**Lab 4 Django**

Andrea Murphy

Project Log: /django/week4

**Subject**: Python and Django

**Purpose**: Become familiar with foreign key relationships and unit testing

**Description**: Update our database to include a foreign key relationship, explore how to retrieve data that is related to another object via a foreign key relationship, add methods to our models, and create automated unit tests.

**Details**

1. App: “resume”
   * Unless otherwise stated, all references below relate to the app “resume” we have been working in previous labs.
2. Models:
   * Resume Model
     + Create a new model named “Resume”
     + It should have at least two fields:
       - First\_name (CharField)
       - Last\_name (CharField)
     + Methods
       - get\_full\_name
         1. Should return “Jon Shallow”
       - get\_last\_name\_first\_name
         1. Should return “Shallow, Jon”
       - get\_experience
         1. Should return all foreign key related “Experience” objects
       - get\_education
         1. Should return all foreign key related “Education” objects
   * Experience Model
     + Modify experience model so that it now includes a foreign key relationship to the “Resume” model. Add a default value (set it to 1), otherwise django will yell at you when you try to migrate
   * Education
     + Modify education model so that it now includes a foreign key relationship to the “Resume” model. Add a default value (set it to 1), otherwise django will yell at you when you try to migrate
3. Django Shell
   * Launch the shell using “python manage.py shell”
   * Import the Resume model
     + from resume.models import Resume
   * Create and save a new resume object
     + new\_resume = Resume(first\_name=’your name’, last\_name=’your\_name’)
     + new\_resume.save()
   * To view the reverse side of a foreign key relationship (from the parent, see the children), django helps us by given us a very useful RelatedManager. They called by default using the following naming convention: <name of the child model>\_set
     + To get the experience objects that are related to a resume:
       - new\_resume.experience\_set.all()
       - The RelatedManager (experience\_set) has all the functional of the default ModelManager (which we seen before called “objects”...resume.objects.all() )
     + To get the education objects that are related a resume
       - new\_resume.experience\_set.all()
4. Urls-Views-Templates:
   * URLS
     + No changes
   * VIEWS
     + Update your views so that the data is all handled via a resume object
       - Get the first resume in the database
       - From that resume, get the related experience and education sets
   * TEMPLATES
     + Updated the template used to display your view as appropriate.
5. Tests
   * Create unit tests using django’s built-in testing framework to test the methods that we defined earlier in the resume model (the four methods we added to the Resume model).
   * This will probably by the most confusing portion for most people. Don’t be afraid to task questions, use slack to collaborate.

**Resources**

1. **Django Project**
   1. Testing <https://docs.djangoproject.com/en/2.0/topics/testing/>
   2. ForeignKey Field <https://docs.djangoproject.com/en/2.0/ref/models/fields/#foreignkey>
2. **Django Boo**k
   1. Ch 4 Django Models <https://djangobook.com/django-models/>
   2. Ch 12 Testing In Django <https://djangobook.com/testing-in-django/>
3. **HighFiveCode**:
   1. Testing: Videos 31-34
   2. Foreign Keys: Video 14
   3. DCC Repo (remember dcc is a django 1.11 project, there may be some subtle differences for 2.0): <https://github.com/highfivecode/DjangoCrashCourse>
   4. DCC Cheat Sheet <https://github.com/highfivecode/guides/blob/master/index/DjangoCrashCourse.md>
4. **Python Docs**
   1. Unittest framework: <https://docs.python.org/3/library/unittest.html>

**Requirements (see previous labs for verbose instructions)**

1. Work inside a venv
2. Put this in a directory “week4” in your django directory
3. Develop and commit incrementally
4. Adhere to python coding style
5. Participate in slack discussions