trix Documentation

Release 1.0a

trix

Oct 05, 2018

Contents

CHAPTER 1

User documentation

1.1 Student docs

1.1.1 Getting started

As a student you may use the Trix system in two modes.

1. Anonymous
2. Authenticated

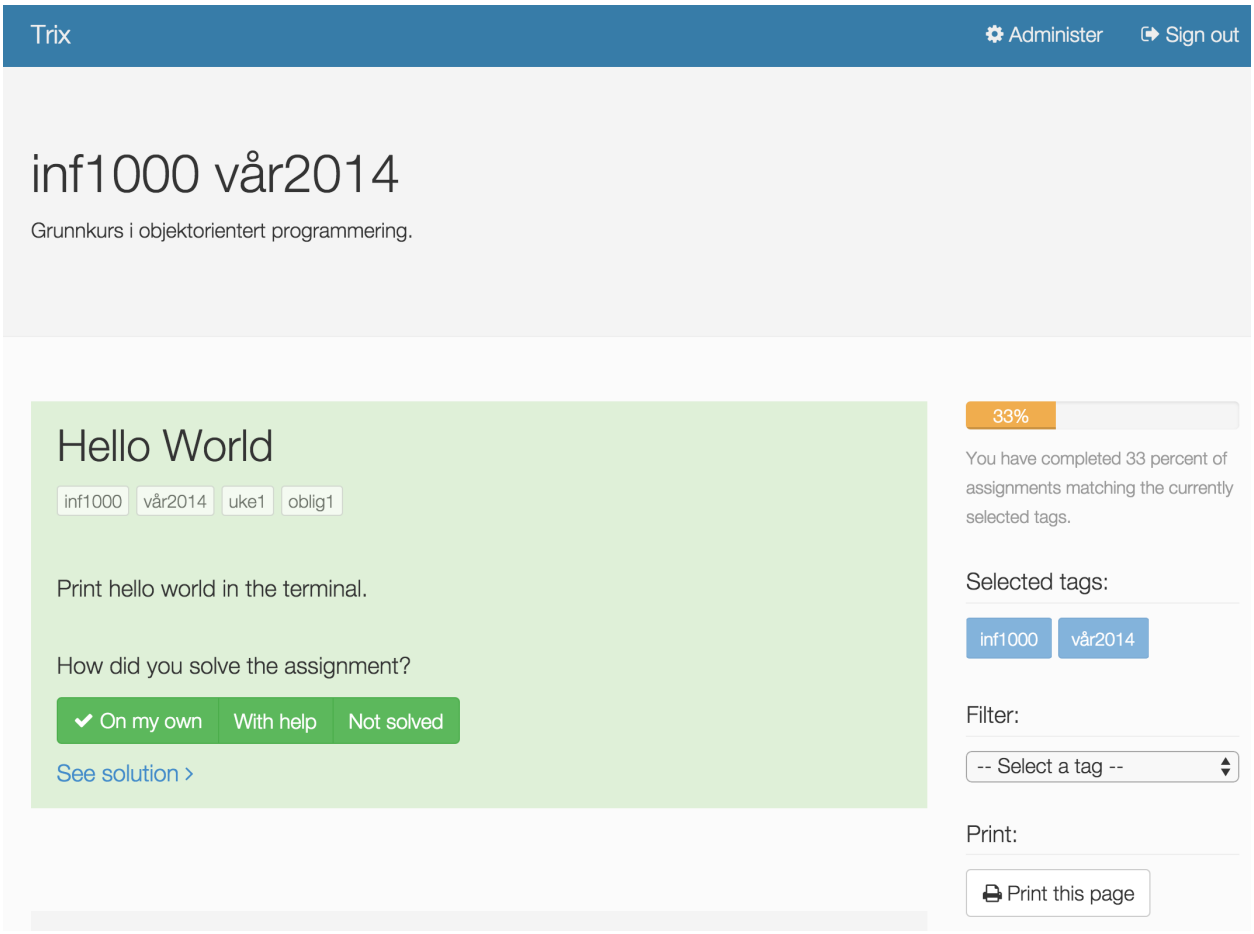
As an anonymous user you will be able to see the assignments and view their solutions. You will not be able to track your progress.

If you log in you will be able to track your progress along with the status on how an assignment was solved. An assigments may be marked as one of the following.

Not solved This is the default status, meaning that you have not marked the assignment.

With help You may mark an assigments as *With help* when you have solved it with the help of others or with extra hint and tips.

On my own An assignments should be marked *On my own* when you completed it all by yourself. This is the highest degree of verification that the subject is understandable for you.



The progress bar in the right side of the view tell in percentage how many of the selected assignments that have been solved on your own.

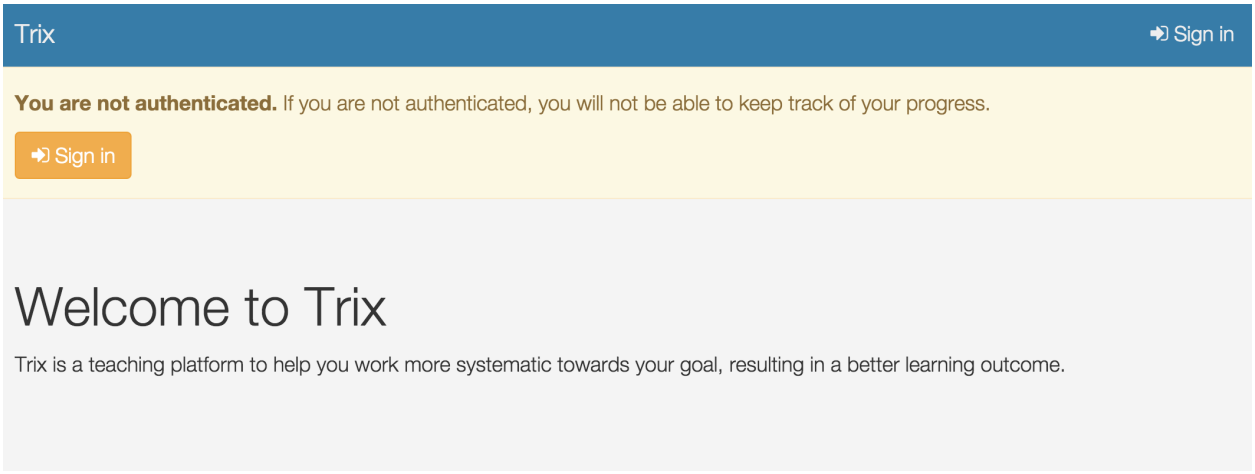
The selected tags lists the active tags that define the selection of assigments.

