1. What is Virtualenv? Steps to Create a Virtual Environment

Answer:

Virtualenv is a tool used to create isolated Python environments. It allows developers to manage dependencies for different projects separately, ensuring that packages required for one project do not interfere with another.

1. Steps to create a virtual environment:

- Install virtualenv (if not already installed): pip install virtualenv
- Navigate to your project directory.
- Create a virtual environment: virtualenv env_name
- Activate the environment:
- - On Windows: env_name\Scripts\activate
- - On macOS/Linux: source env_name/bin/activate
- Deactivate the environment when done: deactivate

2. What Does MVT Stand for in Django? Explain.

Answer:

MVT stands for Model-View-Template. It is a software design pattern used by Django to separate concerns within an application:

- Model: Represents the data structure and handles the database. It defines the schema of the database tables.
- View: Contains the business logic and processes user requests. It retrieves data from the model and passes it to the template.
- Template: Deals with the presentation layer. It defines how data is displayed to the user using HTML.

3. Explain Request-Response Cycle in Django

Answer:

The request-response cycle in Django refers to the process that occurs when a user interacts with a web application:

4. 1. The user sends an HTTP request (e.g., by visiting a URL).
5. 2. The request is routed by the URLconf (urls.py) to the appropriate view function.
6. 3. The view function processes the request (possibly interacting with models).
7. 4. The view returns an HTTP response, typically rendering a template.
8. 5. The response is sent back to the user's browser and displayed.

4. What is the Importance of urls.py in Django Project?

Answer:

The urls.py file is crucial in a Django project as it handles URL routing. It maps URLs to their corresponding view functions, ensuring that the right logic is executed when a particular web address is requested.

- Key roles of urls.py:
- Acts as a directory of all application URLs.
- Enables clean, readable, and SEO-friendly URLs.
- Facilitates modularity by including app-specific URLs using include().