DBI Lab 009: Python and Flask

COMP1048 - Databases and Interfaces

Matthew Pike and Yuan Yao

Task 1: Hello World!

In this task, you will create a simple Flask application that displays "Hello World!" on the screen. When the user navigates to the root URL of the application, the application will display "Hello World!" on the screen. There is no need to use templates,HTML or CSS in this task. Simply return the string "Hello World!".

- **Endpoint**: `http://127.0.0.1:5000/` (root URL)
- **Output**: `Hello World!` in plain text

> **Running your Code**
>
> To run your solution, enter: `python file.py` into your terminal, where `file.py` is the name of the Python source file you are editing.

Task 2: Hello `<name>`

In this task, you will extend the application you created in Task 1 to display "Hello `<name>`!", where `<name>` is some name specified by a URL variable. For example, if the user navigates to the URL `hello/Alice`, the application will display "Hello Alice!" on the webpage. Again, there is no need to use templates, HTML or CSS in this task. Simply return the string "Hello `<name>`!".

- **Endpoint**: `http://127.0.0.1:5000/hello/<name>`
- **Output**: `Hello <name>!` in plain text, where `<name>` is the name specified by the URL variable.

Task 3: Hello `<name>` using a template

Modify the application you created in Task 2 to return a HTML webpage using a template. The functionality of the `/hello/<name>` should not change from that in task 2. The template should be stored in a file called `hello.html` in the `templates` folder. The template should be rendered using the `render_template` function.

- **Endpoint**: `http://127.0.0.1:5000/hello/<name>`
- **Output**: `Hello <name>!` in **HTML**. The exact structure of the HTML is up to you, but it should be valid HTML.

Task 4: Adding style

Modify the application you created in Task 3 to use CSS. The CSS should be stored in a file called `style.css` in the `static` folder. The CSS should be applied to the template you created in Task 3. The CSS file should be linked using the `url_for` function. The CSS rules themselves are not important, and you can use any CSS rules you like. Do not spend more than 5 minutes on writing the CSS rules.

> **Template Inheritance**
> You may find it useful to use template inheritance in this task, as it will simplify tasks 5 and 6.

Task 5: Retrieving data from a database

Before beginning this task, create an SQLite database called `people.db`. The database should contain a table called `people` with the following columns:

- `id` - an integer primary key
- `name` - a text column, not null
- `age` - an integer column, not null

The database should contain the following data:

id	name	age
1	Alice	20
2	Bob	21
3	Charlie	22

Your SQL script should be stored in a file called `people.sql`.

Modify the application you created in Task 4 to retrieve the data from the database and display it on the screen. Your web application should display all columns and all rows of the database on the webpage, when the user navigates to the `/people/` url of your application. You should use a template to display the data. The template should be stored in a file called `people.html` in the `templates` folder. The template should be rendered using the `render_template` function. Use the `url_for` function to link to the CSS file you created in Task 4. You may add additional CSS rules to the CSS file, if you wish, but again do not spend more than 5 minutes on writing these CSS rules.

- **Endpoint**: `http://127.0.0.1:5000/people/`
- **Output**: An HTML table containing all the people stored within the `people` database table.

Task 6 - Add Items to a Database

Modify the application you created in Task 5 to allow the user to add new people to the database. The user should be able to add a new person to the database by navigating to the `/add/` url of your application. The user should be able to enter the name and age of the person, and the person should be added to the database. The user should be redirected to the `/people/` url of your application, where they can see the new person in the database. You should use a template to display the form. The template should be stored in a file called `add.html` in the `templates` folder. The template should be rendered using the `render_template` function.

- **Endpoint**: `http://127.0.0.1:5000/add/`
- **Output**: An HTML form to add a new person to the database.

Submission

Please submit a ZIP file containing the final version of your web application. Your zip should be named according to your student id e.g. `12345678.zip`. Do not use alternative compression formats such as `.rar` or `.7z`.

Your zip should contain the following directory structure:

```
<student-id>    # The root directory should be named after your student id
├── app.py      # The main application file
├── people.sql  # The SQL file used to create the database
├── people.db   # The database file
├── static      # The static folder
│   └── style.css   # The CSS file
└── templates   # The templates folder
    ├── base.html   # The base template from which all other templates inherit
    ├── add.html   # Template for the add person form page (Task 6)
    ├── hello.html # Template for the hello name page (Task 3, Task 4)
    └── people.html    # Template for the people page (Task 5, Task 6)
```

Submitting this assignment will contribute 2% to your overall module grade. Your submission should demonstrate reasonable effort and fulfil the specified requirements set out in this lab sheet in order to receive the full marks.

There is no granularity to the marking, the marking is on a pass-or-fail basis.

Registration is reported to Faculty office on a weekly basis. The submission point is available on Moodle.

Submission Deadline - Friday, 5 December 2022 @ 15:00