HW 2 - Server-Side Basics Done 3 Ways

Due Apr 27 by 11:59pm **Points** 124
Available Apr 14 at 12am - Apr 28 at 11:59pm 15 days

This assignment was locked Apr 28 at 11:59pm.

HW 2 - Server Side Basics Done 3+ Ways

Learning Outcomes

- The student will learn how to build server-side programs in multiple languages often as CGI style programs, demonstrating that the language does not deeply impact the request/response cycle of HTTP.
- The student will analyze the differences across the technologies, hopefully noting the sameness of abilities as it all just relies on HTTP requests/responses just with different levels of abstraction and language syntax.
- The student will analyze three different kinds of logging and analytics platforms to compare prebuilt analytical platforms to our potential projects. An over-all goal of the comparisons will be encourage thinking about buy-build-customize trade-offs for good solution finding in future efforts.

Before You Begin

Read the FULL WRITEUP before starting and work slowly. Many errors we have seen have been due to rushing, skipping steps or not close reads. While bugs may exist, bigger stumbling blocks appear to come from hastiness.

First up, make sure you have all the necessary technology installed on your server! Pick which languages you want to use and get them installed. Once installed sanity check the language there and make sure you are comfortable with the language if it is not already familiar to you. Do a non-web "hello world" or two with focus on reading and writing strings. Once you believe you are comfortable Google " <language> apache cgi hello world." Jumping right to the web execution of the homework will vary in ease from a little bumpy to potentially a grade and time destroying effort. You've been warned.

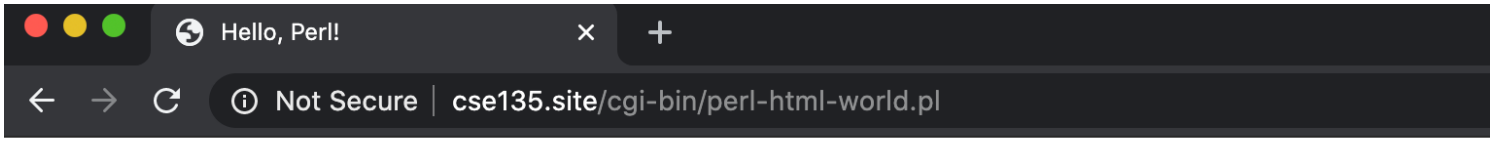
Part 0: Get OUR Code Running

Server Side Programs

Here are various CGI programs, all accomplishing the same thing, but written in different languages

- Perl - CGI
  - Hello, Perl! (source)
  - Hello, Perl! (JSON) (source)
  - Environment Variables (Manually) (source)
  - Environment Variables (CGI.pm) (source)
  - GET Echo (source)
  - POST Echo (source)
  - General Echo (source)
  - Sessioning (In Progress)
- C - CGI
  - Hello, C! (source)
  - Hello, C! (JSON) (source)
  - Environment Variables (source)
  - GET Echo (source)
  - POST Echo (source)
  - General Echo (source)
  - Sessioning (In Progress)
- PHP - Server Side Scripting via a Module
  - Hello, PHP!
  - Hello, PHP! (JSON)
  - Environment Variables
  - GET Echo
  - POST Echo
  - General Echo
  - Sessioning (In Progress)

Go to **cse135.site/cgi.html** (http://cse135.site/cgi.html). You will see example programs done in both Perl and C along with the source code. Download the source files and get them running. Change the code to put your name or team name in the various "Hello Examples."



Thomas was here - Hello, Perl!