You may also filter the selection by selecting tags that will narrow the number of assignments listed to the set that have all the selected tags. As an example a course is already defined by the authors with their *course* and *period* tags. Below this the authors may have marked a number of assignments with the tags *week1* and some other assignments *week2*. You may then filter with these tags to focus on the correct assignments.

1.2 Administrator docs: Edit assignments, permalinks, . . .

1.2.1 Getting started

To be able to perform the administrator actions you need to be logged into the system with an account with administrator rights. Click the button on the frontpage and follow the instructions.



1.2.2 Add/edit assignments

From the trix frontpage, select **Administer** in the page header.

If you have access to more than one course, the next page will ask you to select which one you want to administer. If you have access to only one course, you will be redirected to the administrator pages for that course.

Select **Assignments** from the left hand menu.

Create an assignment

Click the **Create** button at the top of the page. Select a title, assignment text and optionally solution. You can read more about tagging below, and you can leave that field blank for now.

Edit an assignment

From the assignment overview, browse or search for the assignment you want to edit. Click the **Edit** button in the first column below the assignment title.

Tagging

Assignments can be tagged. How you structure your tags is something you and other course administrators need to agree on. Try to use tags that make it easy for users to filter out a useful subset of assignments.

If you want to link to a set of tags and provide a title and description for the page you link to, use the **Permalink** app in the left side menu.

1.2.3 Trix flavoured Markdown

When you write feedback to your students, you use the Markdown text formatting language.

With Markdown, you to write using an easy-to-read, easy-to-write plain text format, and let someone else (Trix) worry about how the results will look. This makes it possible to write feedback text that Trix can optimize for anything from smartphones to large desktop displays.

Basics

Paragraphs

Paragraphs are just one or more lines of consecutive text followed by one or more blank lines:

```
Maecenas faucibus mollis interdum. Vestibulum id ligula porta felis euismod
semper. Vestibulum id ligula porta felis euismod semper. Aenean lacinia
bibendum nulla sed consectetur.

Donec id elit non mi porta gravida at eget metus. Vestibulum id ligula
porta felis euismod semper. Praesent commodo cursus magna, vel scelerisque
nisl consectetur et.
```

Headings

```
# Largest heading
## Second largest heading
...
##### Very small heading
```

Text styles

```
*Italic text*
**Bold text**
```

Links

```
Check out the [https://github.com/devilry/trix2/](Trix website).
```

Lists

Unordered lists (bullet lists):

```
* This
* is
* a
* test
```

Ordered lists (numbered lists):

```
1. Item one
2. Item two
3. Item three
```

Blockquotes

```
As stated on the first page of the 101 guide:

> You have to learn to walk before you can learn how to run
```

Advanced

Escape Markdown characters

If you want to use a special Markdown character in your document (such as displaying literal asterisks), you can escape the character with a backslash. Markdown will ignore the character directly after a backslash. Example:

```
This is how the \_ (underscore) and \* asterisks characters look.
```

Code blocks

You can easily show syntax highlighted code blocks:

````
Java code:
``` java
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello world");
    }
}
```

Python code:
``` python
if __name__ == "__main__":
    print "Hello world"
```

C code:
``` c
#include<stdio.h>
int main() {
    printf("Hello World");
    return 0;
}
```

C++ code:
``` c++
#include <iostream>
int main() {
    std::cout << "Hello World!";
    return 0;
}
```

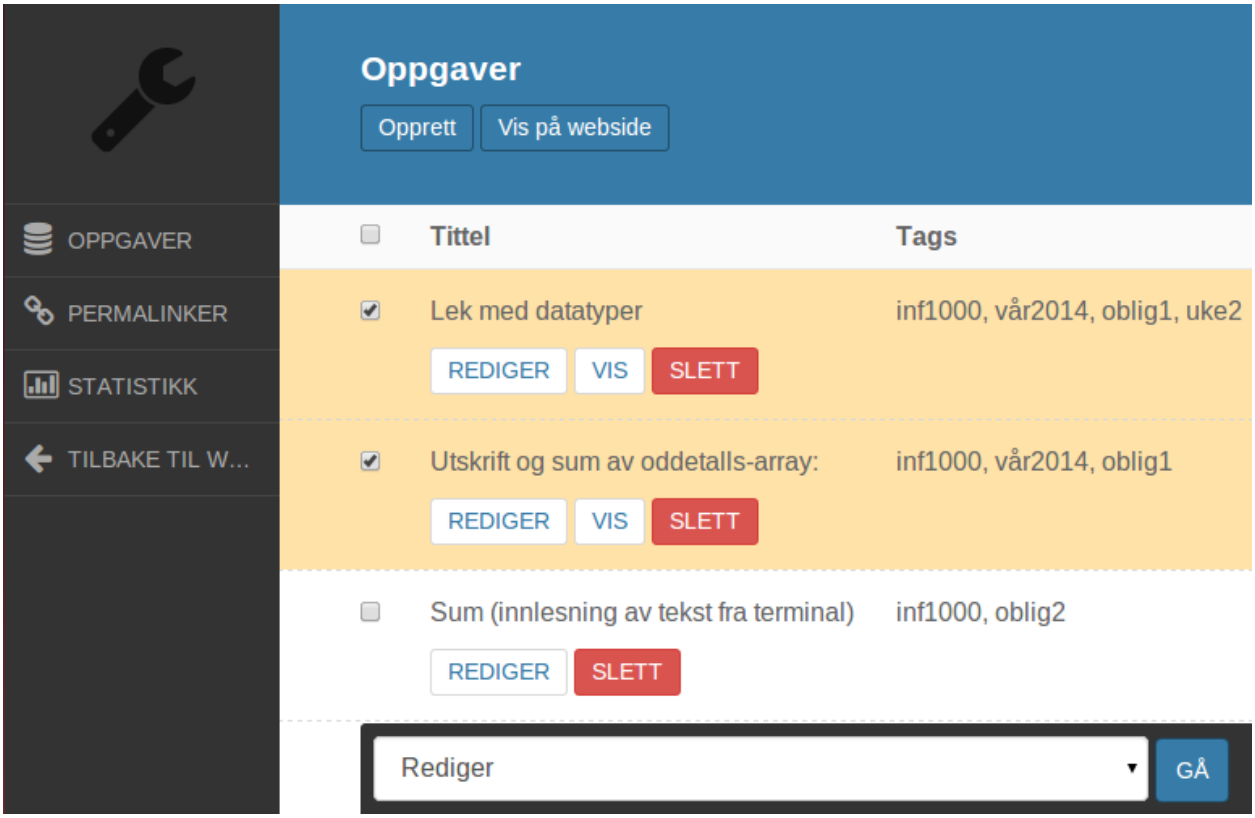
HTML example:
````

