Web Application Development with Python L-4
Training Program
Written Question and Answer

Qs1. How do you create a new project in Django?

To create a new Django project, follow these steps:

cmd.sh

```
# 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 startporject 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
```

Qs2. 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).

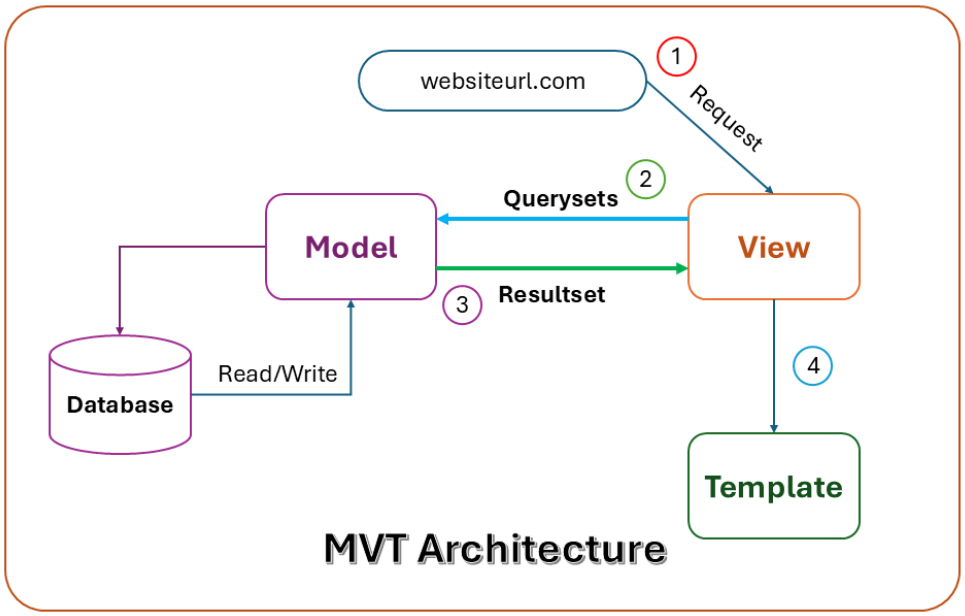
In Django, a **Project** is the entire website or application that serves as a container for multiple **Apps**, handling global configurations like **settings, URLs, and middleware** (the project is created using **django-admin startproject**), while an **App** is a modular, reusable component focused 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**).

Qs3. 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.

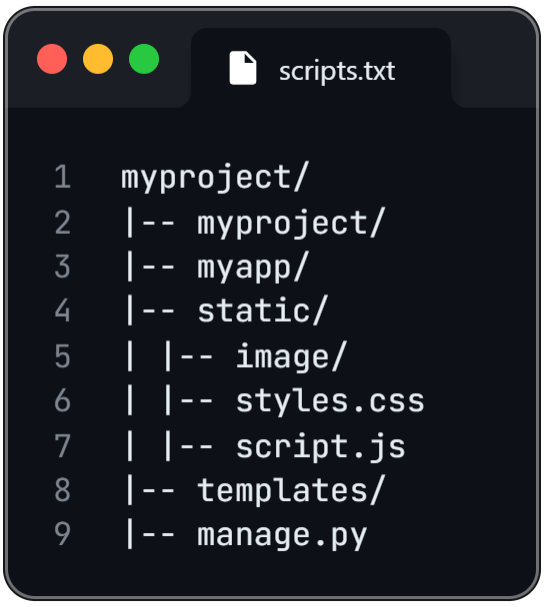
MVT Structure has the following three parts –

- **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.



Qs4. 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.



Qs5. 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

Qs6. 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.

Qs7. 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.

Qs8. 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.

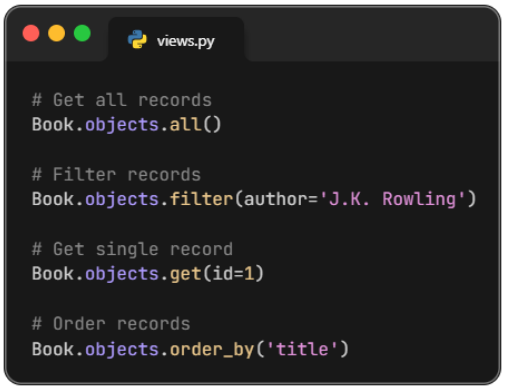
Qs9. What files are created when you start a new app?

