  - Human-readable Title for Field: Organism ID
  - Chado table: organism_dbxref
  - Chado field/column: organism_id
- **In the Record Referral section:**
  - Record to refer to: Organism
  - Field to refer to: organism_id

After saving the field. Edit the record and set the following:

- Change the record mode to: insert or select if duplicate

We are now done! We have created a bulk loader template that reads in a file with three columns containing an NCBI taxonomy ID, a genus and species. The loader places the genus and species in the **organism** table, adds the NCBI Taxonomy ID to the **dbxref** table, links it to the NCBI Taxonomy entry in the db table, and then adds an entry to the **organism_dbxref** table that links the organism to the NCBI taxonomy Id. The following screen shots show how the template should appear:


File Edit View History Bookmarks Tools Help
Edit Template | defa…
localhost:3000/tripal/chado/organism#overlay=admin/tripal/loaders/bulk
Docker start an image
Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help
Hello admin Log out
Add content Find content
Edit shortcuts
Show row weights

		RECORD NAME	CHADO TABLE	MODE
✥	Edit | Delete | Duplicate View Fields | Add Field	1. Organism	organism	insert or select if duplicate
✥	Edit | Delete | Duplicate View Fields | Add Field	2. NCBI Taxonomy DB	db	select once
✥	Edit | Delete | Duplicate View Fields | Add Field	3. Taxonomy ID	dbxref	insert or select if duplicate
✥	Edit | Delete | Duplicate View Fields | Add Field	4. Taxonomy/Organism Linker	organism_dbxref	insert or select if duplicate

▾ CURRENT FIELDS

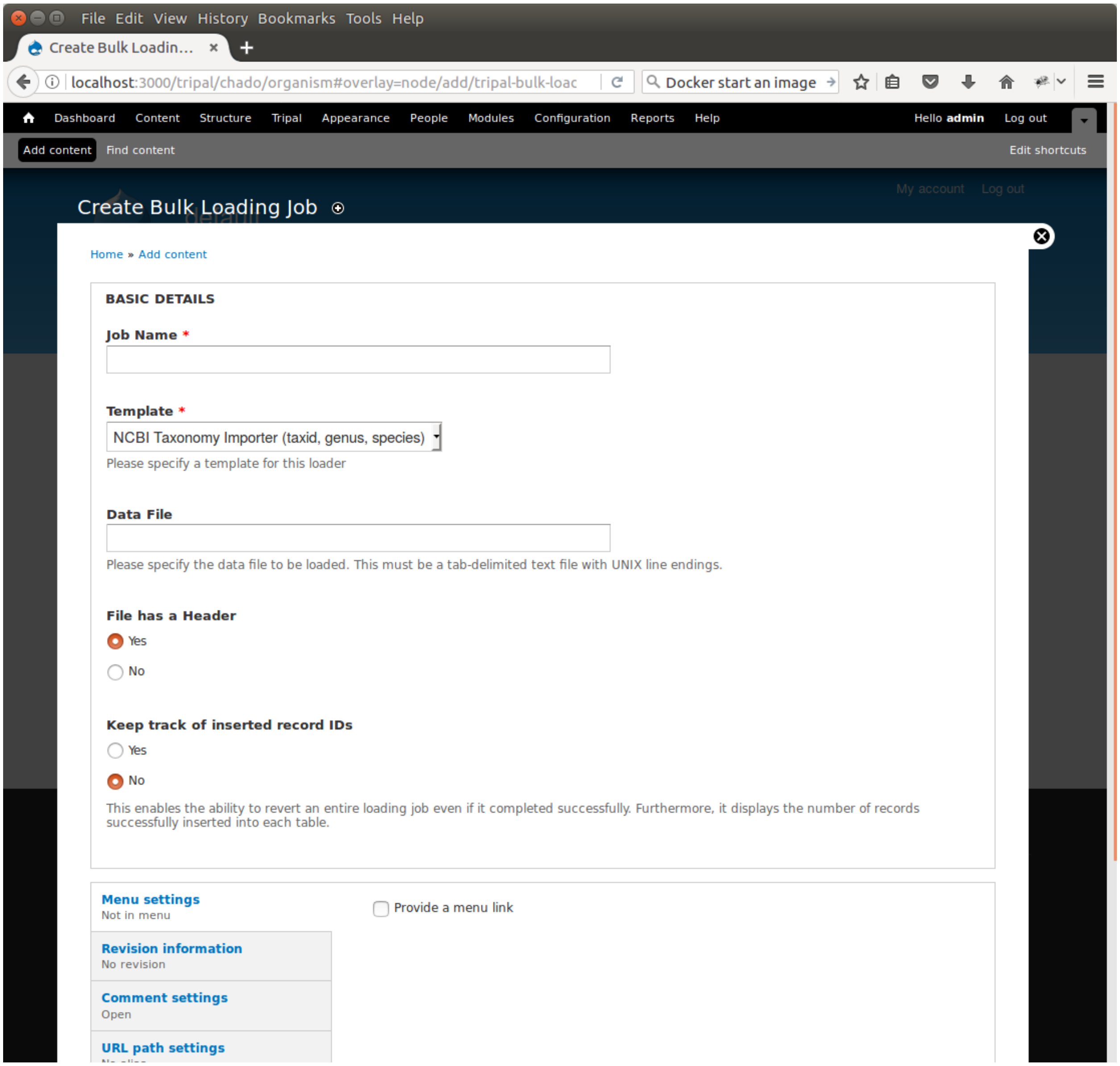
A "Field" is a single column in a database entry. As such a field is always associated with a database entry (record).

RECORD	FIELD OPERATIONS	FIELD NAME	CHADO TABLE	CHADO FIELD	FIELD TYPE	FIELD SETTINGS
Organism *(Insert Or Select If Duplicate)*	Edit | Delete View Record	Genus	organism	genus	▦	2
	Edit | Delete View Record	Species	organism	species	▦	3
NCBI Taxonomy DB *(Select Once)*	Edit | Delete View Record	DB name	db	name	🔒	NCBI Taxonomy
Taxonomy ID *(Insert Or Select If Duplicate)*	Edit | Delete View Record	NCBI Taxonomy DB ID	dbxref	db_id	🔗	NCBI Taxonomy DB (db_id)
	Edit | Delete View Record	Title for Field: Accession	dbxref	accession	▦	1
Taxonomy/Organism Linker *(Insert Or Select If Duplicate)*	Edit | Delete View Record	Accession Ref	organism_dbxref	dbxref_id	🔗	Taxonomy ID (dbxref_id)
	Edit | Delete View Record	Organism ID	organism_dbxref	organism_id	🔗	Organism (organism_id)

To save the template, click the **Save Template** link at the bottom of the page.

1.13.3 Creating a Bulk Loader Job (importing a file)

Now that we have created a bulk loader template we can use it to import a file. We will import the **Fragaria**.txt file downloaded previously. To import a file using a bulk loader template, click the **Add Content** link in the administrative menu and click the **Bulk Loading Job**. A bulk loading job is required each time we want to load a file. Below is a screen shot of the page used for creating a bulk loading job.


File Edit View History Bookmarks Tools Help
Create Bulk Loadin...
localhost:3000/tripal/chado/organism#overlay=node/add/tripal-bulk-loac
Docker start an image
Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help
Hello admin Log out
Add content Find content
Edit shortcuts
My account Log out

Create Bulk Loading Job

Home » Add content

BASIC DETAILS

Job Name *

Template *

NCBI Taxonomy Importer (taxid, genus, species)

Please specify a template for this loader

Data File

Please specify the data file to be loaded. This must be a tab-delimited text file with UNIX line endings.

File has a Header

● Yes

○ No

Keep track of inserted record IDs

○ Yes

● No

This enables the ability to revert an entire loading job even if it completed successfully. Furthermore, it displays the number of records successfully inserted into each table.

Menu settings
Not in menu

□ Provide a menu link

Revision information
No revision

Comment settings
Open

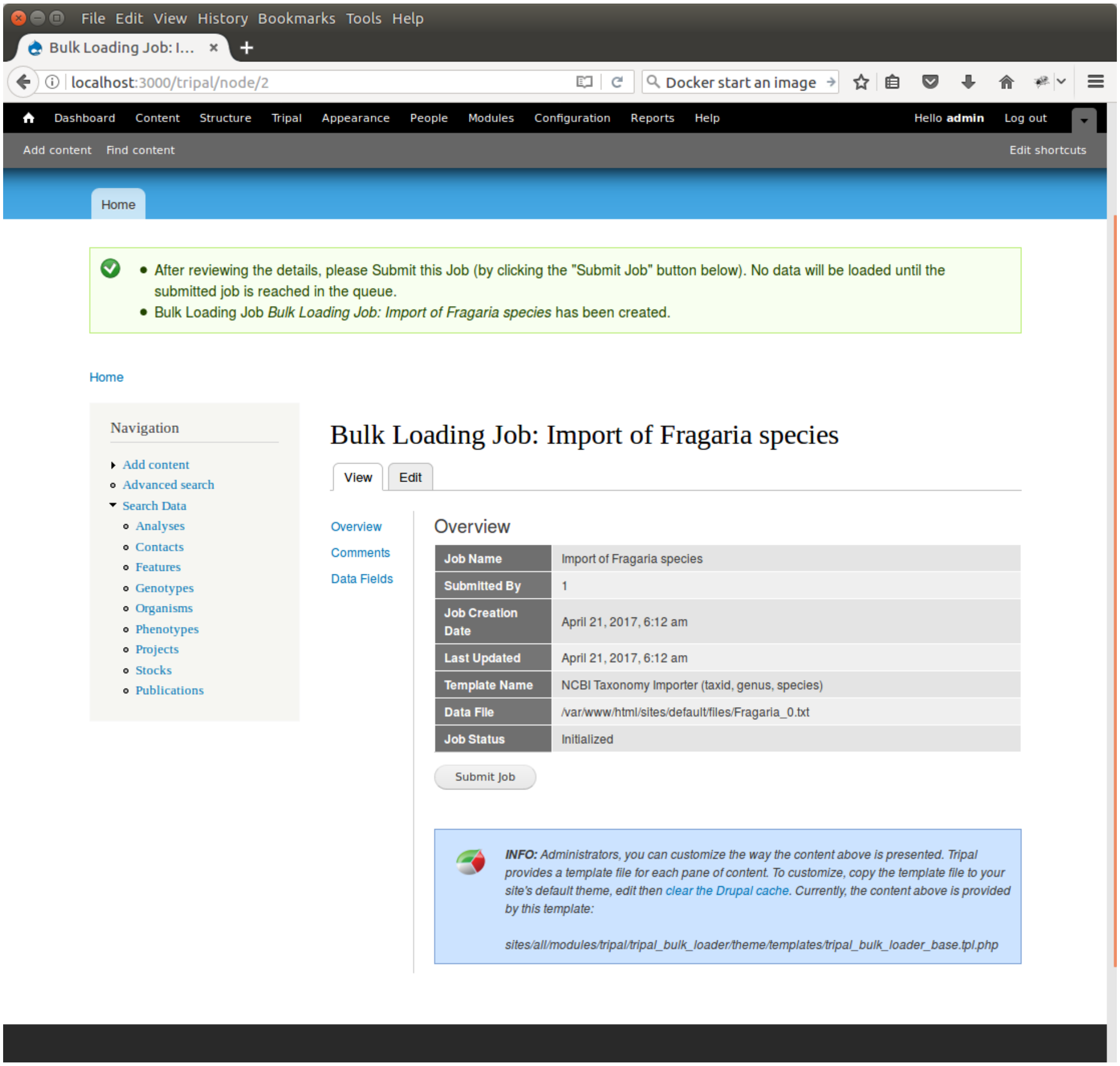
URL path settings

Provide the following values:

- Job Name: Import of Fragaria species
- Template: NCBI Taxonomy Importer (taxid, genus species).
- Data File: [DRUPAL_HOME]/sites/default/files/Fragaria_0.txt
- Keep track of inserted IDs: No
- File has a header: No

Note: Be sure to change the [DRUPAL_HOME] token to where Drupal is installed.

Click **Save**. The page then appears as follows:



File Edit View History Bookmarks Tools Help
Bulk Loading Job: I...
localhost:3000/tripal/node/2
Docker start an image
Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help
Hello admin Log out
Add content Find content
Edit shortcuts
Home

- After reviewing the details, please Submit this Job (by clicking the "Submit Job" button below). No data will be loaded until the submitted job is reached in the queue.
- Bulk Loading Job *Bulk Loading Job: Import of Fragaria species* has been created.

Bulk Loading Job: Import of Fragaria species

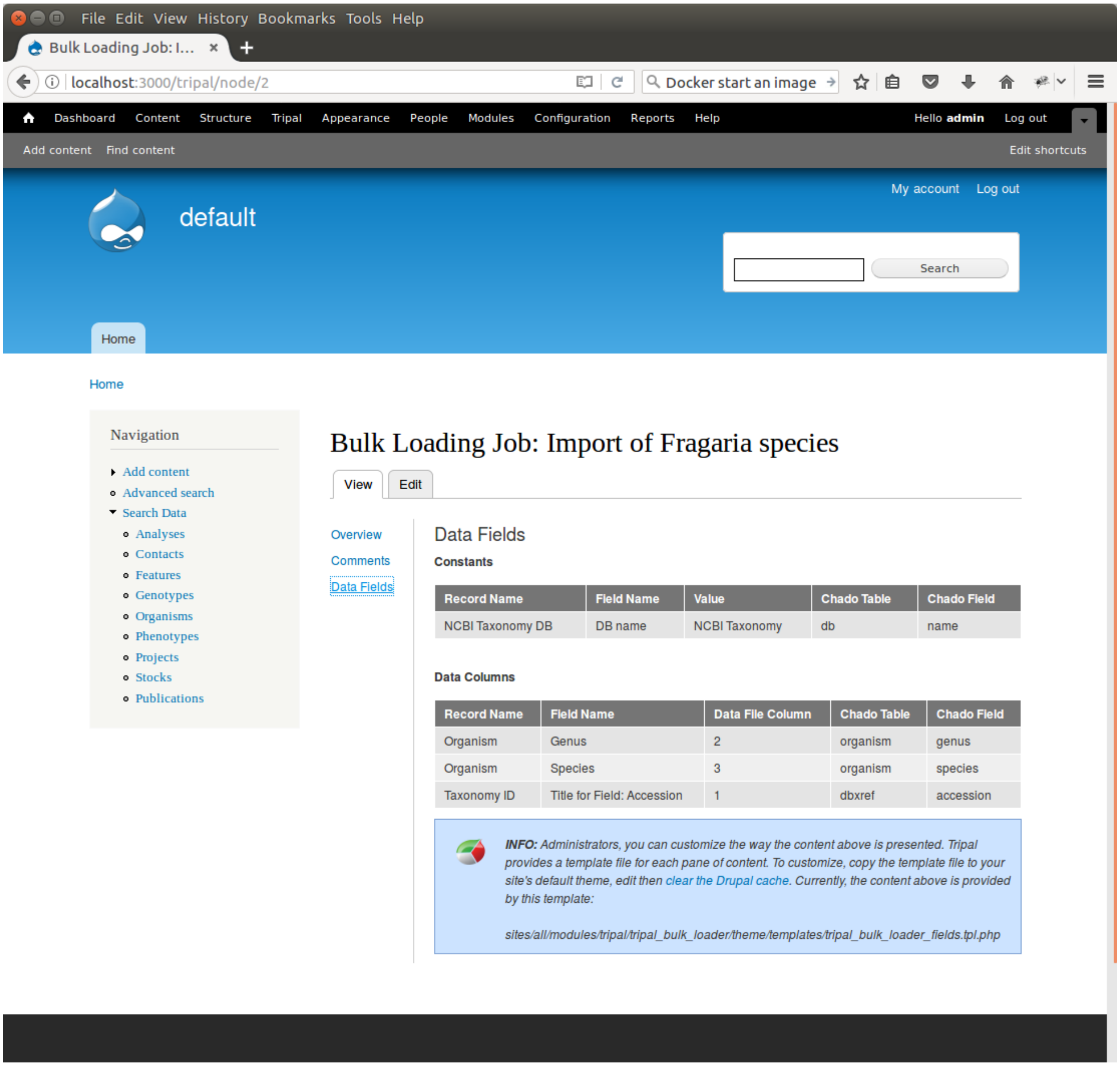
Overview

Job Name	Import of Fragaria species
Submitted By	1
Job Creation Date	April 21, 2017, 6:12 am
Last Updated	April 21, 2017, 6:12 am
Template Name	NCBI Taxonomy Importer (taxid, genus, species)
Data File	/var/www/html/sites/default/files/Fragaria_0.txt
Job Status	Initialized

INFO: Administrators, you can customize the way the content above is presented. Tripal provides a template file for each pane of content. To customize, copy the template file to your site's default theme, edit then clear the Drupal cache. Currently, the content above is provided by this template:

sites/all/modules/tripal/tripal_bulk_loader/theme/templates/tripal_bulk_loader_base.tpl.php

You can see details about constants that are used by the template and the where the fields from the input file will be stored by clicking the **Data Fields** tab in the table of contents on the left sidebar.



Bulk Loading Job: Import of Fragaria species

Data Fields

Constants

Record Name	Field Name	Value	Chado Table	Chado Field
NCBI Taxonomy DB	DB name	NCBI Taxonomy	db	name

Data Columns

Record Name	Field Name	Data File Column	Chado Table	Chado Field
Organism	Genus	2	organism	genus
Organism	Species	3	organism	species
Taxonomy ID	Title for Field: Accession	1	dbxref	accession

INFO: Administrators, you can customize the way the content above is presented. Tripal provides a template file for each pane of content. To customize, copy the template file to your site's default theme, edit then clear the Drupal cache. Currently, the content above is provided by this template:

sites/all/modules/tripal/tripal_bulk_loader/theme/templates/tripal_bulk_loader_fields.tpl.php

Now that we have created a job, we can submit it for execution by clicking the **Submit Job** button. This adds a job to the Tripal Jobs systems and we can launc the job as we have previously in this tutorial:

```
cd /var/www
drush trp-run-jobs --username=admin --root=$DRUPAL_HOME
```

After execution of the job you should see similar output to the terminal window:

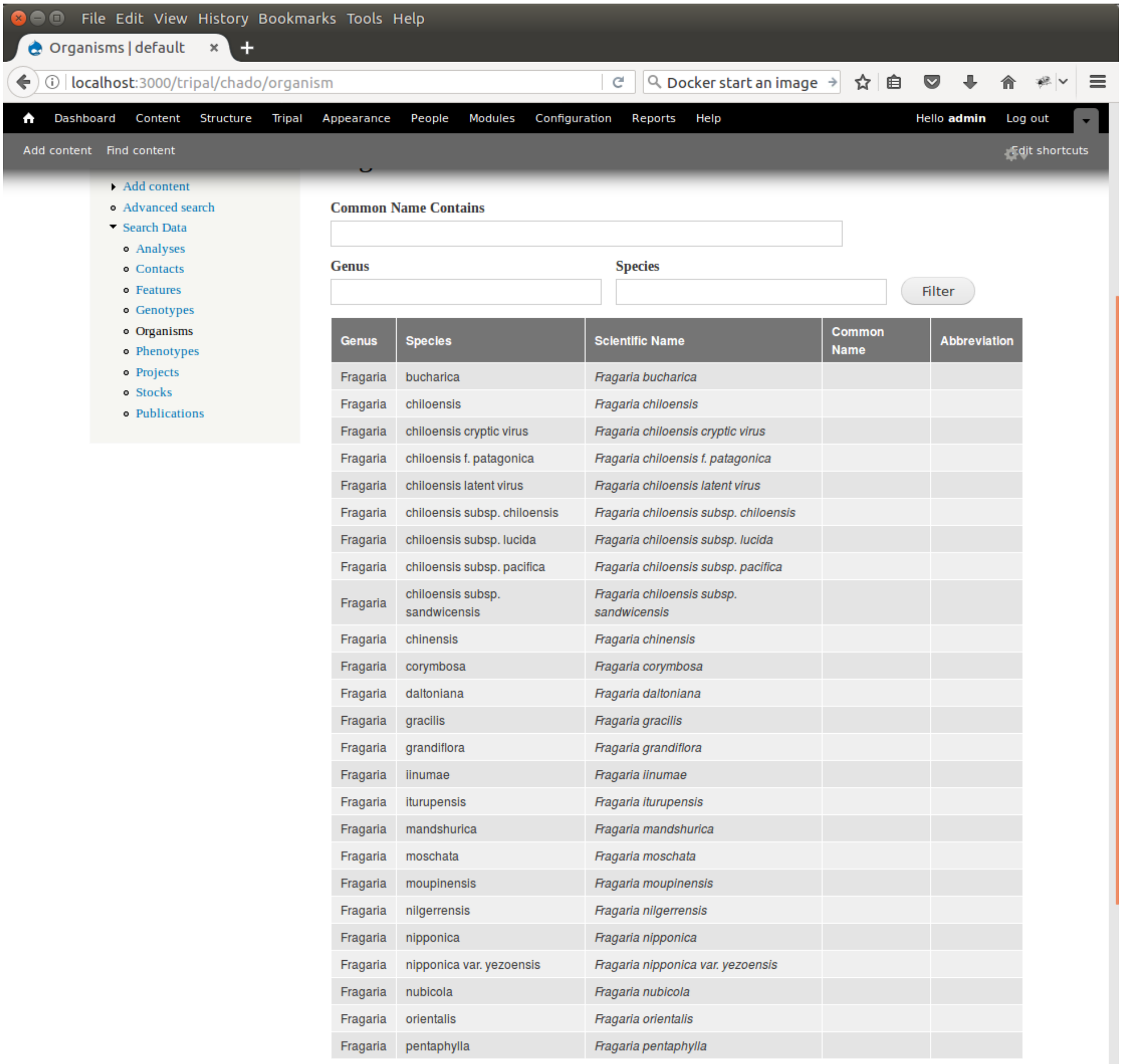
```
Tripal Job Launcher
Running as user 'admin'
-------------------
There are 1 jobs queued.
Calling: tripal_bulk_loader_load_data(2, 7)
Template: NCBI Taxonomy Importer (taxid, genus, species) (1)
File: /var/www/html/sites/default/files/Fragaria_0.txt (46 lines)

Preparing to load...
```

(continues on next page)

(continued from previous page)

```
Loading...
    Preparing to load the current constant set...
        Open File...
        Start Transaction...
        Defer Constraints...
        Acquiring Table Locks...
            ROW EXCLUSIVE for organism
            ROW EXCLUSIVE for dbxref
            ROW EXCLUSIVE for organism_dbxref
    Loading the current constant set...
Progress:
[|||||||||||||||||||||||||||||||||||||||||||||||||||] 100.00%. (46 of 46) Memory:␣
↪33962080
```

Our *Fragaira* species should now be loaded, and we return to the Tripal site to see them. Click on the **Organisms** link in the **Search Data** menu. In the search form that appears, type “Fragaria” in the **Genus** text box and click the **Filter** button. We should see the list of newly added *Fragaria* species.



Common Name Contains

Genus

Species

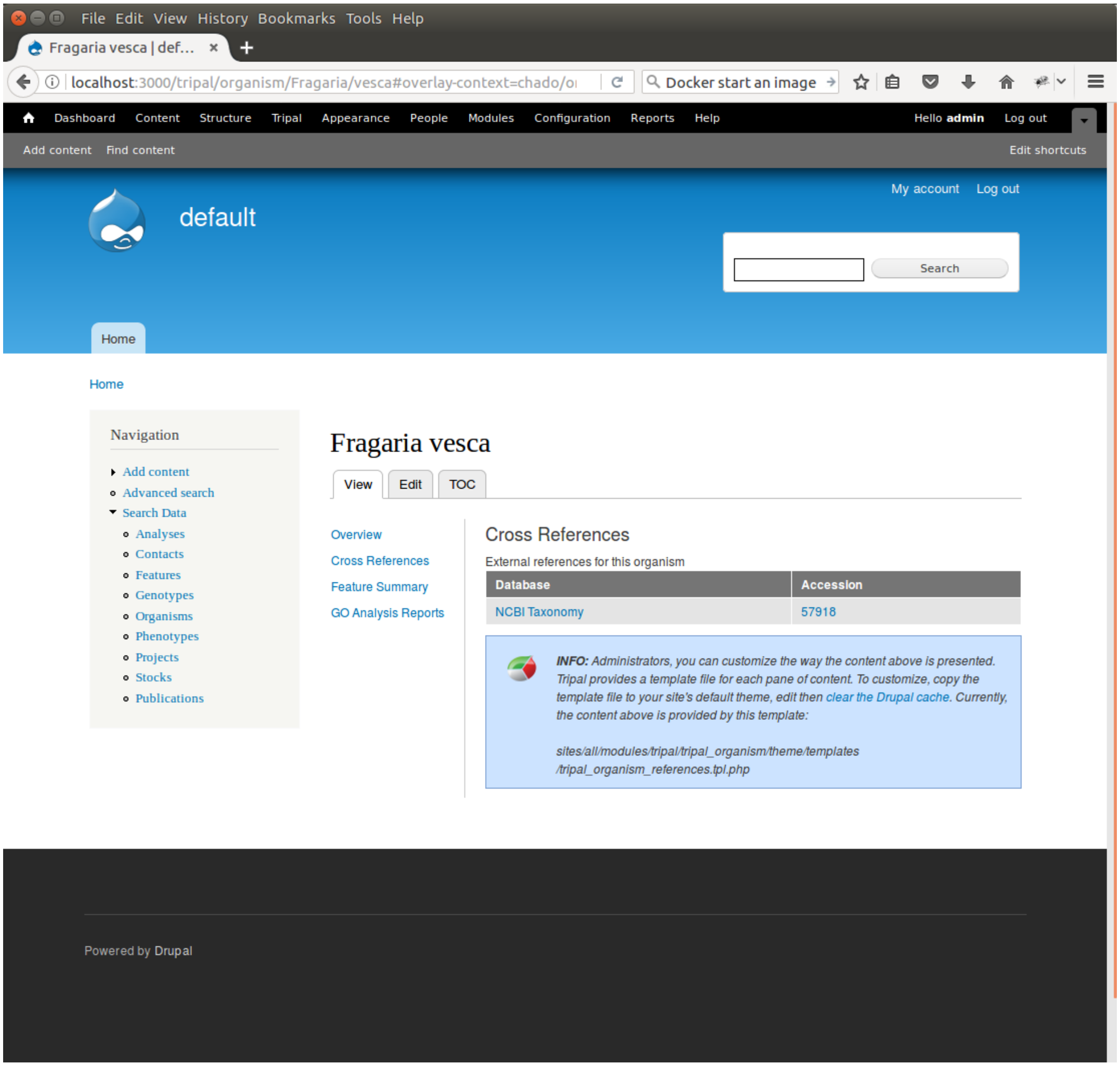
Filter

Genus	Species	Scientific Name	Common Name	Abbreviation
Fragaria	bucharica	*Fragaria bucharica*		
Fragaria	chiloensis	*Fragaria chiloensis*		
Fragaria	chiloensis cryptic virus	*Fragaria chiloensis cryptic virus*		
Fragaria	chiloensis f. patagonica	*Fragaria chiloensis f. patagonica*		
Fragaria	chiloensis latent virus	*Fragaria chiloensis latent virus*		
Fragaria	chiloensis subsp. chiloensis	*Fragaria chiloensis subsp. chiloensis*		
Fragaria	chiloensis subsp. lucida	*Fragaria chiloensis subsp. lucida*		
Fragaria	chiloensis subsp. pacifica	*Fragaria chiloensis subsp. pacifica*		
Fragaria	chiloensis subsp. sandwicensis	*Fragaria chiloensis subsp. sandwicensis*		
Fragaria	chinensis	*Fragaria chinensis*		
Fragaria	corymbosa	*Fragaria corymbosa*		
Fragaria	daltoniana	*Fragaria daltoniana*		
Fragaria	gracilis	*Fragaria gracilis*		
Fragaria	grandiflora	*Fragaria grandiflora*		
Fragaria	iinumae	*Fragaria iinumae*		
Fragaria	iturupensis	*Fragaria iturupensis*		
Fragaria	mandshurica	*Fragaria mandshurica*		
Fragaria	moschata	*Fragaria moschata*		
Fragaria	moupinensis	*Fragaria moupinensis*		
Fragaria	nilgerrensis	*Fragaria nilgerrensis*		
Fragaria	nipponica	*Fragaria nipponica*		
Fragaria	nipponica var. yezoensis	*Fragaria nipponica var. yezoensis*		
Fragaria	nubicola	*Fragaria nubicola*		
Fragaria	orientalis	*Fragaria orientalis*		
Fragaria	pentaphylla	*Fragaria pentaphylla*		

Before the organisms will have Tripal pages, the Chado records need to be **Published**. You can publish them by navigating to **Tripal Content -> Publish Tripal Content**. Select the **organism** table from the dropdown and run the job.

Note: In Tripal 2, records were synced by naviating to **Tripal** → **Chado Modules** → **Organisms**.

Once complete, return to the search form, find a *Fragaria* species that has been published and view its page. You should see a Cross References link in the left table of contents. If you click that link you should see the NCBI Taxonomy ID with a link to the page:



Fragaria vesca

Cross References

External references for this organism

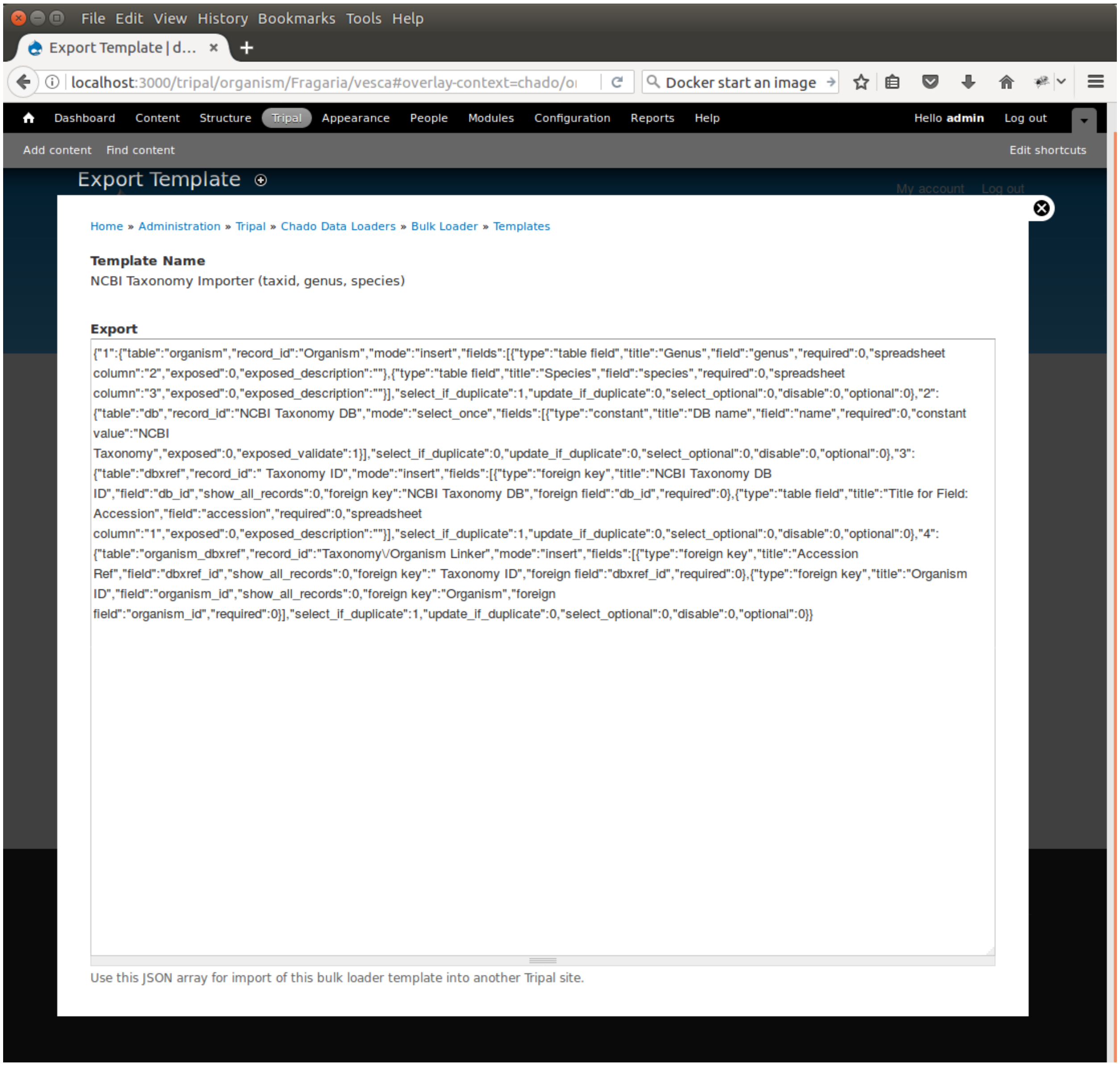
Database	Accession
NCBI Taxonomy	57918

INFO: Administrators, you can customize the way the content above is presented. Tripal provides a template file for each pane of content. To customize, copy the template file to your site's default theme, edit then clear the Drupal cache. Currently, the content above is provided by this template:

sites/all/modules/tripal/tripal_organism/theme/templates /tripal_organism_references.tpl.php

1.13.4 Sharing Your Templates with Others

Now that our template for loading organisms with NCBI Taxonomy IDs is completed we can share our template loader with anyone else that has a Tripal-based site. To do this we simply export the template in text format, place it in a text file or directly in an email and send to a collaborator for import into their site. To do this, navigate to **Tripal** → **Chado Data Loaders** → **Buik Loader** and click the **Tempalate** tab at the top. Here we find a table of all the templates we have created. We should see our template named **NCBI Taxonomy Importer** (taxid, genus, species). In the far right colum is a link to export that template. Licking that link will redirect you to a page where the template is provided in a serialized PHP array.


File Edit View History Bookmarks Tools Help
Export Template | d...
localhost:3000/tripal/organism/Fragaria/vesca#overlay-context=chado/o
Docker start an image
Dashboard Content Structure Tripal Appearance People Modules Configuration Reports Help
Hello admin Log out
Add content Find content
Edit shortcuts
My account Log out

Export Template

Home » Administration » Tripal » Chado Data Loaders » Bulk Loader » Templates

Template Name
NCBI Taxonomy Importer (taxid, genus, species)

Export

```
{"1":{"table":"organism","record_id":"Organism","mode":"insert","fields":[{"type":"table field","title":"Genus","field":"genus","required":0,"spreadsheet
column":"2","exposed":0,"exposed_description":""},{"type":"table field","title":"Species","field":"species","required":0,"spreadsheet
column":"3","exposed":0,"exposed_description":""}],"select_if_duplicate":1,"update_if_duplicate":0,"select_optional":0,"disable":0,"optional":0},"2":
{"table":"db","record_id":"NCBI Taxonomy DB","mode":"select_once","fields":[{"type":"constant","title":"DB name","field":"name","required":0,"constant
value":"NCBI
Taxonomy","exposed":0,"exposed_validate":1}],"select_if_duplicate":0,"update_if_duplicate":0,"select_optional":0,"disable":0,"optional":0},"3":
{"table":"dbxref","record_id":" Taxonomy ID","mode":"insert","fields":[{"type":"foreign key","title":"NCBI Taxonomy DB
ID","field":"db_id","show_all_records":0,"foreign key":"NCBI Taxonomy DB","foreign field":"db_id","required":0},{"type":"table field","title":"Title for Field:
Accession","field":"accession","required":0,"spreadsheet
column":"1","exposed":0,"exposed_description":""}],"select_if_duplicate":1,"update_if_duplicate":0,"select_optional":0,"disable":0,"optional":0},"4":
{"table":"organism_dbxref","record_id":"Taxonomy\/Organism Linker","mode":"insert","fields":[{"type":"foreign key","title":"Accession
Ref","field":"dbxref_id","show_all_records":0,"foreign key":" Taxonomy ID","foreign field":"dbxref_id","required":0},{"type":"foreign key","title":"Organism
ID","field":"organism_id","show_all_records":0,"foreign key":"Organism","foreign
field":"organism_id","required":0}],"select_if_duplicate":1,"update_if_duplicate":0,"select_optional":0,"disable":0,"optional":0}}
```

Use this JSON array for import of this bulk loader template into another Tripal site.

Cut-and-paste all of the text in the **Export** field and send it to a collaborator.

To import a template that may have been created by someone else, navigate to **Tripal** → **Chado Data Loaders** → **Buik Loader** and click the **Tempalate** tab. A link titled Import Template appears above the table of existing importers. The page that appears when that link is clicked will allow you to import any template shared with you.

1.14 Customizing Your Site

Through the combination of Drupal, Chado and the Tripal API, it is possible to fully customize your site and to add new functionality. Tripal does not store or display all data out-of-the box, and every site has its own different look-and-feel. If you want greater functionality beyond what comes with Tripal please see the *Developer's Handbook* for instructions to use the Drupal and Tripal APIs to develop your own extensions to Tripal.

