CMPUT 404 Lab 7

23-24th February 2016

OVERVIEW

Learn how to create a RESTful web application backend using Flask. Integrate a basic angular application front end client to interact (CRUD) with the Flask application's REST api.

STEPS

1. Navigate to a new folder and initialize a new python virtual environment.
2. Install Flask.

```
mkdir -p ~/sandbox/CMPUT404LAB7_W2016
cd ~/sandbox/CMPUT404LAB7_W2016
virtualenv venv
source venv/bin/activate
pip install Flask
```

3. Create a new python file named **hello.py** and edit the contents so it looks like the following:
   https://gist.github.com/awwong1/936a11c8bdbefa7d26b5

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run(debug=True)
```

4. Run the application and navigate to the page in your browser.

```
python hello.py
```

5. Navigate back to your terminal and quit the running flask application. Install the python package **flask-restful**

```
pip install flask-restful
```

6. Open up the **hello.py** file and modify the file so it looks like the following: https://gist.github.com/awwong1/461cf26b7c499023a320

```
from flask import Flask
from flask_restful import Resource, Api

app = Flask(__name__)
api = Api(app)

class HelloWorld(Resource):
    def get(self):
        return {“hello”: “world”}

api.add_resource(HelloWorld, “/”)

if __name__ == “__main__”:
    app.run(debug=True)
```

7. Let’s implement a fully RESTful application for a Todo object. Open up the **hello.py** file and add the following: https://gist.github.com/awwong1/29f3082d64e1ea17fa02

```
from flask_restful import reqparse, abort, Api, Resource

TODOS = {
    'todo1': {'task': 'build an API'},
    'todo2': {'task': '?????'},
    'todo3': {'task': 'profit!'},
}


def abort_if_todo_doesnt_exist(todo_id):
    if todo_id not in TODOS:
        abort(404, message="Todo {} doesn't exist".format(todo_id))

parser = reqparse.RequestParser()
parser.add_argument('task')
```

```
# Todo
# shows a single todo item and lets you delete a todo item
class Todo(Resource):
    def get(self, todo_id):
        abort_if_todo_doesnt_exist(todo_id)
        return TODOS[todo_id]

    def delete(self, todo_id):
        abort_if_todo_doesnt_exist(todo_id)
        del TODOS[todo_id]
        return '', 204

    def put(self, todo_id):
        args = parser.parse_args()
        task = {'task': args['task']}
        TODOS[todo_id] = task
        return task, 201


# TodoList
# shows a list of all todos, and lets you POST to add new tasks
class TodoList(Resource):
    def get(self):
        return TODOS

    def post(self):
        args = parser.parse_args()
        todo_id = int(max(TODOS.keys()).lstrip('todo')) + 1
        todo_id = 'todo%i' % todo_id
        TODOS[todo_id] = {'task': args['task']}
        return TODOS[todo_id], 201

##
## Actually setup the Api resource routing here
##
api.add_resource(TodoList, '/api/todos')
api.add_resource(Todo, '/api/todos/<todo_id>')
```

8. What does the browser show you when you navigate to these pages? Try out the following curl commands.

```
curl http://127.0.0.1:5000/api/todos
curl http://127.0.0.1:5000/api/todos/todo3
curl http://127.0.0.1:5000/api/todos/todo2 -X DELETE -v
curl http://127.0.0.1:5000/api/todos -d "task=something new" -X POST
-v
curl http://127.0.0.1:5000/api/todos/todo3 -d "task=something
different" -X PUT -v
```

9. Download the latest stable version of node.js and install it locally within your home folder somewhere. (Make sure to add the bin directory to your path)

```
wget https://nodejs.org/dist/v4.3.1/node-v4.3.1-linux-x64.tar.xz
tar -xvf node-v4.3.1-linux-x64.tar.xz
mkdir ~/y_u_no_sudo
mv node-v4.3.1-linux-x64 ~/y_u_no_sudo
```

```
# inside the ~/.bash_rc file
PATH=$PATH\:/cshome/<your ccid>/.gem/ruby/1.9.1/bin
PATH=$PATH\:/cshome/<your ccid>/y_u_no_sudo/node-v4.3.1-linux-x64/bin
export PATH
```

10. Test to see if everything works! Source the ~/.bash_rc file and run some commands to check the versions of everything

```
source ~/.bash_rc
npm -v; node -v
```

11. Install the yeoman generator angular dependencies. https://github.com/yeoman/generator-angular

```
npm install -g grunt-cli bower yo generator-karma generator-angular
```

12. Make a new directory named your app name, then cd into it. Run the yeoman generator.