(continues on next page)

(continued from previous page)

```
``` html
<html>
    <body>
        <h1>Hello world</h1>
    </body>
</html>
```

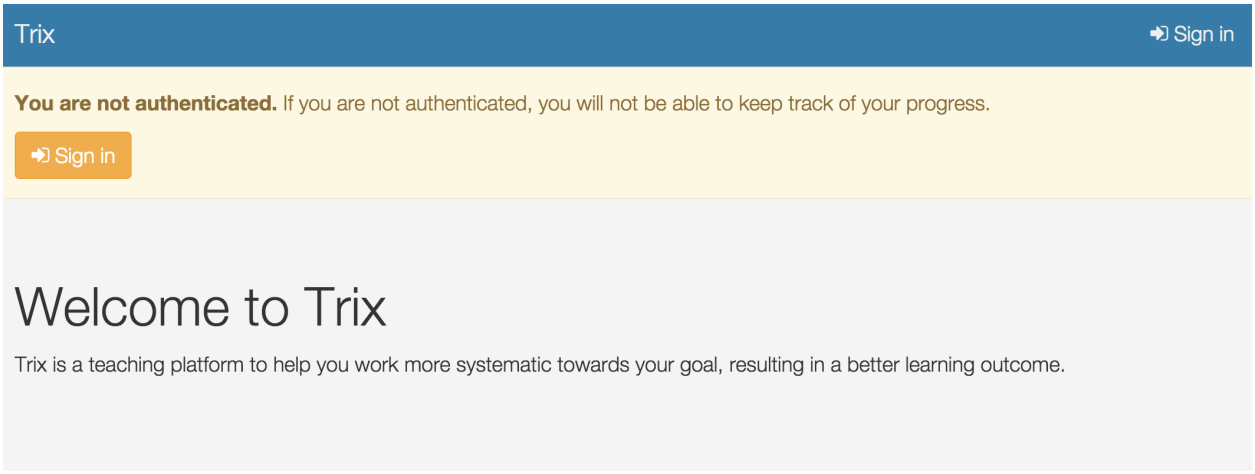
CSS example:
``` css
body {
    background-color: pink;
    color: green;
    font-size: 80px;
}
```

Any code:
```
for x in 1 through 3
    show x
```
```

Trix supports all languages supported by Pygments.

1.2.4 Editing multiple assignments

To ease handling of editing multiple assignments at the same time Trix provides a YAML based format which fully supports the need for bulk managment

Multiple editing will opt in as an action when you check several assignments in the assignment view as shown in the image below.



Format

Read up on YAML: yaml_wikipedia

Each assignment are associated with the following fields:

id : Internal unique identifier.

title : The title of the assignment.

tags : Tags to classify the assignments.

text : The assignment text to describe the problem to solve.

solution : The solution to the problem of the assignment.

The fields are rendered within the YAML format as listed below.

```
id: 4
title: '<title>'
tags: [tag1, tag2, tag3, ...]
text: |-
    <assignment text>
solution: |-
    ``` java
        class Solution {

        }
    ```
```

Example

```
id: 4
title: 'Utskrift og sum av oddetalls-array:'
tags: [inf1000, vår2014, oblig1]
text: |-
  Skriv et program som inneholder en heltalls-array med følgende elementer: 1, 3, 5,␣
↪7, 9, 11, 13, 15, 17, 19.  Programmet skal inneholde en løkke som skriver ut␣
↪indeksen og verdien for alle elementene i arrayen.
solution: |-
  ``` java
  class Oddetall {
    public static void main(String[] args) {
      int[] oddetall = { 1, 3, 5, 7, 9, 11, 13, 15, 17, 19 };
      for (int i = 0; i < oddetall.length; i++) {
        System.out.println("oddetall[" + i + "] = " + oddetall[i]);
      }
    }
  }
  ```
---
id: 1
title: Hello World
tags: [inf1000, vår2014, uke1, oblig1]
text: |-
  Print hello world in the terminal.
solution: |-
  ``` java
  public class HelloWorld {
      public static void main(String [] args) {
          System.out.println("Hello World!");
      }
  }
  ```
```

1.3 Superuser docs: Add/edit courses and users

1.3.1 Getting started

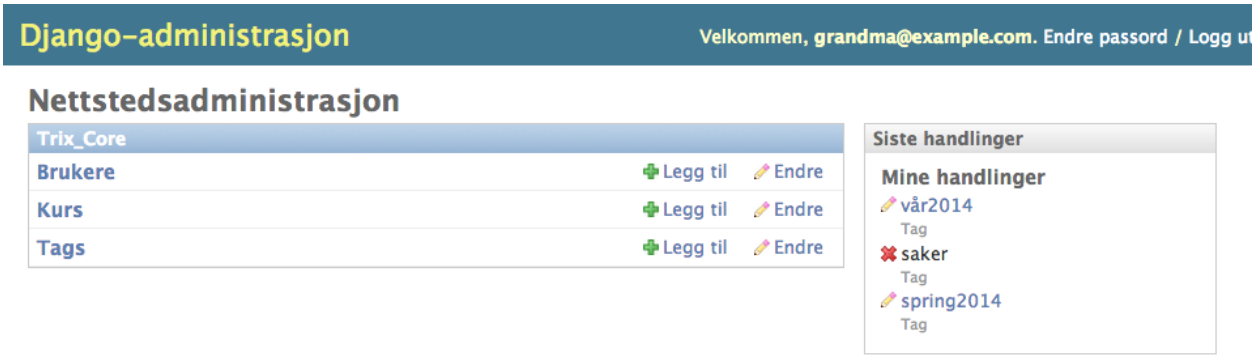
To be able to perform the superuser actions you need to be logged into the system with an account with superuser rights. Click the button on the frontpage and follow the instructions.