This page was generated with the Perl programming langauge

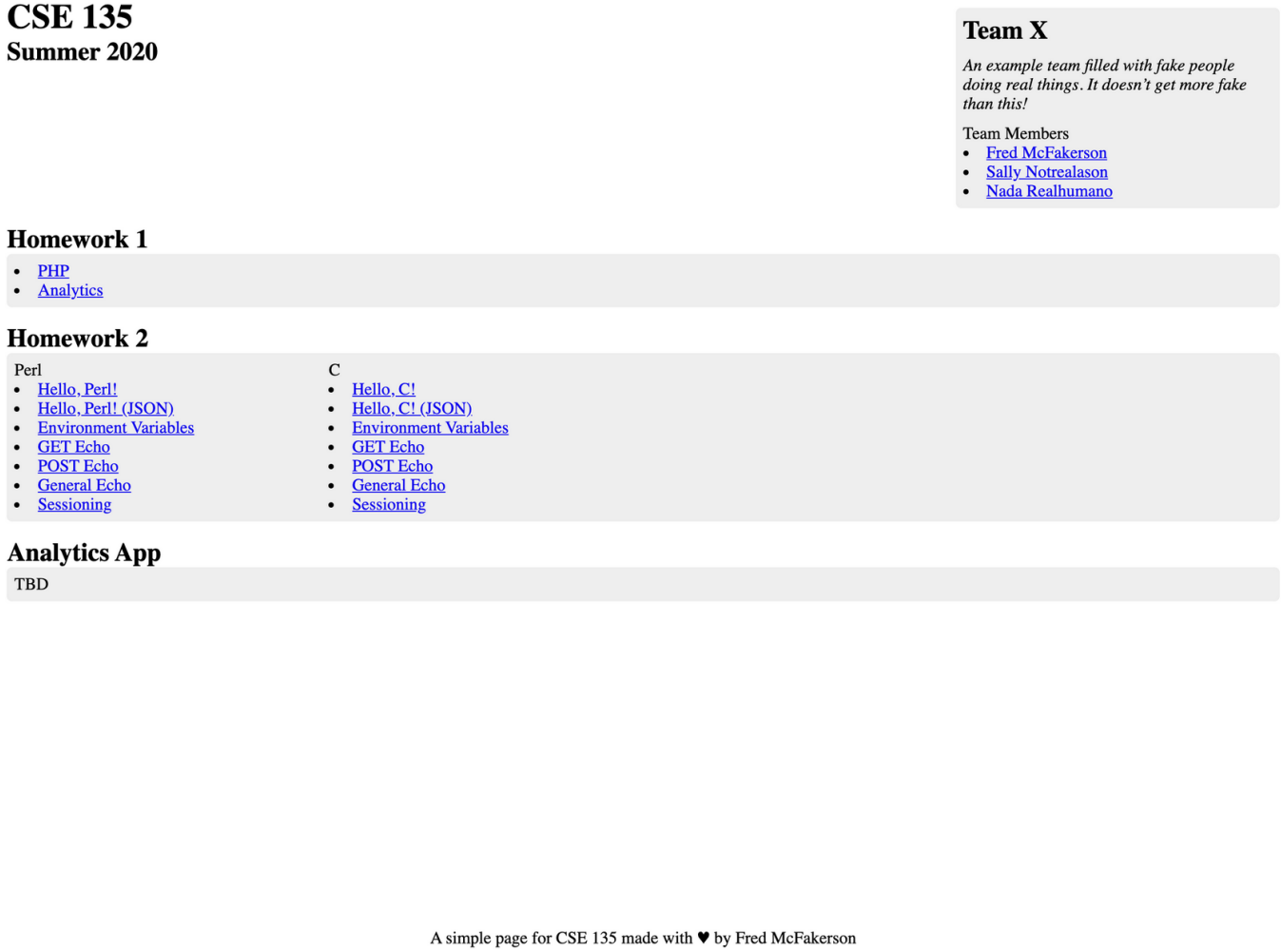
Current Time: Wed Aug 12 18:35:45 2020

Your IP Address: 66.27.63.226

Make sure your domain is visible in the screen capture.

Screenshot Alert! Once you have made this code your own take a screenshot to show that they are running on your domain. An example capture is shown above. You will need to take a screenshot **for each** of the 19 Perl and C files that you download. Title the files accordingly - c-hello-html-world.png, etc.

Additionally, link to each of these pages from index.html.



Tip: CGI Quick Start Guide

The most challenging part of Part 0 is to make sure you have set-up CGI correctly. To get started with serving CGI programs, you'll have to configure your server to execute these script files. The following quick start guide was adapted from the Apache how-to guide here: **https://httpd.apache.org/docs/2.4/howto/cgi.html (https://httpd.apache.org/docs/2.4/howto/cgi.html)**

1. Make a `cgi-bin` directory in your Document Root. This should be at `/var/www/yourdomain.site`, `/var/www/yourdomain.site/public_html`, or something similar. If you don't know what your Document Root is, go to yourdomain.site. It will pull your `index.html` file. Whatever directory that file is in is your Document Root.

2. Now, we need to configure that directory (cgi-bin) to run all programs as executables. To do this we will make changes in your global .conf file - should be at `/etc/apache2/apache2.conf`.

