

Web Application Development with Python L-4

Training Program

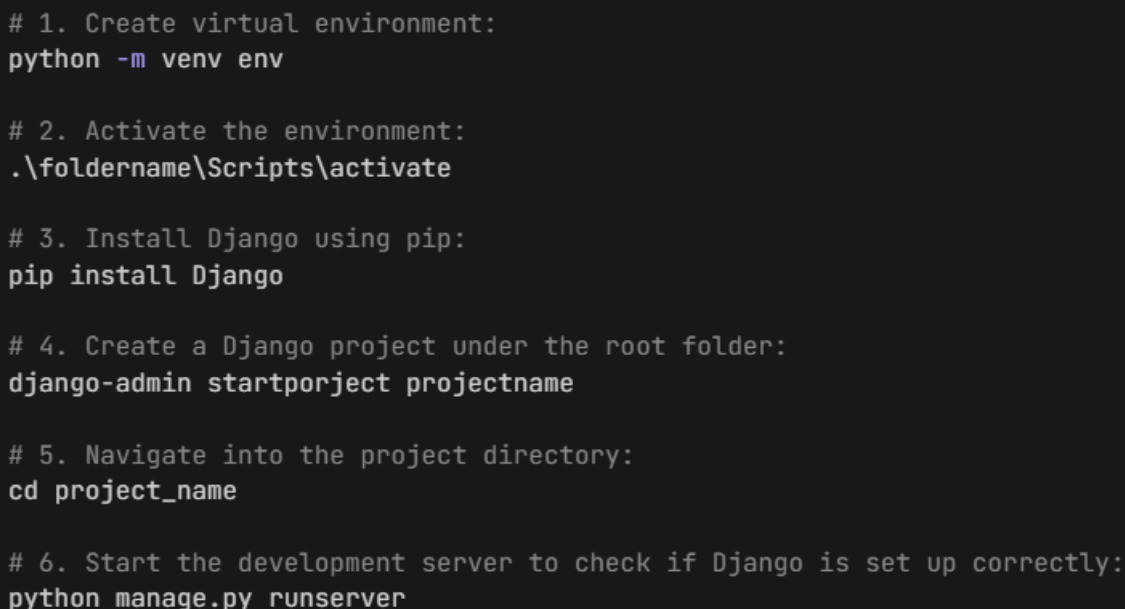
Written Question and Answer

Qs1. What is Django?

Django is a high-level web framework written in Python that allows developers to quickly build secure, scalable, and feature-rich web applications. Django makes it easier to automate repeated operations, resulting in a more efficient development process with fewer lines of code.

Qs2. How do you create a new project in Django?

To create a new Django project, follow these steps:

A terminal window with a dark background and light-colored text. The title bar shows three colored circles (red, yellow, green) and a tab labeled 'cmd.sh'. The terminal contains six numbered steps for setting up Django, each followed by a code snippet.

```
# 1. Create virtual environment:
python -m venv env

# 2. Activate the environment:
.\foldername\Scripts\activate

# 3. Install Django using pip:
pip install Django

# 4. Create a Django project under the root folder:
django-admin startproject projectname

# 5. Navigate into the project directory:
cd project_name

# 6. Start the development server to check if Django is set up correctly:
python manage.py runserver
```

Qs3. Which file is responsible for defining the database settings for a Django project?

Ans: The settings.py file is responsible for defining the database settings for a Django project. In this file, we can find the DATABASES configuration where you specify details such as the database engine, name, user, password, host, and other settings.

Qs4. What is ORM in Django, and how does it work?

ORM (Object-Relational Mapping) in Django is a tool that allows developers to interact with the database using object-oriented programming rather than writing raw SQL queries. Django's ORM automatically generates SQL code based on the defined models, enabling developers to interact with the database through Python objects.

- For example, when creating a model class in Django, it is automatically mapped to a database table. Instead of writing SQL queries, developers use Python methods to create, update, delete, and retrieve data from the database.

Qs5. Explain the purpose of the django admin site?

The Django admin site is a built-in web-based interface that provides a powerful and customizable administrative interface for managing a Django web application.

