Group 20

Sam Westerlund

Sean Hinchee

Requirements:

* Raspberry Pi (Raspbian)
* Apache 2
* Python 3
* JavaScript

Run:

* We self-host at <http://raspi.student.iastate.edu> (on-campus only)

To start the server (from the Pi):

* *sudo service apache2 restart*

To clear the playlists:

* *sudo python /srv/www/port3/cgi/server.py*

Usage:

* Visit <http://raspi.student.iastate.edu/>
* Click on a playlist
* Click on a song within a playlist

Manually modify files/source:

* We use FTP manually as well as the Remote-FTP mod in the Atom editor (<http://atom.io/>)

Tested on:

* Chrome
* Firefox

Checklist

* Self-contained design via the Raspberry Pi
* New server systems
  + Apache
  + FTP
* Co-habitual web app
  + Apache
  + Python 3
* CGI Usage (for uploading files)
* Remote development
  + Atom and FTP were used for remote testing and development

Group 20

Sam Westerlund

Sean Hinchee

Portfolio 3 – A cloud-capable playlist builder and player.

Table of Contents

1. Overview (p.5)
   1. Design principles and adaptation
   2. Self-contained design
   3. Portable design
2. New and Complex (p.6)
   1. Uploading files to the server
   2. Building the server on a Rasberry Pi
      1. Using GNU/Linux
   3. Learning Python 3
   4. Integrating Apache and Python 3
3. Bloom’s Taxonomy (p.7-9)
   1. Originally meant to be a Python 3 app
      1. Python was a new language
   2. Difficulties with file upload
      1. Decided to use CGI via Apache
   3. Evolved into an Apache + Python 3 design
   4. Ported to run on a Raspberry Pi
      1. Had to set up FTP and SSH
   5. Overall integrated design

Overview

Originally, the concept for the portfolio was to build a web-based playlist manager. The first stage of development was to produce an app that was based in Python 3. Although the original web-app stages were successful in development, the first roadblock to be hit was that of file upload.

The resolution of the upload issue was to take a well-documented route of using Apache and CGI to upload files for use with the playlists. This allows users to add their own songs to their playlists at will.

A Raspberry Pi was chosen as the development platform for the app since it is a common platform that is simple to use and is easy to set up with Apache. The self-contained nature of the Raspberry Pi made the project and objectively simple to approach design where anyone could, in theory, set up their own playlist server at home, etc. for personal use with minimal effort.

New and Complex

Building a web application that involved uploading files was a new type of challenge neither of us had attempted prior to portfolio 3. We are on our third portfolio which involves some form of web application, but we had not previously attempted file transfer within a web application. Portfolio 3 was the first time we had tried to perform a connection to the client system to perform actions beyond basic web interaction.

The design choice to utilize GNU/Linux as well as a Raspberry Pi was a new front that neither of us had attempted to work on previously in depth or formally. Raspberry Pi’s are a common and affordable computer that is easy both to set up and to extend.

Python 3 was a new language neither of us had used previously. While Python 2 may have been used on and off, neither of us had utilized Python 3 during our education at ISU nor in personal projects. The learning curve was not terrible, but existent. Resources for learning were plentiful, so we could develop our application at pace with few language-based issues.

Using Apache was a new platform that was unfamiliar as neither of us had used it in a class or formal setting before. Apache was leveraged to utilize the CGI web interface (Common Gateway Interface) so that we could upload files in an easy to develop manner.

Bloom’s Taxonomy