CPSC 449 - Web Back-End Engineering

Project 3, Fall 2020

due October 21 (Section 01) / October 23 (Section 02)

Last updated Friday, October 10 12:00 am PDT

In this project you will learn to make HTTP requests from a Python program and use this ability to store server-side session data in a separate storage service as described in Figure 3-5 of the textbook.

This project may be completed individually, or in a pair as long as both students are enrolled in the same section of the course.

Cookie-based sessions

1. Clone the GitHub repository for cpsc449 and examine the contents of the `counter` directory.

The file `app.py` is a small Flask app that displays a counter that increments every time the page is loaded. A button allows you to reset the counter.

Since the configuration variable `USE_SESSION_STORE` in `counter.cfg` is set to `False`, Flask uses the default session implementation, storing the session variable `count` in a signed cookie on the client side.

2. Start the app and try loading the page in two different browsers (e.g. Firefox and Chrome). Refresh the page a few times in each browser, and notice that they maintain separate counts. Take screenshots of a few different count values.

3. Use your browser's developer tools to examine the `Set-Cookie:` response header, and notice that it changes each time the page is refreshed. Take screenshots showing the change.

You could use curl or HTTPie to examine the network traffic as well, but both utilities require additional command-line switches to save cookies between requests.

Key-value store service

The file `kv.py` implements a simple key-value store service accessible via HTTP. The service exposes the following methods:

> `POST /` - sets a new value for a key
>
> `GET /<key>` - retrieves a value of a key
>
> `DELETE /<key>` - removes a key
>
> `GET /?prefix=PREFIX` - returns a list of all keys matching PREFIX

4. Experiment with these methods until you understand how they work. Note that the comments in `kv.py` for the `set_key()`, `get_key()`, `delete_key()`, and `match()` methods include example API calls using HTTPie.

The Requests library allows you to make API calls from Python. If you are using Tuffix, Requests is already installed.

5. Use Requests to write a Python script `dump.py` to connect to the `kv.py` service and display all of its keys and values. For example, if the store had been populated with

```
$ http localhost:5000 twitter=https://twitter.com/ProfAvery
$ http localhost:5000 github=https://github.com/ProfAvery
$ http localhost:5000 cpsc449=https://sites.google.com/view/cpsc449
```

   Then running the script should yield the following:

```
$ ./dump.py http://localhost:5000
{'twitter': 'https://twitter.com/ProfAvery'}
{'github': 'https://github.com/ProfAvery'}
{'cpsc449': 'https://sites.google.com/view/cpsc449'}
```

Server-side sessions

The file `sessions.py` includes the following class definitions to implement server-side sessions:

> `ServerSideSession` - a dictionary-like object that stores a session id and its session variables
>
> `ServerSideSessionInterface` - replaces the default Flask signed cookie session interface with a session interface that stores a session id in the cookie and the session variables in a separate session store

Server-side sessions require a session store to persist the session variables associated with a session id. The following class definitions set up the infrastructure needed for a session store:

> `SessionStore` - abstract interface whose methods must be overridden to store session information
>
> `KeyValueSessionStore` - custom subclass of `SessionStore` for storing session information in the key-value store service

If you are interested in the details of how Flask's default sessions can be replaced, see the Session Interface section of the Flask documentation.

6. Modify `counter.cfg` to set `USE_SESSION_STORE` to `True`, then start the counter app and the key-value store service with `foreman start`. Load the counter page, and notice that the methods of `KeyValueSessionStore` have not yet been implemented. Take a screenshot showing the failure.

7. Override the `set_key()`, `get_key()`, and `delete_key()` methods in `KeyValueSessionStore` to use the Requests library to store keys and values in the `kv.py` service.

Testing

8. You will know that your implementation of `KeyValueSessionStore` is correct when you can repeat step *(2)* from *Cookie-based Flask sessions* and see the correct session behavior in both browsers.

9. Repeat step *(3)* from *Cookie-based Flask sessions* and notice that the `Set-Cookie:` response header only contains a session id, and that the session id does not change when the page is refreshed and the counter is incremented. Take screenshots showing the same session key with different counter values.

10. Click the *Reset* button, and notice that the `POST` request clears the session cookie and the subsequent `GET` request generates a new session id. Take screenshots of both `Set-Cookie:` headers.

11. Use your `dump.py` script to list the active sessions in the key-value store. Take a screenshot of at least two active sessions for different browsers.

Test platform

You may use any platform for development, but note that per the Syllabus the test environment for projects in this course is a Tuffix VM with Python 3.8.2. It is your responsibility to ensure that your code runs on this platform.

Submission

Submit a compressed tarball (`.tar.gz`, `.tgz`, `.tar.Z`, `.tar.bz2`, or `.tar.xz`) file containing the following through Canvas before class on the due date:

1. The Python source code for `dump.py` and `sessions.py`
2. A `README.TXT` file as described in the Syllabus
3. A document in PDF format showing your screenshots in order with descriptions of their contents.
4. Do **not** include compiled `.pyc` files, the contents of `.git/` directories, SQLite database files, or other binary artifacts.
5. If you include other files in your tarball, I will not examine them unless your `README.TXT` states explicitly that they should be included in evaluation of your project.

If the assignment is completed by a pair, only one submission is required. Be certain to identify the names of both students in your `README.TXT` and in the PDF document.

Failing to comply with these submission instructions will immediately result in **no credit with no further feedback**. That means that you will need to re-submit, and if there are any problems with the project you will not have a chance to correct them.