It provides functionalities like:

- Creating, editing, and deleting data records.
- Viewing all data objects for a specific model.
- Searching and filtering data based on various criteria.

Registering new models to be accessible in the admin interface.

Qs6. What is the purpose of a URL pattern in django?

The purpose of a URL pattern in Django is to map URLs to views. It defines the structure of the URL and specifies which view function or class-based view should handle the corresponding HTTP request.

Example:



```
1 from django.urls import path
2 urlpatterns = [
3     path("", function_name, name="path_name"),
4 ]
```

Qs7. What is the difference between a Project and an App?

A Project is the entire Django web application, including settings and configurations. An App is a modular part of the project that performs a specific function (e.g., blog, shop).

In Django, a **Project** is the entire website or application that serves as a container for multiple **Apps**, handling global configurations like **settings, URLs, and middleware** (the project is created using **django-admin startproject**), while an **App** is a modular, reusable component focused on a specific functionality (e.g., **user authentication or blog posts**) with its own models, views, and templates (the apps are created using **python manage.py startapp**).

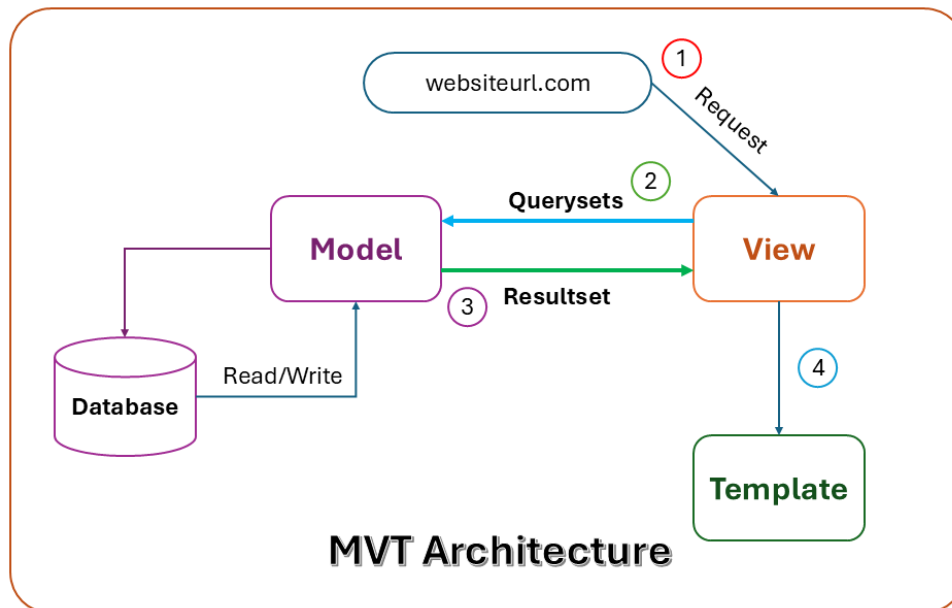
Qs8. Explain the Django MVT architecture.

Django is based on MVT (Model-View-Template) architecture. MVT is a software design pattern for developing a web application.

MVT Structure has the following three parts –

- **Model:** It is a data access layer which handles the data. It Defines the structure of the database and handles database operations. Each model maps to a single database table. Django's ORM (Object-Relational Mapping) translates model instances into database queries.

- **View:** Views contain the logic to determine what data to present and which template to use for rendering that data. It accepts a Web request and delivers a Web response. Views in Django are part of the user interface in a Django application because they return the web pages which contain HTML/CSS/JAVASCRIPT in the template of the Django.
- **Template:** Template in Django contains the static content of a Django project like Html, CSS, and Javascript, along with the image used in the project.



Qs9. Which file is responsible for defining the database settings for a django project?