3. First, configure Apache to permit CGI by adding the following lines in your .conf file

```
LoadModule cgid_module /usr/lib/apache2/modules/mod_cgid.so
ScriptAlias "/cgi-bin/" "/<your path to cgi-bin> (probably var/www/domain.site)/"
```

4. At the end of the file, add a Directory directive. This will look like `<Directory /path/to/cgi-bin> </Directory>` . This will allow you to set Apache config settings on a by-directory basis (so just for our `cgi-bin` directory). Make sure your path to your `cgi-bin` directory is correct - you can use the absolute path (that begins with `/var/www`)

5. Within this Directory directive, we want to allow certain file extensions to execute as CGI. We are able to do this with the following configuration:

```
Options +ExecCGI
AddHandler cgi-script .cgi .pl
```

   Note: the AddHandler will take a list of file extensions - if you wanted .py files to run as scripts because you chose to do your CGI in Python, you would use `AddHandler cgi-script .cgi .py`

6. Now, lets enable CGI by running `sudo a2enmod cgi`.

7. Make sure everything is saved and ready. Use `sudo apachectl configtest` to check for syntax errors, then restart your server with `sudo systemctl restart apache2`

8. To run the given Perl files, make sure you have `.pl` in your `AddHandler` . Put the Perl files in your `cgi-bin` directory. Make sure your `cgi-bin` directory is executable by running `sudo chmod a+x cgi-bin`. Now, if you navigate to `yourdomain.site/cgi-bin/path-to-perl-file.pl` (http://yourdomain.site/cgi-bin/path-to-perl-file.pl) , you should have a functional CGI program.

9. To run the given C files, copy all the files to your `cgi-bin` directory. Then, use gcc to compile the files. The best way to do this is

```
gcc c-file-path.c -o c-file-path.cgi
```

   That will generate an executable with a .cgi extension (so your server should already handle it). Make sure your `cgi-bin` directory is executable by running `sudo chmod a+x cgi-bin`. You should now be able to go to `[yourdomain.site/cgi-bin/path-to-c-file.cgi](<http://yourdomain.site/cgi-bin/path-to-c-file.cgi>)` and see your C program output. You should have cc or gcc installed on your local system from the previous homework.

If you complete this and are still having problems, please visit the Apache guide (linked above) *before* reaching out to the teaching staff for help. There is a troubleshooting guide at the bottom of the page.

Part 1: CGI Done 3 Ways

This part of the homework presents an opportunity to prove to ourselves (or not) the Professor's proposition that languages and architectures used in Web programming are quite related to each other. Our small test programs hopefully will keep us from getting too caught up in the syntax and focus on the parts of Web development that are more similar than different.

You will write **each of the following 7 (small) programs** in different languages. If you are a solo or duo you have to write two languages if you are a group of three you must write 3 languages. Many students will opt for PHP because of ease, however, regardless of ease no extra points are assigned so choose wisely. If you complete your required amount you may do more languages for bonus points. Each extra language is worth 1/3 of the points given for a single required language.

Super Bonus: If you are an extremely adventurous student you may apply to write Apache modules in C. You must contact the Professor to do this for approval and you must have performed the required two or three languages beforehand. A student/group accomplishing the Super Bonus can opt out of a quiz with 100%.

Support Warning: Do **NOT** ask the tutors/TAs for syntax help with the languages - that is on you to go figure out on your own 🙂 Part of the assignment is to teach the form of self sufficiency useful when dealing with dev requests you'll face later on. You can work with your peers and share answers which is also skill. This is not to say the teaching staff may not provide tips but they will not debug for you here or be walking StackOverflows.

Choose from the following languages:

- Perl - Given (do not choose this - you will get a 0)
- C - Given (do not choose this - you will get a 0)

