Pumpkin Spice Latte

Beginner Server Routing

Overview

Simple HTTP + Routing:

- Python
- Ruby
- Java
- Rust
- Node.js

Standard vs Custom Routing

- Currently, when we start our simple HTTP servers, we are taking advantage of the default behavior of browsers that serve up an index.html file (if it exists), otherwise it renders a list of files within the directory.
- If we want to add a path or route, we add another file to that directory.
- There is another way that creates a more custom approach to routing and pathing which opens the door to architecting APIs.
- Typically, developers import packages that allow them to simplify implementing the architected APIs for their respective platform or framework.
- We will be discussing these more in depth as it is a slight waste of time to cover the traditional implementation of routing without packages.
- The only reason to learn traditional routing is to write servers in C++.

Python

Python

- There are a few options we can do with python routing:
    - Flask
    - Django
- We are going to create a file called **requirements.txt** which will install packages onto our machine, so we can import them into our **server.py** file.
- Packages are handled through the packaged manager called pip.

Flask is supposed to be *lightweight* in comparison to Django meaning there is less code that supports its functionalities. However, Django is more robust.

Example

```
$ pip install -r requirements.txt

…

Django==3.2.3

# ...or...

Flask==2.0.0
```

Tutorial

Python

Ruby

Ruby

- Ruby needs to be put on rails in order to be used as a server ergo the name, Ruby on Rails.
- Packages in Ruby are called Gems, and can be installed through the package manager called gem.
- Gems can be standardized for a project via a **setup.rb** file.
- To set up the routes, **config/routes.rb** needs to be available for rails to direct incoming traffic accordingly.

Example:

```
$ gem install rails

…

get '/patients/:id', to: 'patients#show', as: 'patient'

    @patient = Patient.find(params[:id])
```

Tutorial

Ruby

Node.js

Node.js

- The standard package to for routing in Node.js is Express.js.
- You can also create your own package to handle routing that you can register on npm through GitHub.
    - I don’t recommend this, but, if you know how to create parsers really well, it would be a fun challenge to tackle.
- Packages are handled through Yarn or npm using a **package.json** file within Node.js projects.
- Typically, the server is defined in an **app.js**, **index.js**, **main.js**, or **server.js** file. The default is **server.js**.

Example:

```
$ npm i express

...or…

$ yarn add express

var express = require('express');
var app = express();

app.get('/', function (req, res) {
  res.send('hello world')
});
```

Tutorial

Node.js

Java

Java

- There are a few routing libraries that can be imported:
  - Spring
  - Spark
- Both libraries use the compiled version of Java that needs to run as an executable.
- Through implementing the runnable interface, we can filter the requests through regex matching instead of purely serving up a directory.
  - I don't advice doing this for every project as it can cause issues with tech debt later if your project because much larger.
- You'll need to define a **pom.xml** file for the compiled external libraries.

Example:

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>

...or...

<dependency>
    <groupId>com.sparkjava</groupId>
    <artifactId>spark-core</artifactId>
    <version>2.9.3</version>
</dependency>
```

Tutorial

Java

Rust

Rust

- There are a lot of RESTful crates available for Rust:
    - sn_routing
    - yew_router
- Rust using a package manager called Cargo to install crates.
- Cargo requires a *manifest* file that is named **Cargo.toml** in order to install your customized Rust server.
    - Crates are listed under the **[dependencies]** section.
    - **[package]** is where the project's details are listed like the authors, name, and version.

Example:

```
[dependencies]
sn-routing = "0.73.1"

...or...

[dependencies]
yew-router = "0.14.0"
yew = "0.17.0"
```

Tutorial

Rust

Pumpkin Spice Latte Co.

hello@pumpkinspicelatte.org