The settings.py file is responsible for defining the database settings for a Django project. This file is located within the main project directory. Inside settings.py, we'll find a section where we can configure various settings related to the database, including the database engine, database name, user, password, host, and port. These settings are usually specified in a dictionary named DATABASES. Here's a typical example:

```

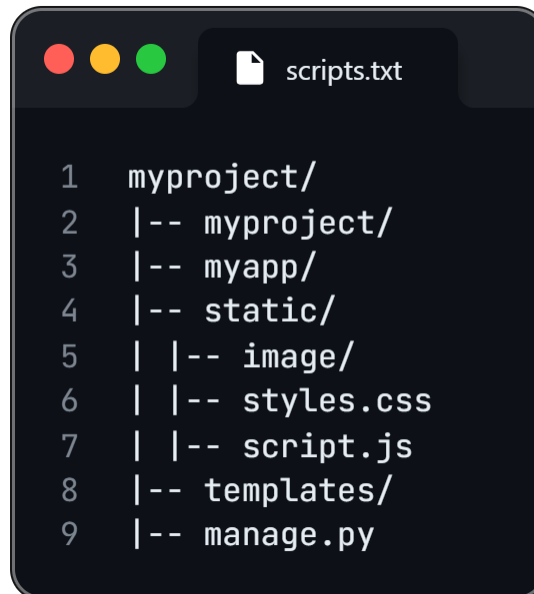
1 DATABASES = {
2     'default': {
3         'ENGINE': 'django.db.backends.sqlite3',
4         'NAME': BASE_DIR / 'db.sqlite3',
5     }
6 }

```

In this example, Django is configured to use SQLite as the database engine, and the database file is db.sqlite3, which is located in the project's base directory. However, depending on your project requirements, you might configure Django to use other database engines like PostgreSQL, MySQL, or others, and the settings in DATABASES would reflect those choices.

Qs10. Where should you store your static files like CSS and JavaScripts in a Django project?

In Django Project, static files such as CSS, image, fonts and JavaScripts files are store 'static' folder in project main directory or within each app. Use {% static %} template tag to reference them in templates.

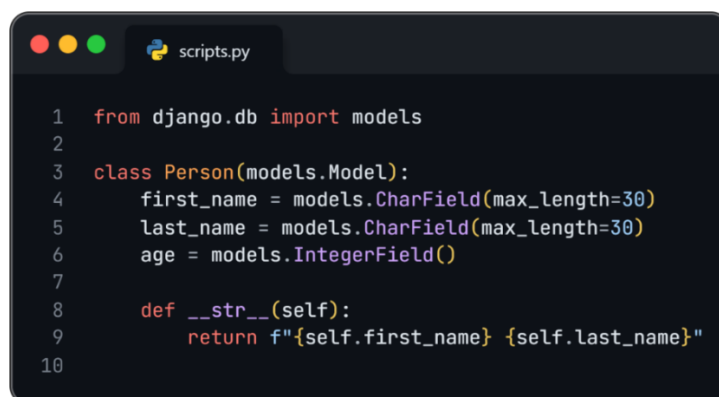


```
1 myproject/
2 |-- myproject/
3 |-- myapp/
4 |-- static/
5 | |-- image/
6 | |-- styles.css
7 | |-- script.js
8 |-- templates/
9 |-- manage.py
```

Qs11. How do you create a new database record (object) using Django ORM?

To create a new database record (object) using Django's ORM (Object-Relational Mapping), we typically follow these steps:

- Import the model class representing the database table where we want to create a new record.



```
1 from django.db import models
2
3 class Person(models.Model):
4     first_name = models.CharField(max_length=30)
5     last_name = models.CharField(max_length=30)
6     age = models.IntegerField()
7
8     def __str__(self):
9         return f"{self.first_name} {self.last_name}"
10
```

- Create an instance of the model class, setting the desired field values.



```
1 from myapp.models import Person
2
3 new_person = Person.objects.create(first_name="John", last_name="Doe", age=28)
4
```

- Call the save() method on the instance to persist it to the database.



```
1 from myapp.models import Person
2
3 new_person = Person(first_name="John", last_name="Doe", age=28)
4 new_person.save()
```

Qs12. What are the key features of Django?

Key features include:

- Object-Relational Mapping (ORM)
- Automatic admin interface
- URL routing
- Template engine
- Form handling
- Authentication system
- Caching framework
- Internationalization support

Qs13. Explain the Django project directory structure.

When you first start a Django project, it comes with some basic files like `manage.py` and `view.py`.

