Midterm Project Report

Advanced Computer Programming

Student Name : Tselmeg Gantulga

Student ID : 113021201

Teacher : DINH-TRUNG VU

2024-04-14

Chapter 1: Introduction

1.1 GitHub

1. Personal GitHub Account: https://github.com/113021201

2. Group Project Repository: https://github.com/113021201/camtonolzhon

1.2 Overview

This project uses Python and GitHub's REST API to fetch detailed information about repositories hosted under a GitHub user account. The extracted information includes repository URL, description, last updated date, languages used, and number of commits. The project uses advanced features like object-oriented programming (OOP) with classes to modularize the code. The main classes used are GitHubAPI (for interacting with the GitHub API) and Repository (for storing repository data). These classes help make the code more organized, reusable, and maintainable.

Key libraries used in the project:

- **requests**: For making HTTP requests to the GitHub API.
- **xml.etree.ElementTree**: To create and write XML data.
- **google.colab.files**: To enable downloading the generated XML file in Google Colab.

Chapter 2: Implementation

Setup / Environment Preparation

Before running the main Python code, I had to install some external libraries. Since I was using **Google Colab**, I used the following commands:

```
# Install Scrapy
!pip install scrapy
```

The scrapy library was installed in case I needed to use it for scraping GitHub pages. However, in the final version of the project, it was not used because all the data was accessed through the GitHub API.

```
# Install requests
!pip install requests
```

The requests library was important for the project. It allowed my program to connect to GitHub and get data like repositories, languages, and commit history using simple HTTP requests.

Installing these libraries was only required once in Google Colab. After that, I could run the rest of the code smoothly.

2.1 Class 1: GitHubAPI This class connects to GitHub's API and retrieves repository data, languages, and commits.

```python
# GitHubAPI Class to interact with GitHub
class GitHubAPI:
    def __init__(self, username, token):
        self.username = username
        self.token = token
        self.headers = {"Authorization": f"token {token}", "Accept": "application/vnd.github.v3+json"}

    def get_repos(self):
        repos_url = f"https://api.github.com/users/{self.username}/repos"
        return requests.get(repos_url, headers=self.headers).json()

    def get_languages(self, repo_name):
        languages_url = f"https://api.github.com/repos/{self.username}/{repo_name}/languages"
        return requests.get(languages_url, headers=self.headers).json()

    def get_commits(self, repo_name):
        commits_url = f"https://api.github.com/repos/{self.username}/{repo_name}/commits"
        return requests.get(commits_url, headers=self.headers).json()
```

2.1.1 Fields

- username: GitHub username.
- token: GitHub personal access token.
- headers: Headers for authentication and API version.

2.1.2 Methods

- get_repos(): Returns the user's repositories.
- get_languages(repo_name): Returns the languages used in a repository.
- get_commits(repo_name): Returns all commits in a repository.

2.2 Class 2: Repository This class stores the data of a repository and converts it into XML.

```python
#  Repository Class to store data
class Repository:
    def __init__(self, url, about, last_updated, languages, num_commits):
        self.url = url
        self.about = about
        self.last_updated = last_updated
        self.languages = languages
        self.num_commits = num_commits

    def to_xml(self):
        repo_elem = ET.Element("repository")
        ET.SubElement(repo_elem, "url").text = self.url
        ET.SubElement(repo_elem, "about").text = self.about
        ET.SubElement(repo_elem, "last_updated").text = self.last_updated
        ET.SubElement(repo_elem, "languages").text = self.languages
        ET.SubElement(repo_elem, "number_of_commits").text = str(self.num_commits)
        return repo_elem
```

2.2.1 Fields

- url: Repository URL.
- about: Description or name.
- last_updated: Last updated date.
- languages: Programming languages used.
- num_commits: Number of commits.

2.2.2 Methods

- to_xml(): Converts all the repository info into an XML format.

2.3 Function 1: GitHubAPI.get_repos() this function returns all the public repositories of user from GitHub.

```python
def get_repos(self):
    repos_url = f"https://api.github.com/users/{self.username}/repos"
    return requests.get(repos_url, headers=self.headers).json()
```

2.4 Function 2: GitHubAPI.get_languages() this function gets the languages used in a specific repository.

```python
def get_languages(self, repo_name):
    languages_url = f"https://api.github.com/repos/{self.username}/{repo_name}/languages"
    return requests.get(languages_url, headers=self.headers).json()
```

2.5 Function 3: GitHubAPI.get_commits() this function fetches all commits of the given repository.

```
def get_commits(self, repo_name):
    commits_url = f"https://api.github.com/repos/{self.username}/{repo_name}/commits"
    return requests.get(commits_url, headers=self.headers).json()
```

2.6 Function 4: Repository.to_xml() it converts the repository information into a clean XML element.

```
def to_xml(self):
    repo_elem = ET.Element("repository")
    ET.SubElement(repo_elem, "url").text = self.url
    ET.SubElement(repo_elem, "about").text = self.about
    ET.SubElement(repo_elem, "last_updated").text = self.last_updated
    ET.SubElement(repo_elem, "languages").text = self.languages
    ET.SubElement(repo_elem, "number_of_commits").text = str(self.num_commits)
    return repo_elem
```

2.7 Function 5: main() this function runs the whole script. It gets the data using GitHubAPI, fills Repository objects, creates the XML structure, and downloads the file.It does the following steps:

2.7.1 Initializes the GitHubAPI class

```
def main():
    # Initialize GitHubAPI class
    github_api = GitHubAPI(username, token)
```

This part initializes the GitHubAPI class, which handles communication with GitHub's API and retrieves data for repositories.