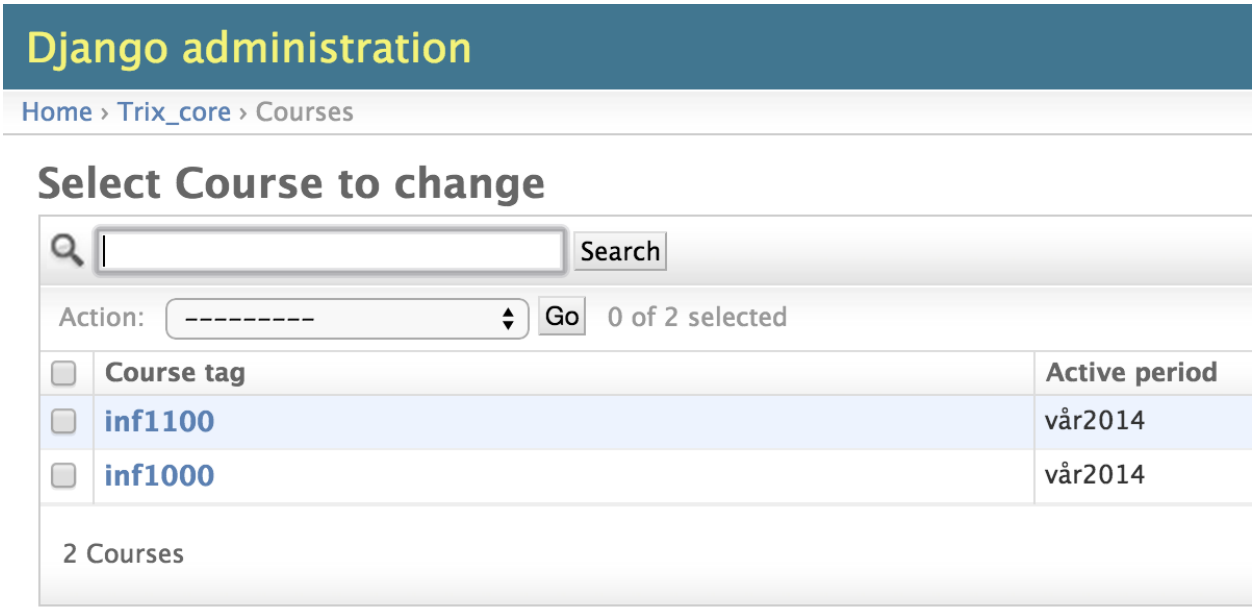
1.3.2 Administer a course

After you haved logged in to the Trix, go to `<domain-where-you-host-trix>/admin/`. So if trix is hosted at `trix.ifi.uio.no`, type in `trix.ifi.uio.no/admin/` in the address bar of your browser.

The page you will see lists every entity in the system that you can manage and edit. Be careful since all changes are stored directly in the database.



In the view above, click **Courses**.



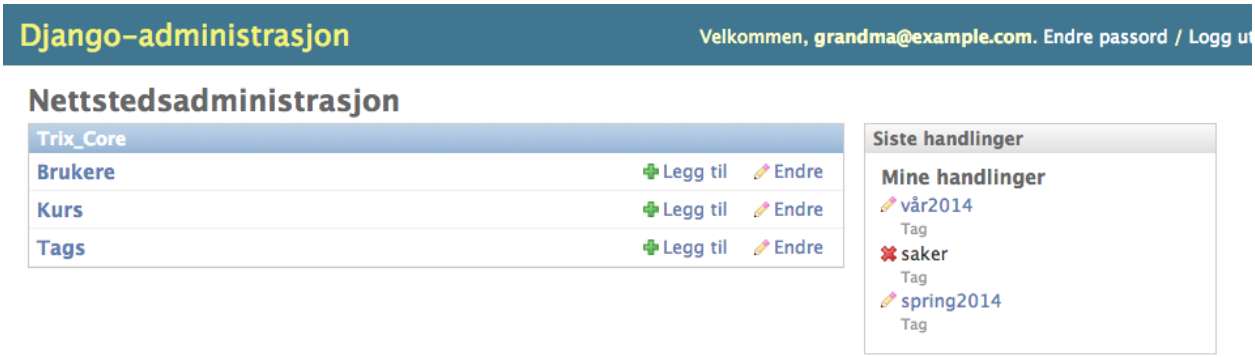
In the view abowe choose the **course** you want to administer and click on the link.

1.3.3 Add a new timeperiod/semester

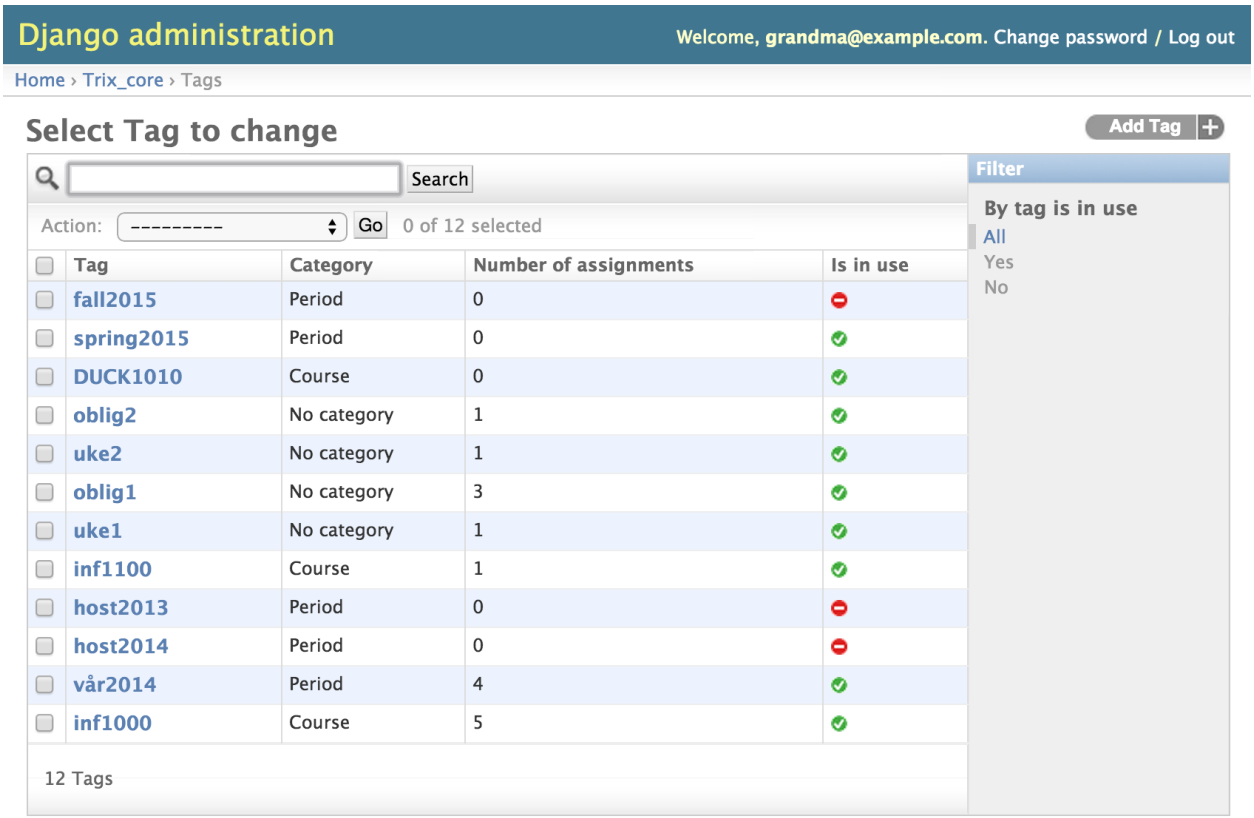
The active timeperiod/semester within a course is defined as a tag with the *period* category. Administrators can access all assignments tagged with a course where they have administrator rights, but students can only access assignments tagged with the active period tag configured for a course.