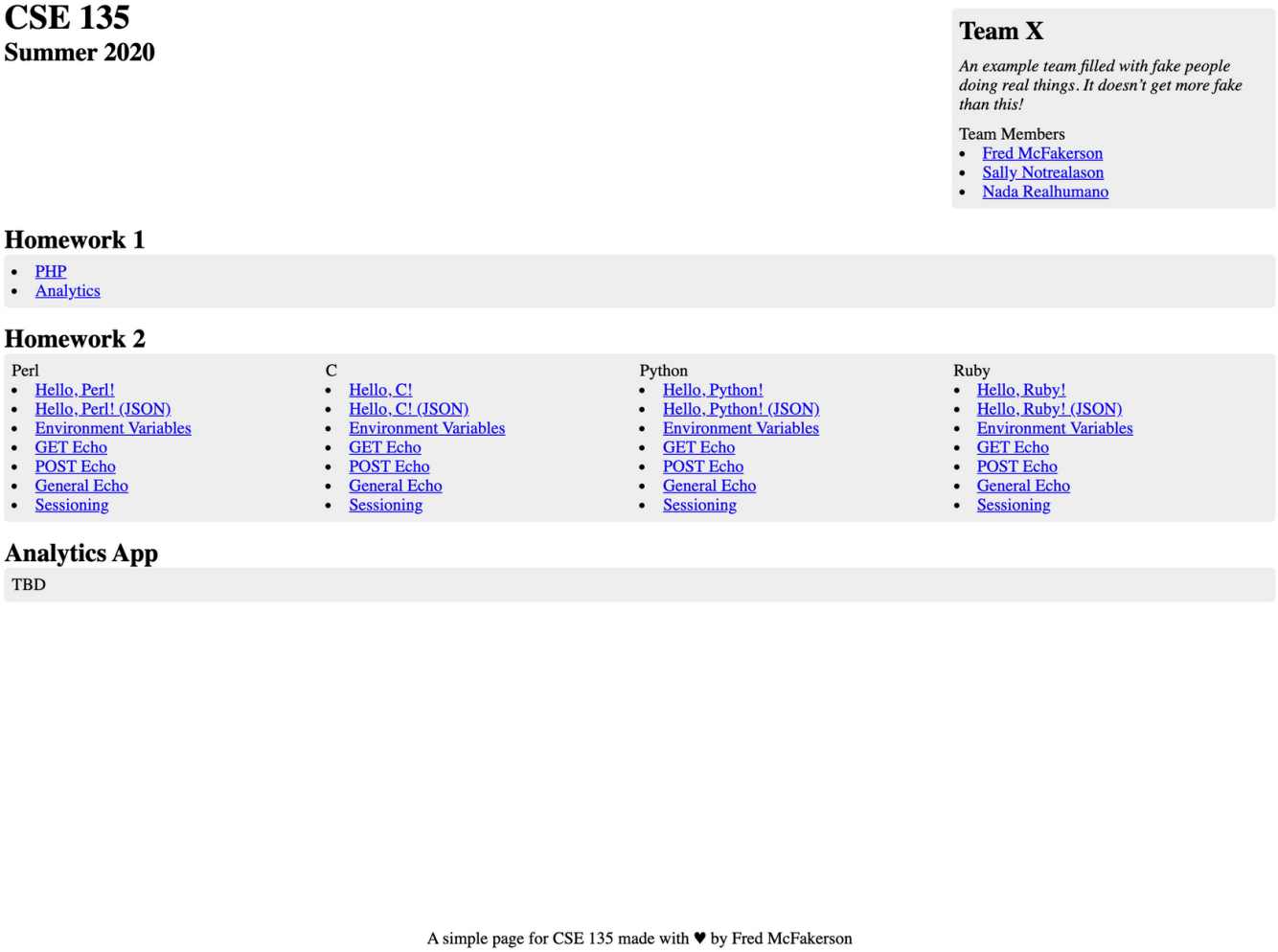
1. PHP
2. NodeJS (without Express - done manually) [must be proxied by Apache!]
3. NodeJS (with Express or similar) [must be proxied by Apache!]
4. JSP (Java)
5. Ruby (Ruby on Rails)
6. Python
7. Go
8. Rust
9. ColdFusion

Write the following programs:

Prefix your file names to keep your directory organized. For example, `php-hello-html-world.php`, `py-hello-html-world.py`, `node-hello-html-world.js`

1. "hello-html-world" – Outputs an html document with a "Hello World" message as well as the current Date/Time and the user's IP address to show that the page was created dynamically.
2. "hello-json-world" - The same as the *hello-html-world* demo, but it returns a JSON packed instead of HTML.
3. "environment" - Outputs an echo of the environment variables of a request (HTTP request headers + Server variables)
4. "get-echo" - Outputs a page with data from the HTTP get request query string
5. "post-echo" - Outputs a page with data from the HTTP post request message body
6. "general-request-echo" - Outputs information of whatever it is sent showing method and payload (DELETE, PUT, HEAD, etc)
7. "state-demo" - On the first page of this demo will be an input for a user to enter their name. This name will be saved in a "session". When the second page of this demo is visited, content from the previous page will be shown via this "session"
    - Three possible ways to implement this: *Cookies*, *Dirty URLs*, or *Hidden Form Fields*
    - **NOTE:** This must be a *SERVER* side session, so do not simply store user info in localStorage. Remember, you cannot trust users, a user could very easily just dump their localStorage.