CHAPTER 2

Developer's Guide

2.1 Introduction to the Tripal API

Tripal provides an Application Programming Interfaces (API) that allows developers to interact with and customize Tripal. Using the API, developers can customize the way data is presented or create custom modules that provide new or different functionality. These custom modules can in turn be shared with the Tripal community. The Tripal API is documented online at http://api.tripal.info/api/tripal/3.x (COMING SOON). This document provides examples and best practices for using the Tripal API.

2.1.1 Requirements

In order to use the Tripal API the developer should have the following skills:

Common skills:

- Knowledge of PHP.
- Knowledge of relational databases and SQL.

To create custom modules:

- Familiarity with Drupal API
- Familiarity with Drupal module development

To use Chado for data storage:

- Knowledge of Chado and relationships between tables (at least tables where data of interest is stored).
- An idea how data is stored in Chado (or a willingness to ask questions)

Things that will make your Tripal development experience positive, fun and rewarding:

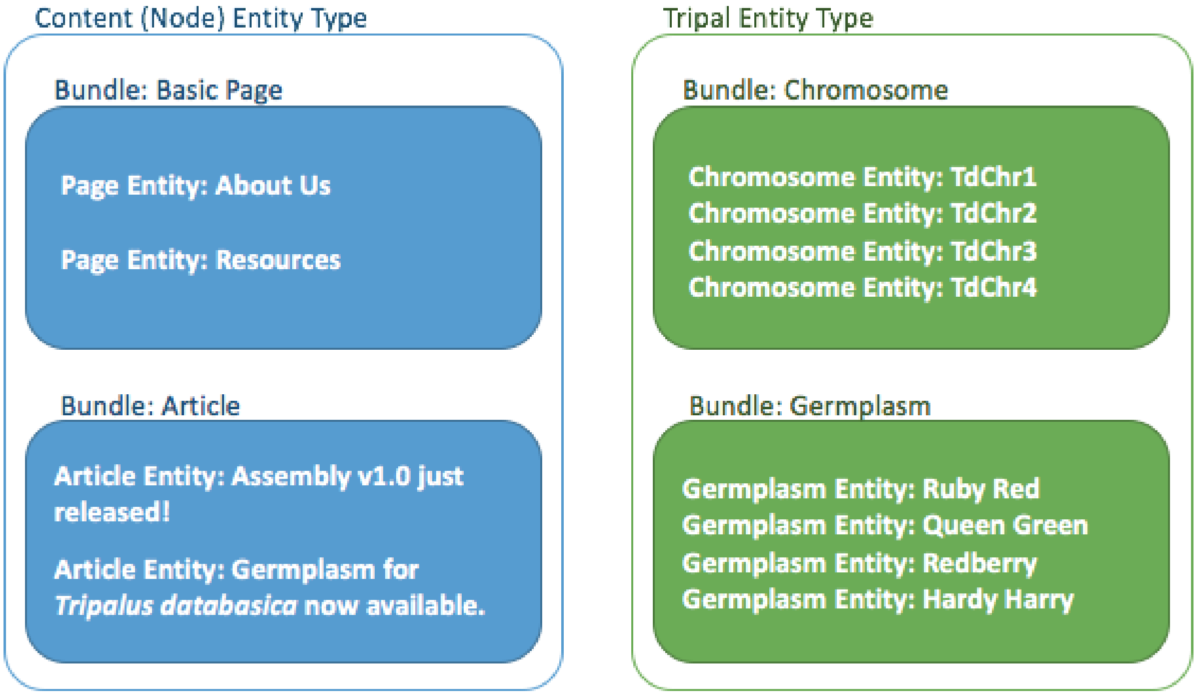
- A desire to write code that can be re-used and shared by other groups
- A desire to share your work with others to support other the Tripal community members.
- A willingness to ask for help on the Tripal Github issue queue if you get stuck, find bugs or desire new features.

2.2 Tripal Data Structures

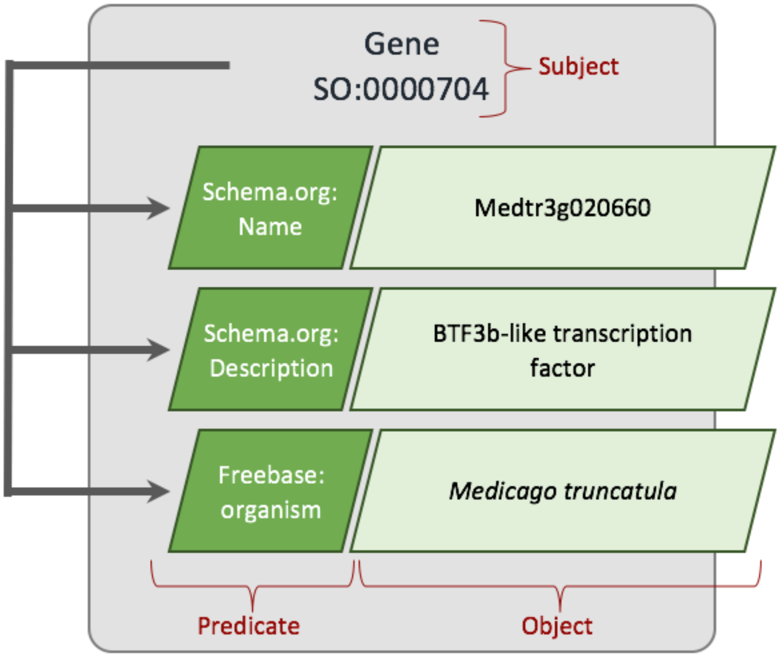
This page explains the relationships between Entity types, Bundles (content type), Entities and Fields. These are data structures provided by Drupal that Tripal v3 uses to expose biological content through your Drupal site, and to provide flexibility in site customization. It is important to understand these terms and their relationships before developing new modules or custom functionality. A quick summary of these Drupal data structures are as follows:

- Entity: a discrete data record. Entities are most commonly are seen as "pages" on a Drupal web site.
- Field: an "atomic" piece of ancillary data that describes, defines or expands the entity. Common fields include the name of an entity, a "body" of descriptive text, etc.
- Bundle: a content type. An entity must always have a content type. Drupal provides several content types by default: Basic Page and Article. Tripal provides biological bundles (e.g. genes, organisms, etc).
- Entity Type: despite the confusing name, an entity type is simply a group of bundles that are somehow related. Drupal provides a "Node" Entity type that includes the Page and Article bundles. All of the Tripal bundles (content types) belong to the TripalEntity Entity type.

The following figure describes the hierarchical relationship between Drupal Entity types (e.g. Node) in comparison with TripalEntity entity types (e.g. Chromosome, Germplasm, etc.).


Content (Node) Entity Type
Bundle: Basic Page
Page Entity: About Us
Page Entity: Resources
Bundle: Article
Article Entity: Assembly v1.0 just released!
Article Entity: Germplasm for *Tripalus databasica* now available.
Tripal Entity Type
Bundle: Chromosome
Chromosome Entity: TdChr1
Chromosome Entity: TdChr2
Chromosome Entity: TdChr3
Chromosome Entity: TdChr4
Bundle: Germplasm
Germplasm Entity: Ruby Red
Germplasm Entity: Queen Green
Germplasm Entity: Redberry
Germplasm Entity: Hardy Harry

Furthermore, fields are "attached" to a Bundle and hold unique values for each Entity. The following figure describes this relationship for a Gene Bundle that has several fields attached: name, description and organism. Note that in this figure the Entity and each of the Fields are defined using a controlled vocabulary term. As a result, bundles and fields can be described using the Semantic Web concepts of a "subject", "predicate" and "object". We will discuss more on controlled vocabularies a bit later in the Handbook.



2.2.1 Bundles (Content Types)

Bundles are types of content in a Drupal site. By default, Drupal provides the Basic Page and Article content types, and Drupal allows a site developer to create new content types on-the-fly using the administrative interface–no programming required. Tripal also provides several Content Type by default. During installation of Tripal the Organism, Gene, Project, Analysis and other content types are created automatically. The site developer can then create new content types for different biological data–again, without any programming required.

In order to to assist with data exchange and use of common data formats, Tripal Bundles are defined using a controlled vocabulary term (cvterm). For example, a "Gene" Bundle is defined using the Sequence Ontology term for gene whose term accession is: SO:0000704. This mapping allows Tripal to compare content across Tripal sites, and expose data to computational tools that understand these vocabularies. You can create as many Tripal Content Types as you would like through Administration > Structure > Tripal Content Types provided you can define it using a controlled vocabulary term.

By default, Tripal uses Chado as its primary data storage back-end. When a bundle is created, the Tripal Chado module allows you to map a Bundle to a table in Chado. Thus, any content type desired can be define as well as how it is stored in Chado–all using the administrative interface.

2.2.2 Entity

An entity is a discrete data record. Entities are most commonly seen as "pages" on a Drupal web site and are instances of a Bundle (i.e content type). When data is published on a Tripal site such as organisms, genes, germplasm, maps, etc., each record is represented by a single entity with an entity ID as its only attribute. All other information that the entity provides is made available via Fields.

2.2.3 Fields

A field is a reusable "data container" that is attached to a Bundle. Programmatically, each field provides one or more primitive data types, with validators and widgets for editing and formatters for display. Each field independently manages the data to which it assigned. Just like with Bundles, Fields are also described using controlled vocabulary terms. For example, a gene Bundle has a field attached that provides the name of the gene. This field only provides the name and nothing more. Tripal uses the schema:name vocabulary term to describe the field.

2.2.4 Field Instances

Fields describe "atomic" units of data that are associated with an entity. For example, a "name" is an atomic unit of data about a Gene or Organism entity. Fields can be reused for multiple Bundles. For example, gene, mRNA, genetic markers and variants all have name data. Despite that all of these Bundles provides a "name", we only need one field to describe that this data is a "name". However, we may want to customize a field specific to each bundle. Therefore, an Instance of a field is attached to a bundle, and field instances can then be customized differently. The most important customization is the one that defines the Chado table from which the data for a field is retrieved. Despite that field instances are attached to bundles, they become visible with Entities. When an entity is loaded for display, Drupal examines all of the fields that are attached to the entity's bundle, and then populates the fields instances with data specific to the entity being loaded.

2.2.5 Entity Types

An entity type is simply a group of Bundles that have some similarity. For examples Drupal provides a Node entity type. The Node entity type contains the Basic Page and Article Bundles. Tripal v2 expanded the Node entity type when creating new content. Tripal v3, however, uses an a new entity type named TripalEntity that provides the Organism, Gene, Analysis, etc. content types. Using these new entity types provides a a more responsive solution then the Node entity type, is more flexible, and supports the new ontology-driven approach of Tripal v3.

2.3 Module Development Best Practice

If you create custom Tripal Modules, here are some best practices and suggestions.

2.3.1 The Drupal Devel Module

Before staring your development work, it is suggested that you download and install the Drupal devel module. This module helps greatly with debugging your custom theme or module. A very useful function of this module is the dpm function. You can use the dpm function to print to the web page an interactive view of the contents of any variable. This can be extremely helpful when accessing Chado data in objects and arrays returned by Tripal.

2.3.2 Add your module to Tripal.info

Add your modules to the Tripal ReadtheDocs *Extension Modules* list. The *Tripal Module Rating System* was designed to give guidance on Tripal module development best practices.

2.3.3 Coding Best Practices

2.3.3.1 Host your code on GitHub

We recommend making your code open source and hosting it on GitHub. It's free, it let's people easily find, use, and contribute to your source code.

2.3.3.2 Associate the GitHub repository with Tripal

Once your module is on GitHub, consider joining the Tripal organization. Your lab group can exist as a team and maintain control over your code, but your projects will be listed in the main Tripal group.

If you'd rather not, you can still tag your project as Tripal by clicking on the Manage Topics Link at the top of your repository.

2.3.3.3 DOIs

When your module is release ready, why not create a Digital Object Identifier (DOI) for it with Zenodo? It's free! Sync your github account and create a new release (Zenodo won't find old releases). You can then display your DOI badge on your module's page.

Additionally, there is a Tripal Community group on Zenodo. You can edit your record to associate your DOI with the Tripal community.

2.3.3.4 Testing and Continuous Integration

Tripal Test Suite is a full-featured testing module that makes writing tests much easier. which will automatically set up a PHPUnit and Travis testing environment for you.

- Test with PHPUnit
- Run tests as you push code with Travis CI

2.3.3.5 Documentation

Every repository can include a README file that will be displayed on the repository page. A README file should at a minimum include:

- An overview of the module
- Instructions on how to install & use the module

Consider documenting your Code itself. Tripal documents in the Doxygen style which allows documentation webpages to be automatically generated. Even if you don't build HTML documentation, the in-line code documentation will be very helpful to contributors.

2.3.3.6 Coding Standards

Drupal has defined coding standards that Tripal modules should meet.

2.4 Accessing Chado

Primarily biological data made available to Tripal is stored in the GMOD Chado schema. As such, you will likely need to interact with Chado at some point. Tripal has developed a number of API functions and classes to make this interaction easier and more generic.

2.4.1 The Chado Query API

Provides an API for querying of chado including inserting, updating, deleting and selecting from specific chado tables. There is also a generic function, `chado_query()`, to execute and SQL statement on chado. It is ideal to use these functions to interact with chado in order to keep your module compatible with both local & external chado databases. Furthermore, it ensures connection to the chado database is taken care of for you.

2.4.1.1 Generic Queries to a specific chado table

Selecting Records

`chado_select_record( [table name], [columns to select], [specify record to select], [options*] )`

This function allows you to select various columns from the specified chado table. Although you can only select from a single table, you can specify the record to select using values from related tables through use of a nested array. For example, the following code shows you how to select the name and uniquename of a feature based on it's type and source organism.

```
$values =  array(
  'organism_id' => array(
    'genus' => 'Citrus',
    'species' => 'sinensis',
  ),
  'type_id' => array (
    'cv_id' => array (
      'name' => 'sequence',
    ),
    'name' => 'gene',
    'is_obsolete' => 0
  ),
);

$result = chado_select_record(
  'feature',                      // table to select from
  array('name', 'uniquename'),    // columns to select
  $values                         // record to select (see variable defn. above)
);
```

Inserting Records

`chado_insert_record( [table name], [values to insert], [options*] )`

This function allows you to insert a single record into a specific table. The values to insert are specified using an associative array where the keys are the column names to insert into and they point to the value to be inserted into that column. If the column is a foreign key, the key will point to an array specifying the record in the foreign table and then the primary key of that record will be inserted in the column. For example, the following code will insert a feature and for the type_id, the cvterm.cvterm_id of the cvterm record will be inserted and for the organism_id, the organism.organism_id of the organism_record will be inserted.

```
$values =  array(
  'organism_id' => array(
      'genus' => 'Citrus',
      'species' => 'sinensis',
   ),
  'name' => 'orange1.1g000034m.g',
  'uniquename' => 'orange1.1g000034m.g',
  'type_id' => array (
      'cv_id' => array (
         'name' => 'sequence',
      ),
      'name' => 'gene',
      'is_obsolete' => 0
   ),
);
$result = chado_insert_record(
  'feature',             // table to insert into
  $values                // values to insert
);
```

Updating Records

`chado_update_record( [table name], [specify record to update], [values to change], [options*] )`

This function allows you to update records in a specific chado table. The record(s) you wish to update are specified the same as in the select function above and the values to be update are specified the same as the values to be inserted were. For example, the following code species that a feature with a given uniquename, organism_id, and type_id (the unique constraint for the feature table) will be updated with a new name, and the type changed from a gene to an mRNA.

```
$umatch = array(
  'organism_id' => array(
    'genus' => 'Citrus',
    'species' => 'sinensis',
  ),
  'uniquename' => 'orange1.1g000034m.g7',
  'type_id' => array (
    'cv_id' => array (
      'name' => 'sequence',
    ),
    'name' => 'gene',
    'is_obsolete' => 0
  ),
);
$uvalues = array(
  'name' => 'orange1.1g000034m.g',
  'type_id' => array (
    'cv_id' => array (
      'name' => 'sequence',
    ),
    'name' => 'mRNA',
    'is_obsolete' => 0
```

(continues on next page)

(continued from previous page)

```
  ),
);
$result = chado_update_record('feature',$umatch,$uvalues);
```

Deleting Records

`chado_delete_record( [table name], [specify records to delete], [options*] )`

This function allows you to delete records from a specific chado table. The record(s) to delete are specified the same as the record to select/update was above. For example, the following code will delete all genes from the organism Citrus sinensis.

```
$values =  array(
  'organism_id' => array(
      'genus' => 'Citrus',
      'species' => 'sinensis',
   ),
  'type_id' => array (
      'cv_id' => array (
         'name' => 'sequence',
      ),
      'name' => 'gene',
      'is_obsolete' => 0
   ),
);
$result = chado_select_record(
   'feature',                      // table to select from
   $values                         // records to delete (see variable defn. above)
);
```

2.4.1.2 Generic Queries for any SQL

Often it is necessary to select from more then one table in chado or to execute other complex queries that cannot be handled efficiently by the above functions. It is for this reason that the `chado_query( [sql string], [arguments to sub-in to the sql] )` function was created. This function allows you to execute any SQL directly on the chado database and should be used with care. If any user input will be used in the query make sure to put a placeholder in your SQL string and then define the value in the arguments array. This will make sure that the user input is sanitized and safe through type-checking and escaping. The following code shows an example of how to use user input resulting from a form and would be called with the form submit function.

```
$sql = "SELECT F.name, CVT.name as type_name, ORG.common_name
         FROM feature F
         LEFT JOIN cvterm CVT ON F.type_id = CVT.cvterm_id
         LEFT JOIN organism ORG ON F.organism_id = ORG.organism_id
         WHERE
           F.uniquename = :feature_uniquename";
$args = array( ':feature_uniquename' => $form_state['values']['uniquename'] );
$result = chado_query( $sql, $args );
foreach ($result as $r) { [Do something with the records here] }
```

If you are going to need more then a couple fields, you might want to use the Chado Variables API (specifically `chado_generate_var()`) to select all of the common fields needed including following foreign keys.

2.4.1.3 Loading of Variables from chado data

These functions, `chado_generate_var()` and `chado_expand_var()`, generate objects containing the full details of a record(s) in chado. These should be used in all theme templates.

This differs from the objects returned by `chado_select_record` in so far as all foreign key relationships have been followed meaning you have more complete details. Thus this function should be used whenever you need a full variable and `chado_select_record` should be used if you only case about a few columns.

The initial variable is generated by the `chado_generate_var([table], [filter criteria], [optional options])` function. An example of how to use this function is:

```
$values = array(
  'name' => 'Medtr4g030710'
);
$features = chado_generate_var('feature', $values);
```

This will return an object if there is only one feature with the name Medtr4g030710 or it will return an array of feature objects if more than one feature has that name.

Some tables and fields are excluded by default. To have those tables & fields added to your variable you can use the `chado_expand_var([chado variable], [type], [what to expand], [optional options])` function. An example of how to use this function is:

```
// Get a chado object to be expanded
$values = array(
  'name' => 'Medtr4g030710'
);

$features = chado_generate_var('feature', $values);

// Expand the organism node
$feature = chado_expand_var($feature, 'node', 'organism');

// Expand the feature.residues field
$feature = chado_expand_var($feature, 'field', 'feature.residues');

// Expand the feature properties (featureprop table)
$feature = chado_expand_var($feature, 'table', 'featureprop');
```

2.4.2 The Chado Schema API

The Chado Schema API provides an application programming interface (API) for describing Chado tables, accessing these descriptions and checking for compliancy of your current database to the chado schema. This API consists of the ChadoSchema class which provides methods for interacting with the Chado Schema API and a collection of supporting functions, one for each table in Chado, which describe each version of the Chado schema. Each function simply returns a Drupal style array that defines the table.

2.4.2.1 Ensuring columns Tables & Columns exist

Generally you can assume the tables and columns in the Chado schema have been unaltered. That said, there are still cases where you might want to check that specific tables and columns exist. For example, when using a custom table, it is best practice to ensure it is there before querying as it can be removed through the administrative interface.

To check the existence of a specific table and column, you can use the following:

```
$chado_schema = new \ChadoSchema();

// Check that the organism_feature_count custom table exists.
$table_name = 'organism_feature_count';
$table_exists = $chado_schema->checkTableExists($table_name);

if ($table_exists) {

  // Check that the organism_feature_count.feature_id column exists.
  $column_name = 'feature_id';
  $column_exists = $chado_schema->checkColumnExists($table_name, $column_name);

  if ($column_exists) {

    [ do your query, etc. here ]

  } else { [warn the admin using tripal_report_error()] }
} else { [warn the admin using tripal_report_error()] }
```

2.4.2.2 Checking the Schema Version

If you are using chado tables specific to a given version of Chado, it is best practice to check the chado version of the current site before querying those tables. You can use the following query to do this:

```
$chado_schema = new \ChadoSchema();
$version = $chado_schema-getVersion();
if ($version == '1.3') {
  [do your chado v1.3 specific querying here]
} else { [warn the admin using tripal_report_error() ] }
```

2.4.2.3 Retrieving a list of tables

To retrieve a list of Chado tables, you can use the following:

```
$chado_schema = new \ChadoSchema();

// All Chado Tables including custom tables
$all_tables = $chado_schema->getTableNames(TRUE);

// All Chado Tables without custom tables
$all_tables = $chado_schema->getTableNames();

// Chado tables designated as Base Tables by Tripal.
$base_tables = $chado_schema->getBaseTables();
```

2.4.2.4 Ensuring your Chado instance is compliant

Checking compliancy of your Chado instance with the released Chado Schema is a great way to **confirm an upgrade has gone flawlessly**. Additionally, while it is not recommended, sometimes customizations to the Chado schema may be necessary. In these cases, you should **ensure backwards compatibility** through compliance checking to confirm Tripal will work as expected.

Chado compliancy testing is provided with Tripal's automated PHPUnit testing. As such, to test compliancy of your specific Chado instance, you first need to install Composer. Luckily this can be as easy as:

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('SHA384', 'composer-setup.php') ===
↪'544e09ee996cdf60ece3804abc52599c22b1f40f4323403c44d44fdfdd586475ca9813a858088ffbc1f233e9b180f061
↪') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-
↪setup.php'); } echo PHP_EOL;"
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

Once you have Composer, you need to install PHPUnit. This is installed locally within your Tripal repository. The following bash snippet shows you how to both install composer locally and run compliance checking.

```
cd [DRUPAL_ROOT]/sites/all/modules/tripal
composer up

# Now run compliance checking
./vendor/bin/phpunit --group chado-compliance
```

2.4.2.5 Schema Definition

To retrieve the schema definition for a specific table, you can execute the following:

```
$table_name = 'feature';
$chado_schema = new \ChadoSchema();
$table_schema = $chado_schema->getTableSchema($table_name);
```

The resulting `$table_schema` variable contains a Drupal-style array describing the schema definition of the table specified by `$table_name`. This is a great tool when trying to develop generic queries, since you can extract information about an unknown table and use it to build a query for that table. For more information on the format of this array, see the Drupal Schema API documentation.

2.5 Creating Custom Modules

Note: This section is under construction

The Tripal API, in conjunction with the Drupal API allow developers to create their own modules that can "plug-in" with Tripal. Developers may create their own modules if they desire new or different functionality. A custom module can also be desired to house a sites "customizations." The Tripal API provides an interface for module developers to interact with the Chado database as well as the other Tripal core functions such as jobs management, materialized views, in-house vocabularies, external database cross-references, properties, etc. An understanding of the Drupal API and Drupal module development is required.

Basic instructions for creating custom Drupal module can be found here: https://www.drupal.org/docs/7/creating-custom-modules. Before you can create custom modules that interact with Tripal, you should review those instructions and have practiced creating at least a simple module. Tripal follows Drupal module development rules and API.

2.6 Creating a Custom Field

The most common way that new content will be added to an existing site is by creating new fields, or field displays. In Tripal v2 customizations were added by editing PHP templates files. These template files were relatively easy to create and customize, but they provided less flexibility and did not integrate well with other Drupal features such as GUI-based page layout and Drupal Views. Tripal v3 fields now provide this flexibility. They also support data exchange and data collections!

By default Tripal v3 provides many fields for display of Chado data. However, you may find that these fields do not display data as you want, or you want to display data that the current fields do not already provide. This section of the Handbook describes how to create new fields that are integrated into the display, search and exchange abilities of both Drupal and Tripal.

If you are already familiar with Drupal fields you may be aware of the API functions and hooks that Drupal provides. However, for the quantity of fields needed to support biological data, the Drupal API hooks quickly become overwhelming. Additionally, documentation for fields in the Drupal API can sometimes be difficult to discover when first working with fields. Therefore, Tripal provides several new PHP classes to simplify creation of fields and to consolidate all functionality into one easy to find set of files. To develop new fields you should be somewhat familiar working with PHP's Object-Oriented Classes. The new classes provided by Tripal are these:

Class Name	Description
TripalField	The TripalField class provides the basic information about a new field. It provides loaders for extracting data from the database and functions for querying data managed by the field.
TripalFieldWidget	The TripalFieldWidget class provides the necessary form elements when editing an Entity to allow the end-user to edit the value of the field (if desired). It provides the necessary validators and submitter functions.
TripalFieldFormatter	The TripalFieldFormatter class provides the visualization of the field when viewed on the page.
ChadoField	The ChadoField class extends the TripalField class and provides the necessary settings to allow the field to map entities to data in Chado
ChadoFieldWidget	Extends the TripalFieldWidget class but currently provides no additional functionality. Use this class when working with Chado data to ensure future backwards compatibility.
ChadoFieldFormatter	Extends the TriplFieldFormatter class but currently provides no additional functionality. Use this class when working with Chado data to ensure future backwards compatibility.

The process for creating a custom field are as follows:

- Determine the controlled vocabulary term that best describes the data your field will create.
- Decide if you need the Chado field classes or the base Tripal field classes. If you intend to work with data housed in Chado then you should use the Chado field classes.
- Decide if you want to build your class manually from the ground up or speed development by using the Staton Lab Fields Generator tool.
- Create new implementations of classes that extend those listed in the table above. If you implement the functions properly your field is plug-and-play! Tripal will find it and be able to use it.

The rest of this section will walk you through these steps.

2.6.1 Selecting Vocabulary Terms

2.6.1.1 Ontologies: what and why?

Tripal 3 requires all bundles and fields to be associated with a Controlled Vocabulary (CV). CVs are dictionaries of defined terms (CV terms) that make data machine-accessible, ensuring uniform terms are used across experiments, organisms and websites. Without CVterms, our scientific knowledge might be split by "dialects". Plant biologists might study temperature stress, while animal biologists study heat shock. Each group might benefit from the knowledge of the other, but they use a different vocabulary to describe the same thing, creating challenges for data discovery and exchange. CV terms make this easier not just for people, but especially for machines. Ontologies take this a step further. Where CVs are controlled lists of CVterms, ontologies are a controlled language, that include hierarchical relationships of terms.

Tripal leverages vocabularies to make use of the Semantic Web. Every bundle and field defined in Tripal will be associated with a CVterm. Therefore, it is important to find community developed terms. The EMBL EBI Ontology Lookup Service provides an easy location to search for and identify terms. When choosing terms for new Bundles and Fields, think carefully about the terms you will use to describe your objects. Selecting the proper CV term that best describes the data may be the most challenging part of creating custom Bundles and Fields!

Before you can create a new Bundle or Field the vocabulary term must be present in your local Tripal site. You can check if a term exists by using the Tripal lookup service on your local site using the URL path cv/lookup (e.g. http://your-site/cv/lookup). If the term is not present then you'll need to add it. You can do so manually by using Tripal's controlled vocabulary admin pages. For creating new bundles this is all you need to do. However, when creating Fields you will want to programmatically add the term. This is important because Fields are meant to be shared. If you create an awesome field that you want to share with others then you need to make sure the terms get added programmatically. The following sections describe how terms are stored in Chado and how you can add them using Tripal API calls.

2.6.1.2 Storage of Terms in Chado

In Chado, CVs are stored by two tables: the **db** and **cv** tables. Chado was designed to store a record for the online database that a vocabulary lives at in the **db** table, and the namespace of a vocabulary in the **cv** table. For example, the sequence ontology uses the namespace, sequence, which is stored in the **cv** table but uses the short name of SO which is stored in the **db** table. As we'll see later, sometimes the distinction between what gets stored in the **cv** vs the **db** tables can get a bit fuzzy with some vocabularies. The terms themselves are stored in the cvterm table. The cvterm table has a foreign key to the **cv** table via the **cv_id** field. Every controlled vocabulary term has an accession. For example the term gene in the Sequence Ontology has an accession number of SO:0000704. Accession numbers consist of two parts: a vocabulary "short name", followed by a unique identifier separated by a colon. Within Chado, the accession for any term is stored in the dbxref table. This table has a foreign key to the **db** table via the **db_id** as well as a foreign key to the cvterm table via the **dbxref_id** field.

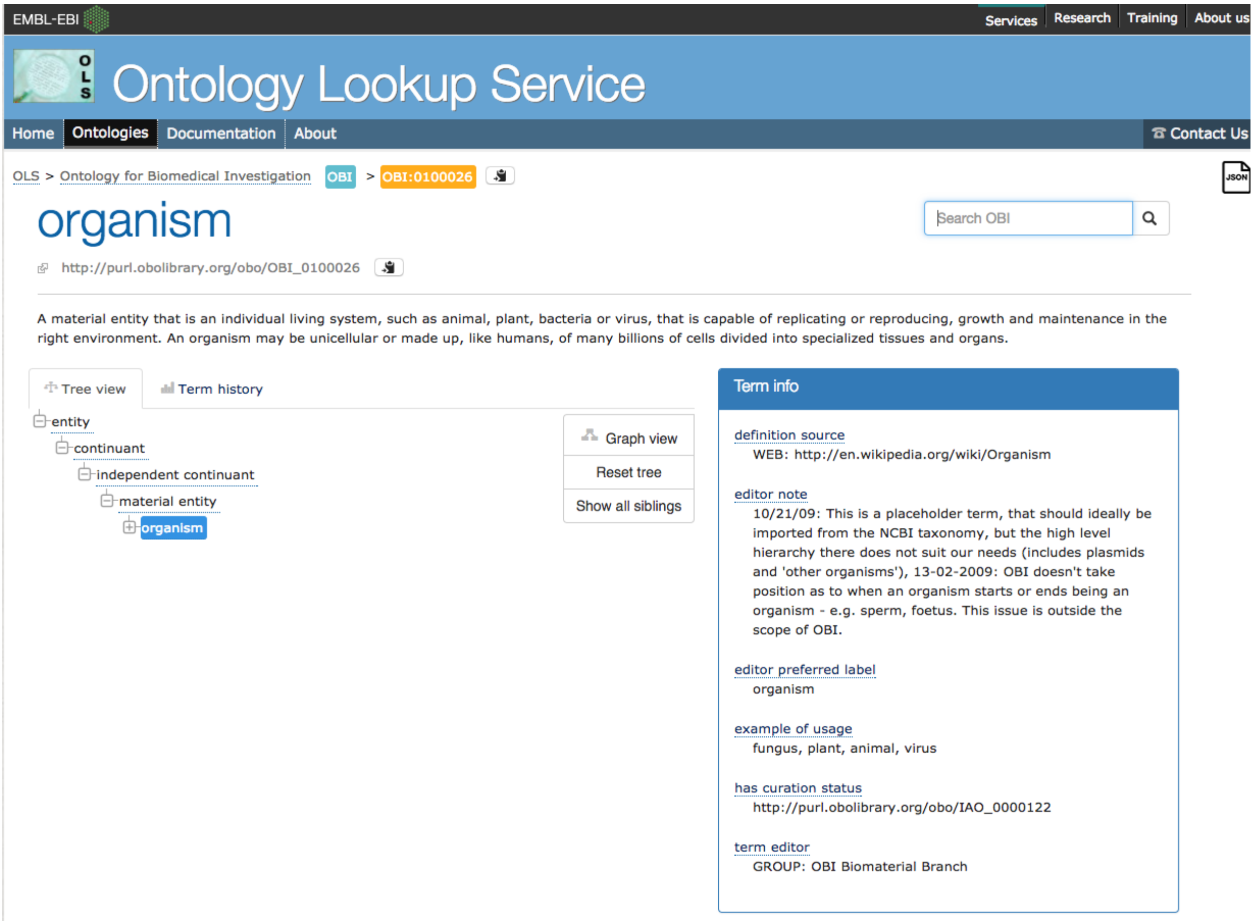
2.6.1.3 Vocabulary Short Names and Namespaces

How can you tell what the **short name** and **namespace** values will be for a vocabulary term that you want to insert into Chado for your custom Bundle or Field? Hint: use the information is in the EMBL-EBI Ontology Lookup Service (OLS). The following sections provide three examples for different cases.

Case 1: Ontologies without a defined namespace

Consider the term for organism.

Notice how the teal box (the **short name**) is OBI, and the orange box contains the **full accession**, OBI:0100026 which includes but the **short name** and the unique term **accession** value. Unfortunately, the OLS does not indicate the **namespace** terms. So, as a rule we will use the short name converted to lower case. Before using this term in a Tripal Bundle or Field you may need to insert this term into Chado. You can do so in your custom module code using the **tripal_insert_cvterm** function. The following provides a demonstration:



organism

http://purl.obolibrary.org/obo/OBI_0100026

A material entity that is an individual living system, such as animal, plant, bacteria or virus, that is capable of replicating or reproducing, growth and maintenance in the right environment. An organism may be unicellular or made up, like humans, of many billions of cells divided into specialized tissues and organs.

definition source
WEB: http://en.wikipedia.org/wiki/Organism

editor note
10/21/09: This is a placeholder term, that should ideally be imported from the NCBI taxonomy, but the high level hierarchy there does not suit our needs (includes plasmids and 'other organisms'), 13-02-2009: OBI doesn't take position as to when an organism starts or ends being an organism - e.g. sperm, foetus. This issue is outside the scope of OBI.

editor preferred label
organism

example of usage
fungus, plant, animal, virus

has curation status
http://purl.obolibrary.org/obo/IAO_0000122

term editor
GROUP: OBI Biomaterial Branch

```
$term= tripal_insert_cvterm([
         'id' => 'OBI:0100026',
         'name' => 'organism',
         'cv_name' => 'OBI',
         'definition' => 'A material entity that is an individual living system, such␣
↪as animal,
         plant, bacteria or virus, that is capable of replicating or reproducing,␣
↪growth and maintenance
               in the right environment. An organism may be unicellular or made up,␣
↪like humans, of many
            billions of cells divided into specialized tissues and organs.',
     ]);
```

Note that in the code above the namespace is provided as the **cv_name** element and the full accessions (including the short name) is provided as the **id** element. In this case the OBI CV already exists by default in the Tripal database, so we did not need to add the vocabulary record. If the OBI did not exist we could have added it using the following API calls. First we insert the "database" record for the ontology.

```
tripal_insert_db(array(
  'name' => 'obi',
  'description' => 'The Ontology for Biomedical Investigation.',
  'url' => 'http://obi-ontology.org/page/Main_Page',
  'urlprefix' => 'http://purl.obolibrary.org/obo/{db}_{accession}',
));
```

Notice here that the **name** element is the **namespace** (short name converted to lower case) for the vocabulary. The url is the web address for the ontology online. The urlprefix is a URL that can be used to construct a link that when clicked will take the user to any term in the vocabulary. Almost all vocabularies will have a common URL for all terms. Tripal will automatically substitute the short name into the **{db}** token and the term **accession** in to the **{accession}** token to generate the URL.