Note: This is very powerful, since publishing/unpublishing assignments only require an administrator to remove/add the active period tag from an assignment.

To add a new period tag, go to `/admin/`:



Click the **Tags** item in the list:



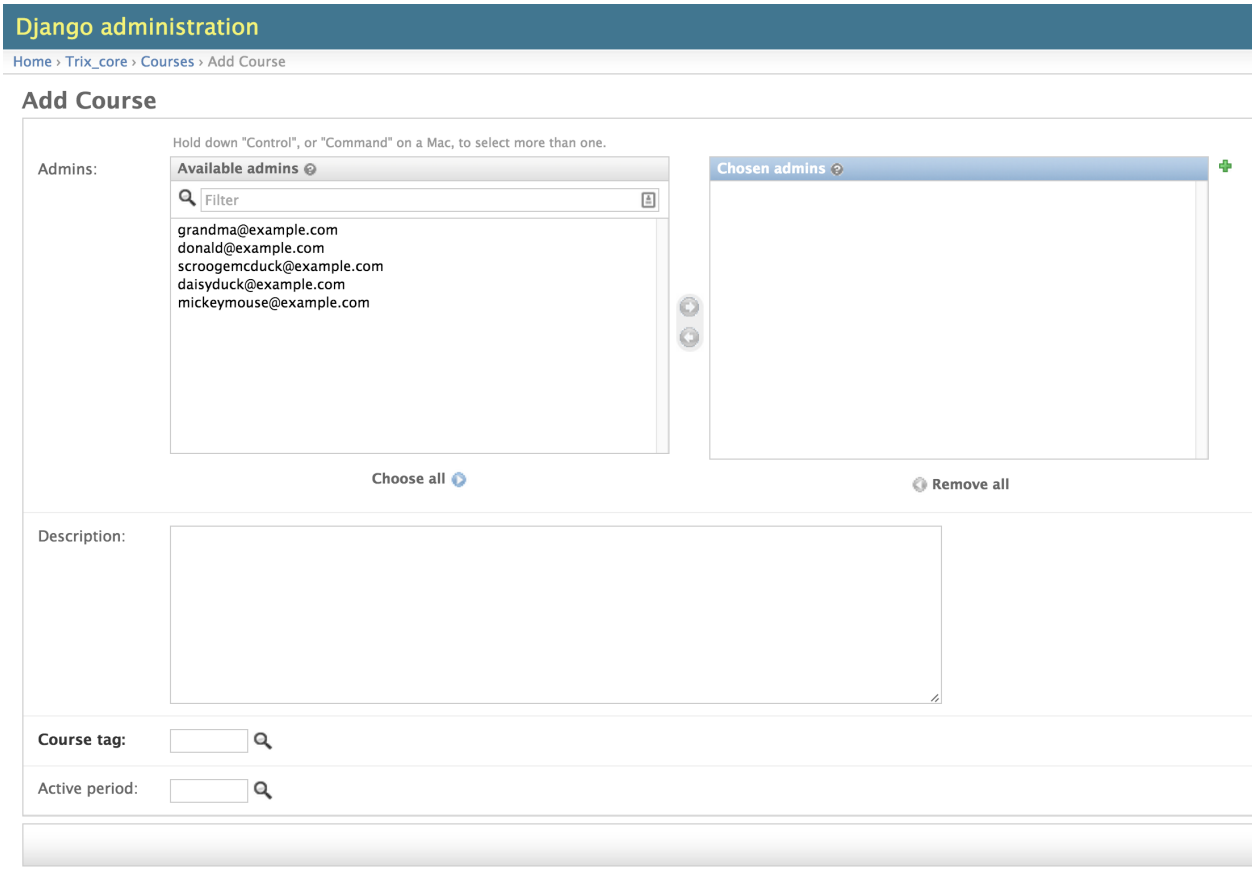
Click **Add Tag** in the upper right corner and then type the name of the tag along with the category *Period*.

The newly added period tag can then be defined as the active period on a course.

1.3.4 Add a new course

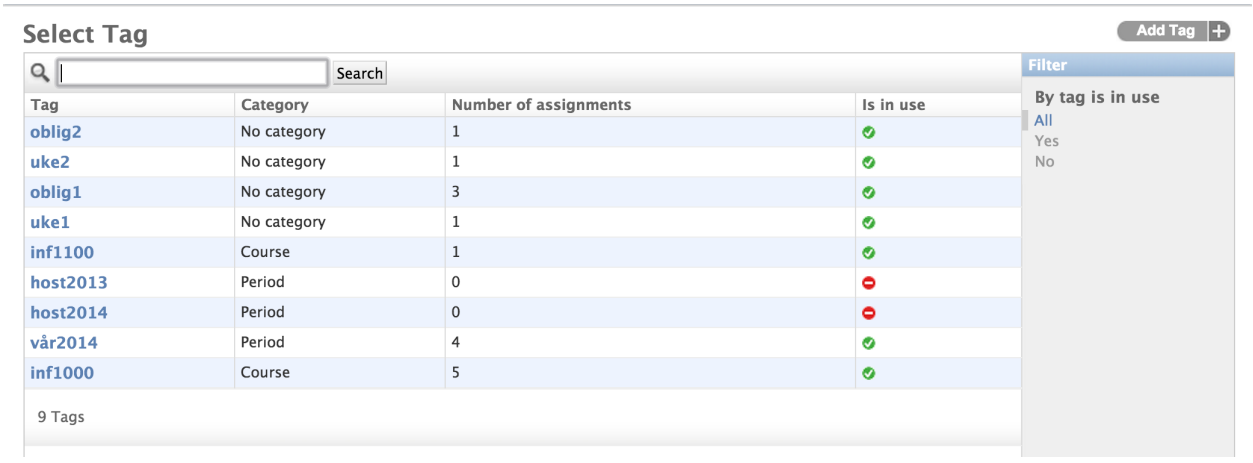
To add a new course you need to navigate to the course list view. This is described in the *Administer a course*. guide. Proceed with the steps outlined below.

