Project Test Cases

For the Better Betting Pool App

By Team C:
Will Blankenship, Ryan Manus, and David Rosenberg
Team contact e-mail: dcrsnbrg@memphis.edu

Submitted
December 1, 2017

Automated Testing

Testing isn't a process that can prove code is error-free; rather, it is intended to check for specific errors. Running ongoing unit tests on the Better Betting Pool project was difficult to maintain, mainly because we are beginners at Django and Python programming. Manual tests, however, were run constantly, checking for proper syntax and verifying proper interaction between objects.

So, in order to understand a bit about the process of automated testing, we followed some online examples to write a series of basic tests, written into tests.py as shown below.

```
1   from django.test import TestCase
2   from django.contrib.auth.models import User
3   from django.utils import timezone
4   from .models import *
5
6   class PlayerModelTests(TestCase):
7       def test_player_fields(self):
8           u = User(username='test', password='testtest', last_login=datetime.datetime(2017, 11,
            17, 18, 00))
9           p = Player(user=u, score=30)
10          one_week_ago = timezone.now().astimezone(timezone.utc).replace(tzinfo=None) - datetime
            .timedelta(days=7)
11          self.assertIs(p.user.username, 'test')
12          self.assertIs(p.score > 0, True)
13          self.assertIs(len(p.user.password)>=8, True)
14          self.assertIs(p.user.last_login > one_week_ago, False)
15
16  class GameModelTests(TestCase):
17      def test_game_fields(self):
18          g = Game(team_1='bears', team_2='sloths', favorite='sloths')
19          self.assertIs(g.team_1 == g.favorite or g.team_2 == g.favorite, True)
```

A player object is created, then tested for the given username, to see if the player's score was a positive number, to verify that the player's password was at least 8 characters, and to check whether the user had logged in over the course of the last week. Next, a game object was created, and a test was run to check whether the favored team was one of the teams playing. Results of the automated testing are as follows:

```
D:\mysite>python manage.py test betting
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
..
----------------------------------------------------------------------
Ran 2 tests in 0.001s

OK
Destroying test database for alias 'default'...
```

Using the Coverage Tool

Upon some investigation, several third-party tools are available to assist with testing. One such tool is called Coverage. It measures the coverage of your unit testing, to see how much of your code has actually been checked by your automated tests. The results of running coverage on our betting pool app after our first sprint are shown below.

```
D:\mysite>coverage report
Name                                                Stmts   Miss  Cover
-----------------------------------------------------------------------
betting\__init__.py                                     0      0   100%
betting\admin.py                                       12      0   100%
betting\apps.py                                         3      0   100%
betting\migrations\0001_initial.py                      7      0   100%
betting\migrations\0002_auto_20171031_0906.py           5      0   100%
betting\migrations\0003_team_game_points.py             5      0   100%
betting\migrations\0004_auto_20171031_0936.py           5      0   100%
betting\migrations\0005_auto_20171031_1039.py           5      0   100%
betting\migrations\0006_game_pub_date.py                7      0   100%
betting\migrations\0007_auto_20171031_1703.py           9      0   100%
betting\migrations\0008_auto_20171031_1714.py           7      0   100%
betting\migrations\0009_auto_20171031_1715.py           7      0   100%
betting\migrations\0010_auto_20171201_1437.py           7      0   100%
betting\migrations\0011_auto_20171201_1437.py           7      0   100%
betting\migrations\0012_auto_20171201_1438.py           6      0   100%
betting\migrations\__init__.py                          0      0   100%
betting\models.py                                      29      3    90%
betting\tests.py                                       17      0   100%
betting\urls.py                                         4      0   100%
betting\views.py                                       17      9    47%
manage.py                                              13      6    54%
mysite\__init__.py                                      0      0   100%
mysite\settings.py                                     18      0   100%
mysite\urls.py                                          3      0   100%
mysite\wsgi.py                                          4      4     0%
-----------------------------------------------------------------------
TOTAL                                                 197     22    89%
```