Dynamic Web Project

WEST KARATE CLUB

John Palmer G00316645 | | 2/12/2016
Data Representation and Querying
Tutor | | Martin Kenirons

Contents

Executive Summary

This report sets out the projects aims and scope, the main aspect which is to create a web based single page application that will dynamically load content into a web based application. This means that the web page does not reload when a link is clicked within the page (i.e. the whole page does not have to reload). The report details how the user should use the application and what happens behind the scenes.

The front cover displays all the different software needed in order to create a web based single page application.

The Assignment

SCOPE

- To develop a single-page web application(SPA)
- The programming language to be used is Python
- Frame work to be used is Flask (Micro Framework)
- Jinja2 Templating
- GitHub

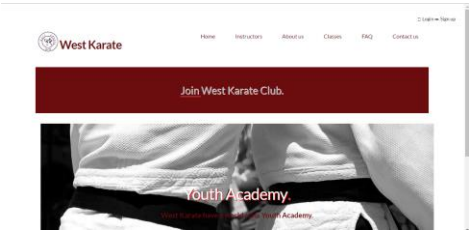
ADDITIONAL PROGRAMMING CODE

- Ajax
- JQuery
- Html5
- CSS3
- JavaScript
- MySQL

THE IDEA

The web application is a club website, in which the clubs information can be accessed. The user will be able to see the class times, the clubs location, a contact form which allows comments, registration of a user and a login. Lastly, instructor's information is controlled with CSS.

The user guide



The index page gives the user a slide show, which displays 3 images relating to the karate club. It displays the club's logo which links to the class time and costs. The navigation bar has buttons to the home page, instructors, classes, about us and contact us. Above the navigation bar there is a registration button and a login.

The user guide for this web application is quite simple as most users are quite familiar with the use of buttons on web sites. If an individual clicks on a button it will display content relating to the name on the button.

For this application I remained with a format that the user would be able to navigate easily in order to give a fluid experience.

The user guide steps

- Main page loads
- User can view the images before clicking on a button
- User can click on the 'Instructors' and the content for those instructors loads without a total page refresh
- The instructor's page has images that when the mouse is hovered over the image of an instructor it flips over to reveal details about him or her
- User can click on the 'About us', and the content of that page again is loaded without a total page refresh
- The 'About us' sets out the club's aims and has a link to the main governing bodies instructors
- User can click on the 'Classes' button. This page contains class times and class etiquette
- User can click on the 'FAQ' button and the content based on questions asked and answered is loaded without a full page refresh
- The user can click on the 'Sign up' button which is located above the navigation bar. It loads a form into the sign up page whereby the user can input a user name, email and password. The user must accept the privacy in order to complete registration. After successful registration, the user is redirected to the home page
- The user can click on the 'Login' button which is located above the navigation bar. The login page allows the user to enter in their username and password
- The user can click on the 'Contact us' button and a Google form and Google map is displayed. There is a full page reload due to the fact that the page has to access Google's Application Programming Interface (API). The contact form allows the user to leave their details and leave any comments. The site administrator will receive an email including the details, which will also be placed into a spreadsheet

THE BACKEND

Flask

The website is hosted on Flask micro frame work and uses Python. In order to install Flask onto my PC, I needed to install a program called Anaconda which uses Flask.

There are over 100 pre-built packages within Anaconda Flask and MySQL being among them. With a simple “conda install <package name>” over 620 more packages can be installed.

Jinja2

The web application also uses Jinja2 templating, which allows the use of a basic page layout that would normally be repeated on each page of the application. The template contains all the header information, such as links to the bootstrap, Content Delivery Network (CDN) and style sheets. It also contains navigation buttons and a footer. It essentially contains any piece of repeat code that would appear on all pages of a standard website. With the use of body tags, Jinja2 allows the insertion of content that is between matching body tags.

Example of a body tag: {% block body %}<content> {% endblock %}

GitHub

The project was set up with GitHub being for a repository. There were uploading issues with the GitHub commit, however, the issues were resolved by using GitHub for desktop on the PC. This is a GUI which allows the file on the PC to sync with the file in the repository.

HTML5 / CSS3

The main pages for the web application were written with HTML5 and CSS3.

Python

The file that runs the server is written in Python, the main file being Hello.py. It contains all the information required for the web application to run by importing all the different packages required.

This part is quite challenging as Python is based on indentation and is unforgiving if there is an error in the code.

An Example of the Python connection code for pyMySQL:

```
def connection():

    conn = pymysql.connect(host='sql8.freemysqlhosting.net', port=3306, user='sql8146091', passwd='gLs1fMyhhT', db='sql8146091')

    c = conn.cursor()

    return c, conn
```

MySQL

The package of MySQL would not install for me initially. In order to work around the problem I installed pyMSQL and then within my Hello.py file I inserted the following code:

```
import pymysql

pymysql.install_as_MySQLdb()
```

I am using a remote MySQL database which holds a database called users and contains the auto incremented ID and user name, email, password which is stored as a sha 256 hash.

In the Hello.py file, the script checks the user name and compares the stored hash. When a user registers the script will ensure the user name is not taken.

JQuery

jQuery is used with the web application in order to load the pages asynchronously and also to block the default action when an anchor tag in the navigation bar is clicked. Without this aspect of the code, the page would execute a Http Get request. However, by blocking the default action and having the pages loaded, you can insert the content without the page being refreshed.

There is also code that is manipulating the back button on the Brower. The jQuery is using the push state, pop state and history in the DOM, by taking a snap shot of the pages and placing them on the stack. With this, when you click on a link the push state takes the url and places it on the history stack. If then, you press the back button it brings the pages back in the sequence they were previously viewed.

Other jQuery code is also used within the web Application for slider and accordions.

Conclusion

Based on the assignment given, I have created a web application that gives the user a fluid experience, in which that user can access all the relevant information for West Karate club in a dynamic, asynchronous website. The user can also contact the club from within the site with the Google form.

I found the project quite challenging but very rewarding. I have now expanded my knowledge including, but not exclusive to Http, Ajax, Python, Flask, Jinja2. Finally, it has added to my overall ability to self-learn and problem solve.

References

- Lab work and lecture notes
- On line tutorials
- Application documents

https://pythonprogramming.net/

http://flask.pocoo.org/

http://jinja.pocoo.org/docs/dev/

https://github.com/PyMySQL/PyMySQL/blob/master/example.py

http://www.w3schools.com/xml/ajax_intro.asp

https://jquery.com/

http://getbootstrap.com/