In the upper right corner of the course list view click **Add Course**



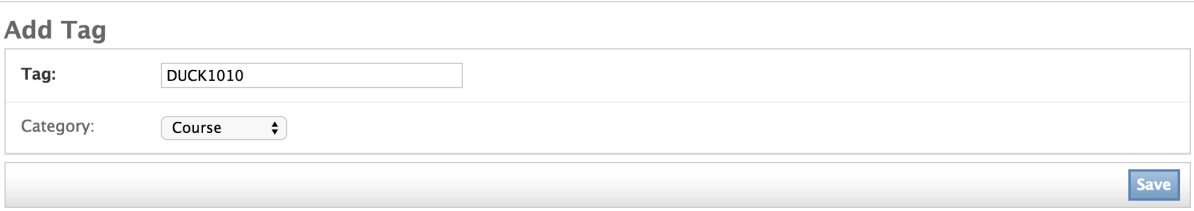
First you may select the administrators of the course and type a description that briefly explaines the course. This text is shown to the students when they access the course page.

Then set the correct course tag that are defining the course. Click the magnifier icon next to the **Course tag:** field



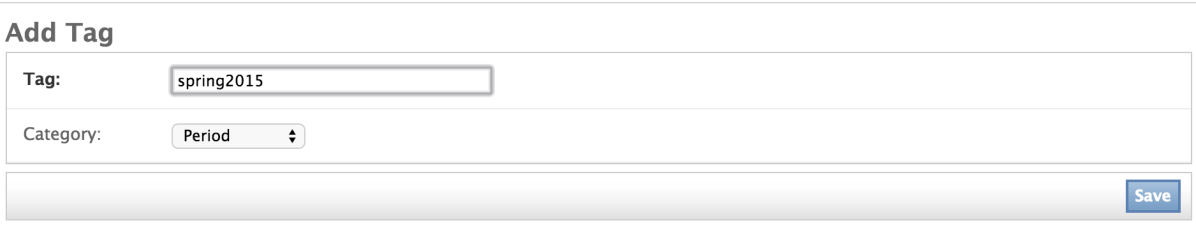
If you already have created a course tag choose and click the right one from the list. Make sure that the tag you are using have the *Category* Course.

To create a new course tag click the **Add Tag** in the upper right corner.



Type the name and set the category to **Course** and click save.

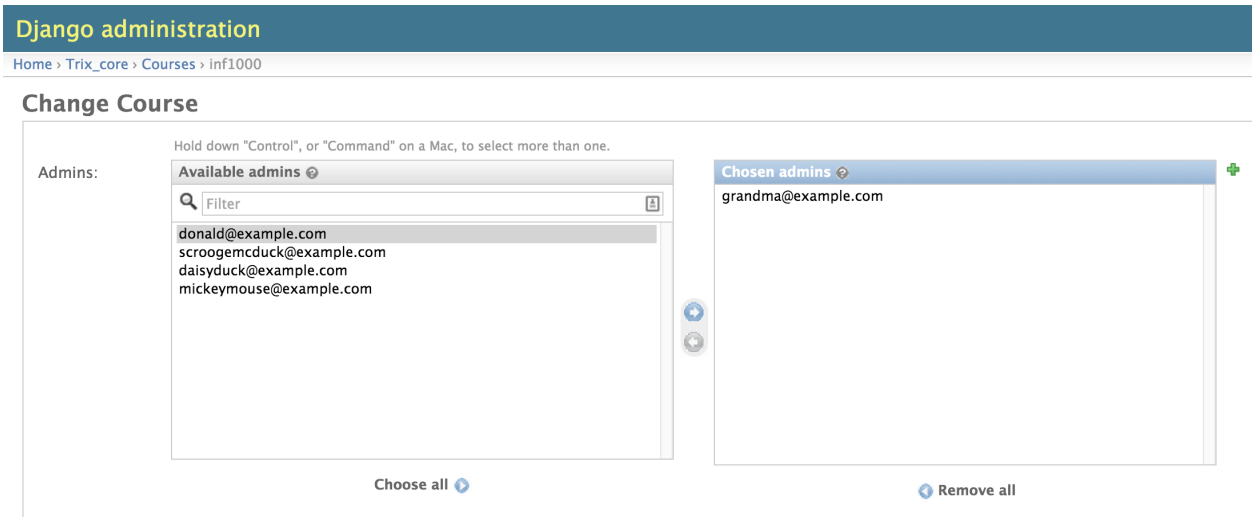
Finally set the period tag by clicking the magnifier icon next to the **Active Period:** field. Choose an existing tag if it's already defined otherwise click **Add Tag**[1] and type the info.



Click save.

1.3.5 Add administrators to a course

The add administrators on a course you need to navigate to the correct course. This is described in the *Administer a course*. guide. Proceed with the steps outlined below.



The panel on the left list users eligible for admin access. Just mark a username by clicking on it[1] and click the arrow button to provide admin access.

The panel on the right side lists the current administrators. To deprive users their administrators rights just mark them and click the arrow button pointing in the opposite direction.

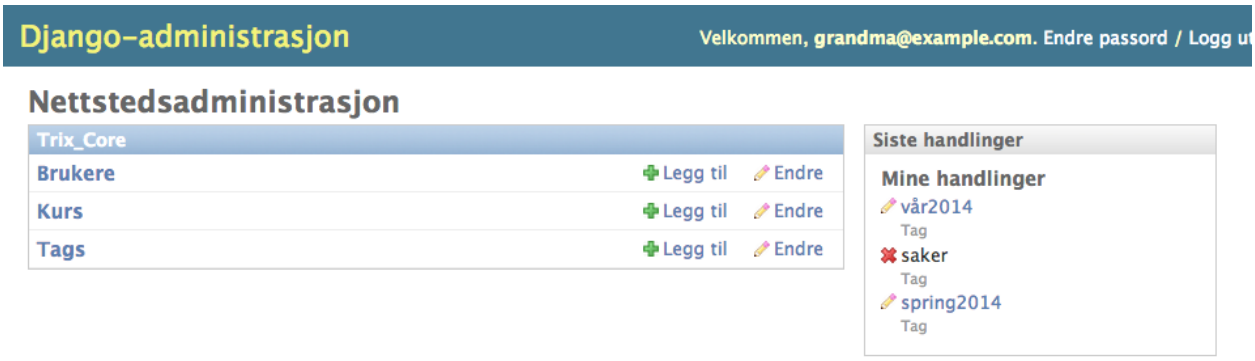
When your administrator list is updated click the save button in the bottom right position of your browser window.

