Python and Django - The elephant in the room (Part 2)

Janosch Maier

April 09, 2015

Contents

1 Introduction

This tutorial let you build a scorecard application. Seminars and lectures can be added and then voted up or down. An overview page shows the courses ranked by the votes. This tutorial was build for the TUM seminar Webtech within the summer term 2015. Refer to https://wwwmatthes.in.tum.de/pages/g78qcvnwz3u3/Web-Technologies-Frameworks-Libraries-and-Plattforms for more information.

1.1 Requirements

To go through this tutorial you will need an installation of Python 3 with Django 1.7. You will probably get Python by your package manager. For help installing Django you should definitely check out https://docs.djangoproject.com/en/1.7/intro/install/. If you encounter any problems during this tutorial, refer back to that page. There is a lot of further information there.

1.2 Setup

You can either clone the application from GitHub or create it yourself.

1.2.1 Clone

To get the application from the GitHub use:

```
git clone https://github.com/Phylu/webtech-django2.git
```

Listing 1: Clone application

1.2.2 Create

If you want to start from scratch, create your development directory and create the application yourself. The application will be stored within the directory scorecard.

```
django-admin startproject webtech
cd webtech
./manage.py startapp scorecard
./manage.py migrate
```

Listing 2: Create application

Edit the file *webtech/settings.py* and add your newly created django app.

```
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'scorecard',
)
```

Listing 3: Register application

You might need to create some directories and template files yourself if you do not use the template from GitHub. This is currently not supported and most likely will never be.

1.3 Test the application

To start the server run the following command. You can then access the django application in your browser via http://127.0.0.1:8000/scorecard

```
./manage.py runserver
```

Listing 4: Run development server

2 Models

2.1 Course model

Within models information is stored for the web application. We use one model witch stores a course and the number of votes it got. A positive vote increases the vote counter. A negative vote decreases it. Create the file *scorecard/course.py* and fill it with the following content:

```
from django.db import models

class Course(models.Model):
    """
    This class stores the information about one course
    course_title     is the field which stores the title of a course
    vote             is the number of votes a course got
    pk               is created automatically as the primary key
    """
    course_title = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
```

Listing 5: scorecard/models.py

Now the model has to be registered. Run:

```
./manage.py makemigrations scorecard
./manage.py migrate
```

Listing 6: Register models

2.2 Admin app for editing models

If you try to access your django application (http://127.0.0.1:8000/scorecard, you remember?), you will get an error message. We have not yet defined the framework to show anything. To see if your model was propperly created and interact with it, django provides an admin interface in its package django.contrib.admin. This one is activated by default if you create a new django application.

Within the admin interface the new model needs to be enabled. You can do this by adding the model to the file *scorecard/admin.py*

```
from django.contrib import admin
from scorecard.models import Course

admin.site.register(Course)
```

Listing 7: scorecard/admin.py

To access it, you need to create a user first (In the version from github, the credentials are admin:admin). Run:

```
./manage.py createsuperuser
```

Listing 8: Create superuser

Now you can access http://127.0.0.1:8000/admin and create some courses as you like. As you can see, the course overview is not named nicely. This is because django does not know how to convert the course objects to a string. Add a `__str__` method to the model to solve this.

```
from django.db import models

class Course(models.Model):
    """
    This class stores the information about one course
    course_title     is the field which stores the title of a course
    vote             is the number of votes a course got
```

```
pk                is created automatically as the primary key
"""
course_title = models.CharField(max_length=200)
votes = models.IntegerField(default=0)

def __str__(self):
    """
    Convert a Course Model to a human readable string
    :return: Returns the title of the course
    """
    return self.course_title
```

Listing 9: scorecard/models.py