Practice with Flask Part 1

Estimated time needed: 20 minutes

Welcome to the first lab of the Capstone course. You will practice working with Flask in this lab. You should know all the concepts you need for this lab from the previous set of videos. Feel free to pause the lab and review the module if you are unclear on how to perform a task or need more information.

Learning Objectives

After completing this lab, you will be able to:

- Create and run a Flask server in development mode
- Return JSON from an endpoint
- Understand the request object

About Skills Network Cloud IDE

Skills Network Cloud IDE (based on Theia and Docker) provides an environment for hands on labs for course and project related labs. Theia is an open source IDE (Integrated Development Environment) that can run on desktop or the cloud. To complete this lab, you will be using the Cloud IDE based on Theia and MongoDB running in a Docker container.

Important Notice about this lab environment

Please be aware that sessions do not persist for this lab environment. Every time you connect to this lab, a new environment is created for you. Any data you save in earlier sessions will be lost. Plan to complete these labs in a single session, to avoid losing your data.

Set Up the Lab Environment

There are some prerequisite preparations required before you start the lab.

Open a Terminal

Open a terminal window using the menu in the editor: **Terminal** > **New Terminal**.

In the terminal, if you are not in the `/home/project` folder, change to your project folder now.

```
cd /home/project
```

Create the lab directory

You can now create a directory for your server file.

```
mkdir lab
```

Change into the `lab` directory:

```
cd lab
```

Check Python version and install Flask

Use the `python3 --version` command to check the version of python3 in the lab enviornment. You should see an output as follows:

```
theia@theiadocker-captainfedo1:/home/project/lab$ python3 --version
Python 3.10.12
```

Next, install Flask version 2.2.2 using the following command:

```
pip install "Flask==2.2.2"
```

If Flask is present on the system, you will see the following message:

```
Requirement already satisfied: Flask==2.2.2 in /home/theia/.local/lib/python3.8/site-packages (2.2.2)
Requirement already satisfied:
...
```

You are now ready to start the lab.

Optional

If working in the terminal becomes difficult because the command prompt is long, you can shorten the prompt using the following command:

```
export PS1="[\[\033[01;32m\]\u\[\033[00m\]: \[\033[01;34m\]\W\[\033[00m\]]\$ "
```

Step 1: Create the Hello World server

Your Tasks

1. Create server.py file.

   First, create an empty file called `server.py` in the terminal or use the file editor menu.

▶ Click here for a hint.

Open `server.py` in the editor

Open **server.py** in IDE

If a new tab called `Python - Get Started` displays after opening this file, you can close it to return to the python file.

2. Import Flask module.

   Next, import the Flask module in this file so you can start coding the server.

▶ Click here for a hint.

3. Create the Flask app

   After importing the Flask module, create your Flask application by initializing the Flask class.

▶ Click here for a hint.

4. Create the main route.

   You can now use the app you created in the previous task to create your first route.

▶ Click here for a hint.

5. Define the method for the main root URL.

   First import Flask in this file.

▶ Click here for a hint.

6. Return the “Hello World” message to the client.

   Return the string “Hello World” to the client.

▶ Click here for a hint.

You are all set to run the server. Use the following command to run the server from the terminal:

```
flask --app server --debug run
```

You should now be able to use the CURL command on `localhost:5000/`. Note that the terminal is already running the server, you can use the `Split Terminal` button to split the terminal and run the following command in the second tab.

Note: Kindly verify the presence of the `Server.py` file in the /home/project/lab directory to prevent encountering a connection refusal error.

```
curl -X GET -i -w '\n' localhost:5000
```

The `-X` argument specifies the `GET` command, and the `-i` argument displays the header from the response.

You should see `Hello World` returned as the output of the CURL command. Note the return status of `HTTP 200 OK` and the `Content-type` of `text/html`. You are asked to return a custom status with JSON instead of plain text in the next part of this lab.

Solution

Double-check that your work matches the solution below.

▶ Click here for the answer.

Step 2: Return JSON

Your Task

Congratulations on creating your first route handler in the Flask server. You can return a number of different content types from the `@app.route()` methods. For the purpose of this project, let's return the following JSON instead of the **Hello World** string.

```
"message": "Hello World"
```

Recall from the videos that there are two ways to return a JSON object from the method:

1. Return a Python dictionary
2. Use the **jsonify()** method on a string

You are being asked to use the first method in this lab.

Hint

You can edit the existing **index** method to return the desired JSON message.

▶ Click here for a hint.

Solution

Double-check that your work matches the following solution.

▶ Click here for the answer.

If you have the server running, you are good to go. If not, you can run the server with the following command again:

```
flask --app server --debug run
```

You should now be able to use the CURL command with `localhost:5000/`. Note that the terminal is running the server, you can use the `Split Terminal` button to split the terminal and run the following command in the second tab.

```
curl -X GET -i -w '\n' localhost:5000
```

You should see `{"message": "Hello World"}` JSON returned as the output of the CURL command. Note the return status of `HTTP 200 OK` and the `Content-type` of `application/json` this time.

Author(s)

CF

© IBM Corporation 2023. All rights reserved.