[1] A period is a tag with a special category that act as a period. The active period is just one of the defined period tags making it possible to easily switch between periods.

[1] Tips: To mark several users hold in the Control(On Windows) / CMD(On Mac) button on your keyboard and click on the users.

1.3.6 Add or remove a superuser

To add or remove a superuser, go to `/admin/`:



Select **Users** from the displayed list. Select the user you want to turn into a superuser. Check or uncheck the **Is superuser** checkbox to turn a user into a superuser, or to remove their superuser rights. Click **Save** to complete the changes.

CHAPTER 2

System administrator documentation

2.1 System administrator docs: Install, update, ...

2.1.1 Setup Trix for production

Install dependencies

1. Python 2.7.X. Check your current version by running `python --version`.
2. PIP
3. VirtualEnv
4. PostgreSQL server — not needed if you just want to build the docs.

Create or select a user that you want to run the Trix server

We recommend using a normal user with no admin/root/sudo privileges to run Trix. You should perform the rest of the steps in this guide as this new user.

Make a directory where you will install trix

This directory MUST NOT be served by a http server like apache. It should be a well protected local directory only acceccible to the user running Trix. Example:

```
$ mkdir ~/trixdeploy
```

We will refer to your trix deploy directory as `~/trixdeploy` for the rest of this guide.

Install Trix

```
$ cd ~/trixdeploy
$ virtualenv venv
$ venv/bin/pip install psycopg2 dj-static trix
```

Create a Django management script

Copy this script into `~/trixdeploy/manage.py`:

```
import os
import sys

if __name__ == "__main__":
    os.environ["DJANGO_SETTINGS_MODULE"] = "trix_settings"
    from django.core.management import execute_from_command_line
    execute_from_command_line(sys.argv)
```

Configure

Trix is configured through a `trix_settings.py` file. Start by copying the following into `~/trixdeploy/trix_settings.py`:

```
from trix.project.production.settings import *
import dj_database_url

# Make this 50 chars and RANDOM - do not share it with anyone
SECRET_KEY = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'

# Database config
DATABASE_URL = 'sqlite:///trixdb.sqlite'
DATABASES = {'default': dj_database_url.config(default=DATABASE_URL)}

# Set this to False to turn of debug mode in production
DEBUG = False
```

LDAP

To enable LDAP authenication, `trix_settings.py` would need to include an authenication backend with LDAP support, the URI for the LDAP server and its DN template, and possiblely some customization to adjust for how the usernames are stored in Trix's database.

As an example, the settings for UiO would need to adjust for the LDAP username not being a full email adresse by overwriting `ldap_to_django_username()` and `django_to_ldap_username()` functions of `django_auth_ldap.backend.LDAPBackend`.

Add the following to you `trix_settings.py` file (but adjust the DN template):

```
AUTHENTICATION_BACKENDS = [
    'trix_uio_ldap_auth.TrixUioLDAPBackend',
]
AUTH_LDAP_SERVER_URI = 'ldaps://ldap.uio.no'
AUTH_LDAP_USER_DN_TEMPLATE = 'uid=hei,cn=people,dc=uio,dc=no'
```

Create a `trix_uio_ldap_auth.py` as follows (but adjust the email suffix):

```
from django_auth_ldap.backend import LDAPBackend

class TrixUioLDAPBackend(LDAPBackend):
    def ldap_to_django_username(self, username):
        return u'{}@example.com'.format(username)

    def django_to_ldap_username(self, username):
        return username.split('@')[0]
```

Consent template

Customising the consent template is highly recommended and can be done by following these steps:

1. Create a directory for custom templates with a subfolder named `trix_student`
2. Create a django html file called `consent_form.django.html` and make it look like this:

```
{% extends "trix_student/consent_form_base.django.html" %}

{% block consent_title %}<h1>Consent title here</h1>{% endblock %}

{% block consent_text %}Lorem Ipsum{% endblock %}

{# If you want to override the buttons you can add this #}
{# {% block consent_buttons %}BUTTONS{% endblock %} #}
```

3. In `trix_settings.py` do:

```
from trix.utils.template import add_custom_template

add_custom_template('custom_template_directory/goes/here')
```

If you wish to disable the consent dialog completely for some reason, add `DISABLE_CONSENT = True` to your `trix_settings.py` file.

Make sure it works

Just to make sure everything works, run:

```
$ cd ~/trixdeploy/
$ venv/bin/python manage.py syncdb --noinput
```

This should create a file named `~/trixdeploy/trixdb.sqlite`. You can remove that file now - it was just for testing.

Collect static files

Run the following command to collect all static files (CSS, javascript, ...) for Trix:

```
$ venv/bin/python manage.py collectstatic
```

The files are written to the `staticfiles` sub-directory (~/trixdeploy/staticfiles).

Configure a database

Configure a Postgres database by editing the `DATABASE_URL` setting in your `trix_settings.py` script. The format is:

```
DATABASE_URL = "postgres://USER:PASSWORD@HOST:PORT/NAME"
```

Configure a SECRET_KEY

Configure the SECRET_KEY (used for cryptographic signing) by editing the `SECRET_KEY` setting in your `trix_settings.py` script. Make it a 50 characters long random string.

Disable debug mode

Before running Trix in production, you **must** set `DEBUG=False` in `trix_settings.py`.

> **Warning:** If you do not disable DEBUG mode in production, you database credentials and SECRET_KEY will be shown to any visitor when they encounter an error.

Run the production server

```
$ DJANGO_SETTINGS_MODULE=trix_settings venv/bin/gunicorn trix.project.production.wsgi␣
↪-b 0.0.0.0:8000 --workers=12 --preload
```

You can adjust the number of worker threads in the `--workers` argument, and the port number in the `-b` argument. You can run this on port 80, but if you want to have SSL support, you will need to use a HTTP proxy server like Apache og Nginx.

2.1.2 Setup Trix on Heroku

Quick setup

Create an Heroku account

Go to https://www.heroku.com and create your Heroku account. Make sure you set it up completely (including setting a SSH key).