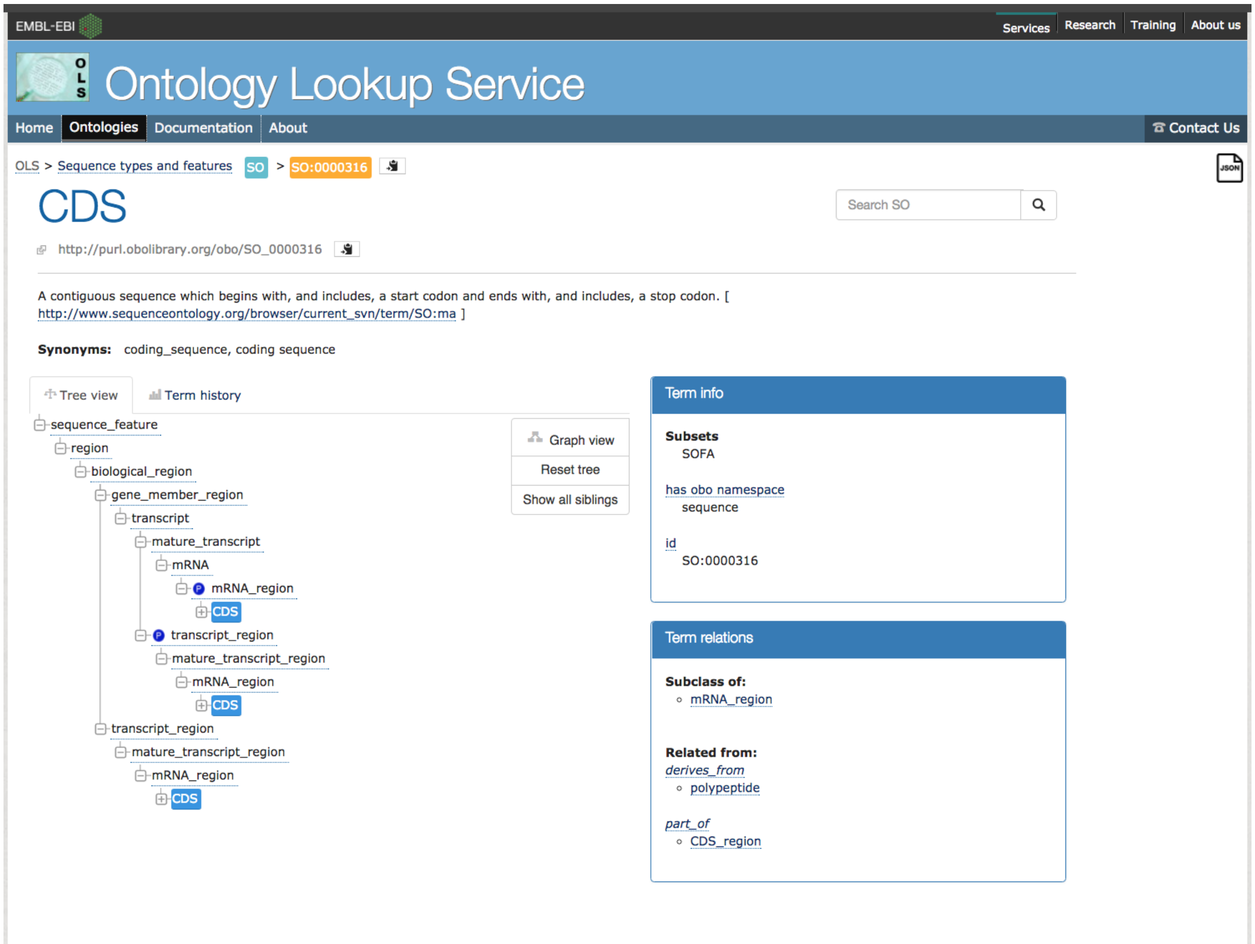
Second, we insert the record for the controlled vocabulary.

```
      tripal_insert_cv(
      'OBI',
      'Ontology for Biomedical Investigation. The Ontology for Biomedical␣
↪Investigations (OBI) is build in a collaborative, international effort and will␣
↪serve as a resource for annotating biomedical investigations, including the study␣
↪design, protocols and instrumentation used, the data generated and the types of␣
↪analysis performed on the data. This ontology arose from the Functional Genomics␣
↪Investigation Ontology (FuGO) and will contain both terms that are common to all␣
↪biomedical investigations, including functional genomics investigations and those␣
↪that are more domain specific.'
);
```

Case 2: Ontologies with a defined namespace

Consider the entry for CDS.

Notice that in the Term Info box on the right there is the term **has_obo_namespace** which is defined as the word: sequence. This is much better than the organism example from the OBI. We now know the correct namespace for the term! By default, Tripal loads the Sequence Ontology during install. However, suppose we did not have this term loaded we could do so with the following:



CDS

http://purl.obolibrary.org/obo/SO_0000316

A contiguous sequence which begins with, and includes, a start codon and ends with, and includes, a stop codon. [http://www.sequenceontology.org/browser/current_svn/term/SO:ma]

Synonyms: coding_sequence, coding sequence

Term info

Subsets
SOFA

has obo namespace
sequence

id
SO:0000316

Term relations

Subclass of:
- mRNA_region

Related from:
derives_from
- polypeptide

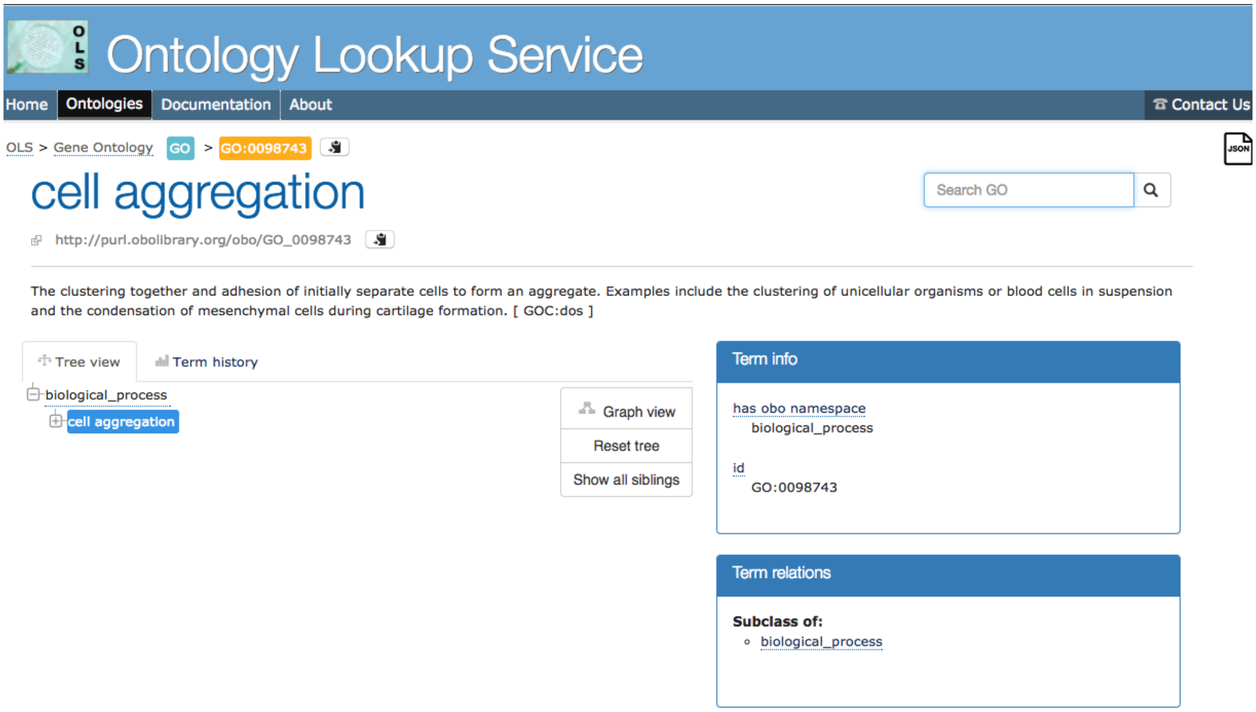
part_of
- CDS_region

```
$term= tripal_insert_cvterm([
        'id' => 'SO:0000316',
        'name' => 'CDS',
        'cv_name' => 'sequence',
        'definition' => 'A contiguous sequence which begins with, and includes, a␣
↪start codon and ends with, and includes, a stop codon. [ http://www.
↪sequenceontology.org/browser/current_svn/term/SO:ma ].',
    ]);
```

Notice in the code above we can properly set the cv_name to sequence.

Case 3: Ontologies with multiple namespaces

Some ontologies are b into sub-ontologies. This includes the Gene Ontology (GO). Let's consider the example GO term cell aggregation. Looking at the EBI entry, the teal box is GO, the orange box is GO:0098743, and the has_obo_namespace is biological_process. However, the GO provides two other namespaces: cellular_component and molecular_function. Be sure to pay attention to these different namespaces if you ever need to manually insert a term.



cell aggregation

http://purl.obolibrary.org/obo/GO_0098743

The clustering together and adhesion of initially separate cells to form an aggregate. Examples include the clustering of unicellular organisms or blood cells in suspension and the condensation of mesenchymal cells during cartilage formation. [GOC:dos]

Term info

has obo namespace
biological_process

id
GO:0098743

Term relations

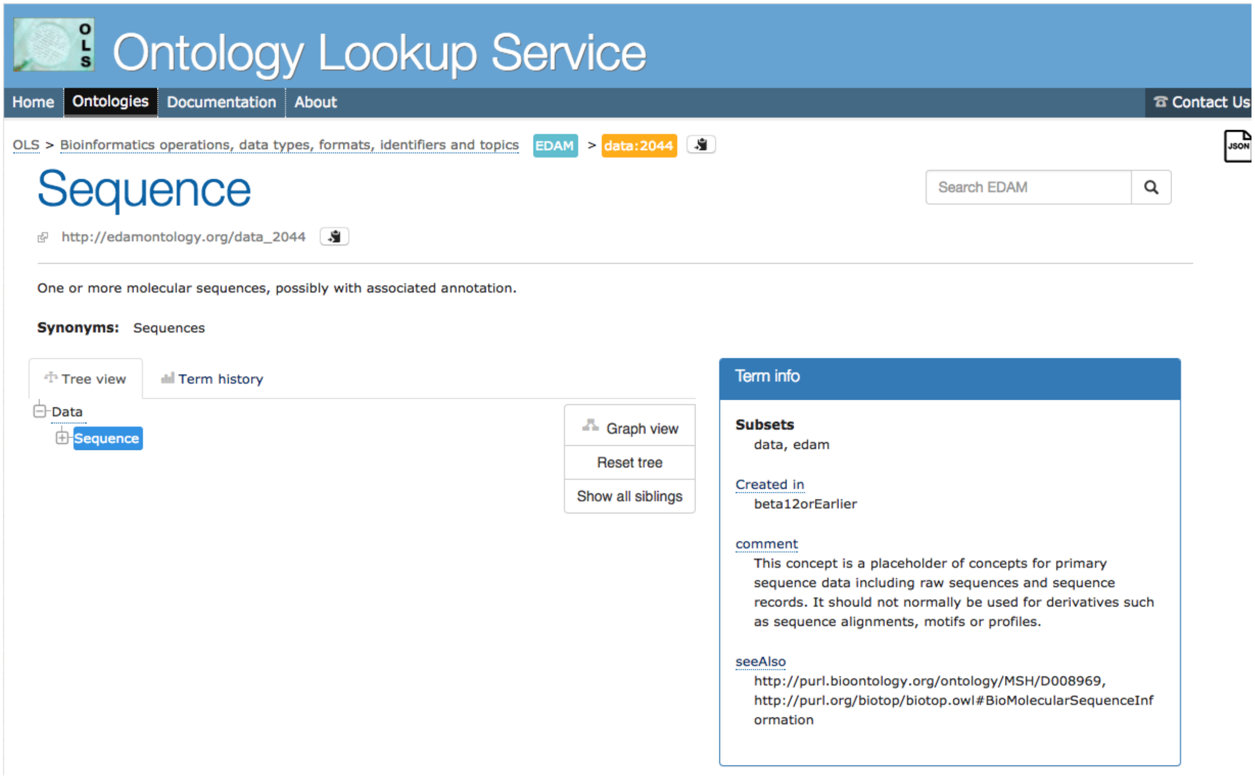
Subclass of:

- biological_process

Case 4: Ontologies with multiple short names

The EDAM ontology builds its term accessions using different short names instead of the ontology. Consider the EDAM term for Sequence. The teal box is EDAM, the orange box is data:2044, and there is no **namespace**.

For this case, the **namespace** is EDAM, the short name is **data**, and the accession is 2044. Unfortunately, this breaks the paradigm that Chado expects. Typically the **short name** is the teal box (EDAM). In order to force Chado to



Ontology Lookup Service

Home | Ontologies | Documentation | About | Contact Us

OLS > Bioinformatics operations, data types, formats, identifiers and topics EDAM > data:2044

Sequence

http://edamontology.org/data_2044

Search EDAM

One or more molecular sequences, possibly with associated annotation.

Synonyms: Sequences

Tree view | Term history

Data
Sequence

Graph view
Reset tree
Show all siblings

Term info

Subsets
data, edam

Created in
beta12orEarlier

comment
This concept is a placeholder of concepts for primary sequence data including raw sequences and sequence records. It should not normally be used for derivatives such as sequence alignments, motifs or profiles.

seeAlso
http://purl.bioontology.org/ontology/MSH/D008969, http://purl.org/biotop/biotop.owl#BioMolecularSequenceInformation

properly handle ontologies like this we are forced to reverse the short name and **namespace** values when creating our record:

```
$term= tripal_insert_cvterm([
  'id' => 'data:2044',
  'name' => 'sequence',
  'cv_name' => 'EDAM',
  'definition' => 'One or more molecular sequences, possibly with associated␣
↪annotation.',
]);

tripal_insert_db(array(
    'name' => 'data',
    'description' => 'Bioinformatics operations, data types, formats, identifiers and␣
↪topics.',
    'url' => 'http://edamontology.org/page',
    'urlprefix' => 'http://edamontology.org/{db}_{accession}',
));

tripal_insert_cv(
    'EDAM',
    'EDAM is an ontology of well established, familiar concepts that are prevalent␣
↪within bioinformatics, including types of data and data identifiers, data formats,␣
↪operations and topics. EDAM is a simple ontology - essentially a set of terms with␣
↪synonyms and definitions - organised into an intuitive hierarchy for convenient use␣
↪by curators, software developers and end-users. EDAM is suitable for large-scale␣
↪semantic annotations and categorization of diverse bioinformatics resources. EDAM␣
↪is also suitable for diverse application including for example within workbenches␣
↪and workflow-management systems, software distributions, and resource registries.'
);
```

Case 5: You really cant find a term!

Sometimes a good CVterm just doesn't exist for what you want to describe. If you can't find a CV term, you can insert a term into the "local" CV. This is meant to be used as a last resort. In these cases, before you use a local term, consider contributing the term to an existing CV or ontology. Any terms that are invented for a local site may mean that the data exposed by your site cannot be discovered by other sites or tools. In this case, the accession will not be numeric, but is the same as the term name.

```
$term= tripal_insert_cvterm([
   'id' => 'local:shame_on_you',
   'name' => 'shame_on_you',
   'cv_name' => 'local',
   'definition' => 'You should really find a good CVterm.',
]);
```

Notice in the above code the **short name** and **namespace** are both "local" as this is a local term on the site.

2.6.2 Manual Field Creation

To show how a TripalField works we will break down a class implementation section by section. Here we will use the **obi__organism** field that comes with Tripal and which extends the ChadoField class. The ChadoField class is almost identical to the TripalField class except that it provides a few extra settings for working with Chado tables. To create your own class you need to create a new class that implements the necessary functions.

Note: Creation of your first field may not seem easy! The following document is a lot to think about and consider. Therefore, when you write your first field, don't try to do everything at once. Take it one piece at a time. The variables and functions described here are in order with the most critical components described first. Take it at an even pace.

2.6.2.1 Directory Structure for Fields

Before we create our class we must first create a proper directory structure. Tripal expects that all new Tripal field classes are located inside of a custom module in the following directory structure:

```
/sites/all/modules/[your_module]/includes/TripalFields/[field_name]/[field_name].inc
/sites/all/modules/[your_module]/includes/TripalFields/[field_name]/[field_name]_
↪widget.inc
/sites/all/modules/[your_module]/includes/TripalFields/[field_name]/[field_name]_
↪formatter.inc
```

In the directories above the token [your_module] can be substituted with the name of your module and [field_name] is the name of your field. You can name your field whatever you like, but you must use this name consistently in other locations throughout the modules. Because all fields are defined by vocabulary terms, it is custom to name your fields with the vocabulary **short name** followed by two underscores followed by the **term name**, hence: obi__organism. Here the ChadoField implementation goes in the [field_name].inc file, the ChadoFieldWidget in the [field_name]_widget.inc file and the ChadoFieldFormatter in the [field_name]_formatter.inc. All new fields must implement all three classes. in the case of our obi__organism field the directory structure is as follows:

```
/sites/all/modules/tripal/tripal_chado/includes/TripalFields/obi__organism/obi__
↪organism.inc
/sites/all/modules/tripal/tripal_chado/includes/TripalFields/obi__organism/obi__
↪organism_widget.inc
/sites/all/modules/tripal/tripal_chado/includes/TripalFields/obi__organism/obi__
↪organism_formatter.inc
```

(continues on next page)

(continued from previous page)

2.6.2.2 Anatomy of the ChadoField Class

The following describes a ChadoField class from top to bottom. The code for the obi__organism field is shown in order that it appears in the class with descriptions provided for the meaning of each piece of code. To write your own class, duplicate the variables and function and customize accordingly. First, let's look at the definition of the class. The following line defines the class and indicates that it extends the ChadoField class:

```
class obi__organism extends ChadoField {
```

Note: In the line above, the class is named obi__organism. This must be the same name as the directory in which the field is located. Otherwise, Tripal won't be able to find the field.

2.6.2.3 Static Member Variables

Next, the TripalField/ChadoField class has a section of public static variables. These are variables that you can customize to describe your field to Tripal. Here you will provide the default label that appears for the field, and a description for the field:

```
// The default label for this field.
public static $default_label = 'Organism';

// The default description for this field.
public static $description = 'The organism to which this resource is associated.';
```

As described in the section titled Tripal Data Structures, fields that are attached to Bundles are "instances" of a field. Every field instance can be customized differently on each bundle. The following section of code allows your field to provide custom settings. Here we want to "hard-code" the term that defines this field using the $default_instance_settings variable:

```
// Provide a list of instance specific settings. These can be accessed within
// the instanceSettingsForm.  When the instanceSettingsForm is submitted
// then Drupal will automatically change these settings for the instance.
// It is recommended to put settings at the instance level whenever possible.
// If you override this variable in a child class be sure to replicate the
// term_name, term_vocab, term_accession and term_fixed keys as these are
// required for all TripalFields.
public static $default_instance_settings  = array(
  // The short name for the vocabulary (e.g. schema, SO, GO, PATO, etc.).
  'term_vocabulary' => 'OBI',
  // The name of the term.
  'term_name' => 'organism',
  // The unique ID (i.e. accession) of the term.
  'term_accession' => '0100026',
  // Set to TRUE if the site admin is allowed to change the term
  // type. This will create form elements when editing the field instance
  // to allow the site admin to change the term settings above.
  'term_fixed' => FALSE,
  // The format for display of the organism.
  'field_display_string' => '<i>[organism.genus] [organism.species]</i>',
);
```

Notice in the code above that the elements **term_vocabulary, term_name** and **term_accession** are used to define the vocabulary term that this field maps to. The term_fixed element allows the term to be changed by the site admin if desired. These elements are required of all TripalFields classes. You must always have these elements. However, the **field_display_string** is a variable unique to this obi__organism field! Because this field is displaying the organism we want to allow the site-admin to customize how the organism name is constructed and displayed. Therefore, the **field_display_string** creates this new setting for us. How this setting is used will be described later.

As you may have noticed, a field requires a widget and a formatter. This is why there are three classes for every field. However, Drupal is flexible and allows fields to be edited or displayed by any number of widgets and formatters. By default, Tripal provides one widget class and one formatter class for every field. When you write a new field you will need to do the same and create a new ChadoFieldWidget and ChadoFieldFormatter class (or the corresponding non-Chado versions if you don't need Chado). The following variables in the class indicate what are the default widget and formatter classes (we have not yet created those, but we know their names!):

```
// The default widget for this field.
public static $default_widget = 'obi__organism_widget';

// The default formatter for this field.
public static $default_formatter = 'obi__organism_formatter';
```

Drupal allows new instances of fields to be attached to any Bundle. This is really useful for fields like the built in Image field that Drupal provides. It can be very handy to attache an instance of an Image field to any content type and viola! your content type now supports images. However, there are some fields that should never be added via the online Drupal interface. Our organism field is a good example. We probably don't want to allow end-users to add an organism field to a Person content type... In this case we will programmatically control which fields are attached to which Bundles. We'll show that later. But for now, let's set the no_ui variable to TRUE to prevent users from adding our new field to any Bundle.

```
// A boolean specifying that users should not be allowed to create
// fields and instances of this field type through the UI. Such
// fields can only be created programmatically with field_create_field()
// and field_create_instance().
public static $no_ui = TRUE;
```

Sometimes a field is meant to provide a visualization or some other functionality. An example of this might be a small search form or link to an analytical service. In these cases we want the field to show up on the web page but it should not appear anywhere else, such as in Tripal's web service that provides access to all content. We can set the no_data variable to TRUE and this will allow it to be seen on the site, but not anywhere else.

```
// A boolean specifying that the field will not contain any data. This
// should exclude the field from web services or downloads.  An example
// could be a quick search field that appears on the page that redirects
// the user but otherwise provides no data.


public static $no_data = FALSE;
```

Note: Be sure to only set this to TRUE when you are absolutely certain the contents would not be needed in web services. Tripal was designed so that what appears on the page will always appear in web services. Aside from the formatting we see on the website, the content should be the same.

Finally, the last item in our Class variables is the **download_formatters**. Tripal provides an API that allows tools to group entities into data collections. Data collections are like "baskets" or "shopping carts". Entities that are in data collections can be downloaded into files. If your field is compatible with specific file downloaders you can specify those here. A file downloader is a special TripalFieldDownloader class that "speaks" certain file formats. Tripal, by default, provides the TripalTabDownloader (for tab-delimited files), the TripalCSVDownloader (for CSV files), a TripalNucFASTADownloader for creating nucleotide FASTA files and a TripalProteinFASTADownloader for protein FASTA files. If your field is compatible with any of these formatters you can specify them in the following array:

If your field is compatible with the TripalTabDownloader, for example, your field will be included as a column in a tab delimited file where each row represents contents for a given entity.

2.6.2.4 The load() function.

The first function we want to implement in our class is the load() function. This function is responsible for querying the database and populating the field value. Data that is loaded into the field must be organized in two ways: 1) a value that is visible to the end-users, and 2) values that are visible to Chado for ensuing update/editing of the correct record in Chado when the field is edited. Our obi__organism field is designed to be used for multiple Bundles therefore the code in our load() function must be able to support any Chado table that has a foreign key relationship with the organism table.

To get started, the load() function receives a single argument. The entity object:

```
public function load($entity) {
```

Because this is a ChadoField and the TripalChado module supports this field and maps entities to their “base” record on Chado, we get something extra… we get the record itself

```
$record = $entity->chado_record;
```

Having the record helps tremendously. Our **obi__organism** field is meant to be attached to genomic feature content types (e.g. genes, mRNA, etc.), germplasm, etc. Therefore, the entity will be a record of one of those types. In the case of a genomic feature, these come from the **feature** table of Chado. In the case of germplasm, these records come from the **stock** table of Chado. Both of these records have an **organism_id** field which is a foreign key to the organism table where we find out details about the organism.

Before we set the values for our field, we need a little bit more information. Remember that all field instances have settings? The Tripal Chado module also populates for us the name of the Chado table and the column that this field maps to. Our obi__organism field can be used for multiple Bundles. A gene bundle would map to the **feature** table of Chado and a germplasm Bundle would map to the **stock** table. We need to know what table and column this field is mapping to: We can get that from the instance object of the class and its settings:

```
$settings = $this->instance['settings'];
$field_table = $this->instance['settings']['chado_table'];
$field_column = $this->instance['settings']['chado_column'];
```

Next, we want to get this field name and its type. We obviously know our field name, it is obi__organism. However, we can get the name programmatically as well. Drupal maintains an “informational” array about our field. Inside of that field array we can find lots of interesting information such as our field name and its type (Bundle). We’ll need this when we set our field value. But rather than hard-code it, let’s grab it programmatically from the field name. It’s best to grab it programmatically because there are cases where the field name could change:

```
$field_name = $this->field['field_name'];
$field_type = $this->field['type'];
```

Now, let’s plan how we want our values to appear in our field. The organism record of Chado v1.3 has a genus, species, abbreviation, infraspecific name, infraspecific type, and a common name. We want these values exposed to the end user. But, wait… when we discussed fields in the Tripal Data Structures section we learned about a name field that provides names for entities. That field only has one value: the name. Our organism field has multiple values (i.e. genus, species, etc.). A field can provide more than just one value but values have to be qualified. We have to provide values in key/value pairs, and the keys must be controlled vocabulary terms. We must use controlled vocabulary terms because we want our field to be searchable by other Tripal sites. For example, the ontology term for the word 'genus' comes from the TAXRANK vocabulary. Fortunately, almost every column of every table in Chado has been mapped to a controlled vocabulary term so we don't need to go hunting for terms. We can use a Chado API function that Tripal provides for getting the ontology terms associated with every column table in Chado. The following code shows these functions retrieving the ontology terms for our values from the organism table:

```
// Get the terms for each of the keys for the 'values' property.
$label_term = 'rdfs:label';
$genus_term = tripal_get_chado_semweb_term('organism', 'genus');
$species_term = tripal_get_chado_semweb_term('organism', 'species');
$infraspecific_name_term = tripal_get_chado_semweb_term('organism', 'infraspecific_
↪name');
$infraspecific_type_term = tripal_get_chado_semweb_term('organism', 'type_id');
```

Notice that for our organism fields we can easily get the ontology terms for them using the API function **tripal_get_chado_semweb_term**. You will also notice a **label_term** variable. Sometimes a user may want to see the full name of the organism and not pieces of it in various elements. Therefore, we will provide a label in our list of values that will concatenate the full organism name. This field is not in our organism table so we hard-code the term 'rdfs:label' which is a term from the Resource Data Framework Schema vocabulary that defines a label.

Next, let's initialize our field's value to be empty. When setting a field value we must do so in the entity object that got passed into our load function. The entity is an object and it stores values using the names of the fields. The following code sets an empty record for our field:

```
// Set some defaults for the empty record.
$entity->{$field_name}['und'][0] = array(
  'value' => array(),
);
```

Notice that our field has some sub elements. The first is 'und'. This element corresponds to the "language" of the text. Drupal supports multiple spoken languages and wants to know the language of text we provide. For Tripal fields we always use 'und' meaning 'undefined'. The next element is the delta index number. Fields have a cardinality, or in other words they can have multiple values. For every value we add we increment that index, always starting at zero. The last element is our 'value' element and it is here where we put our element. You may notice that our **delta** index is hard coded to 0. This is because an entity can only always have one organism that it is associated with. We will never have more than one.

Now that we've got some preliminary values and we've initialized our value array we can start adding values! Before we do though, let's double check that we have a record. If we don't have a record for this entity, we can't get a value.

```
if ($record) {
```

Now if we do have a record we need to get the value The first step is to actually get our organism record. For this we will find the record variable to be really handy. It already comes pre-populated with every Chado record that has a foreign-key relationship with our base record. So, in the case of a gene, the record is stored in the feature table which has an organism_id column which is a foreign key to the organism table. So, we know then that our record object has an organism_id property and we can get our organism from that. The only exception is the biomaterial table which uses a field named taxon_id:

```
if ($field_table == 'biomaterial') {
  $organism = $record->taxon_id;
}
else {
  $organism = $record->organism_id;
}
```

We can easily get all of the values we need from this organism object. We can now access the values for this organism using the Chado organism table column names (e.g. $organism->genus, $organism->species).

```
$label = chado_replace_tokens($string, $organism);
$entity->{$field_name}['und'][0]['value'] = array(
  $label_term => $label,
  $genus_term => $organism->genus,
  $species_term => $organism->species,
);
// The infraspecific fields were introduced in Chado v1.3.
if (property_exists($organism, 'infraspecific_name')) {
  $entity->{$field_name}['und'][0]['value'][$infraspecific_type_term] = NULL;
  $entity->{$field_name}['und'][0]['value'][$infraspecific_name_term] = $organism->
↪infraspecific_name;
  if ($organism->type_id) {
    $entity->{$field_name}['und'][0]['value'][$infraspecific_type_term] =  $organism->
↪type_id->name;
  }
}
```

In the code above we are populating our value array and we're using the controlled vocabulary terms we retrieved earlier as the keys.

Okay, so, we have our values set. However, remember, our fields must support two types of values: 1) those for end users; and 2) those that allow us to save values in Chado if the field is edited. If you look at our value array above you will recognize that the entity to which this field is loading data for is for a feature or stock or library, etc. This field represents the organism for a record from one of those tables. If someone wants to edit the entity and change the organism then effectively we need to change the organism_id of that table. But in our values array we don't have the organism_id we only have data about the organism. How will Tripal know how to change the organism for an entity if edited? To do help Tripal out, we have to create special key/value pair to add to our values. These are values that are not meant to be seen by the end-user. The organism_id is a good example of such a value. To create these values we create a key with a special naming scheme: use "chado-" as a prefix, followed by the table name (e.g. feature), followed by two underscores and finally the column name (e.g. organism_id). The following code shows the creation of this value name:

```
// Set the linker field appropriately.
if ($field_table == 'biomaterial') {
  $linker_field = 'chado-biomaterial__taxon_id';
}
else {
  $linker_field = 'chado-' . $field_table . '__organism_id';
}
```

If our entity were of type "gene" then our **field_table** is feature. Therefore, our **linker_field** variable would be **chado-feature__organism_id**. Next, we need to add this to our value:

```
$entity->{$field_name}['und'][0][$linker_field] = $organism->organism_id;
```

Notice, though, that we did not add this value inside the 'value' key like we did above for our end-user, such as the following:

```
$entity->{$field_name}['und'][0]['value'])
```

Instead, we put it in at the same level as 'value':

```
$entity->{$field_name}['und'][0][$linker_field]
```

We do this because anything in the 'value' element is intended for the end-user. Anything outside of the 'value' is meant for Tripal. Adding the organism ID to this field as a Tripal "hidden" value allows Tripal to recognize where these values really came from. When writing your own fields, you must include any values as "hidden" Tripal values that need to be written to the database table. A good way to remember if your value should be visible to the end-user or hidden for Tripal is to ask yourself these questions:

1. Does the user need this value? If yes, put it in the 'value' element.
2. Does Tripal need the value when writing back to the Chado table? If yes, put it as a hidden element.
3. Does the user need to see the value and will this same value need to be written to the table? If yes, then you have to put the value in both places.

For our **obi__organism** field it is for entities with records in the **feature, stock, library**, etc. tables. Those tables only have an **organism_id** to represent the organism. So, that's the database column this field is supporting. We therefore, need to put that field as a hidden field, and all the others are just helpful to the user and don't get saved in the feature, stock or library tables. So, those go in the values array.

Now, we're at a good stopping point with our field! We can close out our if($record) statement and the function:

```
    }
}
```

2.6.2.5 elementInfo() function

The elementInfo() function is necessary to integrate your new field with Drupal Views and Tripal Web Services. Drupal needs to know what data elements your field provides and Tripal needs to know what vocabulary terms to use for each of the data elements. Related to vocabulary terms, all fields are assigned an ontology term for the field itself. Every field has to have an one. But when a field provides more than just a single data value it must also provide vocabulary terms for any sub elements as well. Our obi__organism field provides the genus, species, etc. sub elements and, therefore, we need to describe these to Drupal and Tripal. The elementInfo() function from the obi_organism field is as follows:

```
/**
 * @see TripalField::elementInfo()
 */
public function elementInfo() {
  $field_term = $this->getFieldTermID();

  $genus_term = chado_get_semweb_term('organism', 'genus');
  $species_term = chado_get_semweb_term('organism', 'species');
  $infraspecific_name_term = chado_get_semweb_term('organism', 'infraspecific_name');
  $infraspecific_type_term = chado_get_semweb_term('organism', 'type_id');

  return array(
    $field_term => array(
      'operations' => array('eq', 'contains', 'starts'),
      'sortable' => TRUE,
      'searchable' => TRUE,
      'readonly' => FALSE,
      'type' => 'xs:complexType',
      'elements' => array(
        'rdfs:label' => array(
          'searchable' => TRUE,
          'name' => 'scientific_name',
          'operations' => array('eq', 'ne', 'contains', 'starts'),
          'sortable' => FALSE,
          'type' => 'xs:string',
          'readonly' => TRUE,
```

(continues on next page)

(continued from previous page)

```
          'required' => FALSE,
        ),
        $genus_term => array(
          'searchable' => TRUE,
          'name' => 'genus',
          'operations' => array('eq', 'ne', 'contains', 'starts'),
          'sortable' => TRUE,
          'readonly' => FALSE,
          'type' => 'xs:string',
          'required' => TRUE,
        ),
        $species_term => array(
          'searchable' => TRUE,
          'name' => 'species',
          'operations' => array('eq', 'ne', 'contains', 'starts'),
          'sortable' => TRUE,
          'readonly' => FALSE,
          'type' => 'xs:string',
          'required' => TRUE,
        ),
        $infraspecific_name_term => array(
          'searchable' => TRUE,
          'name' => 'infraspecies',
          'operations' => array('eq', 'ne', 'contains', 'starts'),
          'sortable' => TRUE,
          'readonly' => FALSE,
          'type' => 'xs:string',
          'required' => FALSE,
        ),
        $infraspecific_type_term => array(
          'searchable' => TRUE,
          'name' => 'infraspecific_type',
          'operations' => array('eq', 'ne', 'contains', 'starts'),
          'sortable' => TRUE,
          'readonly' => FALSE,
          'type' => 'xs:integer',
          'required' => FALSE,
        ),
        'entity' => array(
          'searchable' => FALSE,
        ),
      ),
    ),
  );
}
```

The code above generates and returns an associative array that provides metadata about the field and its elements. The array is structured such that the first-level key is the term for the field. Details about the field are at the second-level and all sub elements are contained in a 'elements' key. In the following code the terms for the field and sub elements are retrieved using TripalField class functions and Tripal API calls:

```
$field_term = $this->getFieldTermID();

$genus_term = chado_get_semweb_term('organism', 'genus');
$species_term = chado_get_semweb_term('organism', 'species');
$infraspecific_name_term = chado_get_semweb_term('organism', 'infraspecific_name');
```

(continues on next page)

(continued from previous page)

```
$infraspecific_type_term = chado_get_semweb_term('organism', 'type_id');

return array( $field_term => array(
  'operations' => array('eq', 'contains', 'starts'),
  'sortable' => TRUE,
  'searchable' => TRUE,
  'readonly' => FALSE,
  'type' => 'xs:complexType',
  'elements' => array(
```

Notice the value for $field_term variable was easily obtained by calling the $this->getFieldTermID function and all of the terms for the elements were obtained using the chado_get_semweb_term function which maps table columns in the Chado database schema to ontology terms. The operations key indicates which search filter operations are supported for the field as a whole. For this example these include 'eq' (for equals), 'contains' and 'starts' (for starts with). The field is sortable and searchable so those values are set to TRUE. Later, we will learn how to implement the sorting, searching and filtering that the field will support. For now we know we want them so we set the values accordingly. Additionally, the field allows updating so 'readonly' is set to FALSE. By convention, the 'type' of a field follows the XML data types for simple types (https://www.w3schools.com/xml/schema_simple.asp) and Complex types (https://www.w3schools.com/xml/schema_complex.asp) that have multiple elements. Because our obi__organism field has subelements and is not a single value, the field type is 'xs:complexType'.

The array keys just mentioned fully describe our field to Drupal and Tripal. Next we will define the sub elements in the same way, and these go in the 'elements' key. First, we will describe the label. Our obi__organism field provides a handy label element that concatenates the genus, species and infraspecific name into one simple string. Therefore, we need to describe this element in the same way we described the field itself. In the code below that the key is set to 'rdfs:label' (which is the controlled vocabulary term for a label) and that the child keys are the same as for the field.

```
'elements' => array(
  'rdfs:label' => array(
    'searchable' => TRUE,
    'name' => 'scientific_name',
    'operations' => array('eq', 'ne', 'contains', 'starts'),
    'sortable' => FALSE,
    'type' => 'xs:string',
    'readonly' => TRUE,
    'required' => FALSE,
  ),
```

Notice that our field will allow searching, provides a variety of search filter options, is sortable and defines the type as 'xs:string'. The remaining elements follow the same pattern. As another example, here is the genus element:

```
$genus_term => array(
   'searchable' => TRUE,
   'name' => 'genus',
   'operations' => array('eq', 'ne', 'contains', 'starts'),
   'sortable' => TRUE,
   'readonly' => FALSE,
   'type' => 'xs:string',
   'required' => TRUE,
 ),
```

The major difference in the code above is that the term is provided by the variable $genus_term.

Finally, our obi__organism field provides an 'entity' element that provides information for a published organism entity. We do not provide any filtering, searching or sorting of those values. So the final element appears as:

```
'entity' => array(
  'searchable' => FALSE,
),
```

In summary, you will always want to describe your field and every element of your field in the array returned by the elementInfo function. However, you do not need to provide sorting, filtering or querying for every element. If your field is read-only and simply provides values you should still describe these elements but you would set the meta data keys appropriately for the behavior of your field. Also, you only need to describe elements in the values array returned by your load function. Remember, there may be other key/value pairs (such as those used to help coordinate inserts/updates into Chado) but those do not need to be described here because they are never seen by the end-user.