```
mkdir lab7app && cd $_ && yo angular lab7app
```

13. For the settings:

```
? Would you like to use Gulp (experimental) instead of Grunt? No
? Would you like to use Sass (with Compass)? Yes
? Would you like to include Bootstrap? Yes
? Would you like to use the Sass version of Bootstrap? Yes
? Which modules would you like to include? angular-animate.js,
angular-aria.js, angular-cookies.js, angular-resource.js,
angular-messages.js, angular-route.js, angular-sanitize.js,
angular-touch.js
```

14. Occasionally, the application hangs after:

```
Execution Time (2016-02-17 20:21:02 UTC)
loading tasks         350ms  ████████████████████████████████ 45%
loading grunt-wiredep   11ms  ██ 1%
wiredep:app           358ms  ████████████████████████████████ 46%
wiredep:test           33ms  ███ 4%
wiredep:sass           28ms  ███ 4%
Total 781ms
```

Once it gets here, and nothing seems to be happening, press Enter a few times.

15. Once the above *npm install* and *bower install* finish, proceed to install Sass

```
gem install --user-install sass
```

16. Run the application and view it within the browser:

```
grunt serve
```

17. Create a new route within the newly scaffolded angular application:

```
yo angular:route todo

# This should have created three files:
   create  app/scripts/controllers/todo.js
   create  test/spec/controllers/todo.js
   create  app/views/todo.html
```

18. Within your **app/index.html** file, modify the navbar such that the Contacts link is replaced with a link to our new todo route.

```
<div class="collapse navbar-collapse" id="js-navbar-collapse">
  <ul class="nav navbar-nav">
  <li><a href="#/">Home</a></li>
  <li><a ng-href="#/about">About</a></li>
  <li><a ng-href="#/todo">Todo</a></li>
</ul></div>
```

19. Modify your **app/views/todo.html** file to look like the following: https://gist.github.com/awwong1/b9911b9d784e9babc930

```
<p>This is the todo view.</p>
<div ng-repeat="(todo_key, todo_value) in todo.todos">
  {{ todo_key }} : {{ todo_value }}
</div>
```

20. Modify your **app/scripts/controllers/todo.js** file to look like the following: https://gist.github.com/awwong1/66936fa0f78629b036d7

```
'use strict';

/**
 * @ngdoc function
 * @name lab7appApp.controller:TodoCtrl
 * @description
 * # TodoCtrl
 * Controller of the lab7appApp
 */
angular.module('lab7appApp')
  .controller('TodoCtrl', function ($http) {
    /*
    // ECHMA 6, but this will break on dist compilation
    $http({
      method: 'GET',
      url: 'http://127.0.0.1:5000/api/todos'
    }).then((response) => {
     console.log(response);
     this.todos = response;
    }, function errorCallback(response) {
    });
    */
```

```
    // ECHMA 5
    var self = this;
    $http({
      method: 'GET',
      url: 'http://127.0.0.1:5000/api/todos'
    }).then(function successCallback(response) {
        console.log(response);
        self.todos = response.data;
    }, function errorCallback(response) {
        // do nothing for now
    });
  });
```

21. Back in your **hello.py** restful backend, install flask-cors and enable CORS on your entire application.