Clone the Trix repo

First you need to checkout the Trix repo:

```
$ git clone https://github.com/devilry/trix2.git
$ cd trix2/
```

Create the Heroku instance

Next, create the heroku instance. We have configured everything for Heroku, so all you need is:

```
$ heroku create
$ heroku config:set DJANGOENV=production
$ heroku config:set DJANGO_SETTINGS_MODULE=trix.project.settingsproxy
$ git push heroku master
$ heroku ps:scale web=1
```

Note: You can create the Heroku instance in Europe with:

```
$ heroku create --region eu
```

Note: The Heroku config for Trix is basically the same as the one in https://devcenter.heroku.com/articles/getting-started-with-django.

Create a demo database

To create the Trix demo database, run:

```
$ heroku run bash
>$ python manage.py syncdb --noinput
>$ python manage.py migrate --noinput
>$ python manage.py runscript trix.project.develop.dumps.dev.data
>$ exit
```

Drop and recreate the database

If you need to drop and recreate the database, run:

```
$ heroku pg:info
```

You will find database name on the first line (all uppercase, something like HEROKU_POSTGRESQL_AQUA_URL). Then you can run:

```
$ heroku pg:reset <database-name>
```

Lastly, repeat *herokucreatedemodb*.

CHAPTER 3

Developer documentation

3.1 Developer docs

3.1.1 Getting started with the Django app and/or the documentation

Install the requirements

Install the following:

1. Python2.7
2. PIP
3. VirtualEnv
4. virtualenvwrapper
5. gettext for Django translations

#. nodejs and npm for our clientside stuff .. #. libjpeg, liblcms1, libfreetype6 and zlib for the required format support in Pillow

Install the system packages on OSX with Homebrew

```
$ brew install npm nodejs
```

You will also have to add gettext to your path if you want to be able to update translation strings. You can either run `brew link gettext --force`, or add `/usr/local/Cellar/gettext/SOMETHING/bin/` to your path.

Install the system packages on Ubuntu

```
$ sudo apt-get install npm nodejs
```

Install in a virtualenv

Create a virtualenv (an isolated Python environment):

```
$ mkvirtualenv trix
```

Note: Whenever you start a new shell where you need to use the virtualenv we created with `mkvirtualenv` above, you have to run:

```
$ workon trix
```

Install the development requirements:

```
$ pip install -r requirements/develop.txt
```

Run npm install (include -g if you want global):

```
$ inv npm-install
```

Run bower install:

```
$ inv bower-install
```

Finally build the static files:

```
$ inv grunt-build
```

Create a database

See *Using and creating database dumps*.

Build the docs

Enable the virtualenv, and run:

```
$ cd docs/
$ inv docs
```

Then open `_build/index.html` in a browser.

3.1.2 Code guidelines

Python code

Follow PEP8. Some parts of PEP8 is just "preferred" or has options. In those cases, we do the following:

- Indent with spaces is REQUIRED, not just preferred.
- You can use 100 chars long lines for Python code.

Django templates

- Indent with 4 space (no tabs).
- Use `.django.html` as filename suffix. This enables us to configure our editors to syntax hilight Django html templates with a different hilighter than AngularJS templates.

3.1.3 Develop the Django project

Note: You should read *Getting started with the Django app and/or the documentation* before you read this.

Note: All the commands in this guide assumes you have *enabled the virtualenv*, and that the CWD is the root of the repo.

Run the Django development server with sqlite database

Run:

```
$ python manage.py runserver
```

to start the Django development server.

Run the Django development server with Postgres

Run:

```
$ DJANGOENV=postgres_develop python manage.py runserver
```

to start the Django development server.

Running tests

To run the tests, we need to use a different settings file. We tell mgp to do this using the `DJANGOENV` environent variable:

```
$ DJANGOENV=test python manage.py test
```

3.1.4 Using and creating database dumps

We use dumpscript from the django-extentions Django app to create our test data. We already have data, so unless you want to add more data, you do not need to know anything more than how to run a Django management task or an invoke task.

Importing the test data

The easiest method of importing the test database is to use the `recreate_devdb` Invoke task:

```
$ inv recreate-devdb
```

Warning: This will destroy your current database.

A slighly more low level method is to use the management command:

```
$ python manage.py runscript trix.project.develop.dumps.dev.data
```

This does exactly the same as the management command, but it does not destroy and re-initialize the database for you first.

Users in the test database

After importing the test data, you will have some new users. Login to the Django admin UI (http://localhost:8000/admin/) with:

```
user: super@example.com
password: test
```

and select Users to list all users. The password of all other users are `test`.

Add new data

To add new data, you just need to do add data to the database manually, or programmatically.

Adding data manually (I.E.: Using the Django admin UI)

To add data manually, you should first run the `recreate-devdb` management command to make sure you start out with the current up-to-date dataset. Then you can use the web-UI or the Django shell to add data. Finally, run:

```
$ inv dump-to-db
```

which is short for:

```
$ python manage.py dumpscript trix_core > trix/project/develop/dumps/dev/data.py
```

Adding data programmatically

Adding data programmatically must be done in `trix/project/develop/dumps/dev/import_helper.py`. See the comment at the top of `trix/project/develop/dumps/dev/data.py` for information about how `import_helper` works.

3.1.5 Adding to this documentation

This documentation uses Sphinx. It is generated from the reStructuredText sources in `docs/*.rst`. See reStructuredText Primer for to learn reStructuredText (very easy to learn the basics).

How to build the docs

This is explained in `README.md` at the root of the git repo.

How to add a new file to the documenation

- Add a new `.rst`-file in `docs/<subdirectory>/`, where `<subdirectory>` is the part of the project where the new document belongs.
- Add the filename without .rst to a toctree in `docs/<subdirectory>/index.rst`

Language and file naming

- Write everything in English.
- Use english lowercase letters and no spaces to name the `.rst`-files. Use - or _ instead of spaces.

Linking to issues and wiki pages on github

You can link to issues and wiki pages using the following syntax:

```
:issue:`<issue number>`
:wikipage:`<page name>`
```

E.g.:

```
:issue:`10`
:wikipage:`SomePage`
```

This is handled by the extlinks Sphinx extension

3.1.6 Git usage in this project

- Commit often, and with good messages.
- Commit messages should allways start with a prefix of where changes have been made. Example:
  - *djangoproject: Added login view, with redirect to reset password if user does not exist in django auth.*
  - *docs: Added description of how to use postgres as development db.*
- Throw your local changes
  - If you want to revert changes made to your working copy, do this: `git checkout .`
  - If you want to revert changes made to the index (i.e., that you have added), do this: `git reset`
  - If you want to revert a change that you have committed, do this: `git revert ...`

CHAPTER 4

Indices and tables

- genindex
- modindex
- search