2.6.2.6 query() function

As described above in the elementInfo function section, some fields and elements of fields are searchable. if the elementInfo array indicates that the field is searchable and has operations (i.e. filters) then we must provide a way for those queries to occur. This is where the query() function is needed. The following is example code from the query function of our obi__organism field:

```
public function query($query, $condition) {
    $alias = $this->field['field_name'];
    $operator = $condition['operator'];

    $field_term_id = $this->getFieldTermID();
    $genus_term = $field_term_id . ',' . chado_get_semweb_term('organism', 'genus');
    $species_term = $field_term_id . ',' . chado_get_semweb_term('organism', 'species
↪');
    $infraspecific_name_term = $field_term_id . ',' . chado_get_semweb_term('organism
↪', 'infraspecific_name');
    $infraspecific_type_term = $field_term_id . ',' . chado_get_semweb_term('organism
↪', 'type_id');

    // Join to the organism table for this field.
    $this->queryJoinOnce($query, 'organism', $alias, "base.organism_id = $alias.
↪organism_id");

    // If the column is the field name then we're during a search on the full
    // scientific name.
    if ($condition['column'] == $field_term_id or
        $condition['column'] == $field_term_id . ',rdfs:label') {
      if (chado_get_version() <= 1.3) {
        $query->where("CONCAT($alias.genus, ' ', $alias.species) $operator :full_name
↪",  array(':full_name' => $condition['value']));
      }
      else {
        $this->queryJoinOnce($query, 'cvterm', $alias . '_cvterm', 'base.
↪infraspecific_type = ' . $alias . '_cvterm.type_id', 'LEFT OUTER');
        $query->where("CONCAT($alias.genus, ' ', $alias.species, ' ', " . $alias . "'_
↪cvterm.name', ' ', $alias.infraspecific_name) $operator :full_name",  array(':full_
↪name' => $condition['value']));
      }
    }

    // If the column is a subfield.
    if ($condition['column'] == $species_term) {
      $query->condition("$alias.species", $condition['value'], $operator);
```

(continues on next page)

(continued from previous page)

```
    }
    if ($condition['column'] == $genus_term) {
      $query->condition("$alias.genus", $condition['value'], $operator);
    }

    if ($condition['column'] == $infraspecific_name_term) {
      $query->condition("$alias.infraspecific_name", $condition['value'], $operator);
    }

    if ($condition['column'] == $infraspecific_type_term) {
      $this->queryJoinOnce($query, 'cvterm', 'CVT', "base.type_id = CVT.cvterm_id");
      $query->condition("CVT.name", $condition['value'], $operator);
    }
  }
```

The code above is how the field tells Drupal and Tripal how to find and filter the records that this field corresponds to. First, we retrieve the field alias and operators, and as with the load and elementInfo functions we get the controlled vocabulary terms for our field and field elements:

```
$alias = $this->field['field_name'];
$operator = $condition['operator'];

$field_term_id = $this->getFieldTermID();
$genus_term = $field_term_id . ',' . chado_get_semweb_term('organism', 'genus');
$species_term = $field_term_id . ',' . chado_get_semweb_term('organism', 'species');
$infraspecific_name_term = $field_term_id . ',' . chado_get_semweb_term('organism',
↪'infraspecific_name');
$infraspecific_type_term = $field_term_id . ',' . chado_get_semweb_term('organism',
↪'type_id');
```

Next, our knowledge of Chado is needed. We know that our obi__organism field will load data from the organism table. Therefore, our search must occur there.

2.6.3 Creating a Custom Widget

In Drupal/Tripal terminology, **widget** refers to the form elements for a specific Tripal Field on the "Edit" form of a piece of Tripal Content. For example, the `obi__organism` field widget creates the "Organism" drop-down on the edit form of a gene. All fields come with a default widget; however, you can create a custom widget if the default one doesn't meet your needs.

Note: This guide assumes you already have your widget class file created. For more information, see *Manual Field Creation* or, *Easier Field Creation: Tripal Field Generator*.

Note: If you are only creating a widget and not the whole field, you still need to follow the expected directory structure. For example, if your widget is going to be named `obi__organism_fancy` then your file would be `[your_module]/includes/TripalField/obi__organism_fancy/obi__organism_fancy_widget.inc`.

2.6.3.1 The Form

The form elements of your widget are defined in the `form()` method of your widget according to the Drupal Form API. As such the `$widget` variable is actually a nested associative array describing what the widget portion of the form should look like. For example, the following is how the `obi__organism` widget creates the drop-down.

```
/**
 * @see TripalFieldWidget::form()
 */
public function form(&$widget, &$form, &$form_state, $langcode, $items, $delta,
↪$element) {

  $field_name = $this->field['field_name'];
  $field_table = $this->instance['settings']['chado_table'];
  $linker_field = 'chado-' . $field_table . '__organism_id';

  // The value presented to the user via load.
  // If $items['delta']['value'] is set then we are updating and already have this
  // information. As such, simply save it again.
  $widget['value'] = array(
    '#type' => 'value',
    '#value' => array_key_exists($delta, $items) ? $items[$delta]['value'] : '',
  );

  // Pull out the value previously saved to be used as the default.
  $organism_id = 0;
  if (count($items) > 0 and array_key_exists($linker_field, $items[0])) {
    $organism_id = $items[0][$linker_field];
  }

  // Define a drop-down form element where the options are organisms retrieved using
  // the Tripal API, the default is what we looked up above, and the title and
  // description are those set when defining the field.
  $widget[$linker_field] = array(
    '#type' => 'select',
    '#title' => $element['#title'],
    '#description' => $element['#description'],
    '#options' => chado_get_organism_select_options(FALSE),
    '#default_value' => $organism_id,
    '#required' => $element['#required'],
    '#delta' => $delta,
  );

}
```

At a minimum, the form must have a `value` element. For Tripal, the `value` element of a field always corresponds to the value that is presented to the end-user either directly on the page (with formatting) or via web services, or some other mechanism. Convention is to store the value of the field as a hidden `value` form element as is shown in the above example.

Note: For more information on how to use the Drupal Form API, check out the official Drupal Documentation.

Note: The current item is saved in `$items[$delta]` as an array where the keys will match those set by the field `load()` function.

2.6.3.2 Validation

The `validate()` function of your widget allows you to confirm that the values entered by the user are valid. It is recommended to consider each form element you created above and consider what is required for that element to be entered "correctly". For example, for an organism drop-down, the organism chosen must exist in our chado database (since this is a `ChadoFieldWidget`). Luckily this doesn't need to be validated since Drupal ensures only elements in our select list are chosen.

Warning: The `value` key of this field must be set in the `$form_state['values']` array to a **TRUE** value (e.g. a string or non-zero integer) anytime data is entered by the user.

Note: For more information on how to validate your data, see the official Drupal Form Validation Documentation

2.6.3.3 Saving the Data

The Drupal Storage Backend handles saving of your widget data. As such, **you do not and should not insert, update or delete the data yourself**. It should happen automatically, assuming you've followed the conventions of the specific storage backend.

Chado Fields utilize the chado storage backend to save your data. Thus to ensure your data is saved, you set the columns of your chado table to the values you want them set via the `$form_state['values']` array using the `chado-[table]__[column]` convention. This should be done at the end of the validation function above, if the data submitted is valid.

For our `obi__organism` example, the drop-down returns the chado organism_id of the record chosen by the user. We would like to save that as the organism_id of the chado table the field references, which the following code specifies.

```php
/**
 * @see TripalFieldWidget::validate()
 */
public function validate($element, $form, &$form_state, $langcode, $delta) {

  $field_name = $this->field['field_name'];
  $field_table = $this->instance['settings']['chado_table'];
  $linker_field = 'chado-' . $field_table . '__organism_id';

  //...
  // Validate your data here
  //...

  // In this case, if you have an organism_id, then your user selected this field.
  $organism_id = $form_state['values'][$field_name]['und'][0][$linker_field];
  if ($organism_id > 0) {
    $form_state['values'][$field_name]['und'][0]['value'] = $organism_id;
    // This is where we tell the storage backend what we want to save.
    // Specifically, that we want to save $organism_id to $field_table.organism_id
    $form_state['values'][$field_name]['und'][$delta][$linker_field] = $organism_id;
  }
}
```

But what do you do if the record you want to link to via foreign key constraint doesn't yet exist? Luckily the Chado Storage API has a solution for this as well. Consider the example of the `sbo__relationship_widget`. When this widget is on the create form for a given content type, we will first need to create the base record before we can create a relationship to it. This is done by setting the values you do know (e.g. chado-feature__type_id and chado-feature__object_id) but not setting the column mapping to the base record. The Chado Storage API will then fill it in automatically once the base record is created.

```
/**
 * @see TripalFieldWidget::validate()
 */
public function validate($element, $form, &$form_state, $langcode, $delta) {

  $field_name = $this->field['field_name'];
  $field_table = $this->instance['settings']['chado_table'];
  $linker_field = 'chado-' . $field_table . '__organism_id';

  //...
  // Validate your data here
  //...

  //...
  // Determine the subject_id, object_id and type_id based on user input.
  // User input is found in $form_state['values'].
  //...

  // If we have all the keys then set the columns as in the obi__organism ex.
  if ($subject_id && $object_id && $type_id) {
    // Set all chado fields to their values.
  }
  // Otherwise, maybe we are creating the entity...
  // The storage API should handle this case and automatically add the key in // once␣
↪the chado record is created... so all we need to do is set the
  // other columns.
  elseif ($subject_name && $object_id && $type_id) {
    $form_state['values'][$field_name][$langcode][$delta]['value'] = 'value must be␣
↪set but is not used';
    $form_state['values'][$field_name][$langcode][$delta]['chado-' . $field_table . '_
↪_' . $object_id_key] = $object_id;
    $form_state['values'][$field_name][$langcode][$delta]['chado-' . $field_table . '_
↪_type_id'] = $type_id;
    // Notice that the subject_id is not set here.
  }
  // Otherwise, we don't have a value to insert so leave them blank.
  else {
    // Set all chado fields to empty string.
  }
```

Drupal typically does not provide a submit hook for fields because, as mentioned above, saving should be done by the storage backend. However, the TripalField provides a TripalFieldWidget::submit() to allow for behind-the-scenes actions to occur. This function should never be used for updates, deletes or inserts for the Chado table associated with the field as these actions should be handled by the storage backend.

However, it is permissible to perform inserts, updates or deletions within Chado using this function. Those operations can be performed if needed but on other tables not directly associated with the field. An example is the chado.feature_synonym table. The chado_linker__synonym field allows the user to provide a brand new synonym and it must add it to the chado.synonym table prior to the record in the chado.feature_synonym table.

2.6.4 Creating a Custom Formatter

The third component of a field is the formatter. Thus far we have introduced how to create a field class and a widget class for a field. The field class is responsible for describing the field, loading data into it, and providing search support. The widget class provided a Drupal form for online editing of the field. Finally, the formatter is responsible for display of the field on a Tripal site.

Note: This guide assumes you already have your formatter class file created. For more information, see *Manual Field Creation* or, *Easier Field Creation: Tripal Field Generator*.

The formatter class is the simplest of all the Tripal field classes. Here we will again use the **obi__organism** field that comes with the `tripal_chado` module.

2.6.4.1 The view() function.

In most cases the only function you need to implement is the `view()` function. This function is called whenever your field needs to be displayed on a page. The following code is from the `obi__organism_formatter.inc` class file.

```
public function view(&$element, $entity_type, $entity, $langcode, $items, $display) {

  if ($items[0]['value']) {
    $content = $items[0]['value']['rdfs:label'];
    if (array_key_exists('entity', $items[0]['value'])) {
      list($entity_type, $entity_id) = explode(':', $items[0]['value']['entity']);
      $content = l(strip_tags($items[0]['value']['rdfs:label']), 'bio_data/' . $entity_id);
    }

    // The cardinality of this field is 1 so we don't have to
    // iterate through the items array, as there will never be more than 1.
    $element[0] = array(
      '#type' => 'markup',
      '#markup' => $content,
    );
  }
}
```

In the code above the input arguments have the following meaning:

- `$element` is the first argument. It is an array into which you should set the contents to be displayed.
- `$entity_type` will always have the value `Tripal Entity`.
- `$entity` is the entity object which contains all information about the entity including the loaded data values.
- `$langcode` is the language. This is used by Drupal to provide translations of data into other spoken languages. By default, Tripal does not use a language, as biological data is generally language agnostic. Consider for example a gene sequence or a feature coordinate.
- `$items` is an array containing all of the loaded data for this field.
- `$display` is the name of the display such as full page, a teaser, etc. Currently, Tripal does not distinguish between displays.

The purpose of the `view()` function is to iterate through the values in the `$items` array, and format them into an appropriate display for viewing. Here you must remember the structure of the data in the `$items` array.

To demonstrate this function, let's look at what we expect in our `$items` array. Using the *Citrus sinesis* organism from the User's Guide. We would expect an items array to look like the following:

```
$items = [
  0 => [
    "value" => [
      "rdfs:label" =>  "Citrus sinensis",
      "rdfs:type" =>  "Organism",
      "local:abbreviation" =>  "C. sinensis",
      "TAXRANK:0000005" => "Citrus",
      "TAXRANK:0000006" => "sinensis",
      "entity" => "TripalEntity:3",
    ],
    "chado-feature__organism_id" => 12,
  ],
];
```

You may recall that the `$items` array structure is the same as that created by the `load()` function described in the *Manual Field Creation* page. Note that each key in the `value` array is an accession for a controlled vocabulary term. These accessions are used to unambiguously describe the value. To display the organism on a page we need the element named `rdfs:label`. Thus, we set the `$content` variable to contain this value as shown on line 4 of the `view()` function above.

Because our organisms are also published entities we want to link to their respective pages each time an organism is displayed. Because the `value` array has an element named `entity` we know that this item is published. Lines 5-6 of the `view()` function shown above use this information to create a clickable link to the organism page. Finally, the `$element` argument is set to provide content of type `markup`. This `$element` array is a Drupal renderable array.

Lastly, notice the element named `chado-feature__organism_id`. This element is at the same level as the `value` element. This data is meant to be used internally by the field. It maps this fields values to the appropriate table in Chado where the data is stored.

Warning: You should never show the user any data that is outside of `value` element. Remember that your field can be shown by other viewers, including web services. By ensuring that data in the `value` element is mean to be displayed we ensure that information on the web page, web services, or any other future form of display is always consistent.

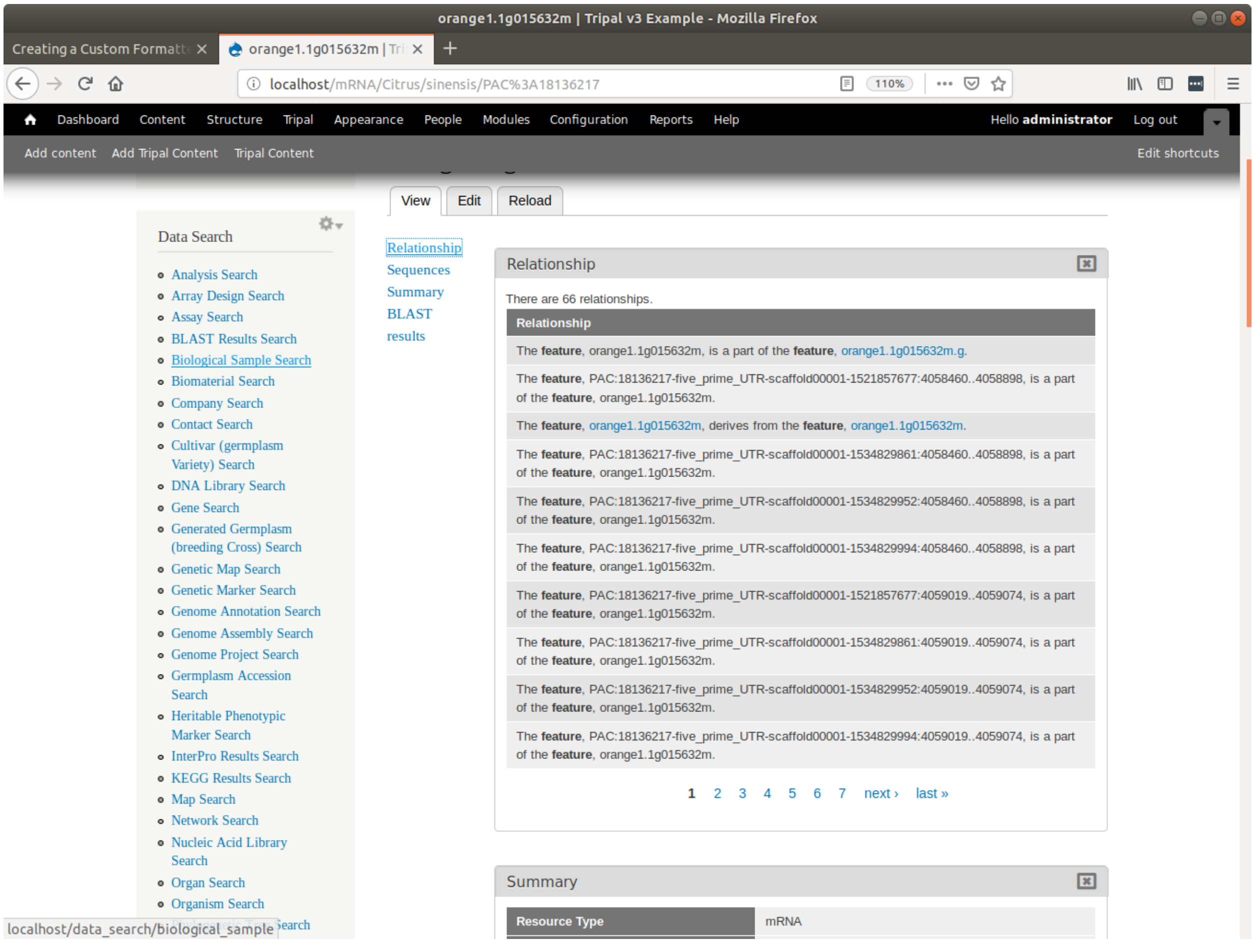
In summary, the following should be observed when processing the `$items` array for viewing:

- A field with only one value (a cardinality of 1) will always have only one element in the `$items` array and can use the index 0. This is what has been done in this example code.
- A field with more than one value can have any number of elements in the `$items` array. You should therefore iterate through all of them.
- For every index in `$item` you should create a matching index in `$element` to display the data found in that `$item`.
- If there are no items, then nothing you return will be displayed.
- For each element in the `$items` array there is a `value` key. Only the data in the `value` key should be shown to the user.
- Each element in the `$items` array may have more than a `value` key. These values are meant to help manage the data.

> **Warning:** You should never have SQL statements or any API calls that retrieve data in the formatter `view()` function. The formatter should strictly format data for viewing.

2.6.4.2 Creating Pagers

The example shown in the previous section was for a field that will always only contain a single element. However some fields may contain a large number of elements. Consider an mRNA and it's relationships to subfeatures: exons, 5' UTRs, 3'UTRs, CDS, etc.). A large mRNA can have many relationships. Alternatively, consider the case where a genetic map content type may have a field that lists all of the markers on the map. Such a list could become extremely long on the page. In these cases it may be best to only list a few items at a time and to provide a pager to let the user cycle through the items. An example of a pager added to the bottom of relationships is shown in the example below.



Relationship

There are 66 relationships.

Relationship
The **feature**, orange1.1g015632m, is a part of the **feature**, orange1.1g015632m.g.
The **feature**, PAC:18136217-five_prime_UTR-scaffold00001-1521857677:4058460..4058898, is a part of the **feature**, orange1.1g015632m.
The **feature**, orange1.1g015632m, derives from the **feature**, orange1.1g015632m.
The **feature**, PAC:18136217-five_prime_UTR-scaffold00001-1534829861:4058460..4058898, is a part of the **feature**, orange1.1g015632m.
The **feature**, PAC:18136217-five_prime_UTR-scaffold00001-1534829952:4058460..4058898, is a part of the **feature**, orange1.1g015632m.
The **feature**, PAC:18136217-five_prime_UTR-scaffold00001-1534829994:4058460..4058898, is a part of the **feature**, orange1.1g015632m.
The **feature**, PAC:18136217-five_prime_UTR-scaffold00001-1521857677:4059019..4059074, is a part of the **feature**, orange1.1g015632m.
The **feature**, PAC:18136217-five_prime_UTR-scaffold00001-1534829861:4059019..4059074, is a part of the **feature**, orange1.1g015632m.
The **feature**, PAC:18136217-five_prime_UTR-scaffold00001-1534829952:4059019..4059074, is a part of the **feature**, orange1.1g015632m.
The **feature**, PAC:18136217-five_prime_UTR-scaffold00001-1534829994:4059019..4059074, is a part of the **feature**, orange1.1g015632m.

1 2 3 4 5 6 7 next › last »

Resource Type	mRNA

To create a pager we first need to calculate the number of items we want to display per page and the total number of pages required to display all of the data.

```
$items_per_page = 10;
$total_records = count($items);
$total_pages = (int) ($total_records / $items_per_page) + 1;
```

Next, we must initialize the pager by calling the `pager_default_initialize` function. We pass it the total number of records, the number of items per page and the index (i.e. `$pelement`) for this pager on the page.

```
$pelement = 0;
$current_page = pager_default_initialize($total_records, $items_per_page, $pelement);
```

The call to `pager_default_initialize` will return the current page. The current page is a numeric number indicating which page the pager is currently showing. The first time the page is loaded this will always be the first page. Each time the user navigates to other pages by clicking the "next" link or the numeric links then this `view()` function is called and the current page is set to the page being viewed. Next, we must theme the pager so that it follows the look-and-feel prescribed for the site. For this we use the Drupal `theme()` function.

```
$pager = theme('pager', array(
  'tags' => array(),
  'element' => $pelement,
  'parameters' => array(),
  'quantity' => $total_pages,
));
```

By default, all links in the pager cause the page to reload. We do not want the page to reload, rather we only want to update the contents of the field. The TripalFieldFormatter class provides a function named `ajaxifyPager` to convert a pager into an AJAX pager:

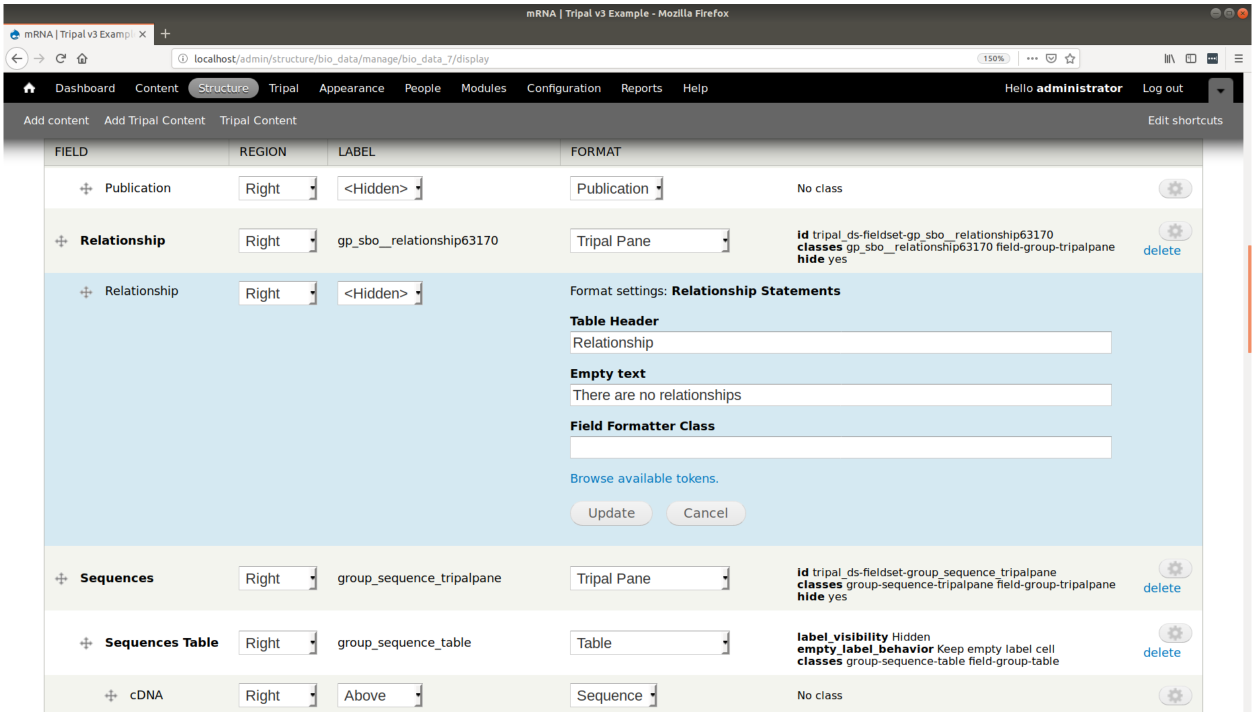
```
$pager = $this->ajaxifyPager($pager, $entity);
```

Now that we have a pager, it has been setup for AJAX and we know the current page that the user is viewing we can now display only the items from the `$items` array that are appropriate for the page being viewed. A common way to provide multiple items on a page is within a table. When we set the `$element` array we need to be sure to provide both the content and the pager:

```
$element[0] = array(
  '#type' => 'markup',
  '#markup' => $content . $pager,
);
```

2.6.4.3 The settingsForm() Function.

Sometimes you may want to provide some control to the site developer for the formatter. For example, the `sbo__relationship_formatter` allows the site developer to customize the title that appears above the table that houses relationships and the text the appears if there are no relationships. By default the title is "Relationships" and the empty text indicates there are no relationships. Both are a bit too generic. The `settingsForm()` function allows you to provide a Drupal form for the field that appears on the **Administer > Structure > Tripal Content Types** on any content type's **manage display** page:



FIELD	REGION	LABEL	FORMAT		
Publication	Right	<Hidden>	Publication	No class	
Relationship	Right	gp_sbo__relationship63170	Tripal Pane	**id** tripal_ds-fieldset-gp_sbo__relationship63170 **classes** gp_sbo__relationship63170 field-group-tripalpane **hide** yes	delete
Relationship	Right	<Hidden>	Format settings: **Relationship Statements** **Table Header** Relationship **Empty text** There are no relationships **Field Formatter Class** Browse available tokens. Update Cancel		
Sequences	Right	group_sequence_tripalpane	Tripal Pane	**id** tripal_ds-fieldset-group_sequence_tripalpane **classes** group-sequence-tripalpane field-group-tripalpane **hide** yes	delete
Sequences Table	Right	group_sequence_table	Table	**label_visibility** Hidden **empty_label_behavior** Keep empty label cell **classes** group-sequence-table field-group-table	delete
cDNA	Right	Above	Sequence	No class	

The form shown in the screenshot above is provided by the `settingsForm()` function. The following code generates this form:

```
public function settingsForm($view_mode, $form, &$form_state) {

  $display = $this->instance['display'][$view_mode];
  $settings = $display['settings'];
  $element = array();
  $element['title'] = array(
    '#type' => 'textfield',
    '#title' => 'Table Header',
    '#default_value' => array_key_exists('title', $settings) ? $settings['title'] : 'Relationship',
  );
  $element['empty'] = array(
    '#type' => 'textfield',
    '#title' => 'Empty text',
    '#default_value' => array_key_exists('empty', $settings) ? $settings['empty'] : 'There are no relationships',
  );

  return $element;
}
```

The form is typical of any form. Note, however that the `#default_value` is set using the current settings values.

A settings form is useful but it only works when Drupal knows what settings you want for your field. You must provide the settings names (e.g. "title" and "empty" in this case) when you attach your field to a given content type (i.e. bundle). You tell Drupal to attach this field to a content type using the `hook_bundle_instances_info` function. See the *Attach Fields to Content Types* to learn more about this function. Briefly, the `display` section of the info array for the `sbo__relationship` field contains the following settings for the `display`:

```
'display' => array(
```

(continues on next page)

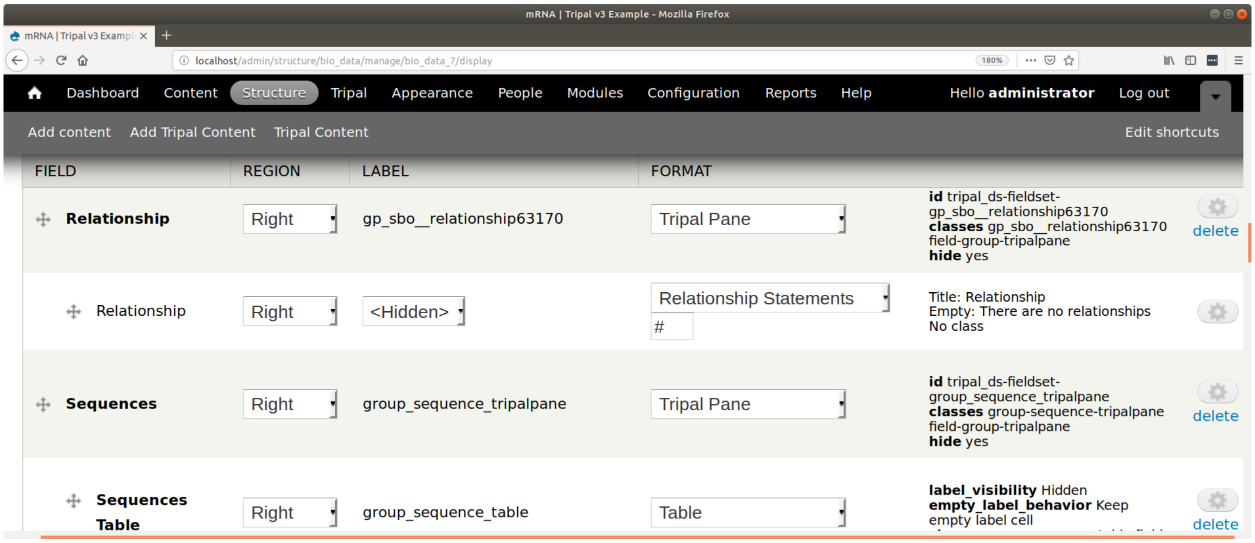
(continued from previous page)

```
  'default' => array(
    'label' => 'hidden',
    'type' => 'sbo__relationship_formatter',
    'settings' => array(
      'title' => 'Relationships',
      'empty' => 'There are no relationships'
    ),
  ),
),
```

> **Warning:** In order for the `settingsForm()` implemented to be available on the "Manage Display" page, you must also implement `settingsSummary()` as described below.

2.6.4.4 The settingsSummary() Function.

The `settingsSummary()` function provides a summary of the current settings values for a field on the **manage display** page. The following shows the same relationship field from the previous section, but with the settings form closed, and a summary of the current values shown:



FIELD	REGION	LABEL	FORMAT		
Relationship	Right	gp_sbo__relationship63170	Tripal Pane	**id** tripal_ds-fieldset-gp_sbo__relationship63170 **classes** gp_sbo__relationship63170 field-group-tripalpane **hide** yes	delete
Relationship	Right	<Hidden>	Relationship Statements #	Title: Relationship Empty: There are no relationships No class	
Sequences	Right	group_sequence_tripalpane	Tripal Pane	**id** tripal_ds-fieldset-group_sequence_tripalpane **classes** group-sequence-tripalpane field-group-tripalpane **hide** yes	delete
Sequences Table	Right	group_sequence_table	Table	**label_visibility** Hidden **empty_label_behavior** Keep empty label cell	delete

An example of the `settingsSummary()` function that generates the summary in the image above is as follows:

```
public function settingsSummary($view_mode) {
  $display = $this->instance['display'][$view_mode];
  $settings = $display['settings'];

  $summary = t('Title: @title<br>Empty: @empty',
      array(
        '@title' => $settings['title'],
        '@empty' => $settings['empty'])
      );

  return $summary;
}
```

2.6.5 AJAX Responsive Formatters

Some fields need to be responsive. For example, a user might select an analysis or organism to display data from. Drupal developers often use AJAX to rebuild the page based on user input.

2.6.5.1 Drupal and AJAX

Drupal has its own special way of doing AJAX! This is important to ensure that changes are executed in the correct order. You should read the documentation carefully! The Drupal AJAX API works best on forms, and field formatters **are not forms**. Instead, they are renderable arrays. As such, rather than accepting $form and &$form_state, they accept &$element, $entity_type, $entity, $langcode, $items, and `$display, where $element is the renderable array.

This means if you want to add an AJAX callback to a field formatter, you need a **separate form function** that gets added in using drupal_get_form(). If you do this, you can build the AJAX as Drupal expects it.

2.6.5.2 Example form and field

Here's an example form file below: as you can see it's a standard form following Drupal AJAX conventions. We provide a rendered_maps fieldset with the prefix defining the wrapper (examplemap-featuremap-organism-selector-wrapper). This is what we want to re-draw depending on what the user selects.

The selector has specified that wrapper, and the AJAX callback function examplemap_organism_featuremap_callback. We then define that function to simply return the piece of the form that should be rebuilt: the rendered_maps fieldset!

```
/**
 * AJAX-enabled form for [field formatter name].
 */
function tripal_example_map_organism_featuremap_selector_form($form, &$form_state,
↪$select) {

  $selected = 0;

  // $form_state['values'] will be set if the form has been submitted via AJAX
  if (isset($form_state['values']['featuremap_select'])) {
    $selected = isset($form_state['values']['featuremap_select']);
  }

  // We need to provide a container for Drupal AJAX to replace.
  // Here we use a fieldset with a set ID which we can refer to below.
  $form['rendered_maps'] = [
    '#type' => 'fieldset',
    '#collapsible' => FALSE,
    '#prefix' => '<div id="examplemap-featuremap-organism-selector-wrapper">',
    '#suffix' => '</div>',
  ];

  // This is the element which will trigger AJAX.
  $form['rendered_maps']['featuremap_select'] = [
    '#type' => 'select',
    '#options' => $select,
    '#title' => 'Please select a map to view',
    '#default_value' => $selected,
```

(continues on next page)

(continued from previous page)

```
    '#ajax' => [
      // Your Drupal AJAX callback
      // which simply returns the form element to be re-rendered.
      'callback' => 'examplemap_organism_featuremap_callback',
      // This should be the ID you set above on your container to be replaced.
      'wrapper' => 'examplemap-featuremap-organism-selector-wrapper',
      'effect' => 'fade',
    ],
  ];

  // Check the AJAX submitted values...
  $chosen = 0;
  if (isset($form_state['values']['featuremap_select'])) {
    $chosen = $form_state['input']['featuremap_select'];
  }

  // If the user chose an option (triggered AJAX).
  if ($chosen != 0) {
    // Then change the form accordingly...
    // Notice that you react to the AJAX change in the form
    // not in the AJAX callback.
    $mini_form = tripal_example_map_genetic_map_overview_form([], $form_state,
↪$chosen);

    $form['rendered_maps']['map'] = $mini_form;

    return $form;
  }

  return $form;
}

/**
 * The callback will return the part of the form you want to re-draw.
 */
function examplemap_organism_featuremap_callback($form, &$form_state) {

  return $form['rendered_maps'];
}
```

In the field formatter, we simply add this form and put the markup in the element:

```
/**
 * In our Our__field_formatter.inc
 */
public function view(&$element, $entity_type, $entity, $langcode, $items, $display) {

  // Select choices would be loaded in the base field's load method.
  $select = $items[0]['select_choices'];

  $form = drupal_get_form('tripal_example_map_organism_featuremap_selector_form',
↪$select);
  $content = drupal_render($form);
  $element[] = [
      '#type' => 'markup',
      '#markup' => $content,
  ];
```

(continues on next page)

(continued from previous page)

```
  return $element;
}
```

2.6.6 Attach Fields to Content Types

In summary, creation of a new field requires creation of three classes that inherit from the TripalField, TripalFieldWidget and TripalFieldFormatter base classes. If the fields are created correctly and placed in the includes/TripalFields directory of your module then Tripal will automatically find them. However, the field is not yet attached to any content type. They must be attached. Fields can be attached programmatically or via the online Drupal interface by a site admin.

2.6.6.1 The hook_bundle_fields_info() function

> The three TripalField classes simply define how the field will function, but Drupal does not yet know about the field. The hook_bundle_fields_info function tells Drupal about your field. It must be implemented in a custom Drupal module, and provides an array that tells Drupal about the fields and the classes to use for the field. Suppose we were creating a field named obi__genus which displays the Genus for a species and we have a custom module named tripal_org2. The hook function would be named tripal_org2_bundle_fields_info():

```
function tripal_org2_bundle_fields_info($entity_type, $bundle) {
  $info = [];

  // Make sure this bundle is an organism (OBI:0100026) then we'll attach our
  // field to display the genus.
  $term = tripal_load_term_entity(array('term_id' => $bundle->term_id));
  $term_accession = $term->vocab->vocabulary . '__' . $term->accession;
  if ($term_accession == 'OBI:0100026') {
    $field_name = 'obi__genus';
    $field_type = 'obi__genus';
    $info[$field_name] = [
      'field_name' => $field_name,
      'type' => $field_type,
      'cardinality' => 1,
      'locked' => FALSE,
      'storage' => [
        'type' => 'field_chado_storage',
      ],
      'settings' => [],
    ];
 }

  return $info
}
```

This function receives as its second argument the $bundle object. This is the bundle that Drupal is requesting new fields for. For this example we only want to attach the field if the content type is the organism content type. The format of the returned $info array should have the field name as the key and an array that follows the instructions provided by Drupal's field_create_field() function.