Create links on your `index.html` page under Homework 2 to link to each of these files on your server. Sort it by language - make sure you use semantic HTML for your link structures (check out lists - or <ol>). Make sure each of the programs can be reached via a link from `index.html`. The graders will *not* grade your programs if they are not reachable from your site homepage. An example of what your homepage should now look like is shown below.



Part 2: Third Party Analytics Done 2 Ways

The applied part of the assignment is demonstrates different approaches to third party analytics. You will go through setting up the most popular script based analytics (Google Analytics) and a powerful replay analytics system (LogRocket). Later assignments may show how we can use these or other services to execute our project in a less "code" focused manner

Step 1: Google Analytics

The first third party analytics system we are going to use is Google Analytics.

Use the Google Analytics Getting Started Guide: **https://support.google.com/analytics/answer/1008015?hl=en# (https://support.google.com/analytics/answer/1008015?hl=en#)** to create or sign in with an account. The

following steps are all enumerated at the above link, but will be provided below with the corresponding links for your ease.

1. Once you log into your account, you need to set up a property - which is essentially just typing in your domain name and clicking Create. A full guide on this is here: **https://support.google.com/analytics/answer/1042508 (https://support.google.com/analytics/answer/1042508)**
2. Next you'll need to add a view, which is also pretty easy to do following these instructions: **https://support.google.com/analytics/answer/1009714 (https://support.google.com/analytics/answer/1009714)** . Make sure you select for a website, and not a mobile app.
3. Then, using the instructions provided here: **https://support.google.com/analytics/answer/1008080 (https://support.google.com/analytics/answer/1008080)** , you're going to add your global site tag to the `<head>` of any HTML document you want to use Google Analytics on. Add it to the bottom of your `<head>` of your index.html. Refresh the page a couple of times and click around - it may take a little while, but your Google Analytics dashboard should start populating with data (if you have an adblocker enabled, it might block your GA script, so try in an incognito window if you aren't seeing data come through).

Screenshot Alert! Take a screenshot of your Google Analytics dashboard (once some data can be seen on it) and call it ga-dashboard.png.

Step 2: LogRocket

For this step, we are going to see and test a live-tracking analytics service called LogRocket. First, go to **logrocket.com (http://logrocket.com)** and click "Get Started Free." Follow the instructions to name your project (feel free to name it yourdomain.site). Use NPM to install LogRocket in your `/var/www/` directory. LogRocket also provides you with code to initialize LogRocket in your JavaScript files. It should look something like :

```
import LogRocket from 'logrocket';
LogRocket.init('<something>');
```

You can get back to this later, but that would be the code to include in your JavaScript files. When you continue to your Dashboard, there are project settings on the left hand side, and you can get into Project Setup, where it provides you with the script tag needed to include LogRocket on your HTML page. It should look something like

```
<script src="<https://cdn.lr-ingest.io/LogRocket.min.js>" crossorigin="anonymous"></script>
<script>window.LogRocket && window.LogRocket.init('<something>');</script>
```

For now, include this at the bottom of your `<head>` tag in `index.html`. Reload your homepage and use DevTools to ensure that LogRocket is functioning correctly, then click and scroll around your page. If you go back to your LogRocket dashboard, there should be a recording of your session.

Screenshot Alert! Take a screenshot of your LogRocket dashboard and label it logrocket.png. Take a screen recording of one of the sessions and save it as either logrocket-session.gif or logrocket-session.mp4

Submission

- **README.md (http://README.md)**
  - a link to yourdomain.site; where under Homework 2, there are links to all of your CGI programs (the programs you/your group wrote, and the C and Perl code we provided to you)
  - any additional notes for the graders
  - **team member names**
  - **your IP address of server, ssh key, grader log in information to the site and server**
- .png files for each of the Perl (10) and C (9) CGI programs (19 total)
- The source code for your programs (the 2 or 3 languages you chose)
- ga-dashboard.png
- logrocket.png
- logrocket-session.gif or logrocket-session.mp4