```
pip install -U flask-cors

# editing hello.py
from flask.ext.cors import CORS

app = Flask(__name__)
api = Api(app)
CORS(app)
```

22. Initialize the rest of the CRUD actions for the REST api in the **app/scripts/controllers/todo.js** file.
https://gist.github.com/awwong1/54571b633d1735a8f979

```
'use strict';

/**
 * @ngdoc function
 * @name lab7appApp.controller:TodoCtrl
 * @description
 * # TodoCtrl
 * Controller of the lab7appApp
 */
angular.module('lab7appApp')
  .controller('TodoCtrl', function ($http) {
```

```
var self = this;
var myUrl = "http://127.0.0.1:5000/api";
self.todos = {};

// Initialize all the todos, get the list of todos. (Read)
function initTodos() {
  $http.get(myUrl + '/todos').then(
 function successCallback(response) {
      self.todos = response.data;
    }, function errorCallback(response) {}
  );
};
this.initTodos = initTodos;

// Edit a todo by its key (Update)
function editTodo(key) {
  $http.put(myUrl + '/todos/' + key, self.todos[key]).then(
 function successCallback(response) {},
    function errorCallback(response) {}
  );
};
this.editTodo = editTodo;

// Delete a todo by its key (Delete)
function deleteTodo(key) {
  $http.delete(myUrl + '/todos/' + key).then(
 function successCallback(response) {
   delete self.todos[key];
 },
    function errorCallback(response) {}
  );
};
this.deleteTodo = deleteTodo;

// Create a todo (Create)
function createTodo(key, value) {
  var data = {'task': value}
  $http.post(myUrl + '/todos', data).then(
    function successCallback(response) {
```

```
          self.initTodos();
        },
        function errorCallback(response) {}
      );
    };
    this.createTodo = createTodo;

    this.initTodos();
  });
```

23. Initialize the rest of the CRUD actions for the REST api in the **app/views/todo.html** file. https://gist.github.com/awwong1/ef1070fe1968f6c98167

```
<div class="row">
  <h3>New Todo Item</h3>
  <div class="col-xs-12">
    <div class="input-group">
      <input type="text" class="form-control" placeholder="Task
Value"
        aria-describedby="task_value_label" id="task_value"
        ng-model="task_value">
      <span class="input-group-btn">
        <button class="btn btn-success"
          ng-click="todo.createTodo(task_name, task_value)">
       Add New Item
      </button>
      </span>
    </div>
  </div>
</div>
<hr>
<div class="input-group" ng-repeat="(todo_key, todo_value) in
todo.todos">
  <span class="input-group-addon" id="{{ todo_key }}_label">
    {{ todo_key }}
  </span>
  <input type="text" class="form-control"
    id="{{ todo_key }}"
    ng-model="todo.todos[todo_key].task"
```

```
    ng-change="todo.editTodo(todo_key)"
    aria-describedby="{{ todo_key }}_label">
  <span class="input-group-btn">
    <button class="btn btn-danger" type="button"
      ng-click="todo.deleteTodo(todo_key)">
      Delete
    </button>
  </span>
</div>
```

24. We don't need grunt to serve our client and our flask application to serve our backend. We should consolidate both of these such that they are both served by Flask. Run the grunt build command within the angular application.

```
grunt build
```

25. Ensure that a **dist** folder is created. Run the serve dist command as opposed to just the serve command. What's the difference?

```
grunt serve
grunt serve:dist
```

26. Edit the **hello.py** python application to serve the dist folder at root instead of the hello world object.
    https://gist.github.com/awwong1/c91e6620cdcdb5ac3914

```
from flask import Flask, send_from_directory

…

##
## Actually setup the Api resource routing here
##
api.add_resource(TodoList, '/api/todos')
api.add_resource(Todo, '/api/todos/<todo_id>')

@app.route('/')
def root():
     return send_from_directory('dist', 'index.html')
```

```
@app.route('/<path:path>')
def root_path(path):
      return send_from_directory('dist', path)

if __name__ == '__main__':
      app.run(debug=True)
```

27. Finalized project can be found at https://github.com/awwong1/CMPUT404LAB7_W2016

QUESTIONS

Question 1:

What does REST stand for? What does it mean?

Question 2:

What does CRUD stand for?

Question 3:

In general, what do HTTP 1XX status codes mean? HTTP 2XX? HTTP 3XX? HTTP 4XX? HTTP 5XX?

Question 4:

What are the benefits of using yeoman generators to scaffold our application? What are the cons?

Question 5:

What is a XSS attack? What does CORS stand for and what does it mean?

Question 6:

Why should client side javascript and css be minified before deployment?