The settings indicate the field name, the field type, the cardinality (how many values are allowed), any default settings and the storage type. Because we expect our data to come from Chado we set the field_chado_storage as the type. The locked setting is set to FALSE indicating that Drupal will allow the field to be deleted if the site developer desires.

When the site administrator navigates to **Administer > Structure > Tripal Content Types**, clicks on a content type, and then the **manage fields** tab, a link appears at the top titled **Check for new fields**. When that link is clicked, this hook function is called.

2.6.6.2 Programmatically Attaching Fields

You probably want to programmatically attach fields to content types if your have existing data that you know should be made available. For example, an organism always has a genus and only one genus. If we have a field that displays the genus for an organism then we will want it automatically attached on installation of our module. We can do this programmatically using two hook functions: `hook_bundle_fields_info()` and `hook_bundle_instances_info()`. Both functions are required to attach a field to a content type.

The hook_bundle_instances_info() function.

The previous hook tells Drupal that our field exists and is allowed to be connected to the organism bundle. Next we need to create an actual instance of this field for the bundle. We do this with the `hook_bundle_instances_info()` function. The format is the same as the previous hook but the info array is different. For example:

```php
function tripal_org2_bundle_instances_info($entity_type, $bundle) {
  $info = []

  // Make sure this bundle is an organism (OBI:0100026) then we'll attach our
  // field to display the genus.
  $term = tripal_load_term_entity(array('term_id' => $bundle->term_id));
  $term_accession = $term->vocab->vocabulary . '__' . $term->accession;
  if ($term_accession == 'OBI:0100026') {

    $field_name = 'obi__genus';
    $is_required = FALSE;
    $info[$field_name] =  [
      'field_name' => $field_name,
      'entity_type' => $entity_type,
      'bundle' => $bundle->name,
      'label' => 'Genus',
      'description' => 'The genus for the organism',
      'required' => TRUE,
      'settings' => [
        'auto_attach' => TRUE,
        'chado_table' => 'organism',
        'chado_column' => 'genus',
        'base_table' => 'organism',
        'term_accession' => '0000005',
        'term_vocabulary' => 'TAXRANK',
        'term_name' => 'Genus',
      ],
      'widget' => [
        'type' => 'obi__genus_widget',
        'settings' => [
          'display_label' => 1,
        ),
      ],
      'display' => [
        'default' => [
          'label' => 'inline',
```

(continues on next page)

(continued from previous page)

```
          'type' => 'obi__genus_formatter',
          'settings' => [],
        ],
      ],
    ];
  }
  return $info;
}
```

The format of the returned `$info` array should have the field name as the key and an array that follows the instructions provided by Drupal's field_create_instance() function.

Unique to this info array are the settings related to Chado. Because we expect our data to be loaded from Chado we must specify these settings:

- `base_table`: the name of the base table to which the record will be associated. In our case the `organism` table of Chado is the base table.
- `chado_table`: the name of the actual table form which the value of the field will be loaded or saved to. In our case the `organism` table is also the `chado_table`.
- `chado_column`: the name of the column in the `chado_table` where the data is loaded from. if the `base_table` and `chado_table` are the same then this is the name of the column. In our case the `genus` columns. If the base and chado tables are different then it is the name o the primary key column in the `chado_table`
- `auto_attach`: set this to TRUE if you want the field to automatically be added to an entity when it is generated for viewing. Set it to FALSE to allow the field to be added via AJAX. For fields that require time to load setting to FALSE is preferred.

Note: A base table is one that contains the primary records to which ancillary data (e.g. properties, cross references, CV terms, publications, contacts, etc) are associated via linker tables. For example some base tables include: `feature`, `organism`, `stock`, `library`, etc.). The `base_table` and `chado_table` will always be the same when you are mapping a field to data in a column in a base table. If your field maps data to a "linker" table where ancillary data is stored then the `chado_table` will be the linker table.

Notice as well that the `display` and `widget` sections list the name of our TripalEntityWidget and TripalEntityFormatter classes respectively. This tells drupal to use our widget and formatter classes by default.

When the site administrator navigates to **Administer > Structure > Tripal Content Types**, clicks on a content type, and then the **manage fields** tab, a link appears at the top titled **Check for new fields**. When that link is clicked, this hook function is called.

Note: Both hook functions must be properly constructed for the field to be automatically attached to the content type.

2.6.6.3 Allowing Manual Attachment of Fields

Not all fields are created equal. Some fields can be added by the site developer to a bundle and some cannot. When the `TripalField` class is implemented for a class the `$no_ui` parameter is set to indicate if a field can be added via the web interface or not. See the *Manual Field Creation* page for more details. But in short the following setting does not allow a field to be added using the web interface

```
public static $no_ui = TRUE;
```

The following setting will allow the field to be added:

```
public static $no_ui = FALSE;
```

Next, we must let Drupal know that our field exists. We do this by adding an entry to the `$info` array in the `hook_bundle_fields_info()` function described above. This lets Drupal know about our field. However, because we are not programmatically creating an instance of the field on a content type, but allowing the user to create them we do not need to implement the `hook_bundle_instances_info()` function. Instead, we must implement `hook_bundle_create_user_field()`. This function is called when the user attempts to add our new field to a bundle. One field that comes with Tripal is the `chado_linker__prop` field. Most Chado base tables have an associated property table (e.g. `organismprop`, `featureprop`, `stockprop`, etc). By default, the `tripal_chado` module automatically adds this field to all bundles that have existing properties. It adds a new instance for every property type. However, new properties can be added to bundle, and the site admin may want to add those properties via the user interface rather. Therefore, this field has the `$no_ui` set to TRUE and uses the `hook_bundle_create_user_field()` to create the new field instance for the user.

The following code is a snippet from the `tripal_chado_bundle_create_user_field` function of the `tripal_chado` module. Note that it uses the `field_create_field` function and the `field_create_instance` functions directly. The arrays passed to these functions are identical to the `$info` arrays of both the `hook_bundle_fields_info` and `hook_bundle_instances_info` functions described above.

```
function tripal_chado_bundle_create_user_field($new_field, $bundle) {

  // Get the table this bundle is mapped to.
  $term = tripal_load_term_entity(array('term_id' => $bundle->term_id));
  $vocab = $term->vocab;
  $params = array(
    'vocabulary' => $vocab->vocabulary,
    'accession' => $term->accession,
  );
  $chado_table = $bundle->data_table;
  $chado_type_table = $bundle->type_linker_table;
  $chado_type_column = $bundle->type_column;
  $chado_type_id = $bundle->type_id;
  $chado_type_value = $bundle->type_value;

  // We allow site admins to add new chado_linker__prop fields to an entity.
  // This function will allow us to properly add them.  But at this point we
  // don't know the controlled vocabulary term.  We'll have to use the
  // defaults and let the user set it using the interface.
  if ($new_field['type'] == 'chado_linker__prop') {
    $table_name = $chado_table . 'prop';

    if (chado_table_exists($table_name)) {
      $schema = chado_get_schema($table_name);
      $pkey = $schema['primary key'][0];
      $field_name = $new_field['field_name'];
      $field_type = 'chado_linker__prop';

      // First add the field.
      field_create_field(array(
        'field_name' => $field_name,
        'type' => $field_type,
```

(continues on next page)

(continued from previous page)

```
        'cardinality' => FIELD_CARDINALITY_UNLIMITED,
        'locked' => FALSE,
        'storage' => array(
          'type' => 'field_chado_storage',
        ),
      ));

      // Now add the instance
      field_create_instance(array(
        'field_name' => $field_name,
        'entity_type' => 'TripalEntity',
        'bundle' => $bundle->name,
        'label' => $new_field['label'],
        'description' => '',
        'required' => FALSE,
        'settings' => array(
          'auto_attach' => TRUE,
          'base_table' => $chado_table,
          'chado_table' => $table_name,
          'chado_column' => $pkey,
          'term_vocabulary' => '',
          'term_accession' => '',
          'term_name' => ''
        ),
        'widget' => array(
          'type' => 'chado_linker__prop_widget',
          'settings' => array(
            'display_label' => 1,
          ),
        ),
        'display' => array(
          'default' => array(
            'label' => 'inline',
            'type' => 'chado_linker__prop_formatter',
            'settings' => array(),
          ),
        ),
      ));
    }
    else {
      drupal_set_message('Cannot add a property field to this entity. Chado does not support properties for this data type.', 'error');
    }
  }
}
```

Note: It is possible to have a field that is both programmatically attached to some content types but is also allowed to be attached to another content type by the site admin using the web interface. To do this, programmatically add the field to the content types using the `hook_bundle_instances_info` function and also implement the `hook_bundle_create_user_field` function to support manual adding.

2.6.7 Easier Field Creation: Tripal Field Generator

The Staton Lab has created a specialized tool for automatically generating a ChadoField or TripalField and populating them with the basic controlled vocabulary and database information. Instructions are available at https://github.com/statonlab/fields_generator.

3
Tripal Fields Generator
Quickly define Tripal fields, minimizing CVterm usage error.
2
4

2.7 Exporting Field Settings With Drupal Features

This guide will demonstrate using the Drupal Features to export and import Tripal Bundle Field settings.

2.7.1 Why Drupal Features?

Consider a case where we have a development site where we configure a bundle (let's say the vanilla Analysis bundle) to have a custom set of Tripal Panes, with fields carefully reorganized into the panes. In particular, we attach a Drupal File field to it, so we can host FASTA files easily.

Once we've configured the field settings, how do we get them to the live site? One option is to open our field configuration admin UI on both sites, and copy the details one at a time. This method is time consuming and error prone, although it is relatively safe: we aren't liable to accidentally break our database this way. Alternatively, trying to transfer via writing database exports is dangerous, and liable to accidentally break our site.

Is there a better way? By exporting the field configuration as a feature!

Note: Drupal 8 has a feature designed to handle development deployment: Configuration Management! Drupal Features remains relevant for sharing field configurations across sites, so this guide may remain useful when Tripal moves to Drupal 8.

Warning: Note that I'm going to talk about features a lot in this post. **In the context of the Feature module, features are exported configurations within your site**. It's important that we don't confuse this with Chado features, which are genes, mRNAs, polypeptides, etc.

2.7.1.1 Version control

Because features get exported as their own module, this means you can treat them as such. You can initialize a git repo, store them on GitHub, make discrete versions, and in general benefit from version control for something which otherwise would only be done via the UI.

2.7.1.2 How many Features?

The features documentation links to a great article about how to manage multiple features. The suggestion that each bundle be its own module is particularly helpful for Tripal, since we have so many bundles and its a reasonable, discrete way to manage and deploy configuration. This means that mRNA and gene should be separate feature modules even though they are both `chado.feature` content types.

2.7.1.3 A Hazard: Mapping bundle IDs

The main hurdle to overcome when mapping features is converting the field machine names across site. This is because each field instance is specific to the bundle it's attached to, and bundle machine-names are from the bundle ID. We can't assume bundle IDs are consistent across sites!

So what do we do? Interestingly, roles have a similar problem, and a guide is available for dealing with them.

The general strategy is:

- remove the exported id value from the `features.inc` file
- use hook alter to fetch the ID on the target deployment setup

In our example below, the bundle ID's match on our site. For default Tripal bundles, this should generally be the case.

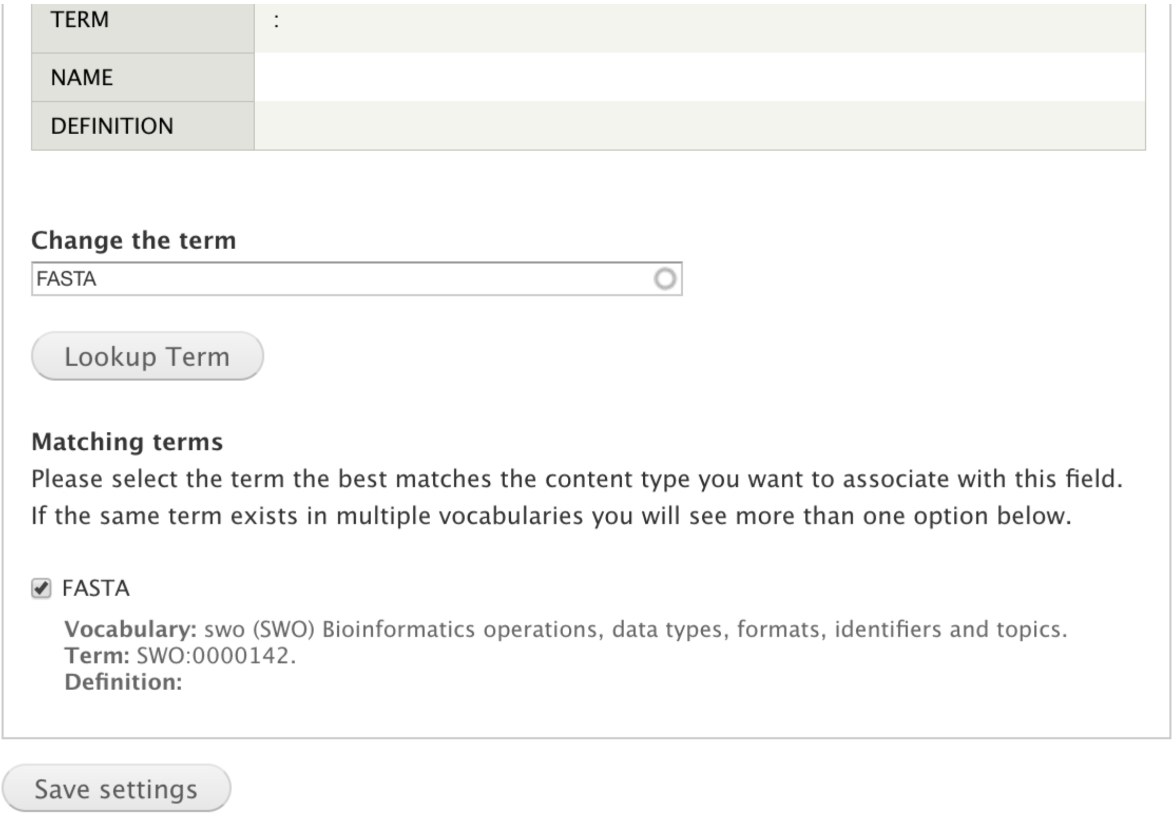
2.7.2 Drupal Features Tutorial

In this tutorial, we'll add a FASTA file field to the Analysis bundle, and export the configuration.

2.7.2.1 Configuring the bundle fields

To begin, let's quickly configure a bundle. Navigate to the structure of your site and select Analysis (**Admin ->Structure -> Tripal Content -> Analysis -> Manage Fields**). Scroll to the bottom and add a new field of type File, with a machine name of `field_fasta_file`, and click **Save**. Be sure to change the **Allowed extensions** parameter to accept `.fasta`, otherwise, it will only allow *.txt* files to be uploaded. You may also want to increase the file size limit, as the default 2MB can be too small for many FASTA files.

We now have to pick a CV term for our field. Because we are providing a FASTA file field, we can use a term such as FASTA (SWO:0000142). Please see *Manually Adding a Term* for help loading a term. Once the term is in our DB, we can assign it to this field.



TERM	:
NAME	
DEFINITION	

Change the term

FASTA

Lookup Term

Matching terms

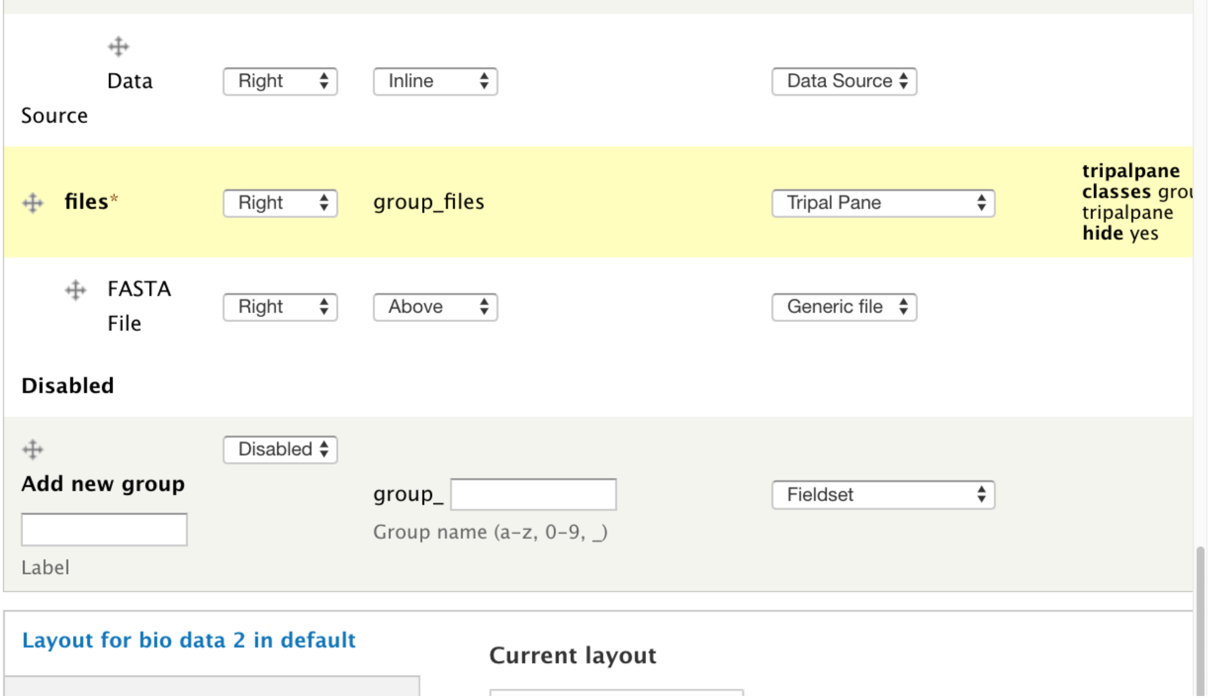
Please select the term the best matches the content type you want to associate with this field. If the same term exists in multiple vocabularies you will see more than one option below.

☑ FASTA

Vocabulary: swo (SWO) Bioinformatics operations, data types, formats, identifiers and topics.
Term: SWO:0000142.
Definition:

Save settings

Now lets configure our field display. Click the **Manage Display Tab** at the top, and create and enable a “Files” Tripal Pane, placing our new field in the Pane.

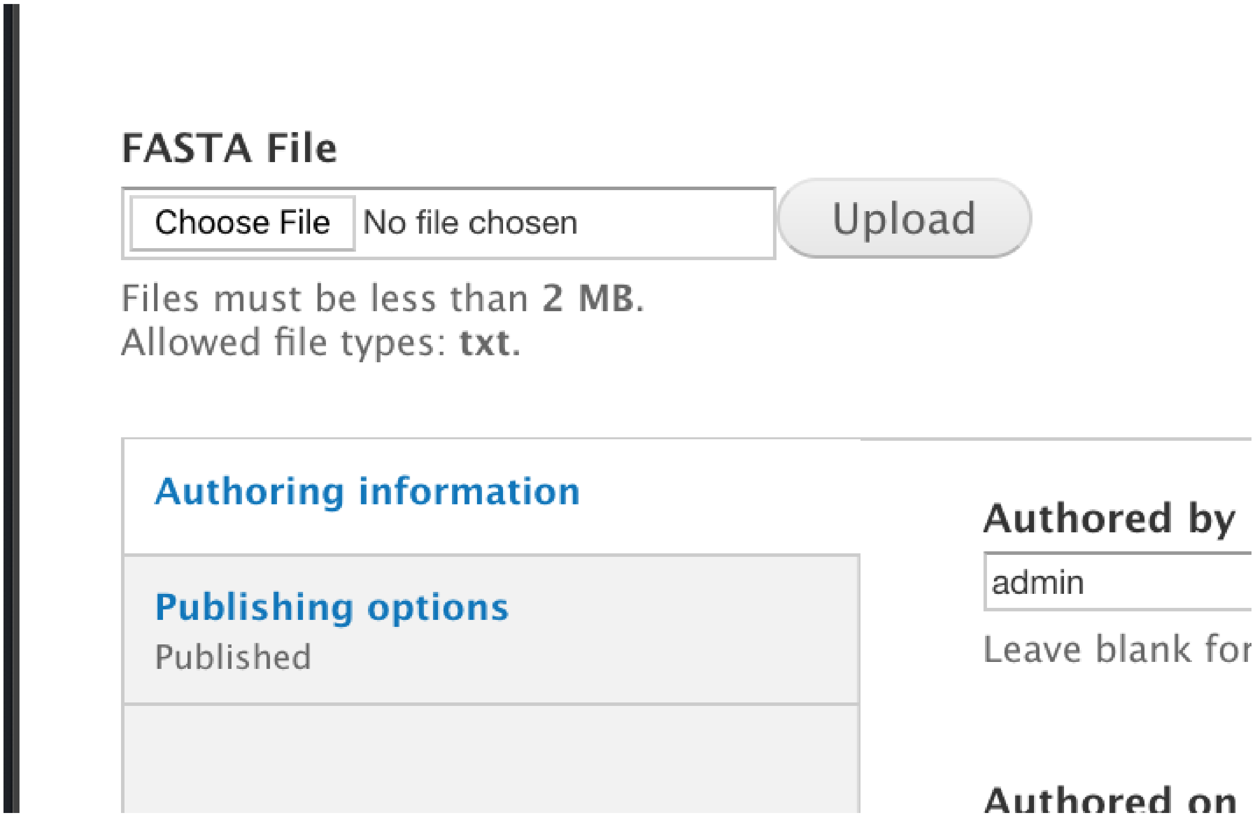


Data Source	Right	Inline	Data Source	
files*	Right	group_files	Tripal Pane	**tripalpane classes** grou tripalpane **hide** yes
FASTA File	Right	Above	Generic file	
Disabled				
Add new group Label	Disabled	group_ Group name (a-z, 0-9, _)	Fieldset	

Layout for bio data 2 in default

Current layout

You can verify your new field is enabled and working by creating a new Analysis and inspecting the "FASTA file" field.



FASTA File

Files must be less than **2 MB**.
Allowed file types: **txt**.

2.7.2.2 Exporting the bundle field displays

Once we are happy with our bundle field configuration, we can export the display settings using the Drupal Features module.

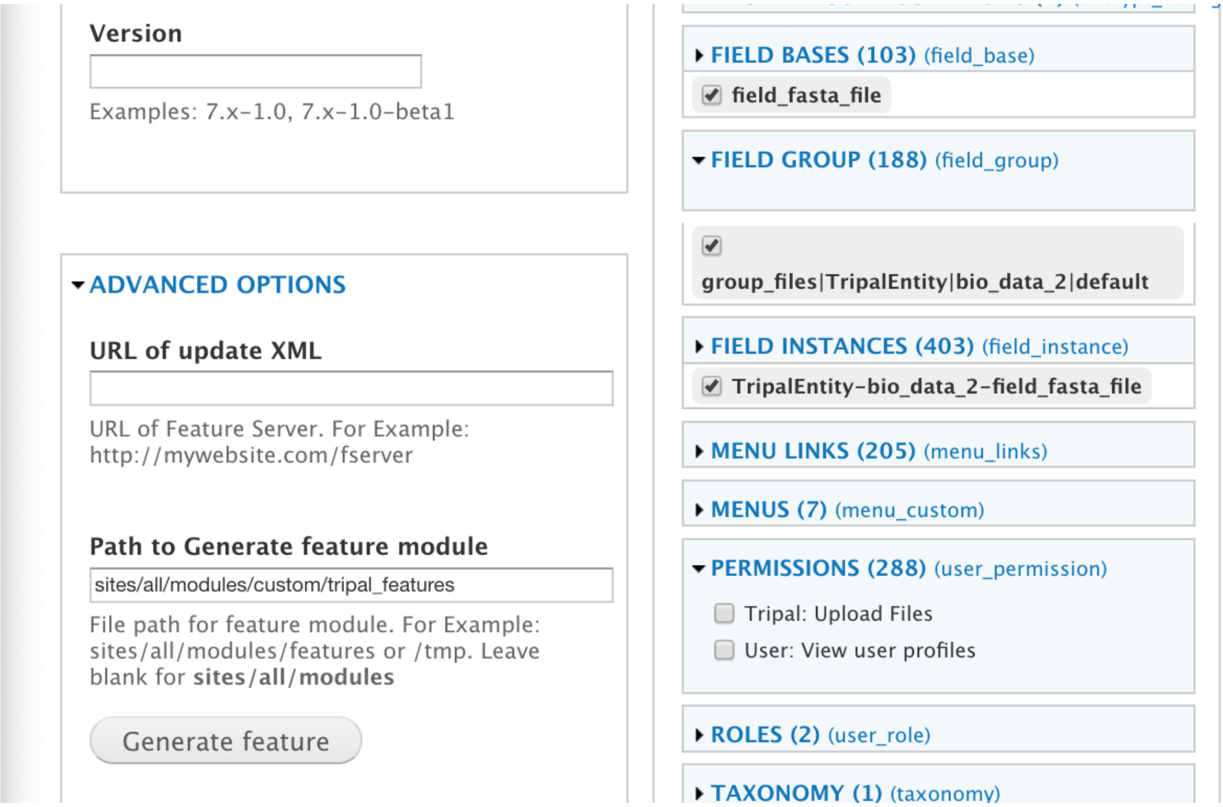
First, we enable the Features module using drush: `drush pm-enable features -y`. This adds a Features area under **Admin -> Structure**. Navigate there and choose **Create Feature**.

The field information we're looking for is in **Field Bases**, **Field Group**, and **Field Instances**. We can search for FASTA to find the field base and instance, and "files" (the name of our group) to find the field group.

Note: Both **Field Bases** and **Field Instances** will contain the machine name of the field you want to export. **Field Bases** contains the site-wide information for a field and **Field Instances** contains the bundle-specific (i.e. Tripal Content Type) settings.

Field Group will contain the machine name of the Tripal Pane and allows you to export the grouping settings you set on the **Manage Display Tab**.

I've also specified a custom path to keep all my Tripal features together under advanced options.



Version

Examples: 7.x-1.0, 7.x-1.0-beta1

▾ **ADVANCED OPTIONS**

URL of update XML

URL of Feature Server. For Example: http://mywebsite.com/fserver

Path to Generate feature module

sites/all/modules/custom/tripal_features

File path for feature module. For Example: sites/all/modules/features or /tmp. Leave blank for **sites/all/modules**

Generate feature

▸ **FIELD BASES (103)** (field_base)

☑ field_fasta_file

▾ **FIELD GROUP (188)** (field_group)

☑ group_files|TripalEntity|bio_data_2|default

▸ **FIELD INSTANCES (403)** (field_instance)

☑ TripalEntity-bio_data_2-field_fasta_file

▸ **MENU LINKS (205)** (menu_links)

▸ **MENUS (7)** (menu_custom)

▾ **PERMISSIONS (288)** (user_permission)

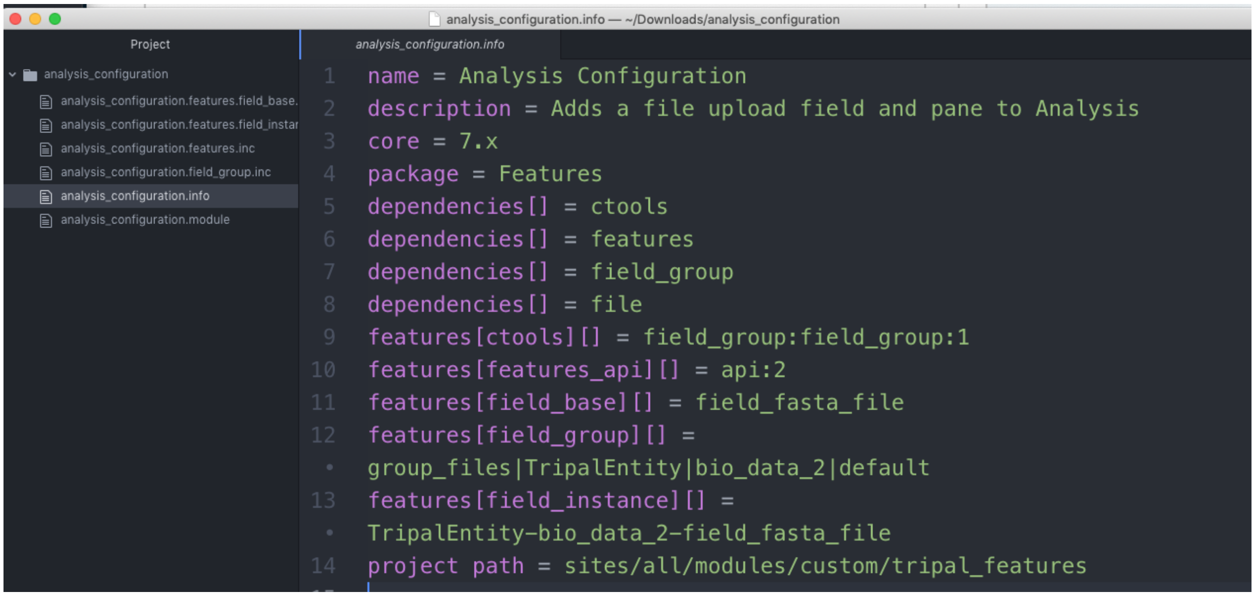
☐ Tripal: Upload Files

☐ User: View user profiles

▸ **ROLES (2)** (user_role)

▸ **TAXONOMY (1)** (taxonomy)

If we download and unzip our feature module, we can see it includes all the trappings of a Drupal module.



analysis_configuration.info — ~/Downloads/analysis_configuration

Project

- analysis_configuration
  - analysis_configuration.features.field_base.
  - analysis_configuration.features.field_instar
  - analysis_configuration.features.inc
  - analysis_configuration.field_group.inc
  - analysis_configuration.info
  - analysis_configuration.module

analysis_configuration.info

```
name = Analysis Configuration
description = Adds a file upload field and pane to Analysis
core = 7.x
package = Features
dependencies[] = ctools
dependencies[] = features
dependencies[] = field_group
dependencies[] = file
features[ctools][] = field_group:field_group:1
features[features_api][] = api:2
features[field_base][] = field_fasta_file
features[field_group][] = group_files|TripalEntity|bio_data_2|default
features[field_instance][] = TripalEntity-bio_data_2-field_fasta_file
project path = sites/all/modules/custom/tripal_features
```

Warning:

As you can see, it makes the assumption that `bio_data_2`, the bundle ID for Analysis on our source site, is the correct bundle to configure fields for. However, Tripal makes no guarantee that will hold true on our target site. One solution would be to manually relabel `bio_data_x` to the correct bundle ID. On a smaller scale, this is a reasonable solution. If you aren't sure what your bundle ID is,

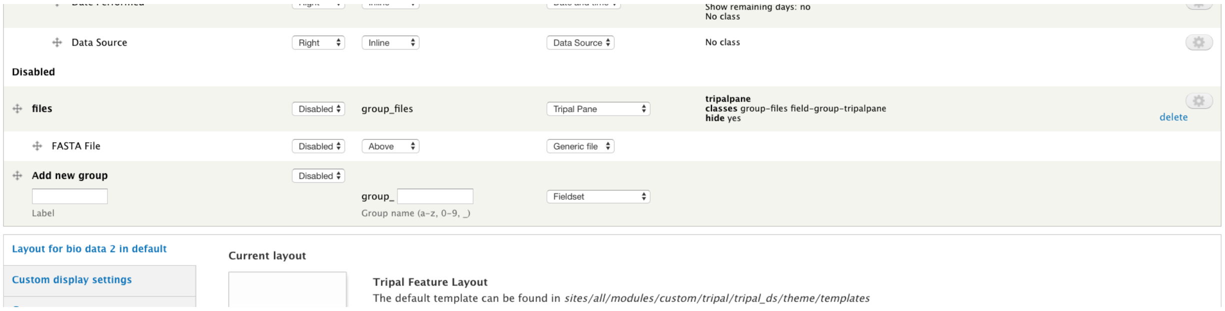
> look in the URL when configuring the fields for it: my constructed URL for example was `admin/structure/bio_data/manage/bio_data_2/fields`.
>
> In our case, the site we want to import to has the same Analysis bundle ID, so no further action is necessary.

2.7.2.3 Importing the feature configuration

Go to our target site, all we need to do is download and unpack the `.tar` file we generated and enable the module (assuming the bundle ID issue is addressed). I downloaded my file to `/var/www/html/sites/all/modules/custom/analysis_configuration.tar`, decompressed it (`tar -xvf analysis_configuration.tar`), and enabled it (`drush pm-enable tripal_configuration`).

The field should now appear when you go to create a new analysis on your target site. To check for yourself, create a new Analysis with dummy information: you'll be able to upload a file for the new file field.

Unfortunately, the field still gets imported **disabled** due to Tripal preference, so we have to go to the display settings on our target site and enable the Tripal pane/field.



				Show remaining days: no No class	
Data Source	Right	Inline	Data Source	No class	
Disabled					
files	Disabled	group_files	Tripal Pane	**tripalpane** **classes** group-files field-group-tripalpane **hide** yes	delete
FASTA File	Disabled	Above	Generic file		
Add new group Label	Disabled	group_ Group name (a-z, 0-9, _)	Fieldset		

Layout for bio data 2 in default

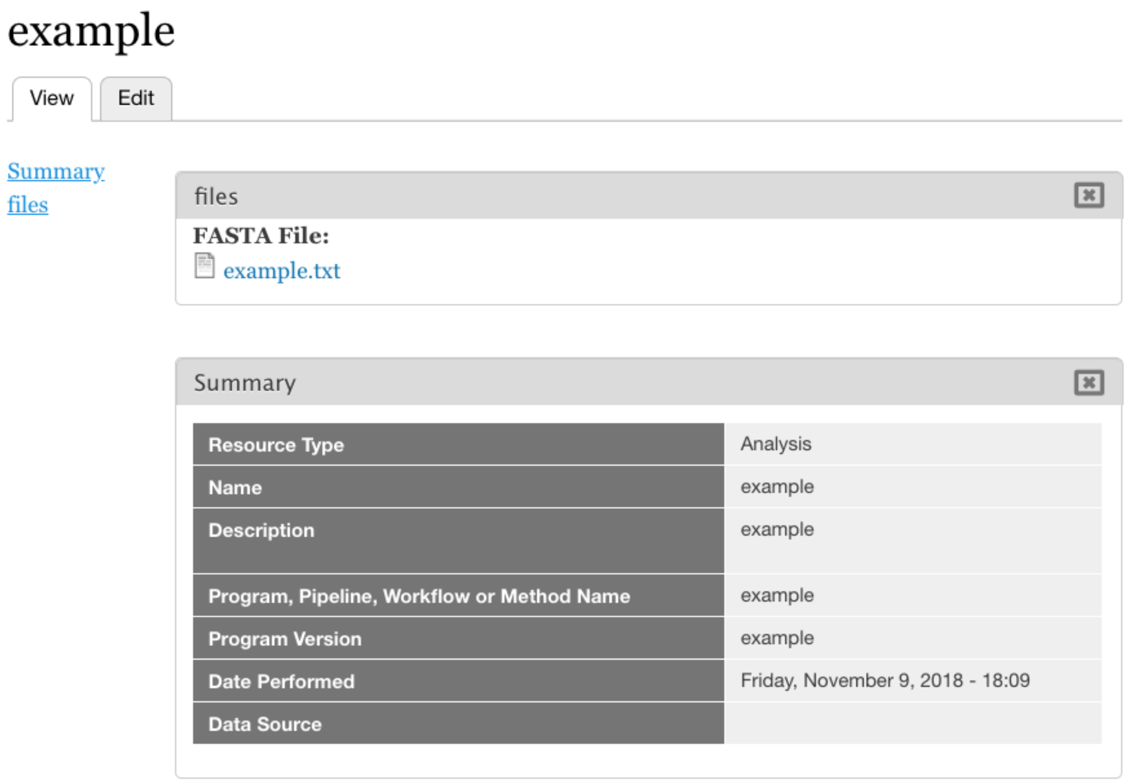
Custom display settings

Current layout

Tripal Feature Layout

The default template can be found in *sites/all/modules/custom/tripal/tripal_ds/theme/templates*

Drag the disabled Tripal pane/field group out of the disabled area, click save, and re-visit your newly created Analysis. The files pane and uploaded FASTA file will now appear.



example

files

FASTA File:

example.txt

Summary

Resource Type	Analysis
Name	example
Description	example
Program, Pipeline, Workflow or Method Name	example
Program Version	example
Date Performed	Friday, November 9, 2018 - 18:09
Data Source	

2.8 Creating Custom Data Loaders

Note: This guide is also available as a video on youtube.