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)

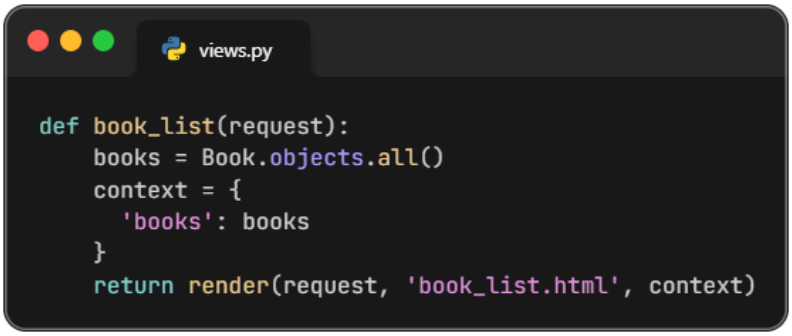
Qs10. What is a QuerySet in Django?

A QuerySet is a collection of database queries that can be used to retrieve, filter, and manipulate data from the database.



Qs11. How do you pass data to a template?

Through the context dictionary in a view we can pass data to a template:



Qs12. What are template tags?

In Django, template tags are special syntax elements enclosed within **{% %}** that allow you to add logic and control flow within your templates, similar to programming constructs, but specifically for the template language. They enable tasks like looping, conditional logic, and other operations that go beyond simply displaying data.

◇ Types of Template Tags:

- **Control flow:** {% if %}, {% for %}, {% elif %}, {% else %}
- **Inclusion:** {% include %}, {% extends %}, {% block %}
- **Utility:** {% csrf_token %}, {% url %}, {% static %}

index.html

```
{% if user.is_authenticated %}
  <p>Welcome, {{ user.username }}!</p>
{% else %}
  <p>Please log in.</p>
{% endif %}
```

Qs13. What is the difference between null=True and blank=True??

- **null =** True sets DB column to NULL.
- **blank =** True allows form field to be empty

Qs14. How to create custom user model?

To create a custom user model in Django, we should extend **AbstractUser** or **AbstractBaseUser** and define our model within an app. Then, specify this new model in your **settings.py** file using the **AUTH_USER_MODEL** setting.

models.py

```
from django.contrib.auth.models import AbstractUser
from django.db import models

class User(AbstractUser):
    phone_number = models.CharField(max_length=20, blank=True)
    date_of_birth = models.DateField(null=True, blank=True)
```

Qs15. Explain ForeignKey, OneToOneField, and ManyToManyField??

◇ ForeignKey

A ForeignKey represents a many-to-one relationship. This means many records in one model can be related to one record in another model.

Example use case: Many books written by one author.

◇ OneToOneField

A OneToOneField represents a one-to-one relationship. This means one record in a model is related to only one record in another model.

Example use case: Each user has one profile.

◇ ManyToManyField

A ManyToManyField represents a many-to-many relationship. This means multiple records in one model can be related to multiple records in another model.

Example use case: Students enrolled in multiple courses, and each course has multiple students.

Qs16. What does CRUD stand for?

CRUD stands for **Create, Read, Update, and Delete** — the four basic operations used to manage data in a database.

Qs17. How do you perform Read operation in Django?

Reading data (retrieving objects) is done using Django QuerySets:

- Model.objects.all() – to get all records
- Model.objects.get(id=1) – to get a single object
- Model.objects.filter(status='active') – to get filtered objects

Qs18. Explain the concept of middleware in Django.

Middleware in Django is a framework of hooks that process requests and responses globally across the application. When a request comes in, it passes through a series of middleware components before reaching the view. After the view processes the request and returns a response, that response again passes through the middleware before being sent to the client.

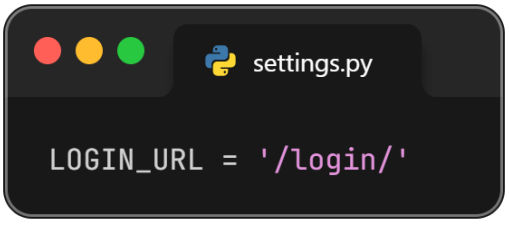
This mechanism allows middleware to perform tasks such as user authentication, session handling, logging, request/response modification, and security checks like CSRF protection. Middleware components are added to the **MIDDLEWARE** list in the **settings.py** file, and Django processes them in the order they appear.

Qs19. What is the difference between create() and create_user()

- **create()** is a generic method used to create any model instance. It saves the data as-is without special handling. It does not hash passwords, so it's not secure for creating user accounts.
- **create_user()** is a specialized method provided by Django's UserManager for creating User objects, specifically designed for the User model or custom user models. It automatically handles important user-related setup like password hashing, setting is_active=True, and more.

Qs20. What is the purpose of LOGIN_URL?

In Django, the LOGIN_URL setting tells the framework where to redirect users if they are not logged in but try to access a view that requires authentication.



Qs21. How does django handle user authentication?

Django handles user authentication using its built-in authentication system. It provides:

- A **User model** to store usernames, passwords (hashed), and other details.
- **Authentication backends** to verify credentials (e.g., authenticate() checks username & password).

- **Login & logout functions** (login(), logout()) to manage sessions.
- **Middleware** that attaches request.user and request.session to every request.
- **Decorators & mixins** like @login_required and LoginRequiredMixin to protect views.

