Homework 2: Django Calculator

Due date: February 1, 2016 at 11:59pm

In this homework, you will extend your first homework assignment to implement the calculator functions by sending web request to a Django application running server-side. This assignment introduces Django, a popular web framework written in Python, which you will use for the next series of homeworks.

For this assignment, you will use the front end you created in the last assignment and add functionality by writing the server-side routes that render and perform the calculator tasks.

The learning goals for this assignment are to:

- Learn how requests are routed and processed in a typical MVC web application.

- Demonstrate an understanding of typical data interactions between a web client and a web server, including the difference between HTTP GET and POST parameters and the use of HTML forms as input to a stateless web application.

- Demonstrate thorough, manual validation of HTTP request parameters by a web application.

- Gain hands-on experience with Django, a production quality web framework.

Specifications

This section describes the behavior of the calculator that you will implement. The calculator itself behaves as a simple four-function calculator.

- The calculator will use the front end that you wrote for the first homework.
- A "current value" is displayed at all times as a single integer value. This value is initially zero (0).
- The calculator does integer math (e.g. 9 ÷ 4 = 2).
- If the user attempts to divide by 0 or send malformed input, reset the state and display an error message until the next button is clicked.
- No operator precedence. Do the operations in the order they are encountered.

Below are a some examples of the expected behavior of the calculator (you may use these as test cases for your own calculator).

Button Pressed	2	4	+	1	9	×	2	0	-	1	0	=
Display	2	24	24	1	19	43	2	20	860	1	10	850

Button Pressed	6	8	-	9	7	=	1	5	÷	1	3	=
Display	6	68	68	9	97	-29	1	15	15	1	13	1

Requirements

Your submission must also follow these requirements:

- Your submission must follow all of the requirements specified in the first homework assignment.
- You may not use any external libraries (e.g. jQuery, Bootstrap, etc.).
- You may not use the Python `eval()` function in the implementation of your calculator.
- Your calculator must be able to support numbers up to ±1,000,000. We will not test numbers beyond that range, so you don't have to worry about overflow.
- Your application should run with **Django 1.8.x** or **Django 1.9.x**.
- The empty URL (i.e. http://localhost:8000/ if the server is run on port 8000) must route to your application's main page in a cleared state.
- You are not to store anything in the database. The server should be *stateless*: the server may not store any data between web requests. To meet this requirement your server will need to send extra data (in addition to the displayed calculator value) to the client with each response. The client will need to resubmit that extra data with its next request.
- Your application must run on the web server. The web browser must simply submit requests to the web server (after each button- click) and display the response; the client may not use JavaScript or perform any processing of the calculator data.
- Your application may not crash as a result of any input sent to the server-side or because of any actions the user performs. Note that the user can send any data (malformed or otherwise) to the server, and you must anticipate and validate these requests.
- Cite all external resources used and any additional notes you would like to convey to your grader in the `README.md` file.

Configuring the .gitignore

Starting this assignment, you may notice that running your application generates a lot of `.pyc` files containing Python bytecode, as a byproduct. To avoid committing these files, we recommend that you set up and configure a .gitignore file on your repository.

Each line in a `.gitignore` file specifies a (regular expression) pattern. If a file matches any of the patterns specified in the repository's .gitignore, then it will not show up in the output of a command like `git log`. Note that files that are already tracked by Git are not affected.

To configure a git repository to ignore all `.pyc` files, for example, its `.gitignore` file must have the following line within it:

```
*.pyc
```

In addition to `.pyc` files, a common type of file to ignore is temporary files such as editor swap files (`.swp`, or `*~` if you use Emacs) or system specific files (such as `.DS_Store`).

You can find `.gitignore` files in the repositories posted with class examples.

Grading criteria

This section contains some specific criteria we will follow while grading your homework. However, keep in mind that *for substantial credit your solution must clearly demonstrate the learning goals for this assignment, which are described above in the introduction.*

Committing your work [10pt]

As with the previous homework, we will be evaluating your version control usage. Keep in mind that good version control usage typically means (1) incremental, modular commits with (2) descriptive and useful commit messages.

Note that we will also start more heavily penalizing any extraneous files committed in your repository (editor swap files, `.pyc`, etc.), so make sure that your `.gitignore` file (described above) is properly set up.

Specification fulfillment [20pt]

Validation [20pt]

We will test your server-side application with a variety of inputs and requests, even requests that could not have been sent by the buttons on your calculator. Note that your calculator must validate all input and respond appropriately.

Routing and configuration [10pt]

Coverage of technologies [40pt]

Turning in your work

Your submission should be turned in via Git and should consist of a Django project/application. Please your code in the “hw2” directory. Use the following commands to create the files:

```
cd hw2
django-admin.py startproject webapps .      <= Notice the “.”
python manage.py startapp calculator
```

The “.” at the end of the “startproject” command will put the project in the current directory.

Here is an example directory structure (some directories/files omitted) of a submission.

```
[YOUR-ANDREW-ID]/hw2/
    |-- webapps/
        |-- settings.py
        |-- urls.py
        |-- [etc.]
    |-- calculator/
        |-- migrations/
        |-- static/
        |-- templates/
        |-- models.py
        |-- views.py
        |-- [etc.]
    |-- manage.py
    |-- db.sqlite3
    |-- README.md
```