The `TripalImporter` class can be extended to create your own data loader. This class provides many conveniences to simplify loader construction. For example, it simplifies and unifies input form development, automatically handles files upload by the user, provides job submission, logging and progress updates. Using the TripalImporter to create your loader also makes it easy to share your loader with other Tripal users!

To document how to create a new importer, we will describe use of the `TripalImporter` class within the context of a new simple importer called the `ExampleImporter`. This importer will read in a comma-separated file containing genomic features and their properties (a fictional "Test Format" file). The loader will split each line into feature and property values, and then insert each property into the `featureprop` table of Chado using a controlled vocabulary term (supplied by the user) as the `type_id` for the property.

Note: Prior to starting your data loader you should plan how the data will be imported into Chado. Chado is a flexible database schema and it may be challenging at times to decide in to which tables data should be placed. It is recommended to reach out to the Chado community to solicit advice. Doing so will allow you to share your loader will other Tripal users more easily!

2.8.1 Create a Custom Module

To create your own importer, you first need to have a custom extension module in which the loader will be provided. If you do not know how to create a module, see the section titled **Creating a Custom Module** for further direction. Providing your new importer within a custom module will allow you to more easily share your loader with other Tripal users. Any site that downloads and enables your extension module will be able to use your data loader. For this document we will describe creation of a new importer in a module named `tripal_example_importer`.

2.8.2 Create the Class File

To define a new class that extends `TripalImporter`, you should create a new class file with a `.inc` extension within your custom module in the directory: `includes/TripalImporter/`. If this is your first importer, then you will need to create this directory. For the example described here, we will create a new `TripalImporter` class named `ExampleImporter`. Therefore, we must name the file the same as the class (with the .inc extension) and place the file here: `tripal_example_importer/includes/TripalImporter/ExampleImporter.inc`. Initially, our new class is empty:

```
class ExampleImporter extends TripalImporter {
}
```

There is no need to include the importer via a `require_once` statement in your module file. Placing it in the `/includes/TripalImporter/` directory of your module is all you need for Tripal to find it. Tripal will automatically place a link for your importer at `admin -> Tripal -> Data Loaders`.

Note: If after creation of your importer file, Tripal does not show a link for it in the Data Loaders page, check that you have named your class file correctly and it is in the path described above. Sometimes a clear cache is necessary (`drush cc all`).

2.8.3 Static Variables

The next step in creation of your importer is setting the static member variables. Open the `TripalImporter` class file that comes with Tripal and found here `tripal/includes/TripalImporter.inc`. Copy the `public static` member variables at the top of the class into your own class. For your importer, override any of the `public static` variables that need to be different from the default.

Note: For the sake of simplicity in this document, many of the default settings are not changed, and therefore, not all are included.

Our `ExampleImporter` class now appears as follows:

```
/**
 * @see TripalImporter
 */
 class ExampleImporter extends TripalImporter {

  /**
   * The name of this loader.  This name will be presented to the site
   * user.
   */
  public static $name = 'Example TST File Importer';
```

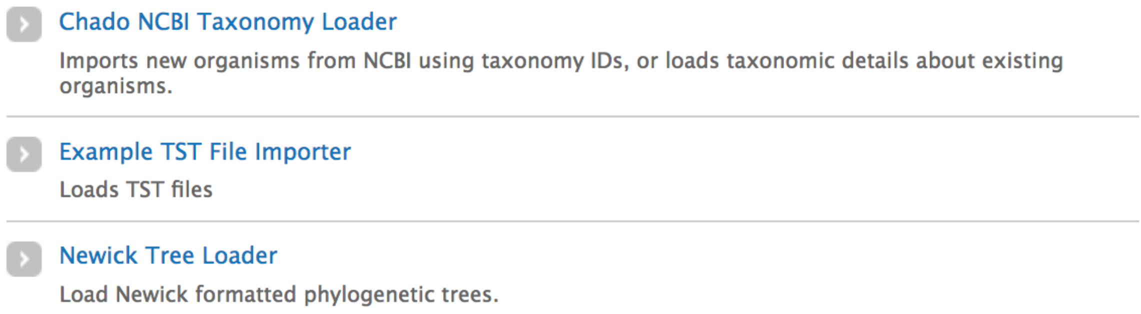
(continues on next page)

(continued from previous page)

```
  /**
   * The machine name for this loader. This name will be used to construct
   * the URL for the loader.
   */
  public static $machine_name = 'tripal_tst_loader';

  /**
   * A brief description for this loader.  This description will be
   * presented to the site user.
   */
  public static $description = 'Loads TST files';

  /**
   * An array containing the extensions of allowed file types.
   */
  public static $file_types = ['txt', 'tst', 'csv'];

  /**
   * Provides information to the user about the file upload.  Typically this
   * may include a description of the file types allowed.
   */
  public static $upload_description = 'TST is a fictional format.  Its a 2-column, CSV file.  The columns should be of the form featurename, and text';

  /**
   * Indicates the methods that the file uploader will support.
   */
  public static $methods = [
    // Allow the user to upload a file to the server.
    'file_upload' => TRUE,
    // Allow the user to provide the path on the Tripal server for the file.
    'file_local' => TRUE,
    // Allow the user to provide a remote URL for the file.
    'file_remote' => TRUE,
  ];
}
```

Warning: The variables that are `private static` **should not** be copied and should not be changed. Only copy and change the `public static` member variables.

Now that we've given our importer a name and description, it will show up at `/admin/tripal/loaders`:



Chado NCBI Taxonomy Loader

Imports new organisms from NCBI using taxonomy IDs, or loads taxonomic details about existing organisms.

Example TST File Importer

Loads TST files

Newick Tree Loader

Load Newick formatted phylogenetic trees.

2.8.4 Form Components

By default, the `TripalImporter` class will provide the necessary upload widgets to allow a user to upload files for import. The static variables we set in the previous step dictate how that uploader appears to the user. However, for this example, our importer needs additional information from the user before data can be loaded. We need to provide additional form widgets.

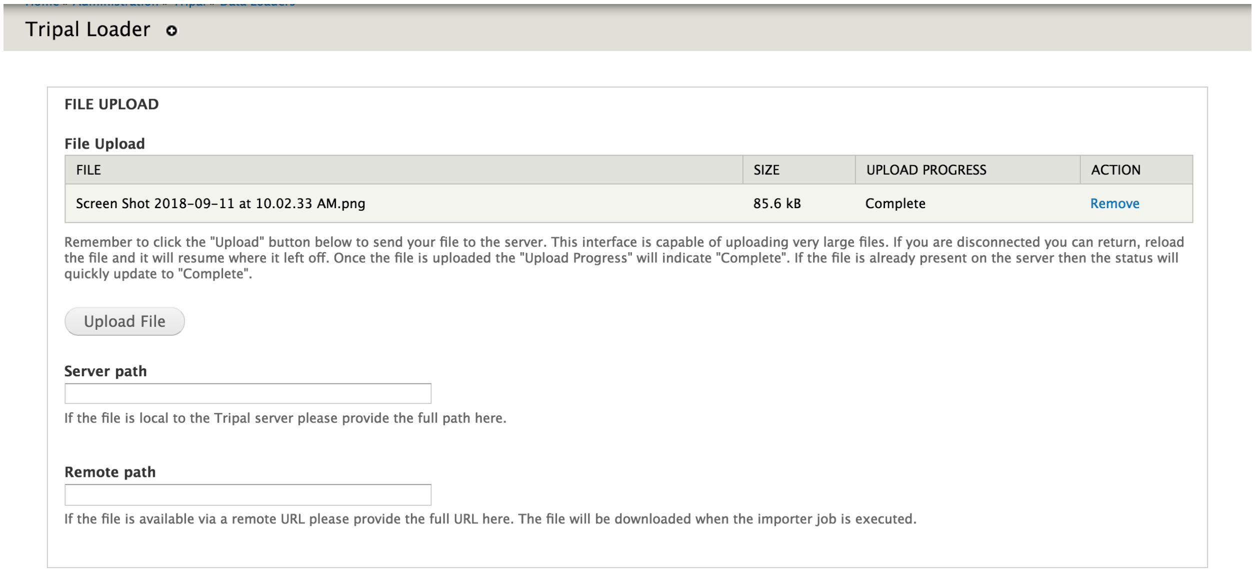
Typically, to create forms, Drupal provides form hooks: `form`, `form_validate`, `form_submit`. The **TripalImporter** wraps these for us as class functions named `form`, `formValidate` and `formSubmit`. We can override these class functions to provide additional widgets to the form.

Note: Typically we only need to implement the `form` and `formValidate` functions. The `formSubmit` does not need to be modified.

Note: If you are not familiar with form creation in Drupal you may want to find a Drupal reference book that provides step-by-step instructions. Additionally, you can explore the API documentation for form construction for Drupal 7. Here, this example expects you are comfortable with form construction in Drupal.

2.8.4.1 The form function

To provide custom widgets for our importer we need to implement the `form` function. However, let's review the current form provided by the TripalImporter for us already. Using the static variables settings specified above the form automatically provides a **File Upload** field set, and an **Analysis** selector. The **File Upload** area lets users choose to upload a file, provide a **Server path** to a file already on the web server or a **Remote path** for files located via a downloadable link on the web. The **Analysis** selector is important because it allows the user to specify an analysis that describes how the data file was created.



Tripal Loader

FILE UPLOAD

File Upload

FILE	SIZE	UPLOAD PROGRESS	ACTION
Screen Shot 2018-09-11 at 10.02.33 AM.png	85.6 kB	Complete	Remove

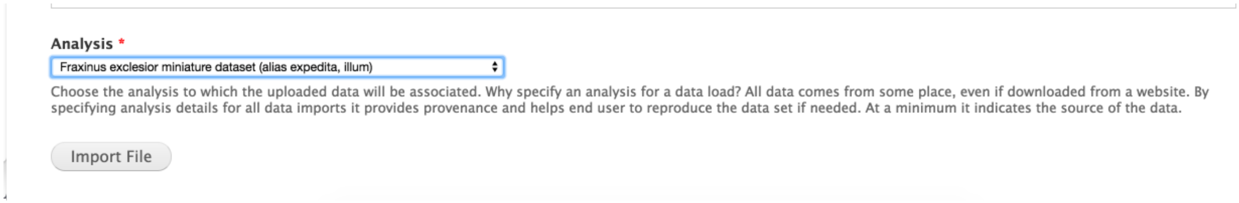
Remember to click the "Upload" button below to send your file to the server. This interface is capable of uploading very large files. If you are disconnected you can return, reload the file and it will resume where it left off. Once the file is uploaded the "Upload Progress" will indicate "Complete". If the file is already present on the server then the status will quickly update to "Complete".

Server path

If the file is local to the Tripal server please provide the full path here.

Remote path

If the file is available via a remote URL please provide the full URL here. The file will be downloaded when the importer job is executed.



Analysis *

Fraxinus exclesior miniature dataset (alias expedita, illum)

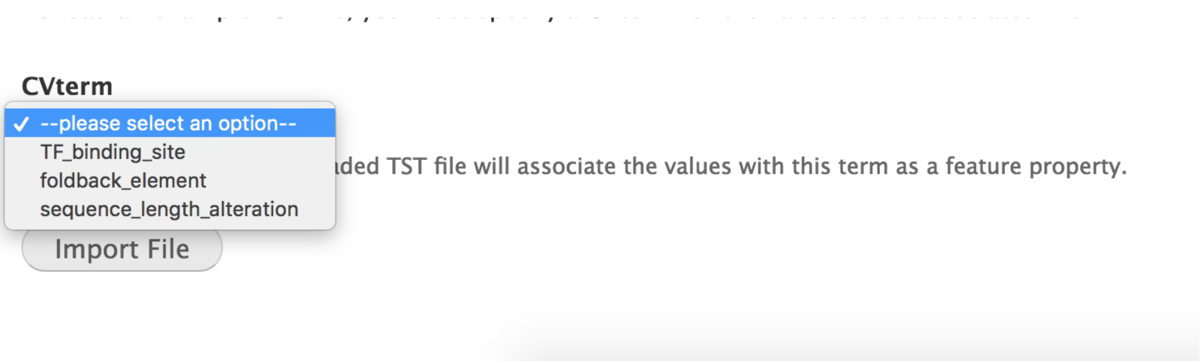
Choose the analysis to which the uploaded data will be associated. Why specify an analysis for a data load? All data comes from some place, even if downloaded from a website. By specifying analysis details for all data imports it provides provenance and helps end user to reproduce the data set if needed. At a minimum it indicates the source of the data.

For our example TST file importer these upload options are sufficient. However, for our data import we want the user provide a CV term. We want our importer to read the file, split it into feature and values, and insert properties into the `featureprop` table of Chado using the the CV term as the `type_id` for the table.

To add a widget that allows the user to provide a CV term, we must implement the `form` function and include code using Drupal's Form API that will add the widget.

```
public function form($form, &$form_state) {


  // For our example loader let's assume that there is a small list of
  // vocabulary terms that are appropriate as properties for the genomics
  // features. Therefore, we will provide an array of sequence ontology terms
  // the user can select from.
  $terms = [
    ['id' => 'SO:0000235'],
    ['id' => 'SO:0000238'],
    ['id' => 'SO:0000248']
  ];

  // Construct the options for the select drop down.
  $options = [];
  // Iterate through the terms array and get the term id and name using
  // appropriate Tripal API functions.
  foreach ($terms as $term){
    $term_object = chado_get_cvterm($term);
    $id = $term_object->cvterm_id;
    $options[$id] = $term_object->name;
  }

  // Provide the Drupal Form API array for a select box.
  $form['pick_cvterm'] =  [
    '#title' => 'CVterm',
    '#description' => 'Please pick a CVterm.  The loaded TST file will associate the values with this term as a feature property.',
    '#type' => 'select',
    '#default_value' => '0',
    '#options' => $options,
    '#empty_option' => '--please select an option--'
  ];

  // The form function must always return our form array.
  return $form;
}
```

Our form now has a select box!



ded TST file will associate the values with this term as a feature property.

Using AJAX in forms

Note: This section is not yet available. For now, check out the Drupal AJAX guide https://api.drupal.org/api/drupal/includes%21ajax.inc/group/ajax/7.x

2.8.4.2 The formValidate function

The `formValidate` function is responsible for verifying that the user supplied values from the form submission are valid. To warn the user of inappropriate values, the Drupal API function, `form_set_error()` is used. It provides an error message, highlights in red the widget containing the bad value, and prevents the form from being submitted–allowing the user to make corrections. In our example code, we will check that the user selected a CV term from the `pick_cvterm` widget.

```php
public function formValidate($form, &$form_state) {

  // Always call the TripalImporter (i.e. parent) formValidate as it provides
  // some important feature needed to make the form work properly.
  parent::formValidate($form, $form_state);

  // Get the chosen CV term form the form state and if there is no value
  // set warn the user.
  $chosen_cvterm = $form_state['values']['pick_cvterm'];
  if ($chosen_cvterm == 0) {
    form_set_error('pick_cvterm', 'Please choose a CVterm.');
  }
}
```

The implementation above looks for the `pick_cvterm` element of the `$form_state` and ensures the user selected something. This is a simple example. An implementation for a more complex loader with a variety of widgets will require more validation checks.

Note: If our importer followed best practices, it would not need a validator at all. The cvterm select box in the form could be defined as below. Note the `'#required' => True` line: this would handle the validation for us. For this tutorial, however, we implement the validation ourselves to demonstrate the function.

```php
// Provide the Drupal Form API array for a select box.
$form['pick_cvterm'] =  [
  '#title' => 'CVterm',
  '#description' => 'Please pick a CVterm.  The loaded TST file will associate the values with this term as a feature property.',
```

(continues on next page)

(continued from previous page)

```
  '#type' => 'select',
  '#default_value' => '0',
  '#options' => $options,
  '#empty_option' => '--please select an option--'
  '#required' => True
];
```

When an importer form is submitted and passes all validation checks, a job is automatically added to the **Tripal Job** system. The `TripalImporter` parent class does this for us! The **Tripal Job** system is meant to allow long-running jobs to execute behind-the-scenes on a regular time schedule. As jobs are added they are executed in order. Therefore, if a user submits a job using the importer's form then the **Tripal Job** system will automatically run the job the next time it is scheduled to run or it can be launched manually by the site administrator.

2.8.5 Importer Execution

The `form` and `formValidate` functions allow our Importer to receive an input file and additional values needed for import of the data. To execute loading a file the `TripalImporter` provides several additional overridable functions: `run`, `preRun` and `postRun`. When the importer is executed, the `preRun` function is called first. It allows the importer to perform setup prior to full execution. The `run` function is where the full execution occurs and the `postRun` function is used to perform "cleanup" prior to completion. For our `ExampleImporter` class we only need to implement the `run` function. We have no need to perform any setup or cleanup outside of the typical run.

2.8.5.1 The run function

The `run` function is called automatically when Tripal runs the importer. For our `ExampleImporter`, the run function should collect the values provided by the user, read and parse the input file and load the data into Chado. The first step, is to retrieve the user provided values and file details. The inline comments in the code below provide instructions for retrieving these details.

```
/**
 * @see TripalImporter::run()
 */
public function run() {

  // All values provided by the user in the Importer's form widgets are
  // made available to us here by the Class' arguments member variable.
  $arguments = $this->arguments['run_args'];

  // The path to the uploaded file is always made available using the
  // 'files' argument. The importer can support multiple files, therefore
  // this is an array of files, where each has a 'file_path' key specifying
  // where the file is located on the server.
  $file_path = $this->arguments['files'][0]['file_path'];

  // The analysis that the data being imported is associated with is always
  // provided as an argument.
  $analysis_id = $arguments['analysis_id'];

  // Any of the widgets on our form are also available as an argument.
  $cvterm_id = $arguments['pick_cvterm'];

  // Now that we have our file path, analysis_id and CV term we can load
```

(continues on next page)

(continued from previous page)

```
  // the file.  We'll do so by creating a new function in our class
  // called "loadMyFile" and pass these arguments to it.
  $this->loadMyFile($analysis_id, $file_path, $cvterm_id);
}
```

Note: We do not need to validate in the `run` function that all of the necessary values in the arguments array are valid. Remember, this was done by the `formValidate` function when the user submitted the form. Therefore, we can trust that all of the necessary values we need for the import are correct. That is of course provided our `formValidate` function sufficiently checks the user input.

2.8.5.2 Importing the File

To keep the `run` function small, we will implement a new function named `loadMyFile` that will perform parsing and import of the file into Chado. As seen in the code above, the `loadMyFile` function is called in the `run` function.

Initially, lets get a feel for how the importer will work. Lets just print out the values provided to our importer:

```
public function loadMyFile($analysis_id, $file_path, $cvterm){
  var_dump(["this is running!", $analysis_id, $file_path, $cvterm]);
}
```

To test our importer navigate to `admin > Tripal > Data Importers` and click the link for our TFT importer. Fill out the form and press submit. If there are no validation errors, we'll receive notice that our job was submitted and given a command to execute the job manually. For example:

> drush trp-run-jobs –username=admin –root=/var/www/html

If we execute our importer we should see the following output:

```
Calling: tripal_run_importer(146)

Running 'Example TST File Importer' importer
NOTE: Loading of file is performed using a database transaction.
If it fails or is terminated prematurely then all insertions and
updates are rolled back and will not be found in the database

array(4) {
  [0]=>
  string(16) "This is running!"
  [1]=>
  string(3) "147"
  [2]=>
  string(3) "695"
  [3]=>
  string(72) "/Users/chet/UTK/tripal/sites/default/files/tripal/users/1/expression.tsv
↪"
}

Done.

Remapping Chado Controlled vocabularies to Tripal Terms...
```

As you can see, running the job executes our run script, and we have all the variables we need to load the data. All we need to do now is write the code!

To import data into Chado we will use the Tripal API. After splitting each line of the input file into a genomic feature and its property, we will use the `chado_select_record` to match the feature's name with a record in the `feature` table of Chado, and the `chado_insert_property` to add the property value.

```
public function loadMyFile($analysis_id, $file_path, $cvterm_id){

  // We want to provide a progress report to the end-user so that they:
  // 1) Recognize that the loader is not hung if running a large file, but is
  //    executing
  // 2) Provides some indication for how long the file will take to load.
  //
  // Here we'll get the size of the file and tell the TripalImporter how
  // many "items" we have to process (in this case bytes of the file).
  $filesize = filesize($file_path);
  $this->setTotalItems($filesize);
  $this->setItemsHandled(0);

  // Loop through each line of file.  We use the fgets function so as not
  // to load the entire file into memory but rather to iterate over each
  // line separately.
  $bytes_read = 0;
  $in_fh = fopen($file_path, "r");
  while ($line = fgets($in_fh)) {

    // Calculate how many bytes we have read from the file and let the
    // importer know how many have been processed so it can provide a
    // progress indicator.
    $bytes_read += drupal_strlen($line);
    $this->setItemsHandled($bytes_read);

    // Remove any trailing white-space from the line.
    $line = trim($line);

    // Split line on a comma into an array.  The feature name appears in the
    // first "column" of data and the property in the second.
    $cols = explode(",", $line);
    $feature_name = $cols[0];
    $this_value = $cols[1];

    // Our file has a header with the name 'Feature name' expected as the
    // title for the first column. If we see this ignore it.
    if ($feature_name == 'Feature name'){
       continue;
    }

    // Using the name of the feature from the file, see if we can find a
    // record in the feature table of Chado that matches.  Note: in reality
    // the feature table of Chado has a unique constraint on the uniquename,
    // organism_id and type_id columns of the feature table.  So, to ensure
    // we find a single record ideally we should include the organism_id and
    // type_id in our filter and that would require more widgets on our form!
    // For simplicity, we will just search on the uniquename and hope we
    // find unique features.
    $match = ['uniquename' => $feature_name];
    $results = chado_select_record('feature', ['feature_id'], $match);

    // The chado_select_record function always returns an array of matches. If
    // we found no matches then this feature doesn't exist and we'll skip
```

(continues on next page)

(continued from previous page)

```
    // this line of the file.  But, log this issue so the user knows about it.
    if (count($results) == 0) {
      $this->logMessage('The feature, !feature, does not exist in the database',
        ['!feature' => $feature_name], TRIPAL_WARNING);
      continue;
    }

    // If we failed to find a unique feature then we should warn the user
    // but keep on going.
    if (count($results) == 0) {
      $this->logMessage('The feature, !feature, exists multiple times. ' .
        'Cannot add a property', ['!feature' => $feature_name], TRIPAL_WARNING);
      continue;
    }

    // If we've made it this far then we have a feature and we can do the
    // insert.
    $feature = $results[0];
    $record = [
      'table' => 'feature',
      'id' => $feature->feature_id
    ];
    $property = [
      'type_id' => $cvterm_id,
      'value' => $this_value,
    ];
    $options = ['update_if_present' => TRUE];
    chado_insert_property($record, $property, $options);
  }
}
```

2.8.6 Logging and Progress

During execution of our importer it is often useful to inform the user of progress, status and issues encountered. There are several functions to assist with this. These include the `logMessage`, `setTotalItems` and `setItemsHandled` functions. All three of these functions were used in the sample code above of the `loadMyFile` function. Here, we provide a bit more detail.

2.8.6.1 The logMessage function

The `logMessage` function is meant to allow the importer to provide status messages to the user while the importer is running. The function takes three arguments:

1) a message string.

2) an array of substitution values.

3) a message status.

The message string contains the message for the user. You will notice that no variables are included in the string but rather tokens are used as placeholders for variables. This is a security feature provided by Drupal. Consider these lines from the code above:

```
$this->logMessage('The feature, !feature, does not exist in the database',
  ['!feature' => $feature_name], TRIPAL_WARNING);
```

Notice that `!feature` is used in the message string as a placeholder for the feature name. The mapping of `!feature` to the actually feature name is provided in the array provided as the second argument. The third argument supports several message types including `TRIPAL_NOTICE`, `TRIPAL_WARNING` and `TRIPAL_ERROR`. The message status indicates a severity level for the message. By default if no message type is provided the message is of type `TRIPAL_NOTICE`.

Any time the `logMessage` function is used the message is stored in the job log, and a site admin can review these logs by clicking on the job in the `admin > Tripal > Tripal Jobs` page.

Note: You should avoid using `print` or `print_r` statements in a loader to provide messages to the end-user while loading the file. Always use the `logMessage` function to ensure all messages are sent to the job's log.

2.8.6.2 The setTotalItems and setItemsHandled functions

The `TripalImporter` class is capable of providing progress updates to the end-user while the importer job is running. This is useful as it gives the end-user a sense for how long the job will take. As shown in the sample code above for the `loadMyFile` function, The first step is to tell the `TripalImporter` how many items need processing. An **item** is an arbitrary term indicating some measure of countable "units" that will be processed by our importer.

In the code above we consider a byte as an item, and when all bytes from a file are read we are done loading that file. Therefore the `setTotalItems` function is used to tell the importer how many bytes we need to process. As we read each line, we count the number of bytes read and provide that number to the `setItemsHandled` function. The `TripalImporter` class will automatically calculate progress and print a message to the end-user indicating the percent complete, and some additional details such as the total amount of memory consumed during the loading.

Note: All importers are different and the "item" need not be the number of bytes in the file. However, if you want to provide progress reports you must identify an "item" and the total number of items there are for processing.

2.8.7 Testing Importers

Unit Testing is a critically important component of any software project. You should always strive to write tests for your software. Tripal provides unit testing using the `phpunit` testing framework. The Tripal Test Suite provides a strategy for adding tests for your new Importer. It will automatically set up and bootstrap Drupal and Tripal for your testing environment, as well as provide database transactions for your tests, and factories to quickly generate data. We will use the Tripal Test Suite to provide unit testing for our `ExampleImporter`.

Note: Before continuing, please install and configure Tripal Test Suite.

For instructions on how to install, configure, and run Tripal Test Suite, please see the Tripal Test Suite documentation.

2.8.7.1 Example file

When developing tests, consider including a small example file as this is good practice both to ensure that your loader works as intended, and for new developers to easily see the expected file format. For our `ExampleImporter`, we'll include the following sample file and store it in this directory of our module: `tests/data/example.txt`.

Table 1: Example input file

Feature name	CVterm value
test_gene_1	blue
test_gene_2	red

2.8.7.2 Loading the Importer

Testing your loader requires a few setup steps. First, TripalImporters are not explicitly loaded in your module (note that we never use `include_once()` or `require_once` in the `.module` file). Normally Tripal finds the importer automatically, but for unit testing we must include it to our test class explicitly. Second, we must initialize an instance of our importer class. Afterwards we can perform any tests to ensure our loader executed properly. The following function provides an example for setup of the loader for testing:

```
private function run_loader(){

  // Load our importer into scope.
  module_load_include('inc', 'tripal_example_importer', 'includes/TripalImporter/
↪ExampleImporter');

  // Create an array of arguments we'll use for testing our importer.
  $run_args = [
    'analysis_id' => $some_analysis_id,
    'cvterm' => $some_cvterm_id
  ];
  $file = ['file_local' => __DIR__ . '/../data/exampleFile.txt'];

  // Create a new instance of our importer.
  $importer = new \ExampleImporter();
  $importer->create($run_args, $file);

  // Before we run our loader we must let the TripalImporter prepare the
  // files for us.
  $importer->prepareFiles();
  $importer->run();
}
```

Note: We highly recommend you make use of database transactions in your tests, especially when running loaders. Simply add `use DBTransaction;` at the start of your test class. Please see the Tripal Test Suite documentation for more information.

2.9 Creating Custom Web Services

2.9.1 Introduction

New in Tripal v3 are RESTful web services. These web-services are designed to allow end-users to access data programmatically using any programming language of their choice. Web services in Tripal v3 are provided by the **tripal_ws** module. By default the module provides the "content" service. This service provides access via a RESTful interface to all of the content published on a Tripal site. It is meant to respond immediately as the site admin makes changes to published data and is expected to always provide access to the same data displayed on the site (nothing more and nothing less).

Tripal v3 has been redesigned from Tripal v2 to be fully organized around controlled vocabularies. All data made available via Tripal is expected to be described by the terms of a controlled vocabulary or ontology. For example, all content types in Tripal are assigned a controlled vocabulary term that describes what the content is. Additionally, each field of data attached to a content type also is described using a controlled vocabulary term. If a field provides more than just a single data value (i.e. it provides a list or nested structured array of data of key/value pairs) then each of the keys for those pairs must also be a controlled vocabulary term. This requirement allows Tripal to fully describe the data it houses to any other Tripal site and any other script or service that can read the web services. As long as the client application understands the vocabulary term it will understand the meaning of the data. Using controlled vocabulary terms to describe all data allows a Tripal site to participate in the Semantic Web.

Finally, the Tripal RESTful services are meant to be discoverable. In some cases, when a web services is designed, the only way to understand the structure of it and the operations that it provides are for a programmer to read online documentation for the service before she can write the client application. However, to better support automatic data discovery without human intervention by a client the Tripal web services have been designed to be discoverable. To this end, Tripal uses the Hydra Core Vocabulary specification. It fully describes all of the services, their operations, and the resulting value. A client application that understands the Hydra language can therefore learn to use the web service without human intervention. However, in practice, its a good idea to continue to provide online documentation for humans. And this User's Guide *provides those instructions* for the default Tripal content service.

This documentation provides instructions to construct your own custom web services and making that service available through Tripal's existing web service infrastructure. This will enable your web service to be discoverable and to provide a consistent experience to end-users. Before proceeding with the following instructions, please review the **Structure of a Web Service Response** section on the User's Guide *Web Services page*.

2.9.2 Getting Started

Before creation of your new web services you must first consider the following in your design:

1. What functionality will your service provide. This will dictate the URL paths that you will need and what operations (Create, Read, Update, Delete) your service will support.
2. What controlled vocabularies do you need to use to describe the data that your service may provide.
3. What human-readable label can you give to your service.
4. What one-word describes the type of service you are creating.
5. What human-readable description should you provide to users of your service.

When you are ready to begin construction of your web service, you must first create a custom Drupal module. The assumption is that you already know how to create new Drupal modules, or you have access to an existing one into which you can add your new web services. Prepare your custom module by creating the following directory structure:

```
[module_name]/includes/TripalWebService
```

Where [module_name] is the name of your custom module. It is inside of this directory that you will place the code for your new web services. Tripal is designed to recognize this directory, discover the web services described inside of it and to automatically make them available! You need only program the service and Tripal does the rest.

Note: When selecting a version number it is best practice to start with 0.1. The first number represents the "major" number and the second number is the "minor" number. When minor bug fixes or changes to the service occur the minor number should be incremented. When major changes occur that affect the structure of the web service or it's operations then the major number should be incremented. The service can be named whatever you like. This class file should be named according to this schema with a **.inc** extension.

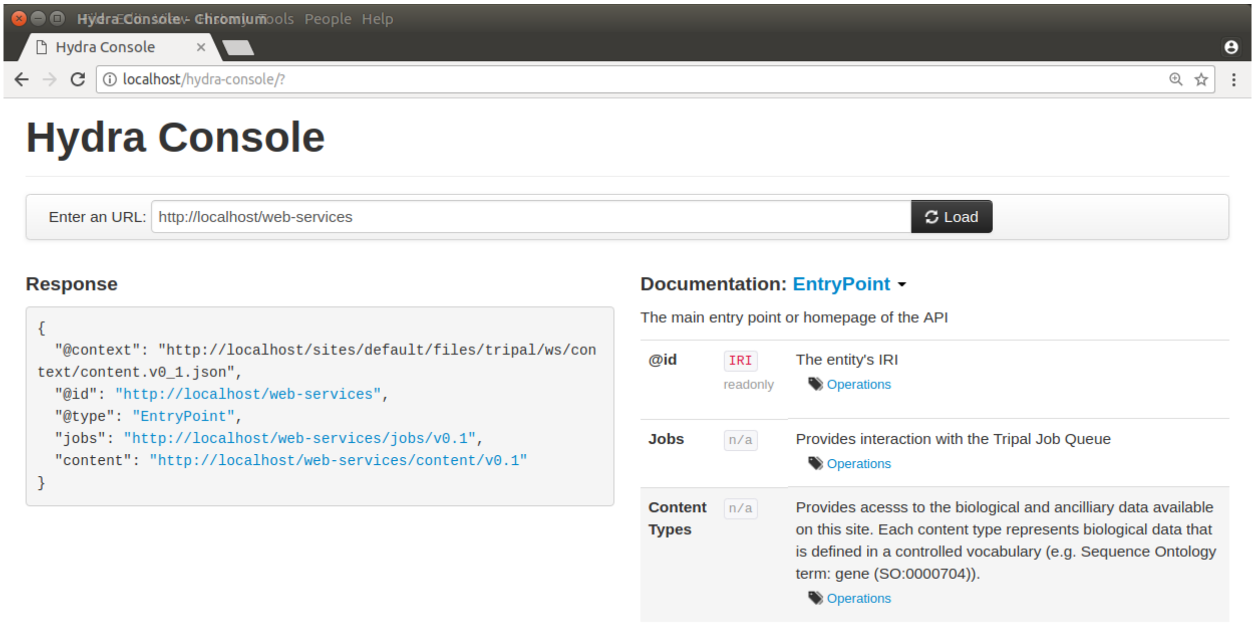
For this tutorial, suppose we wanted to create a web service that allowed someone to interact with the Tripal Job queue. We could name our new class the **TripalJobService_v0.1** class and we would create this class in the file:

```
[module_name]/includes/TripalWebService/TripalJobService_v0_1.inc
```

Within this file we will implement our class with the following structure:

```
class TripalJobService_v0_1 extends TripalWebService {

  /**
   * The human-readable label for this web service.
   */
  public static $label = 'Jobs';

  /**
   * A bit of text to describe what this service provides.
   */
  public static $description = 'Provides interaction with the Tripal Job Queue';

  /**
   * A machine-readable type for this service. This name must be unique
   * among all Tripal web services and is used to form the URL to access
   * this service.
   */
  public static $type = 'jobs';

  /**
   * Implements the constructor
   */
  public function __construct($base_path) {
    parent::__construct($base_path);
  }
}
```

This is all we need for Tripal to recognize our new service! Notice that the class implementation extends the TripalWebSerivce class and it sets a few static variables that defines the label, description and name for this service. Finally, this class defines the constructor which simply calls the parent class constructor. Be sure to always call the parent constructor when you implement your own service. We can now use the Hydra console to see our service. Note, that the hydra console must be able to have access to your site. For this tutorial, the Tripal site is temporarily hosted on the local machine, and hence Hydra has been installed locally. To see if your new service shows up, enter the URL for your site into the Hydra console and you should see it appear:



Hydra Console

Enter an URL: http://localhost/web-services

Response

```
{
  "@context": "http://localhost/sites/default/files/tripal/ws/con
text/content.v0_1.json",
  "@id": "http://localhost/web-services",
  "@type": "EntryPoint",
  "jobs": "http://localhost/web-services/jobs/v0.1",
  "content": "http://localhost/web-services/content/v0.1"
}
```

Documentation: EntryPoint

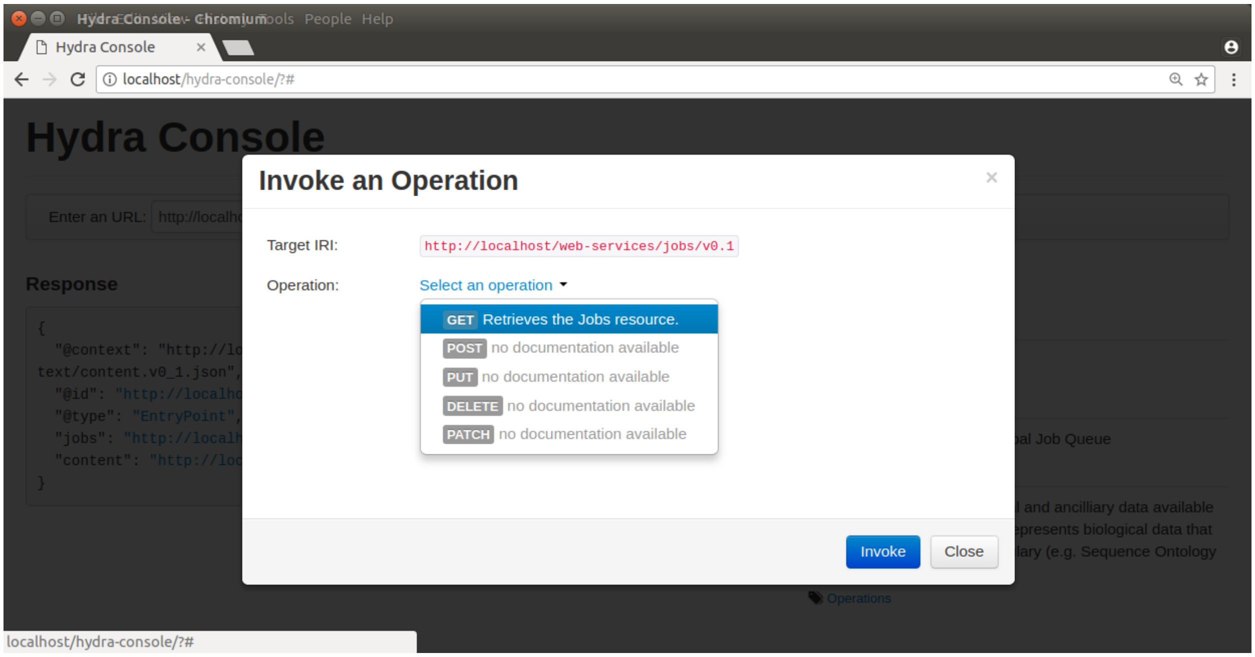
The main entry point or homepage of the API

@id	IRI readonly	The entity's IRI Operations
Jobs	n/a	Provides interaction with the Tripal Job Queue Operations
Content Types	n/a	Provides acess to the biological and ancilliary data available on this site. Each content type represents biological data that is defined in a controlled vocabulary (e.g. Sequence Ontology term: gene (SO:0000704)). Operations

Notice in the above screen shot that our **jobs** service is now present in the **Response** section, and in the **Documentation** section! The **Response** section shows the JSON array returned by Tripal and the **Documentation** section displays the information about our service.

