University of Puerto Rico
Mayagüez Campus
Mayagüez, Puerto Rico
Department of Electrical and Computer Engineering

Programming Languages Assignment 6 - Server Language

by
Alberto Perez Roman
Noel Valentín Roman

For: Prof. Wilson Rivera
Course: Programming Languages

Assignment 6 - ServLang: A Server Language

This project serves as a programming language that facilitates communication between devices by creating local servers and allowing communication with external servers.

Introduction

In recent years, communication of servers between devices has become an increasingly popular trend, with Alexa and Google home. Creating servers, routes and data could be time-consuming and irritating; this program seeks to simplify all of this. Using this program, all these processes can be done with a minimal amount of lines of code. Each line focuses on a specific task, and takes care of it. Each task can be executed with just one line of code: creating a server, adding a route, reading data and starting the servers, among other actions.

For simplicity and to speed up the process, as well as to test the methods, the user could also write a group of lines of code in the main.py file so that the system executes them all at once without the need of the constant input in the program by executing the RunClass file.

Video Demo

Video demonstrating the program and the parser can be seen here.

This video demonstrates the successful execution and creation of the parser, as well as the actions executed successfully through the main.py method, for simplicity. This sample method creates servers, routes and data, updates data, and successfully starts the server.

Technologies

- Python 3.7 or higher - Interpreted, high-level, general-purpose programming language
- certifi - Collection of Root Certificates for validating the trustworthiness of SSL certificates while verifying the identity of TLS hosts
- chardet - Universal Character Encoding Detector

- Click - Python package for creating command line interfaces in a composable way
- Flask - Python web framework built with a small core and easy-to-extend philosophy. Creates server.
- idna - Support for the Internationalised Domain Names in Applications (IDNA) protocol as specified in RFC 5891
- itsdangerous - Various helpers to pass data to untrusted environments and to get it back safely
- Jinja2 - A fast, expressive, extensible templating engine
- MarkupSafe - Implements a text object that escapes characters so it is safe to use in HTML and XML
- ply - Implementation of lex and yacc for Python
- requests - The only Non-GMO HTTP library for Python
- urllib3 - A package that collects several modules for working with URLs
- Werkzeug - A comprehensive Web Server Gateway Interface (WSGI) web application library

Installation

Step 1 - Creating a virtual environment

A virtual environment is recommended for the use of this program. To create a virtual environment, the command line can be used (if using an IDE such as PyCharm, the Command line is found on the Terminal section of the interface).

First, the user must make sure that they have the virtualenv package installed in Python; this package manages virtual environment creation within Python. To install the virtualenv package, the user must run the following:

```
python -m pip install virtualenv
```

Now that the package is installed, the user can now create virtual environment. To create a virtual environment called venv, the user would have to run the following:

```
python -m virtualenv venv
```

It is important to note that after creating the virtual environment, the virtual environment must be activated each time a new terminal is started. The command used to activate the recently created venv is the following:

- Windows

```
venv\Scripts\activate
```

- MacOS/Ubuntu

```
source venv/bin/activate
```

Step 2 - Installing Dependencies

In order to successfully run the application made, the dependencies described in the Technologies section must be installed into the computer. This would involve utilizing the pip install command in the command line, used for installing packages. To simplify the installation process and avoid having to execute a significant amount of installation commands, the requirements.txt file was created. By executing this file, all of the dependencies needed to run the program will be installed. In other words, the only thing needed to have all the dependencies installed is to run the requirements.txt file included in the project folder. To execute this file, run the following:

```
pip install -r requirements.txt
```

Language Syntax

The API is as follows:

- createServer(port=3000) - Initializes the server and assigns port
- json : {string:string} - Creates JSON object
- serverInstance : setRoutes(url=string) - Creates route instance
- routeInstance : createData(object=data) - Creates data
- routeInstance: readData(body=data) - Reads data
- serverInstance : start - Starts running server
- test = httpGet(from=string) - Gets post request
  Note: the url being used is https://reqres.in/api/users?page=2

Coding Example:

```
//if port parameter is empty by default is 80
server = createServer (port= 3000);

//Create variable for data storage
data = json: {};
data2 = json: {"name": "John", "lastName": "Doe", "age": "67"};

//Server receives
send = server: setRoutes(url= "/app/send");

//uses the request body in a JSON format
send: createData(object= data);  //object: object to save into
```

```
//Server sends
get = server: setRoutes(url= "/app/get");
get: readData(body= data); //data in JSON format

server: setRoutes(url= "/empty"): readData(body= data2);
server: setRoutes(url= "/empty"): createData(object= data2);

server: start;

//Communicating with other server
test = httpGet(url= "https://reqres.in/api/users?page=2");
//receives JSON from another server
```

Language Specifications

After installing the program by following the instructions shown in the Installation section, the user could either create a group of executable lines of code in main.py and run that method, or run the RunClass.py file and execute each line one by one. The syntax and the executable actions are defined in the Language Syntax section.

Authors and Acknowledgement

- Noel Valentin - noelvalentin
- Alberto Perez - AlbertoPerez8