2.7.2 Calls the methods to retrieve repository information like languages and commits

```
# Get repository data
repos_resp = github_api.get_repos()
```

The function calls the get_repos(), get_languages(), and get_commits() methods of the GitHubAPI class to fetch repository data, languages used, and the number of commits for each repository.

```
# Get languages and commits
languages_resp = github_api.get_languages(repo['name'])
languages_text = ', '.join(languages_resp.keys()) if languages_resp else "No languages detected"

commits_resp = github_api.get_commits(repo['name'])
num_commits = len(commits_resp)
```

2.7.3 Creates Repository objects to store each repository's details

```
    # Create Repository instance
    repository = Repository(
        url=repo.get("html_url"),
        about=about_text,
        last_updated=repo.get("updated_at"),
        languages=languages_text,
        num_commits=num_commits
    )
```

For each repository, a Repository object is created, storing details such as the repository URL, description, last updated time, languages used, and the number of commits.

2.5.4 Converts the repository details into XML format and saves them to a file

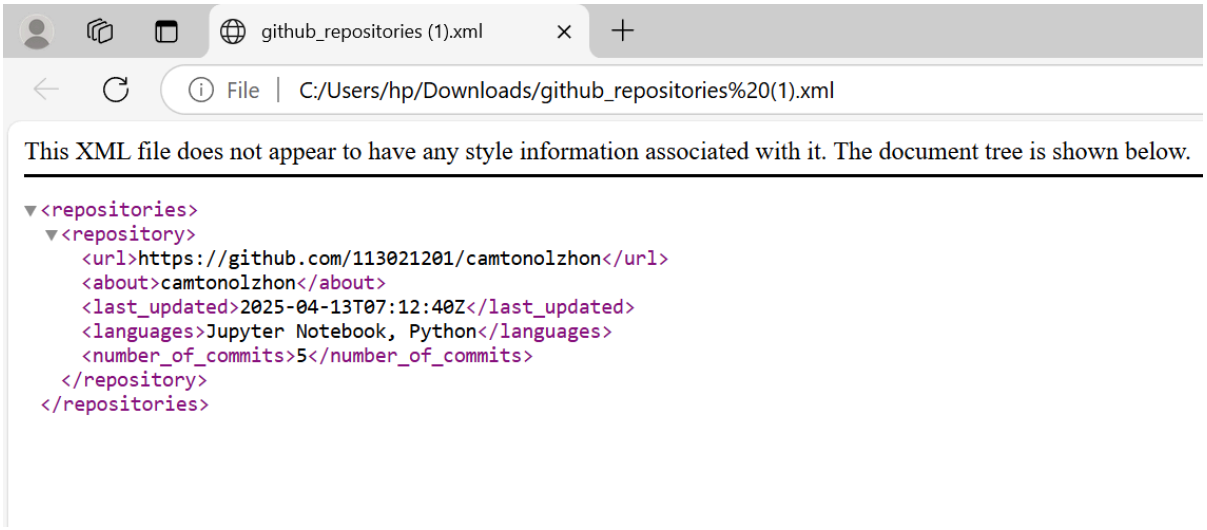
```
        # Add the repository to the XML structure
        root.append(repository.to_xml())

    #  Save to XML file
    tree = ET.ElementTree(root)
    tree.write("github_repositories.xml")

    # Download XML file
    files.download('github_repositories.xml')

# Run the main function
main()
```

Each Repository object is converted into XML format using the to_xml() method, and the resulting XML data is saved to a file named **github_repositories.xml.**

After the XML file is created, the function triggers the file download process.

Chapter 3: Results

After running my Python script, I successfully created an XML file named github_repositories.xml. This file contains all the repositories from my GitHub account.



The data was collected using the GitHub API, processed with Python, and formatted into XML using xml.etree.ElementTree. The final file was downloaded automatically in Google Colab.

The results showed that the program works correctly for different repositories. It handled cases where the description was missing and listed the correct languages and commit counts for each repository. The XML structure was clean and valid.

Chapter 4: Conclusion

In this work, I created a Python program that connects to GitHub, gets information about my public repositories, and saves it into an XML file. To make the code more organized, I used two classes: `GitHubAPI` and `Repository`. One class handles talking to GitHub and getting data, and the other helps store that data and turn it into XML.

By doing this project, I practiced using APIs, writing functions, and organizing my code using classes. I also learned how to work with different types of data like JSON and XML. This project helped me understand how to write better and cleaner Python code, and it gave me real experience working with online data and turning it into something useful.