2.9.3 Documenting the Services

Our service appears in the Hydra console but if we try to use Hydra to perform a **GET** operation on our Jobs service there will be no interesting response and no documentation. Try this by clicking on the link in the **Response** section for our Jobs service, and selecting the GET operation.



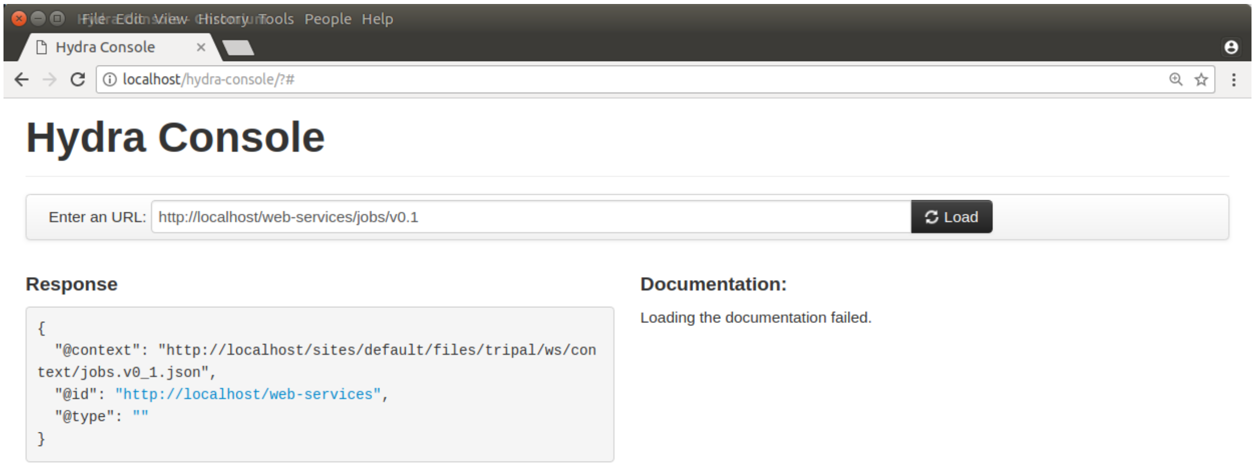
Invoke an Operation

Target IRI: http://localhost/web-services/jobs/v0.1

Operation: Select an operation

- GET Retrieves the Jobs resource.
- POST no documentation available
- PUT no documentation available
- DELETE no documentation available
- PATCH no documentation available

You will see that our service provides nothing:



File Edit View History Tools People Help
Hydra Console
localhost/hydra-console/?#

Hydra Console

Enter an URL: http://localhost/web-services/jobs/v0.1

Response

```
{
  "@context": "http://localhost/sites/default/files/tripal/ws/con
text/jobs.v0_1.json",
  "@id": "http://localhost/web-services",
  "@type": ""
}
```

Documentation:

Loading the documentation failed.

Before we create our service we should have planned the design of our service. Suppose for now we just wanted to provide read-only access to job information. Our design for the web service is quite simple and consists of the these resources (URLs and operations):

TABLE!

Before we begin implementation of our web service we must first document these resources. To do this we must add a new function to our TripalJobService_v0_1 class named **getDocumentation**. The following code shows this initial implementation of that function in our class:

```
/**
    * @see TripalWebService::getDocumentation()
    */
   public function getDocumentation() {
       return parent::getDocumentation();
   }
```

Notice currently all this function does is call the parent getDocumentation function. Now, the first thing we need to document is our web service classes. A web service class (not to be confused with the PHP class) simply refers to a resource. A resource is any distinct URL within web services. So, according to our design above we have two resources and hence two classes: a jobs collection resource and a job resource. Tripal must describe all of the classes (i.e. resources) using the Hydra method. This makes the web service discoverable. For example, with the content web service, Tripal provides a resource for each content type. The Gene content type that Tripal provides is described using the Hydra method in a JSON array with the following:

```
{
  "@id": "http://www.sequenceontology.org/browser/current_svn/term/SO:0000704",
  "@type": "hydra:Class",
  "hydra:title": "Gene",
  "hydra:description": "A region (or regions) that includes all of the sequence␣
↪elements necessary to encode a functional transcript. A gene may include regulatory␣
↪regions, transcribed regions and\/or other functional sequence regions. [SO:immuno_
↪workshop]",
  "subClassOf": "hydra:Resource",
  "supportedOperation": [
    {
      "@id": "_:gene_retrieve",
      "@type": "hydra:Operation",
      "method": "GET",
      "label": "Retrieves the Gene resource.",
```

(continues on next page)

(continued from previous page)

```
      "description": null,
      "statusCodes": [],
      "expects": null,
      "returns": "http://www.sequenceontology.org/browser/current_svn/term/SO:0000704"
     }
  ]
}
```

In the above array, notice the @id is URL that represents a unique identifier for the class. The @type will always be 'hydra:Class' because we are documenting a resource. Then there is information about the class defined using the 'hydra:title' and 'hydra:description'. The 'subclassOf' is always set to 'hydra:Resource'. Next is the list of supported operations for this resource. Remember, in our design we only want to support the GET operation for our Jobs service, so just like in the example above, the method we will support is GET. The key/value pairs for the GET method are described using Hydra terms.

For our services we need to provide the information to Tripal so that it can generate these Hydra JSON arrays that document our service. Tripal provides some easy API functions to help with this. The first is the **addDoc** member function. This function will take as input the class details, operations and properties necessary to generate the documentation for our class. First, lets use this function to document our Jobs Collection resource. Below is sample code that will do this for us.

```
public function getDocumentation() {
$term = tripal_get_term_details('local', 'computational_jobs');
$details = array(
    'id' => $term['url'],
    'title' => $term['name'],
    'description' => $term['definition'],
);
$operations = array(
    'GET' => array(
        'label' => 'Computational Jobs',
        'description' => 'Retrieves the list of computational jobs that have been submitted on this site.',
        'returns' => $term['url'],
        'type' => '_:computational_jobs_retrieve',
    ),
);
$properties = array(
);
$this->addDocClass($details, $operations, $properties);
return parent::getDocumentation();
```

In the code above we add the documentation for our Job Collection class. There are three different arrays, one for the class details, one for the operations that the class supports and the third for properties. For now, the properties array is left empty. We'll come back to that later. All classes must use a controlled vocabulary term. Notice that the term used for this class is a term local to the database named 'computational_jobs'. Normally when creating a class we would try to use a term from a published controlled vocabulary. A large number of these vocabularies can be searched using the EBI Ontology Lookup Service. Unfortunately, an appropriate term could not be found in a published vocabulary, so we had to create a local term. We can use Tripal's API functions to easily add new terms. The following code should be placed in the install() function of your module to ensure the term is available:

```
$term = tripal_insert_cvterm(array(
        'id' => 'local:computational_job',
        'name' => 'Computational Job',
        'cv_name' => 'local',
```

(continues on next page)

(continued from previous page)

```
        'definition' => 'A computational job that executes a specific task.',
    ));
    $term = tripal_insert_cvterm(array(
        'id' => 'local:computational_jobs',
        'name' => 'Computational Jobs',
        'cv_name' => 'local',
        'definition' => 'A set of computational jobs where each job executes a␣
↪specific task.',
    ));
```

You'll notice in the code above that the @id of the Class is the URL of the term. Using the **tripal_get_term_details** function we can get the URL for the term. The URL serves as the unique identifier for this term. We simply set the title and description for the class using the term details. For the operation, we can specify any of the HTTP protocols (e.g. GET, PUT, PUSH, DELETE and PATCH). Here we are currently only supporting read-only operations so we only need to provide a 'GET' operation. Our jobs collection resource is now documented!

2.9.4 Implementing a Collection Resource

Now that our job collection resource is documented we can implement the resource using the **handleRequest** function. We currently only support two paths for our web services as indicated in our design table above. Those include the default path where our job collection resource is found and an additional path with the job ID appended where individual job resources are found. First, we will implement the Job Collections resource:

```
/**
 * @see TripalWebService::handleRequest()
 */
public function handleRequest() {

    // Get the content type.
    $job_id = (count($this->path) > 0) ? $this->path[0] : '';

    // If we have a content type then list all of the entities that belong
    // to it.
    if (!$job_id) {
        $this->doJobsList();
    }
}
```

in the code above need to determine if the resource is a job collection or a job resource. To do that we can check to see if a job_id was provided. The TripalWebService class provides as a member element the full URL path broken into an array of elements. Because our job_id would always be in the first element of the path (after our base path for the service) we can use **$this->path[0]** to look for a job_id. If one is not provided then we can execute a function called **doJobsList** and the code for that is as follows:

```
/**
 * Generates the job collection resource.
 */

private function doJobsList() {
    // If the user has specified a limit or page number then use those to
    // get the specific jobs.
    $limit = isset($this->params['limit']) ? $this->params['limit'] : '25';
    $page = isset($this->params['page']) ? $this->params['page'] : 0;

```

(continues on next page)

(continued from previous page)

```
    // Get the list of jobs for the given page, and the total number.
    $offset = $page * $limit;
    $jobs = tripal_get_jobs($offset, $limit);
    $num_records = tripal_get_jobs_count();

    // Set the current resource to be a new TripalWebServiceCollection resource,
    // and pass in the current service path, and set the pager.
    $service_path = $this->getServicePath();
    $this->resource = new TripalWebServiceCollection($service_path, $this->params);
    $this->resource->setType('local:computational_jobs');
    $this->resource->initPager($num_records, $limit, $page);

    // Now add the jobs as members
    foreach ($jobs as $job) {
        $member = new TripalWebServiceResource($service_path);
        $member->setID($job->job_id);
        $member->setType('local:computational_job');
        $member->addProperty('schema:ItemPage', url('admin/tripal/tripal_jobs/view/' . $job->job_id, array('absolute' => TRUE)));
        $this->resource->addMember($member);
    }
}
```

The first few lines of code above are as follows:

```
$limit = isset($this->params['limit']) ? $this->params['limit'] : '25';
$page = isset($this->params['page']) ? $this->params['page'] : 0;
```

Remember, that we wanted to allow for paging of our job collection. We could have hundreds or thousands of jobs over time and we do not want to slow the page load by loading all of those jobs. Therefore the page and limit parameters that can be added to the URL are available via the params member as a set of key/value pairs. Next, using Tripal API function calls, we get the list of jobs that the user has requested to see (or the first page by default):

```
$offset = $page * $limit;
  $jobs = tripal_get_jobs($offset, $limit);
  $num_records = tripal_get_jobs_count();
```

Now that we have our list of jobs to use in the collection we next need to build the resource. We do this by setting the resource member of our TripalJobService_v0_1 class. Tripal provides an easy way for constructing a collection via a class named TripalWebServiceCollection. This class provides the necessary functions to easily create a collection resource that in the end will generate the appropriate JSON for us. To create a collection we first instantiate a new instance of the TripalWebServiceCollection class and pass it the URL path that it corresponds to (in this case our base service path for the service). We assign this new object to the resource member of our class.

```
$service_path = $this->getServicePath();
$this->resource = new TripalWebServiceCollection($service_path, $this->params);
```

Next we need to indicate what type of collection this is. Remember the controlled vocabulary terms we created previously? We need to use those again to set the type. Our term for a job collection is: local:computational_jobs. So, we need to use this to set the type:

```
$this->resource->setType('local:computational_jobs');
```

Now, because we have instantiated a TripalWebServiceCollection object it can handle creation of the pager for us. We just need to tell it how many total records there are, the page and number of records per page (i.e. limit):

```
$this->resource->initPager($num_records, $limit, $page);
```

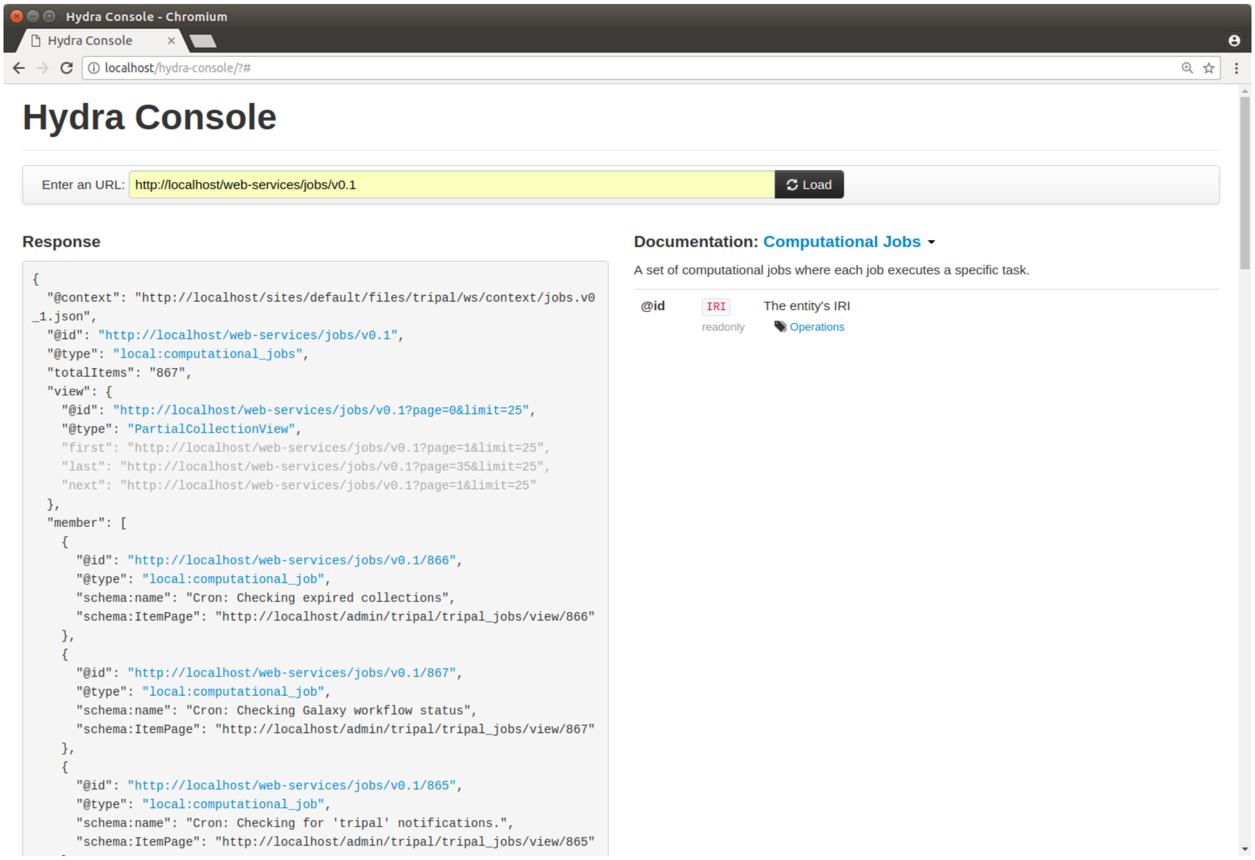
Lastly, we need to add our "members" of the collection. These are the jobs from our query. The following for loop iterates through all of our jobs and creates new member objects that are added to our collection:

```
foreach ($jobs as $job) {
    $member = new TripalWebServiceResource($service_path);
    $member->setID($job->job_id);
    $member->setType('local:computational_job');
    $member->addProperty('schema:name', $job->job_name);
    $member->addProperty('schema:ItemPage', url('admin/tripal/tripal_jobs/view/' .
↪$job->job_id, array('absolute' => TRUE)));
    $this->resource->addMember($member);
}
```

Notice in the code above that each job is an instance of the class called TripalWebServiceResource. We use this class because each element of the collection is a reference to a resource and we reference the ID and the type. In the code above we create the new member resource, we set it's type to be the vocabulary term 'local:computational_job' and eventually, use the addMember function of our TripalWebServiceCollection to add it to the collection.

Also in the code above is a new function named addProperty. We want to add some additional information about the job to help the end-user understand what each job is and how to get to it. Here we add two properties, one that is the job name and another that is the page URL for the job on our Tripal site. With these properties the client can quickly see the title and can go see the job on the Tripal site by following the given URL. Note, a resource always has two identifying pieces of information: the ID and the Type. So, everything else is added as a property of the resource. Also notice that the first argument when using the addProperty function is a controlled vocabulary term. Here we've used the terms **schema:name** and **schema:ItemPage**. These terms are from the Schema.org vocabulary and the define what these properties are: a name and an item's page.

Now that we have what looks like enough code to handle the job collection resource, we can return to Hydra to test if our new resource is working. The screenshot below shows the results:



Enter an URL: http://localhost/web-services/jobs/v0.1

```
{
  "@context": "http://localhost/sites/default/files/tripal/ws/context/jobs.v0
_1.json",
  "@id": "http://localhost/web-services/jobs/v0.1",
  "@type": "local:computational_jobs",
  "totalItems": "867",
  "view": {
    "@id": "http://localhost/web-services/jobs/v0.1?page=0&limit=25",
    "@type": "PartialCollectionView",
    "first": "http://localhost/web-services/jobs/v0.1?page=1&limit=25",
    "last": "http://localhost/web-services/jobs/v0.1?page=35&limit=25",
    "next": "http://localhost/web-services/jobs/v0.1?page=1&limit=25"
  },
  "member": [
    {
      "@id": "http://localhost/web-services/jobs/v0.1/866",
      "@type": "local:computational_job",
      "schema:name": "Cron: Checking expired collections",
      "schema:ItemPage": "http://localhost/admin/tripal/tripal_jobs/view/866"
    },
    {
      "@id": "http://localhost/web-services/jobs/v0.1/867",
      "@type": "local:computational_job",
      "schema:name": "Cron: Checking Galaxy workflow status",
      "schema:ItemPage": "http://localhost/admin/tripal/tripal_jobs/view/867"
    },
    {
      "@id": "http://localhost/web-services/jobs/v0.1/865",
      "@type": "local:computational_job",
      "schema:name": "Cron: Checking for 'tripal' notifications.",
      "schema:ItemPage": "http://localhost/admin/tripal/tripal_jobs/view/865"
```

Documentation: Computational Jobs

A set of computational jobs where each job executes a specific task.

@id IRI readonly The entity's IRI

It's clear that our resource is working! However, there are some issues. The URL for the ItemPage does not show as clickable, and we're missing descriptions of the properties in the Documentation column on the right. We'll fix that issue a bit later. For now, this looks good.

2.9.5 Simplifying the Property Keys

Note: The rest of the guide is under construction

2.9.6 Implementing a Resource

2.9.7 Documenting Properties

2.9.8 More on Parameters

2.9.9 Implementing Other Operations

2.9.10 Controlling Access to Resources

Debugging and Troubleshooting

2.10 Continuous Integration

Continuous integration refers to an active code base where multiple developers are integrating their changes continuously into a single cohesive project. One of the biggest keys to making continuous integration work is testing before integrating! Tripal Test Suite makes developing tests using PHPUnit much easier and this tutorial will show you how to ensure those tests are run every time you push to GitHub!

2.10.1 Using GitHub Workflows

First, what is a GitHub Workflow? From GitHub, "a workflow is a configurable automated process made up of one or more jobs". I like to think of it as configuring a bioinformatics pipeline. For the purposes of this tutorial, we are telling GitHub how to execute our PHPUnit tests automatically.

You can learn more about GitHub Action Workflows from GitHub directly. We will cover it breifly here specifically for Tripal PHPUnit testing but the official tutorial is still recommended.

2.10.1.1 Telling GitHub about your workflow

This is as simple as creating your workflow file in the right place. You want to create a new file named *phpunit.yml* at *.github/workflows/* in your repository. GitHub will automatically know this file describes a workflow and will execute the workflow based on the configuration you provide.

2.10.1.2 Configuring a workflow to run PHPUnit tests

All configuration will be done in the file created in step 1.

First we provide the name, when we want to run the workflow (i.e. on push and PRs) and the basic structure of a workflow:

```
name: PHPUnit

# Controls when the workflow will run.
# Run this workflow every time a new commit is pushed to your repository
on: [push, pull_request]

jobs:
  # This key is the name of the job.
  run-tests:
    # The type of system that the job will run on.
    runs-on: ubuntu-latest
```

We want our tests to be run five times, for each of PHP 7.1, 7.2, 8.0, 8.1, and 8.2. This is done by specifing a matrix build which can be done as follows:

```
jobs:
  # This key is the name of the job.
  run-tests:
    # The type of system that the job will run on.
    runs-on: ubuntu-latest

    # Matrix Build for this job.
    strategy:
      matrix:
```

(continues on next page)

(continued from previous page)

```
        php-versions: ['7.1', '7.2', '8.0', '8.1', '8.2']

    # Name the matrix build so we can tell them apart.
    name: PHPUnit Testing (PHP ${{ matrix.php-versions }})
```

We also want to tell GitHub to setup a PostgreSQL server for us. This is done using services:

```
jobs:
  # This key is the name of the job.
  run-tests:
    # The type of system that the job will run on.
    runs-on: ubuntu-latest

    # Service containers to run with `run-tests`
    services:
      # Label used to access the service container
      postgres:
        # Docker Hub image
        image: postgres
        env:
          POSTGRES_USER: tripaladmin
          POSTGRES_PASSWORD: somesupersecurepassword
          POSTGRES_DB: testdb
        # Set health checks to wait until postgres has started
        options: >-
          --health-cmd pg_isready
          --health-interval 10s
          --health-timeout 5s
          --health-retries 5
        ports:
          # Maps tcp port 5432 on service container to the host
          - 5432:5432
```

Finally we can actually tell GitHub what steps we want to run using this beautiful container we have setup! We want to:

```
jobs:
  # This key is the name of the job.
  run-tests:
    # The type of system that the job will run on.
    runs-on: ubuntu-latest

    steps:
    # 1) Checkout the repository and setup workspace.
    - uses: actions/checkout@v2

    # 2) Setup PHP according to the version passed in.
    - name: Setup PHP
      uses: shivammathur/setup-php@v2
      with:
        php-version: ${{ matrix.php-versions }}
        extensions: mbstring, intl, php-pgsql, php-gd, php-xml
        ini-values: memory_limit=2G
        coverage: xdebug
        tools: composer, phpunit
```

(continues on next page)

(continued from previous page)

```
    # 3) Install Drush/Drupal/Tripal
    - name: Setup Drush, Drupal 7.x, Tripal 3.x
      id: tripalsetup
      uses: tripal/setup-tripal-action@7.x-3.x-1.0
      with:
        postgres_user: tripaladmin
        postgres_pass: somesupersecurepassword
        postgres_db: testdb

    # 4) Install Tripal Extension Module.
    - name: Install Tripal Extension Module
      id: installextension
      env:
        DRUSH: ${{ steps.tripalsetup.outputs.drush_path }}
        DRUPAL_ROOT: ${{ steps.tripalsetup.outputs.drupal_root }}
      run: |
        mkdir -p $DRUPAL_ROOT/sites/all/modules/example_module
        cp -R * $DRUPAL_ROOT/sites/all/modules/example_module
        cd $DRUPAL_ROOT
        $DRUSH en -y example_module

    # 5) Runs the PHPUnit tests.
    # https://github.com/mheap/phpunit-github-actions-printer is used
    # to report PHPUnit fails in a meaningful way to github in PRs.
    - name: PHPUnit Tests
      env:
        DRUSH: ${{ steps.tripalsetup.outputs.drush_path }}
        DRUPAL_ROOT: ${{ steps.tripalsetup.outputs.drupal_root }}
      run: |
        cd $DRUPAL_ROOT/sites/all/modules/example_module
        composer require --dev mheap/phpunit-github-actions-printer --quiet
        composer update --quiet
        ./vendor/bin/phpunit --printer mheap\\GithubActionsReporter\\Printer
```

In step 4 above, I have provided an example of what installing your extension module might look like. The run section will need to be modified according to your module and should include downloading and installing any dependencies, applying any patches and installing your module. If your tests require configuration then that should also be included here.

In step 5 we run our PHPUnit tests! All you need to change here is the directory name for your module. The *mheap\GithubActionsReporter\Printer* will ensure any errors reported by PHPUnit are shown on your PR and Action summary.

All steps before step 4 should be generic for any extension module! You can find the full configuration file on the README of the SetupTripalAction.

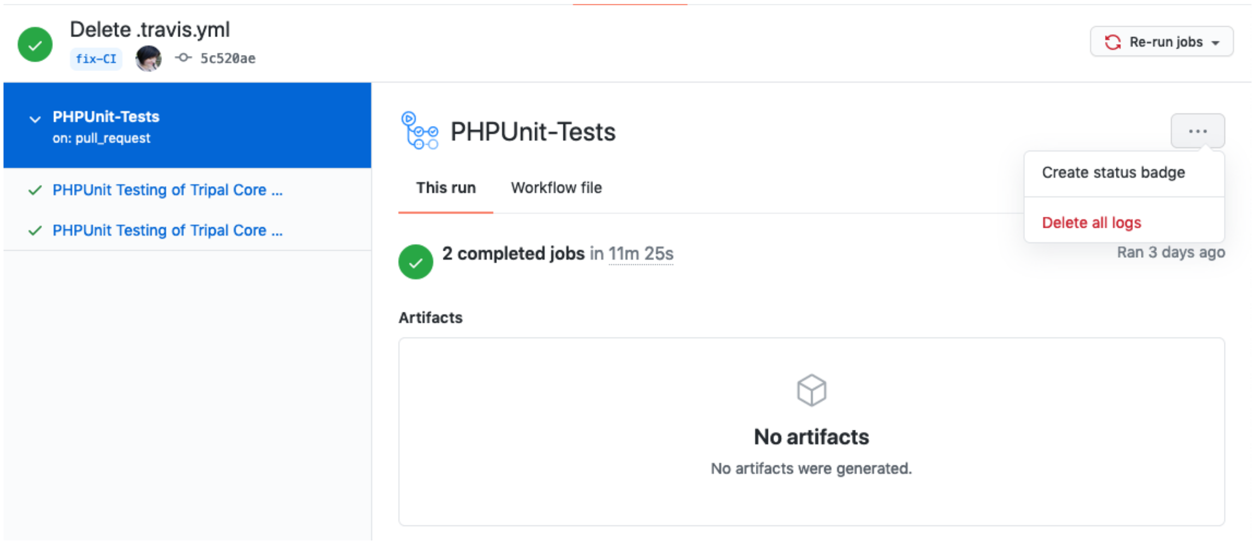
2.10.2 Checking your Action

Everytime you push commits to your repository and when you create a pull request, your action will be run. Thus to test your action, commit your phpunit.yml file created above to trigger the GitHub action. Then click on “Actions” at the top of your repository to see it in progress.

If you created a pull request, you will see your workflow action in the checks section at the bottom of your pull request. From here you can click on Details to see the full running of the job.

2.10.3 Adding the Badge to your README

Click on Actions at the top of your repository, then click on one of the PHPUnit jobs which have already been submitted. This brings you to the job summary page where you will see a button with three dots in the top right corner. Click on this and then “Create status badge” to get the markdown to add to your README.



2 completed jobs in 11m 25s

No artifacts were generated.

2.11 Using ReadTheDocs

Note: For Tripal’s own ReadTheDocs guidelines: *Contributing to the Documentation*.

2.11.1 What is ReadTheDocs?

Documentation is important. It tells users how to use our product, and developers how to read our code. We recommend hosting documentation for your custom module somewhere easily accessible, like ReadTheDocs.

ReadTheDocs (RTD) uses the Sphinx framework to build a website in a Continuous Integration setup. Your RTD-compatible documentation is added directly to your module and when code changes are pushed to GitHub, the documentation website as defined by sphinx is built, and the “live” documentation website is updated.

Benefits to housing documentation inside of your module code are:

- Code changes can include documentation updates **in the same pull request**.
- Changes to the documentation is **subject to review, just like code changes**.
- Documentation changes are under **version control**.

2.11.2 How do I add ReadTheDocs to my project?

Below is a quick overview of steps for integrating your module’s documentation with RTD:

- Set up your ReadTheDocs account and import your project.
- Install Sphinx.

- Create a `docs` folder at the root of your project and navigate into it.
- Run the quickstart command: `sphinx-quickstart`. - This creates necessary site configuration files (`conf.py`) as well as the make script to build your site.
- Write your documentation (we're using reStructuredText (RST) format): - Create an `index.rst` as the home page. - Link other RST documents in your `index.rst`.
- Once the guide is on your master branch on GitHub, ReadTheDocs will handle the rest!

For a detailed walkthrough, please see the official ReadTheDocs getting started guide.

For RTD integration we recommend using reStructuredText (RST) to write your documentation. It might seem a little more complicated than markdown (and it is), but it's also more powerful. The choice is yours for which format to use.

ReadTheDocs also provides **versioning** tools, allowing you to post releases of the documentation so users can go back and find older documentation with almost no effort on your part.

2.11.3 What should my documentation include?

We suggest that your module include the following sections:

- Overview
- Installation and Setup Guide
- User's Manual

The Overview section should describe what features your module offers.

The Installation and setup section should guide a site administrator through installing and setting up your module. Any site-wide settings that need to be configured, environmental variables set, or anything else not handled by the automated Drupal install should be covered.

The User's guide should walk through the day-to-day usage of your module. This may include using custom importers, dashboards, or simply summaries of the new content this module provides.

2.12 Video Tutorials

Note: The following videos are provided by members of the community. While we don't ensure the videos are up to date, we would appreciate being contacted through the issue queue if you find a broken link. Also, if you have your own video and would like to contribute, we would love to hear from you!

2.12.1 Testing

The Tripal Test Suite module is used by the core Tripal module, and many custom modules, to easily set up PHPUnit and Travis CI.

- Creating and running Test Suite Tests
- Test Suite Factories and DB Transactions
- Test Suite Seeding and Publishing Data
- Test Driven Development live demo

2.12.2 Developer Tools

- Tripal 3 - Setting Up A Website From Scratch With TripalDock
- Tripal 3 - TripalDock Commands And Site Structure

2.12.3 Other

- Loading Biomaterials with Tripal Analysis Expression

CHAPTER 3

Extension Modules

The below modules are Tripal 3 compatible and grouped roughly by category:

3.1 Administrative

The following modules provide support to Tripal site administrators.

3.1.1 Private BioData

tripal GOLD

Make individual pages private while others of the same Tripal Content Type remain public! This module provides an additional permission, [Content type]: View Private Content for each Tripal Content Type and a TripalField to indicate which exact pages should be private.

Documentation Repository

3.1.2 Tripal Alchemist

Tripal Alchemist allows you to transform entities from one type to another. Define multiple bundles with the same base table and easily convert existing entities to the new type.

Documentation Repository

3.1.3 Tripal Curator

Tripal Curator is a Toolbox for curating Chado Properties. Split properties into multiple child properties, and mass reassign property CVterms.

Documentation Repository

3.1.4 Tripal HeadQuarters

tripal GOLD

Tripal HeadQuarters (HQ) provides an administrative layer for Tripal, giving users limited access to content creation forms which must be approved by an admin before they are inserted into the database. Admins can use Chado-specific permissions to define organism or project-specific administrative rights.

Documentation Repository

3.2 Analysis/Annotation

The following modules provide support for loading annotation or analysis' into Chado and displaying them on Tripal pages.

3.2.1 Tripal Analysis Expression

A module for loading, annotating, and visualizing NCBI Biomaterials and expression data.

Documentation Repository

3.2.2 Tripal Analysis Blast

This module extends the Tripal Analysis Module and provides a method for loading XML results from the NCBI blast program. Blast results appear on each feature page.

Documentation Repository

3.2.3 Tripal Analysis KEGG

This module provides a method for loading of KEGG ortholog assignments derived from the KEGG Automated Annotation Server (KAAS). KEGG assignments appear on each feature page, and a full KEGG report is available for browsing results once uploaded.

Repository

3.2.4 Tripal Analysis Interpro

This module provides a method for loading XML results from the InterProScan program. The module can load InterProScan XML v4 or InterProScan XML v5 generated from the command-line or web-based versions of InterProScan. Additionally, GO terms mapped by InterProScan can optionally be assigned to features.

Documentation Repository

3.2.5 Tripal CV-Xray

Tripal CV-Xray maps content annotations onto controlled vocabulary trees. The end result is a browseable CV field that lets users explore ontologies and find associated content.

Documentation Repository

3.2.6 Tripal OrthoQuery

The OrthoQuery module identifies orthologous genes via a Tripal database, standardizes the data for comparative analysis, performs the OrthoFinder analysis through the Tripal Galaxy API, sends the data to the user's database profile, and provides interactive visualizations. Visualization features focus on facilitating the interrogation of large gene families, examining relationships among families, and allowing direct query of the stored orthogroups and their function. Orthoquery is currently in development.

Repository

3.3 Breeding

The following modules provide support for managing breeding data.

3.3.1 BIMS

A module for an online breeding management system which allows breeders to store, manage, archive and analyze their public or private breeding program data.

Documentation Repository

3.4 Data Loading/Collection

The following modules provide interfaces for collection and/or loading of biological data.

3.4.1 Genotype Loader

tripal GOLD

A Drush-based loader for VCF files that follows the genotype storage rules outline by ND genotypes. It has been optimized to handle very large files and supports customization of ontology terms used.

Documentation Repository

3.4.2 Mainlab Chado Loader

MCL (Mainlab Chado Loader) is a module that enables users to upload their biological data to Chado database schema. Users are required to transfer their biological data into various types of data template files. MCL, then, uploads these data template files into a Chado schema.

Documentation Repository

3.4.3 Raw Phenotypes

This module was designed to aid in collection and further analysis of raw phenotypic data. It supports Excel drag-n-drop uploads with immediate validation and researcher feedback. Additionally, it provides summary charts and download functionality.

Documentation Repository

3.4.4 Tripal BibTeX

A BibTEX importer for Tripal Publications. Currently this module only provides a Drush function (`tripal-import-bibtex-pubs`; `trpimport-bibtex`) for import of BibTEX files.

Documentation Repository

3.4.5 Tripal Plant PopGen Submission

tripal SILVER

The Tripal Plant PopGen Submit (TPPS) Module supports a flexible submission interface for genotype, phenotype, environmental, and metadata for population, association, or landscape genetics studies. The portal walks the user through specific questions and collects georeferenced coordinates on plant accessions and also supports ontology standards, including the Minimal Information About a Plant Phenotyping Experiment (MIAPPE) (http://www.miappe.org/) and standard genotyping file formats, such as VCF.

Documentation Repository

3.4.6 Migrate Chado

This module is a collection of destination plugins to import biological data to a Chado database using Drupal Migrate. The Migrate module provides a flexible framework for migrating content into Drupal from other sources (e.g., when converting a web site to Drupal). Content is imported and rolled back using a bundled web interface (Migrate UI module) or included Drush commands (strongly recommended).

Documentation Repository

3.5 Developer Tools

The following modules provide support to both core and extension Tripal developers.

3.5.1 TripalDock

TripalDock is a command line tool that helps with creating and running Tripal sites using Docker. This tool is designed for developers and is not suitable for production.

Documentation Repository

3.5.2 Tripal Download API

This module provides an API for downloading Tripal/Chado data. Since download functionality is often sought after for Tripal sites and Views Data Export is not currently meeting our needs, this module aims to provide an API to aid module and site developers in making efficient, user friendly downloads available.

Documentation Repository

3.5.3 Tripal D3.js API

Provides d3.js integration for Tripal. It provides an API for developing consistent biological diagrams with a common configuration, as well as, providing some common diagrams such as pie, bar, column and pedigree diagrams.