1. Which command is used to create a new Django project?

a) django-admin startproject
b) django-admin createproject
c) manage.py startproject
d) django-admin runproject
Answer: a

2. What is the default User model in Django?

a) django.contrib.auth.models.AuthUser
b) django.contrib.auth.models.User
c) django.db.models.User
d) django.contrib.admin.models.User
Answer: b

3. What does the collectstatic command do?

a) Collects all models
b) Collects static files into one directory
c) Deletes old static files
d) Creates migration files
Answer: b

4. Which decorator is used to restrict a view to logged-in users only?

a) @require_login
b) @authenticated
c) @login_required
d) @user_required
Answer: c

5. What is the default template engine used in Django?

a) Jinja2
b) Mako
c) Django Template Language (DTL)
d) Chameleon
Answer: c

6. Which setting specifies the login URL in Django?

a) LOGIN_PAGE
b) LOGIN_URL
c) LOGIN_REDIRECT
d) LOGIN_REQUIRED
Answer: b

7. What is the default primary key field type in Django models?

a) UUIDField
b) CharField
c) IntegerField
d) AutoField
Answer: d

8. Which middleware is responsible for handling sessions in Django?

a) SessionMiddleware
b) AuthMiddleware
c) CacheMiddleware
d) RequestMiddleware
Answer: a

9. Which of the following is NOT a Django field type?

a) CharField
b) IntegerField
c) BooleanField
d) FloatTextField
Answer: d

10. What is the default session engine in Django?

a) File-based sessions
b) Database-backed sessions
c) Cache-based sessions
d) Cookie-based sessions
Answer: b

11. Which of the following is used for class-based views in Django?

a) TemplateView
b) ListView
c) DetailView
d) All of the above
Answer: d

12. Which function is used to check user credentials in Django?

a) login()
b) authenticate()
c) check_password()
d) validate_user()
Answer: b

13. Which class is used to create REST API serializers for Django models?

a) Serializer
b) ModelSerializer
c) RestSerializer
d) APISerializer
Answer: b

14. What does @csrf_exempt do in Django views?

a) Enables CSRF protection
b) Disables CSRF protection for a view
c) Redirects to login page
d) Logs user out
Answer: b

15. Which setting enables Django's debug mode?

a) DEBUG = True
b) DEBUG_MODE = True

c) DEVELOPMENT = True
d) TESTING = True
Answer: a

16. Which function logs in a user and creates a session?

a) login()
b) authenticate()
c) authorize()
d) session_start()
Answer: a

17. What does select_related() do in Django ORM?

a) Joins foreign key relationships in a single query
b) Filters queryset by related fields
c) Prefetches related objects in separate queries
d) Deletes related objects
Answer: a

18. What does prefetch_related() do in Django ORM?

a) Joins foreign key relationships in a single query
b) Prefetches related objects in separate queries
c) Filters queryset by related fields
d) Deletes related objects
Answer: b

19. Which class is used to customize the Django admin interface for a model?

a) AdminView
b) ModelAdmin
c) AdminModel
d) AdminCustomizer
Answer: b

20. Which ORM method raises DoesNotExist if no object is found?

a) filter()
b) get()
c) all()
d) first()
Answer: b

21. What is the default cache backend in Django if none is specified?

a) Memcached
b) File-based cache
c) LocMemCache (in-memory cache)
d) Redis
Answer: c

22. Which DRF class is used to create a read-only API endpoint for a model?

a) ReadOnlyModelViewSet
b) ModelViewSet
c) APIView
d) GenericAPIView
Answer: a

23. What is the purpose of SerializerMethodField in DRF?

a) To validate fields
b) To define a read-only field with a custom method
c) To serialize all model fields automatically
d) To create writable fields
Answer: b

24. Which DRF class is used to handle GET, POST, PUT, DELETE methods for APIs?

a) ViewSet
b) GenericAPIView
c) APIView
d) ModelSerializer
Answer: c

25. Which setting is used to enable Django REST Framework in INSTALLED_APPS?

a) 'rest_framework'
b) 'django_rest'
c) 'restframework'
d) 'django.api'
Answer: a

26. Which of these is a valid DRF permission class?

a) IsAuthenticated
b) IsAuthorized
c) HasPermission
d) AuthRequired
Answer: a

27. Which decorator is used to make a DRF view function-based?

a) @api_view
b) @function_view
c) @drf_view
d) @view_api
Answer: a

28. Which DRF class provides automatic CRUD operations for a model?

a) APIView
b) ModelViewSet
c) GenericAPIView
d) ReadOnlyModelViewSet
Answer: b

29. Which Django setting is used to specify a custom user model?

a) AUTH_USER_MODEL
b) CUSTOM_USER_MODEL
c) USER_MODEL
d) USER_AUTH_MODEL
Answer: a