Documentation – JHU CS Submitter

Kyle Verdeyen, Independent Research, Summer 2017

Dr. Joanne Selinski

* Django in a nutshell:
  + Django is a framework for ORM and web applications. Every folder (/mainform, /table, etc) is its own “application.” Each application usually represents its own page or site function.
  + Generally in an app:
    - models.py – Code for objects, like a constructor for OOP. These are the fields for the SQL database.
    - urls.py - Handles URL management using regex to develop URL slugs. Defines what template to use.
    - views.py – Tells the application what to do when it gets a request. This is where models, urls, forms…etc meet up.
    - forms.py –Form code, this case using a ModelForm to convert our class model into a form which can then be displayed by a template
    - /templates/ - HTML for the page. Django can use hooks to run loops, display data, etc.
* Creating a new app/module
  + python manage.py startapp xxx
    - Then add urls.py (required), forms.py (optional), /templates/ (optional) to the app’s folder.
    - Register the application in /cs\_submitter/settings.py and /cs\_submitter/urls.py.
    - At minimum the application needs a view and a URL to refer to.
    - If you want the application’s model to be editable from the admin page, add admin.site.register(XXX) to admin.py and register the application as xxx.apps.xxxConfig in settings.py.
* How to add or remove field
  + To add/remove field to submission and/or edit form, files that need modification:
    - /…/mainform/models.py
    - /…/mainform/forms.py
    - /…/mainform/templates/form/index.html
    - /…/classeditor/views.py
    - /…/mainform/templates/mainform/mainformmodel\_update\_form.html (if desired, lists all fields from MainFormModel by default)
* Updating table HTML
  + To add/remove field to table:
    - /…/table/templates/table.html
      * Currently available fields: 'is\_new\_course', 'class\_number', 'section\_number', 'course\_title', 'course\_description', 'cs\_area', 'course\_instructor', 'start\_time', 'end\_time', 'is\_monday', 'is\_tuesday', 'is\_wednesday', 'is\_thursday', 'is\_friday', 'semester', 'course\_area', 'num\_credits', 'enrollment\_limit', 'final\_exam'.
* SQL migrations
  + If a field is added, deleted, or otherwise modified in the model, Django must generate migrations to the SQL table. Run:
    - python manage.py makemigrations
    - python manage.py migrate
  + May ask for a default value if adding something. You can usually make this anything you want, however Python standards still exist (e.g. a Boolean must have values of 0 or 1)
* How to delete section instance
  + Go to admin page, click ‘Main form models’. Can edit, add, or delete from this menu.
* How to access admin
  + Either click admin link or go to site.com/admin.
    - **Username: admin Password: jhucs2017**
* To add/delete/change admin or other users:
  + Can use admin page, or:
  + Add:
    - python manage.py createsuperuser, follow instructions
  + Delete:
    - python manage.py shell
    - from django.contrib.auth.models import User
    - users = User.objects.all()
    - print users
    - user = users[x] (whichever user you want)
    - user.delete()
      * If you know the username already you can do User.objects.get(username=”xxx”).delete()
  + Modify:
    - python manage.py shell
    - from django.contrib.auth.models import User
    - users = User.objects.all()
    - print users
    - user = users[x] (whichever user you want)
    - user.set\_password(‘xxx’)
    - user.save()
* How to deploy
  + In debug:
    - python manage.py runserver
  + For production:
    - Copy files into server’s folder.
    - python manage.py check --deploy (don’t worry about HTTPS/SSL stuff since we didn’t’ use it)
    - <https://docs.djangoproject.com/en/1.11/howto/deployment/checklist/>
    - In /cs\_submitter/settings.py uncomment the deployment settings and comment out the debug settings (see file)
    - Chmod all files to proper settings.
    - Should be able to run site now??? I was never able to get it running on my ugrad account (tested to run on PythonAnywhere), maybe the professor that taught the web development class for Django could help on how to deploy to cs.jhu.edu. Instructions found on the internet can only take you so far, as the JHU servers are a specialized case and run differently than a standalone web server.
* Rundown of filesystem
  + /cs\_submitter/
    - /classeditor/
      * /\_\_pycache\_\_/
      * /migrations/
      * \_\_init\_\_.py
      * admin.py
      * apps.py
      * forms.py
        + Uses the MainFormModel to make a new instance of the form available for editing, named EditForm.
      * models.py
      * tests.py
      * urls.py
        + Any /edit/x link takes the respective entry from the SQL db to copy into EditForm.
      * views.py
        + Uses UpdateView and the MainFormModel to modify an entry in the database. Any fields added/removed in the mainform need to be changed here too.
    - /cs\_submitter/
      * /\_\_pycache\_\_/
      * \_\_init\_\_.py
      * settings.py
        + Settings configuration, has 2 sets of configs, one for debug/dev and one for deployment. **Main file.**
      * urls.py
        + Base URL configuration. Directs all applications and their respective urls.py **Main file.**
      * wsgi.py
    - /homepage/
      * /\_\_pycache\_\_/
      * /migrations/
      * /templates/
        + /homepage/

index.html

HTML for the homepage

* + - * \_\_init\_\_.py
      * admin.py
      * apps.py
      * models.py
      * tests.py
      * urls.py
        + Any blank URL triggers the ‘index’ view.
      * views.py
        + index, renders index.html template for this app on any request.
    - /mainform/
      * /\_\_pycache\_\_/
      * /migrations/
      * /templates/
        + /form/

index.html

Template HTML for the submission form.

* + - * + /mainform/

mainformmodel\_update\_form.html

Template HTML for the EDITING FORM. This is a weird place to put HTML used by a different app, but that’s how Django wanted it…

* + - * \_\_init\_\_.py
      * admin.py
        + This app is registered in admin page.
      * forms.py
        + Uses MainFormModel and django’s ModelForm to generate a form based on the model.
      * models.py
        + Contains MainFormModel. **The bread and butter of this entire operation.**
      * tests.py
      * urls.py
        + Directs requests (to site.com/form) to this app’s index.html template.
      * views.py
        + Logic for course submission. Uses POST. Very basic form verification, more could be added here. If the form is “valid” (only checks if there are no illegal args in the fields) then the site will save the course to the db and redirect to the table. If invalid, the site will not proceed, instead displaying an error next to the bad field with all data previously entered still available.
    - /table/
      * /\_\_pycache\_\_/
      * /migrations/
      * /templates/
        + table.html

Template HTML for the table.

* + - * \_\_init\_\_.py
      * admin.py
      * apps.py
      * models.py
      * tests.py
      * urls.py
        + Directs requests (site.com/table) to this app’s table.html. Notice since we aren’t using index the template doesn’t need a folder with the app name on it.
      * views.py
        + Any request to /table gets table.html and ALL objects from the database.
    - db.sqlite3
      * The database. Implementation uses SQLite for local db storage, however Django supports many other SQL DBs, including MySQL (which JHU uses). Editable in settings.py.
    - manage.py
      * Type python manage.py --help for all commands.
      * Most commonly used will be startapp, makemigrations, migrate, runserver (for debug), check --deploy, and shell.