Documentation Repository

3.5.4 Tripal Fields Generator

This is a CLI tool to help automate the generation of Tripal fields.

Documentation Repository

3.5.5 Tripal Rapid Installer

Provides rapid installation of a Tripal site using Drush.

Documentation Repository

3.5.6 Tripal Test Suite

Tripal Test Suite is a composer package that handles common test practices such as bootstrapping Drupal before running the tests, creating test file, creating and managing database seeders (files that seed the database with data for use in testing) and much more.

Documentation Repository

3.6 Third-party Integration

The following modules provide integration with external third-party tools. For example, they may allow you to easily embed the tool in Drupal/Tripal pages and/or expose data from the tool on your Tripal site.

3.6.1 BrAPI

This module provides a Breeding API end point on your Tripal site as well as a user query interface and an auto-query system to integrate external BrAPI end point data into your site dynamically. An administrative interface allows you to adjust the module settings according to the way you use Chado. A couple of hooks are also provided for module developers in order to allow customization/extension of calls.

Documentation Repository

3.6.2 Tripal Blast

tripal GOLD

This module provides a basic interface to allow your users to utilize your server's NCBI BLAST+. There is a simple interface allowing users to paste or upload a query sequence and then select from available databases and a tabular results listing with alignment information and multiple download formats (HTML, TSV, GFF3, XML) available.

Documentation Repository

3.6.3 Tripal Galaxy

tripal GOLD

This module is for integration of Tripal and the Galaxy Project. It facilitates the creation of forms on your Tripal site that submit jobs to a galaxy instance.

Documentation Repository

3.6.4 Tripal JBrowse

This module provides integration between Tripal sites and pre-existing GMOD JBrowse instances. It allows you to create pages on your Tripal site with an embedded JBrowse instance by filling out a simple form.

Documentation Repository

3.6.5 VCF Filter

This modules provides a form interface so users can custom filter existing VCF files and export in a variety of formats. The form simply provides an interface to VCFtools and uses the Tripal Download API to provide the filtered file to the user.

Documentation Repository

3.7 Searching

3.7.1 CartograTree

CartograTree is a web-based application that allows researchers to identify, filter, compare, and visualize geo-referenced biotic and abiotic data. Its goal is to support numerous multi-disciplinary research endeavors including: phylogenetics, population structure, and association studies.

Documentation Repository

3.7.2 Mainlab Chado Search

Mainlab Chado Search is a module that enables advanced search function for biological data stored in a Tripal/Chado database. By default, a set of search interfaces are provided, such as 'Gene Search' for searching genes and/or transcripts, 'Marker Search' for searching genetic markers, and 'Sequence Search' for searching any sequences stored in the Chado feature table. Searches for other data types, such as QTL, Map, Trait, Stock, Organism are also provided but may require modification to the materialized view to adjust for site-specific data storage.

Documentation Repository

3.7.3 Tripal ElasticSearch

The Tripal ElasticSearch module allows you to easily manage the indexing and display of ElasticSearch on your Tripal website. It also easily enables Cross-Site Querying, allowing you to connect to other Tripal sites and provide additional search results to your users.

Documentation Repository

3.7.4 Tripal MegaSearch

Tripal MegaSearch is a tool for downloading biological data stored in a Tripal/Chado database. The module was designed to be generic and flexible so it can be used on most Tripal sites. Site administrators may choose from 1) a set of predefined materialized views or 2) Chado base tables as the data source to serve data. If neither data source is desired, developers may create their own materialized views and serve them through Tripal MegaSearch via a flexible dynamic query form. This form allows filters to be added dynamically and combined using 'AND/OR' operators. The filters correspond to the underlying data source columns so the user can filter data on each column.

Documentation Repository

3.7.5 Tripal Sequence Similarity Search

This module supports sequence similarity search on a Tripal website through a new dual application option. The Tripal module provides access to the speed increase available through Diamond for BLASTP/BLASTX style searches as well as traditional NCBI BLAST for BLASTN. Both applications are integrated into a single interface that provides file upload or copy/paste sequence support for the query and access to formatted databases for NCBI BLAST or Diamond. The target databases can be customized for the categories of whole genome, gene, protein, and transcriptome/unigene. The administration interface allows the admin user to set what pre-indexed databases are available (which show up in a dropdown menu). The module supports execution of the searches on a remote machine so that the search is not running directly on the limited resources typically associated with web servers.

Documentation Repository

3.8 Visualization/Display

The following modules provide specialized displays for Tripal content types.

3.8.1 Analyzed Phenotypes

tripal GOLD

This module provides support and visualization for partially analyzed data stored in a modified GMOD Chado schema. It is meant to support large scale phenotypic data through backwards compatible improvements to the Chado schema including the addition of a project and stock foreign key to the existing phenotype table, optimized queries and well-chosen indexes.

Documentation Repository

3.8.2 CvitEmbed

This module integrates CViTjs with Tripal to provide whole-genome visualizations. It creates one page per plot and makes them accessible via the Drupal Navigation menu.

Documentation Repository

3.8.3 Mainlab Tripal Data Display

Mainlab Tripal Data Display contains a set of Drupal/PHP templates that organize and extend the default display of the biological data hosted on a Tripal-enabled site (i.e. http://tripal.info). Supported data type includes organism, marker, QTL, germplasm (stock), map (featuremap), project, heritable phenotypic marker (MTL), environment (ND geolocation), haplotype block, polymorphism, eimage, generic gene (genes created by parsing Genbank files using the Mainlab `tripal_genbank_parser module), feature, and pub. Each of the templates can be turned on/off as desired.

Documentation Repository

3.8.4 ND Genotypes

tripal GOLD

This module provides support and visualization of genotypic data stored in a modified GMOD Chado schema. The 3.x branch of this module represents a shift towards support for large scale genotypic datasets through backwards compatible improvements to the Chado schema including a new gathering table for genotypes (genotype_call) modeled after the Chado phenotype table, optimized queries and well-chosen indexes.

Documentation Repository

3.8.5 Phylotree

This extension provides a simple file formatter for Newick tree files using the Javascript library Phylotree for display.

Documentation Repository

3.8.6 Tripal Fancy Fields

This module provides additional fields for use with Tripal 3. The current version provides a single-series chart field that can be displayed as a pie, donut, or bar chart, as well as, a simple table.

Documentation Repository

3.8.7 Tripal MapViewer

Tripal MapViewer module displays map data stored in Chado. MapViewer provides interfaces to view all linkage groups of a map, choose a linkage group and zoom in to a specific region of a linkage group, compare maps that share the same markers and change colors of markers/QTL.

The interface can be integrated into Tripal map page and hyperlinked to/from any Tripal page that are displayed in maps (marker, QTL, heritable morphological marker and/or gene). MapViewer also provides a genome comparison view with JBrowse integration and Dot plot and Correspondence Matrix support. The admin page allows site developers some flexibility in the display pattern.

Documentation Repository

3.9 In Development

The following modules are not yet ready for production or not fully Tripal 3 compatible.

3.9.1 Tripal Apollo

Tripal Apollo lets you manage user accounts for your JBrowse Apollo instances on your Tripal site. Provides a form to request apollo access, an Apollo instance content type that connects to Chado Organisms, and an administrative interface for managing requests.

Documentation Repository

3.9.2 Tripal Multi-Chado

The Multi-Chado module is an extension for Tripal 2.x and 3.x (dev branch under testing) that can be used to install more than one Chado instance across different schemata of your choice and it also enables the use of different PostgreSQL database credentials allowing the administrator to do fine tuning of database accesses. With it you can use a *same Drupal instance* for both a *public* and a *private* Chado instance, have *different releases* or *separated species*, provide a *sandbox*, run *tests* on a blank instance and more (dev-staging-prod, etc.).

Documentation Repository

If you don't see the module you are looking for here, try a Tripal-specific search on GitHub.

Informational links for this extension module list:

3.10 Tripal Module Rating System

This module rating system is meant to aid Tripal Site Administrators in choosing extension modules for their site. It is also meant to guide developers in module best practices and celebrate modules which achieve these goals.

3.10.1 Bronze

tripal BRONZE

- Has a public release.
- Should install on a Tripal site appropriate for the versions it supports.
- Defines any custom tables or materialized views in the install file (if applicable).
- Adds any needed controlled vocabulary terms in the install file (Tripal3).
- Provides Installation and admin instructions README.md (or RTD).
- Has a license (distributed with module).

3.10.2 Silver

tripal SILVER

- Follows basic Drupal Coding standards; specifically, code format and API documentation.
- **Uses Tripal API functions. Specifically, it should use the**
  - Chado Query API for querying Chado (if using Chado as the storage system). (API, *Tutorial*)
  - Tripal Jobs API for long running processes. (API)

    - TripalField class to add data to pages (Tripal3). (*Tutorial*)
- Provides ways to customize the module (e.g. drush options, field/formatter settings, admin UI).
- **Latest releases should follow Drupal naming best practices.**
    - e.g. first release for Drupal 7 should be: `7.x-1.0`.

3.10.3 Gold

- Extensive documentation for the module (similar to Tripal User's Guide). (Tutorial)
- Unit testing is implemented using PHPUnit with the TripalTestSuite or something similar.
- Continuous integration is setup (e.g. such as with TravisCI).
- Imports data via Tripal's importer class (Tripal3) (*Tutorial*).
- **Tripal 3 fields are (*Tutorial*)**
    - Fully compatible with web services.
    - The elementInfo function is fully implemented.
    - The query and queryOrder functions fully implemented.
- Web Services uses Tripal's Web Service Classes (Tripal3). (*Tutorial*)
- Code sniffing and testing coverage reports (optional but encouraged).
- Drupal.org vetted release (optional but encouraged).

3.10.4 Rate your Extension Module!

We encourage Tripal module developers to rate their modules. This can be done by *Adding your Module to this list!*

The following badges are for inclusion on your module README and documentation; however, they are only valid if your module has been included in *Extension Modules* with the given rating.

reStructuredText

```
.. image:: https://tripal.readthedocs.io/en/7.x-3.x/_images/Tripal-Bronze.png
  :target: https://tripal.readthedocs.io/en/7.x-3.x/extensions/module_rating.html
↪#Bronze
  :alt: Tripal Rating: Bronze
```

Markdown

```
[![Tripal Rating Bronze Status](https://tripal.readthedocs.io/en/7.x-3.x/_images/
↪Tripal-Bronze.png)](https://tripal.readthedocs.io/en/7.x-3.x/extensions/module_
↪rating.html#Bronze)
```

HTML

```
<a href='https://tripal.readthedocs.io/en/7.x-3.x/extensions/module_rating.html#Bronze
↪'>
    <img src='https://tripal.readthedocs.io/en/7.x-3.x/_images/Tripal-Bronze.png' alt=
↪'Tripal Rating: Bronze' />
</a>
```

Note: Replace all instances of `Bronze` with either `Silver` or `Gold` for those badges.

3.11 Adding your Module to this list!

We would love for you to contribute your own module to this list! This is done by creating a Pull Request (PR) to Tripal modify our documentation.

3.11.1 Instructions

1. From the current page, click the category in the list above that best fits your module.
2. Click the "Edit on Github" link at the top of the page.
3. Add your module using the following template.

```
Module Name
------------

This module loads in X, Y, and Z.  It provides admin for A and B, and user area C.

`Documentation <https://yourmodule.readthedocs.io/en/latest/index.html>`__
`Repository <https://github.com/you/yourmodule>`__
```

4. Rate your module using the *Tripal Module Rating System* and mention in your PR description which requirements your module meets.

3.11.2 Guidelines

- Make sure to follow alphabetical order when choosing where on the category page to add your module.
- Please write two sentences MAXIMUM about the function of the module.
- Include links to both the documentation (even if it's your README) and the repository (e.g. Github, Gitlab)
- If your module doesn't fit well in any of the existing categories, still pick the best one but then feel free to suggest a new category in the PR description.
- Extension Modules must be publicly available for download

CHAPTER 4

Tripal Community

4.1 Guidelines for Contribution to Tripal

The following guidelines are meant to encourage contribution to Tripal source-code on GitHub by making the process open, transparent and collaborative. If you have any feedback including suggestions for improvement or constructive criticism, please comment on the Github issue. **These guidelines apply to everyone contributing to Tripal whether it's your first time (Welcome!) or project management committee members.**

Note: These guidelines are specifically for contributing to Tripal. However, we encourage all Tripal extension modules to consider following these guidelines to foster collaboration among the greater Tripal Community.

Note: Guidelines serve as suggestions (**should**) or requirements (**must**). When the word "should" is used in the text below, the stated policy is expected but there may be minor exceptions. When the word "must" is used there are no exceptions to the stated policy.

4.1.1 Github Communication Tips

- Don't be afraid to mention people (@username) who are knowledgeable on the topic or invested. *We are academics and overcommitted, it's too easy for issues to go unanswered: don't give up on us!*
- Likewise, don't be shy about bumping an issue if no one responds after a few days. *Balancing responsibilities is hard.*
- Want to get more involved? Issues marked with "Good beginner issue" are a good place to start if you want to try your hand at submitting a PR.
- **Everyone is encouraged/welcome to comment on the issue queue! Tell us if you**
  - are experiencing the same problem
  - have tried a suggested fix

  - know of a potential solution or work-around
  - have an opinion, idea or feedback of any kind!
- **Be kind when interacting with others on Github! (see Code of Conduct below for further guidelines). We want to foster a**
  - Constructive criticism is welcome and encouraged but should be worded such that it is helpful :-) Direct criticism towards the idea or solution rather than the person and focus on alternatives or improvements.

4.1.2 Bugs

- **Every bug should be reported as a Github issue.**
  - Even if a bug is found by a committer who intends to fix it themselves immediately, they **should** create an issue and assign it to themselves to show their intent.
- **Please follow the issue templates as best you can. This information makes discussion easier and helps us resolve the proble**
  - Also provide as much information as possible :-) Screenshots or links to the issue on a development site can go a long way!
- Bonus points for unit tests to ensure the bug does not return :-)

4.1.3 Feature Requests

- Every feature request should start as an issue so that discussion is encouraged :-)
- **Please provide the following information (bold is required; underlined strengthens your argument):**
  - **Use Case:** fully describe why you need/want this feature
  - Generally Applicable: Why do you feel this is generally applicable? Suggest other use cases if possible. Mention (@) others that might want/need this feature.
  - Implementation: Describe a possible implementation. Bonus points for configuration, use of ontologies, ease of use, permission control, security considerations
- **All features should be optional so that Tripal admin can choose to make it available to their users.**
  - When applicable, new features should be designed such that Tripal-site admin can disable them.
  - Bonus points: for making new features configurable and easily themed.
- **Feature requests will be voted on by the project management and advisory/steering committees to determine if it should b**
  - Votes should be based on whether this feature is generally applicable and doesn't exclude existing users and not be biased by the needs of your own Tripal site.
- If a feature isn't suitable for inclusion within Tripal core, use the issue discussion as a springboard to create a Tripal extension module!

Note: In the future there will be a set of guidelines for what should be included in core. This will make the process of requesting new features more streamlined, inclusive and transparent.

4.1.4 Pull Request (PR) Guideline

The goal of this document is to make it easy for **A)** contributors to make pull requests that will be accepted, and **B)** Tripal committers to determine if a pull request should be accepted.

- **PRs that address a specific issue must link to the related issue page.**
  - In almost every case, there should be an issue for a PR. This allows feedback and discussion before the coding happens. Not grounds to reject, but encourage users to create issues at start of their PR. Better late than never :).
- **Each PR must be tested/approved by at least 1 contributor, if approved, a "trusted committer" will merge the PR.**
  - Testers **should** describe how the testing was performed if applicable (allows others to replicate the test).
  - While Tripal's review body is small, the code review must be a thorough functional test.
  - At the Project Management Committee's (PMC) discretion, a PR may be subject to a non-functional review. Generally these are small and obvious commits.
  - Tripal's guiding philosophy is to encourage open contribution. With this in mind, committers should **work with contributors** to resolve issues in their PRs. PRs that will not be merged should be closed, **transparently citing** the reason for closure. In an ideal world, features that would be closed are discouraged at the **issue phase** before the code is written!
  - The pull request branch should be deleted after merging (if not from a forked repository) by the person who performs the merge.
- **PRs that include new functionality must also provide Unit Tests.**
  - Tests **must** test the new functionality added.
  - Bonus points for testing all surrounding functionality.
  - For example, when adding feature X to custom tables, the PR must include tests for feature X and we would be greatly appreciative if it included tests for custom tables in general :-D.
- PRs **should** pass all Travis-CI tests before they are merged.
- **Branches should follow the following format:**
  - `[issue\_number]-[tripal\_version]-[short\_description]`
  - `tripal\_version` being Tv2, Tv3, etc.
  - `-[short\_description]` being optional but highly encouraged
- **Must** follow Drupal code standards
- PRs for new feature should remain open until adequately discussed (see guidelines below) and approved by a vote (all members of the PMC must vote in favour).

4.1.4.1 How to create a PR

There are great instructions on creating a PR on Digital Ocean: How To Create a Pull Request on GitHub.

The tl;dr version:

1. Fork the repository or update an existing fork
2. Clone the fork
3. Create a branch specific to your change: `[issue\_number]-[tripal\_version]-[short\_description]`

4. Make your changes, committing often with useful commit messages.
5. Push your changes to your fork.
6. Create a PR by going to your fork: target should be `tripal:7.x-3.x`. For specifics, see guidelines above.

4.1.5 How PRs and Issues are Handled

The Project Management Committee (PMC) and trusted committers will follow specific rules when working with all issues and pull requests. The rules are listed below. Anyone may provide bug fixes in which case some of the following will apply to all:

- **Every task related to Tripal (bug, feature requests, documentation, discussions) should be in Github, either as it's own issue or grouped with like tasks into a single issue.** This effectively puts our todo list on github making it transparent to anyone who wants to help. It has the benefit of showing how active our community is, keeps everyone informed with where Tripal is headed and makes it easy for others to chime in with experience, comments and support.
- **Guidelines for Tagging Issues:**
  - The first committer who comments on an issue should tag it with the version of Tripal it applies to.
  - Issues with a suggested fix or work-around should be tagged with "Fix Required" to let others know a PR is needed.
  - Only tag an issue with "bug" once it has been shown to be reproducible. If it's not reproducible by a committer but you feel it is a bug then tag it as "potential bug".
  - If multiple users have commented that a bug affects them, tag it as "affects multiple users".
  - Issues that require a PR and someone with relatively little Tripal experience could fix should be tagged with "Good beginner issue"
  - All feature requests should be tagged as an "enhancement"
  - If you are the first reviewer to confirm a PR works, tag it with "Reviewer #1 Approval"
- **Guidelines for Discussion:**
  - Issues that do not require discussion (PRs still require 2 reviews): minor bug fixes, changes to inline comments, addition of unit tests, minor code typos
  - Issues that require discussion: major changes, new features, and issue at the discretion of the PMC - Add the "discussion" tag to any issue requiring discussion - Discussion Tag is removed when adequate discussion has taken place (at the discretion of the person who added the tag) - Additionally, new features require that all members of the PMC have had a chance to contribute to the discussion and feel satisfied.
- **Please use the assignment feature to clarify who will be contributing the code to prevent duplication of effort.**
  - When assigning yourself, comment on what your timeline is. This allows others to jump in if they have time sooner.
  - If you would like to **take over a PR assigned to someone else** , comment asking for an update and offer your services.
  - If the author of the issue plans on contributing the fix themselves but is not a committer, they should indicate that in the issue. A committer will assign them the issue.
- **When you start working on an issue, you should create the branch and push to it regularly. If you are working on a fork, y**

  - Committers can work on a fork or directly. If the branch is on tripal/tripal, then other committers should contribute via PR unless otherwise agreed

- If an issue is identified as being relevant to another repository (ie a tripal module, not core), a new issue **should** be created, cross referenced, and the original issue should be closed encouraging discussion in the module.

4.1.6 Code of Conduct

- Be nice! If that's insufficient, Tripal community defers to https://www.contributor-covenant.org/

4.1.7 Testing/CI

Comprehensive guides to testing are available in the *Unit Tests for Tripal* section. Below are guiding principles.

- All tests pass.
- Tests don't modify db: use transactions and factories.
- Tests are organized properly: by submodule and function.
- Tests run quietly.

4.2 Unit Tests for Tripal

This guide is for developers looking to contribute code to the core Tripal project. It introduces the testing philosophy and guidelines for Tripal core. Tripal uses Tripal Test Suite, which brings bootstraps your Tripal site for PHPUnit. It also provides conveniences like name spacing, seeders, transactions, and data factories.

4.2.1 Tripal Test Suite

For a basic introduction of Tripal Testing, please see the Test Suite documentation.

4.2.1.1 Installation

After cloning the Tripal Github repo, you will need to install the developer dependencies required to run tests locally. To do this, you'll need to install Composer, and then execute `composer install` in your project root.

Remember to run `composer update` to update Tripal TestSuite before writing and running new tests. This is especially important when running pull requests that contribute unit tests. If tests are passing on the Travis environment but not on your machine, running composer update might resolve the problem.

4.2.2 Testing criteria

For facilitate accepting your pull requests, your code should include tests. The tests should meet the following guidelines:

- All tests pass
- Tests pass in all environments (Travis)
- Tests don't modify the database (use transactions or clean up after yourself)
- Tests are properly organized (see organization section below)

- Tests run quietly

4.2.3 Test organization

Tests should be placed in `tests/`. This root directory contains the following files:

- `bootstrap.php`: Test directory configuration. Don't modify this.
- `DatabasSeeders/`: Database seeders, for filling Chado with permanent test data.
- `DataFactory.php`: Data factories, for providing test-by-test Chado data.
- `example.env`: An example environment file. Configure this to match your development site and save as `.env`. Read more here: https://tripaltestsuite.readthedocs.io/en/latest/environment.html

Test files must end with `Test.php` to be recognized by PHPUnit. The tests themselves should be organized by submodule, and category.

4.2.3.1 Submodules

- tripal
- tripal_bulk_loader
- tripal_chado
- tripal_chado_views
- tripal_daemon
- tripal_ds
- tripal_ws

4.2.3.2 Categories

- API
- theme
- views
- drush
- fields
- entities
- admin
- loaders

So for example, tests for the file `tripal/api/tripal.jobs.api.inc` should go in `tests/tripal/api/TripalJobsAPITest.php`. tests that don't fit in any of these categories should be placed in `tests/[submodule]/`.

In order for tests to run locally, you'll need an environmental file `tests/.env` with the project root, base url, and locale. See `tests/example.env` for an example.

4.2.4 Writing tests

4.2.4.1 Tagging tests

You should tag your test with relevant groups. For example, our Tripal Chado API tests should be tagged with `@group api`. We don't need to tag it with `@group chado` because it is in the *testsuite* (the submodule folder) Chado.

If your test is related to a specific issue on the Tripal repo, thats great! You can use the `@ticket` tag to link it: ie, `@ticket 742` for issue number 742.

4.2.4.2 Defining the test class

The test class file should extend `StatonLab\TripalTestSuite\TripalTestCase` instead of `TestCase` to take advantage of the Tripal Test Suite tools. Tests should use a database transaction to ensure the database state is the same at the start and end of the test case. Your test class name should match the file.

```
use StatonLab\TripalTestSuite\DBTransaction;
use StatonLab\TripalTestSuite\TripalTestCase;

class TripalChadoOrganismAPITest extends TripalTestCase {
      use DBTransaction;
}
```

4.2.4.3 Defining individual tests

An ideal test operates *independently* of other tests: by default, unit tests run in random order. How, then, do we provide our test with relevant data? We use **Factories**, which you can read about on in the Tripal Test Suite documentation. In the below example, we create an organism with known information, and assert that we can retrieve it with the Chado API functions.

```
namespace Tests\tripal_chado\api;

use StatonLab\TripalTestSuite\DBTransaction;
use StatonLab\TripalTestSuite\TripalTestCase;

class TripalChadoOrganismAPITest extends TripalTestCase {

  use DBTransaction;

  /**
   * Test tripal_get_organism.
   *
   * @group api
   */
  public function test_tripal_get_organism() {

    $genus_string = 'a_genius_genus';
    $species_string = 'fake_species';

    $organism = factory('chado.organism')->create([
      'genus' => $genus_string,
      'species' => $species_string,
    ]);

```

(continues on next page)

(continued from previous page)

```
  $results = [];

  $results[] = tripal_get_organism(['organism_id' => $organism->organism_id]);
  $results[] = tripal_get_organism([
    'genus' => $genus_string,
    'species' => $species_string,
  ]);

  foreach ($results as $result) {
    $this->assertNotFalse($result);
    $this->assertNotNull($result);
    $properties = get_object_vars($result);
    $this->assertArrayHasKey('genus', $properties);
    $this->assertEquals($genus_string, $result->genus);
  }
 }

 public function test_tripal_get_organism_fails_gracefully() {
  $result = tripal_get_organism([
    'genus' => uniqid(),
    'species' => uniqid(),
  ]);

  $this->assertNull($result);
 }
}
```

Note: You typically will want at least one test per public method in your file or class. Tests should start with `test_`, otherwise it wont run by default in PHPUnit (you can also declare that it is a test in the method documentation using `@test`.

4.2.4.4 Testing quietly

Tests should run quietly. If the output goes to standard out, you can use `ob_start()` and `ob_end_clean()`.

```
ob_start();//dont display the job message
$bool = tripal_chado_publish_records($values);
ob_end_clean();
```

If the message comes from the Tripal error reporter, you must use `"TRIPAL_SUPPRESS_ERRORS=TRUE"` to suppress the Tripal error reporter message.

```
/**
 * Test chado_publish_records returns false given bad bundle.
 *
 * @group api
 */
public function test_tripal_chado_publish_records_false_with_bad_bundle() {
 putenv("TRIPAL_SUPPRESS_ERRORS=TRUE");//this will fail, so we suppress the tripal error reporter
 $bool = tripal_chado_publish_records(['bundle_name' => 'never_in_a_million_years']);
 $this->assertFalse($bool);
```

(continues on next page)

(continued from previous page)

```
  putenv("TRIPAL_SUPPRESS_ERRORS");//unset
}
```

4.3 Contributing to the Documentation

The Tripal documentation is written in **Restructured Text**, compiled with Sphinx, and built/hosted with ReadTheDocs. The `docs` directory, when compiled, is hosted at https://tripal.readthedocs.io/en/latest/.

For minor changes, you can simply Edit the file using the Github editor, which will allow you to make a Pull Request. Once approved, your changes will be reflected in the documentation automatically!

4.3.1 Guide

4.3.1.1 Install Sphinx

For minor changes, you don’t need to build the documentation! If you want to see how your changes will look on the built site, however, you will need Sphinx installed.

For more information, please see the Sphinx setup guide: http://www.sphinx-doc.org/en/master/usage/quickstart.html

4.3.1.2 Building your changes

For more extensive edits, or when contributing new guides, you should build the documentation locally. From the `docs` root (eg `/var/www/html/sites/all/modules/tripal/docs/`, execute `make html`. The built site will be in `docs/_build/html/index.html`.

4.3.1.3 Tripal conventions

Please follow these guidelines when updating our docs. Let us know if you have any questions or something isn’t clear.

Please place images in the same folder as the guide text file, following the convention [file_name].[n].[optional description].[extension]. For example, `configuring_page_display.3.rearrange.png` or `configuring_page_display.1.png` are both located in `docs/user_guide/` and are part of the `configuring_page_display.rst` guide.

We currently use the following syntax:

```
Title of File (using title case)
=================================

Introduction text.

Section Title
-------------

We use double backticks to indicate ``inline-code`` including file names, function␣
↪and method names, paths, etc.

Longer code-blocks should begin with the ``.. code-block:: [type]`` directive and␣
↪should be indented at least one
```

(continues on next page)

(continued from previous page)

```
level. There should also be a blank line before and after it as shown below.

.. code-block:: sql
  if ($needs_documentation) {
      use $these_guidelines;
      $contribute_docs = $appreciated;
  }

Section 1.1 Title
^^^^^^^^^^^^^^^^^

The use of appropriate sections makes reading documentation and later specific␣
↪details easier. Sub sections such
as this one will be hidden unless the main section is already selected.
```

4.4 Tripal Governance

We wish to maintain Tripal as an open source project and therefore want to empower Tripal adopters and developers as much as possible in the development of Tripal, while keeping the project coherent, focused, and useable to a wide range of adopters. As the Tripal community grows, it is prudent to set up a formal governance model. This document describes this model.

The Tripal project recognizes these roles:

- **End-Users**: They are users of Tripal-based websites (not developers of a site).
- **Adopters**: They have downloaded Tripal. Maybe they even have a site!
- **Extension Contributors**: they extend Tripal through modules, themes, views, data loaders, fields, and/or libraries.
- **Core Code Contributors**: contribute code to the Tripal "core", comments, discussion, questions, bug reports.
- **Core Code Committers**: write access to the Tripal "core" repository.
- **Tripal Project Management Committee (PMC)**: make *code relevant* decisions, (i.e. ensure code standards, robustness, and long-term design objectives are maintained).
- **Tripal Advisory Committee (TAC)**: Provide guidance to the PMC for *policy-level* and *future planning* recommendations.

4.4.1 Adopters

Any person who wishes to or has downloaded/set up a Tripal site. Everyone in this group is invited to the monthly user meetings and is encouraged to ask questions and make suggestions.

4.4.2 Extension Contributors

These are developers who are extending Tripal. Extension contributors are encouraged to make their extensions available on the Tripal GitHub organization and list them on the Tripal Documentation. Extension contributors are also encouraged to use the Tripal Module Rating System as a guideline for developing high quality modules, which are easier to adopt by the greater Tripal community.

4.4.3 Core Code Contributors

Core Code Contributors are active members of the Tripal community who make suggestions to improve and/or contribute code to Tripal Core. Core Code Contributors are encouraged to submit pull requests to the Tripal core and attend monthly user calls. For more information, see the Guidelines for Contributing to Tripal core.

Responsibilities include:

- Monitor Tripal core issue queue.
- Submit pull requests.

4.4.4 Committers

These are dedicated Tripal developers who are trusted to commit pull requests directly to the Tripal core repository. They are encouraged to be active in the Tripal community and routinely review pull requests. Developers are added to to committers group by unanimous agreement from the PMC.

Responsibilities include:

- Monitor Tripal core issue queue.
- Review and merge pull requests.

See the guidelines for contributors for more details.

4.4.5 The Tripal Project Management Committee (PMC)

This group consists of experienced Tripal developers.

Responsibilities include:

- Ensure good practices, for example, submitting errors, questions, and requests via GitHub.
- Monitor issue queue (though this responsibility isn't limited to the PMC).
- Resolve questions and differences of opinions among Contributors.
- **Work with the TAC to make decisions about significant new features. Examples:**
  - a new core module,
  - designing a module-specific REST API,
  - new technologies or libraries used by the core code.
- Avoid feature bloat; judge what contributions benefit many versus those that are specific to the needs of the contributor.
- Final approval of submitting guidelines (see guidelines for contribution).
- Set coding standards.
- Ensure Tripal remains generic and useful to a wide range of groups.

The PMC will strive to obtain consensus, and all members ensure that the Tripal community and the TAC are informed on decisions. Any individual member can call a meeting. The term will be two years with the possibility of extension. At least one member will serve on the TAC; this person will be elected by vote within the PMC.

4.4.5.1 Communication and Meetings

The PMC will meet as necessary. It is expected that frequent decisions will need to be made, some through GitHub issue comments, Slack, e-mail, or conference calls.

4.4.6 Tripal Advisory Committee (TAC)

The Tripal Advisory Committee (TAC) provides leadership, guidance, future planning, advocacy and coordination for Tripal. The TAC acts in an advisory capacity only by determining sets of recommendations for Tripal. All recommendations will be provided to the PMC. Topics include recommended technologies, overall approach, software architecture, development priorities, timelines, funding strategies, best practices, support for a fair and focused open source development strategy.

At least one member of the PMC must be on the TAC to ensure that the reality of what is and is not feasible for the developers is not lost. Additionally, close communication between the TAC and PMC is critical, as is transparency of the TAC discussions to the entire Tripal community. All members of the PMC are welcome at TAC meetings.

4.4.6.1 Membership

The TAC should include "internal" and "external" members. Internal members are individuals who manage Tripal websites or lead teams engaged in active development, possibly with funding to do so. External members may be outside the Tripal community altogether, and may include government, non-profit, or industry representatives who are stakeholders for one or more Tripal databases (but not active managers of a Tripal site) and/or specialists in such disciplines as cyberinfrastructure, bioinformatics, genomics, breeding.

- Terms are for two years.
- Two year memberships can be renewed for individuals who wish to stay on to complete a particular objective.
- **Membership is capped at 15.**
  - **Initial Setup:**
    - Start small and move larger as needed.
    - Set minimum sizes for number of internal and external members.
    - Committee should be organized before inviting external members.
    - Stagger ends of terms to ensure continuity.
- The minimum number of internal members is 3.
- The number of internal members should not be less than 1/2.
- The target number of external members is 5.
- If the TAC decides to replace a leaving member, the current members will develop a list of possible candidates. The chair will contact each in turn until the membership slot is filled.
- Members will be asked to serve by the current TAC.

Responsibilities include:

- Serving a minimum two year term, beginning with the yearly board meeting (see below) in conjunction with the January Plant and Animal Genome Conference in San Diego.
- Respond to issues in a timely manner when contacted directly. Members are strongly encouraged to become part of the TAC GitHub group, and if they wish to comment or discuss agenda items directly with the community, to do so in the GitHub issue queue (instead of the email list serve).

- Attend the annual January meeting at PAG and at least three of the quarterly meetings.
- Review agenda and supporting materials prior to meetings.
- Stay informed about Tripal, its member databases, developers, and users.

In addition, internal members are responsible for:

- Actively communicating with the Tripal community, both to collect ideas and concerns and to inform the community about TAC plans for Tripal.
- Engaging in the Tripal Core GitHub Issue queue on "discussion" issues.

4.4.6.2 TAC Chair

The board will be led by a chair to be elected by TAC members at the January meeting annually (see below). One or more vice-c

- Organize, announce and lead TAC meetings.
- Write the meeting agenda and post to Tripal.info.
- Provide supporting materials for review at least 1 week before TAC meetings.
- Ensure that the agenda items that would benefit from review by the community are posted to the GitHub Tripal core issue queue. Ensure that any GitHub issue discussions are linked on the agenda and available for review by the TAC.
- Ensure meeting notes are taken by someone present at the meeting and posted to Tripal.info.
- Call for votes on TAC recommendations when community voting is required.
- Call additional meetings if needed.
- Facilitate communication between the TAC and PMC.
- Filling vacant slots on the TAC.
- The chair has voting privileges.

4.4.6.3 TAC Meeting Agenda Items

Strongly encouraged to be posted to the GitHub Tripal core issue queue as well as to tripal.info, to inform and solicit community comment. TAC meeting agendas will include issues tagged "TAC Next Meeting" on the GitHub Tripal core issue queue. Other agenda items may be added by the TAC chair or members, or by the PMC. These issues will be closed after the meeting.

4.4.6.4 Communication and Meetings

The primary TAC meeting will be held in January of each year, at the Tripal codefest. In-person attendance is strongly encouraged, but a teleconference option will be provided. Each issue on the agenda will be discussed, and if needed, the chair will call for a vote to determine the final recommendation of the TAC. Votes carry based on simple majority. All discussion, votes and objections will be recorded in meeting notes, which will be posted on Tripal.info.

Additional teleconference TAC meetings will be held once per quarter (April, July, October). These could be held in place of the monthly Tripal User's Meeting to avoid meeting overload.

TAC meetings outside the above schedule may be called by the TAC chair. These will only be called in urgent situations. In less urgent situations, the TAC chair or the TPMC can contact the internal members of the TAC and request a meeting or solicit comments via email, GitHub issue, or Slack.

At any time the TPMC may communicate with members of the TAC with expertise in specific areas if needed to assist in decision making.

4.4.7 Changes to this Document

These guidelines define the structure for official Tripal Management and Governance. If you have comments or questions, please post them as a Github issue or ask them at the user's meeting. Changes to this document will be made after adequate discussion has occurred and the project management committee has voted in favor of the change.

4.5 Funding Proposal Development

Tripal fully supports all community members who want to submit grant proposals for Tripal extensions. Extension development is outside the purview of the Tripal governance structure, although the Tripal Project Management Committee (TPMC) and Tripal Steering Committee (TSC) are available to help with grant submissions, through alignment with Tripal core long term goals, letters of support, etc.

PIs planning to write a grant that would fund any development related to Tripal core are strongly encouraged to engage the TPMC and TSC early in the planning process to ensure the grant aligns with the long term community goals. Any core development plans are still required to go through the Tripal governance structure. Specifically, only development approved by the TPMC and TSC will be implemented or accepted in core. PIs, especially if funded, are strongly encouraged to engage with the Tripal community and to request membership in the TPMC and/or TSC.