1. **init.py** - It's an empty Python file. It is called when the package or one of its modules is imported. This file tells the Python interpreter that this directory is a package and that the presence of the `__init.py_` file makes it a Python project.
2. **manage.py** - This file is used to interact with your project from the command line utility. with the help of this command, we can manage several commands such as:
 - `manage.py runserver`
 - `manage.py makemigration`
 - `manage.py migrate` etc
3. **setting.py** - It is the most important file in Django projects. It holds all the configuration values that your web app needs to work, i.e. pre-installed, apps, middleware, default database, API keys, and a bunch of other stuff.
4. **views.py** - The View shows the user the model's data. The view knows how to get to the data in the model, but it has no idea what that data represents or what the user may do with it.
5. **urls.py** - It is a universal resource locator which contains all the endpoints, we store all links of the project and functions to call it.
6. **models.py** - The Model represents the models of web applications in the form of classes; it contains no logic that describes how to present the data to a user.
7. **wsgi.py** - WSGI stands for Web Server Gateway Interface, This file is used for deploying the project in WSGI. It helps communication between your Django application and the web server.
8. **admin.py** - It is used to create a superuser Registering model, login, and use the web application.
9. **app.py** - It is a file that helps the user to include the application configuration for their app.

Qs14. Importance of virtual environment setup for Django.

A virtual environment allows you to establish separate dependencies of the different projects by creating an isolated environment that isn't related to each other and can be quickly enabled and deactivated when you're done.

It is not necessary to use a virtual environment without a virtual environment we can work with Django projects. However, using `virtualenv` is thought to be the ideal practice. Because it eliminates dependencies and conflicts.

Qs15. Give a brief about the Django admin interface.

Django provides us with a default admin interface that is very helpful for creating, reading, updating, and deleting model objects that allow the user to do administrative tasks. It reads a set of data that explains and provides information about data from the model in order to create a quick interface where the user can alter the application's contents. This is an in-built module.

Qs16. What do the following commands do?

- `python manage.py makemigrations`
- `python manage.py migrate`

The **makemigration** command scans the model in your application and generates a new set of migrations based on the model file modifications. This command generates the SQL statement, and when we run it, we obtain a new migration file. After running this command, no tables will be created in the database.

Running **migrate** command helps us to make these modifications to our database. The migrate command runs the SQL instructions (produced by makemigrations) and applies the database changes. After running this command, tables will be created.

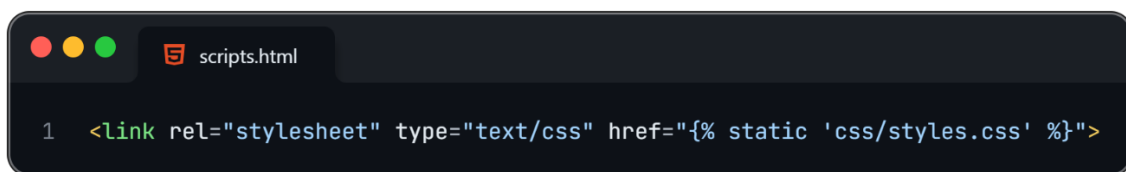
Qs17. What files are created when you start a new app?

A new app contains:

- **admin.py** (for admin configuration)
- **apps.py** (app configuration)
- **models.py** (for database models)
- **tests.py** (for tests)
- **views.py** (for view functions)

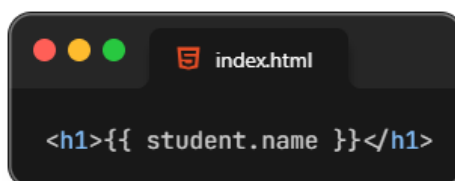
Qs18. What is a static file URL in web development?

In web development, a static file URL refers to the web address (URL) used to access static files such as images, stylesheets (CSS), JavaScript files, fonts, and other assets that do not change dynamically based on user input or server-side processing.



Qs19. How to display dynamic data in templates?

Use `{{ variable }}` in HTML.



Qs20. How do you include an app in your Django project?

Add the app name to `INSTALLED_APPS` in `settings.py`

