**DJANGO INTERVIEW**

**1-What Is the Command to Install Django & To Know About Its Version?**

Command To Install Django:

**pip install django**

Command To Check Django Version:

**python -m django –version**

Command To Check all the versions of installed modules:

**pip freeze**

**2-What Is the Command to Create a Project & An App in Django?**

Command To Create a Project

**django-admin startproject project Name**

Command To Create an App:

**python manage.py startapp App name**

where nitmanis project name & nitapp is app name. A

**3-What Is the Command to Run a Project in Django?**

Command To Run a Project:

**python manage.py runserver**

By default, this command starts the development server on the internal IP at port 8000.If you want to change the server's port, pass it as a command-line argument. For instance, this command starts the server on port 8080:

**python manage.py runserver 8080**

If you want to change the server's IP, pass it along with the port, use:

**python manage.py runserver 0.0.0.0:8000**

**4-What is the command for create migrations in Django?**

Command to create a migration file inside the migration folder:

**python manage.py makemigrations**

After creating a migration, to reflect changes in the database permanently execute migrate command:

**python manage.py migrate**

To see raw SQL query executing behind applied migration execute the command:

**python manage.py sqlmigrate app\_name migration\_name**

**python manage.py sqlmigrate nitapp 0001**

To see all migrations, execute the command:

**python manage.py showmigrations**

To see app-specific migrations by specifying app-name, execute the command.

**python manage.py showmigrations nitapp**

**5- What is the Command to Create a Superuser in Django?**

Command To Create a SuperUser:

**python manage.py createsuperuser**

Enter your desired username and press enter.

**Username: admin**

You will then be prompted for your desired email address:

Email address: **admin@example.com**

‘The final step is to enter your password twice,

the second time as a confirmation of the first.

Password: **HEREREKEE**

Password (again): **\*#\*##\*\*#\***

Superuser created successfully.

**6-What is the Django Command to View a Database Schema of an existing Database?**

Command to view a database schema of an existing (or legacy) database:

**python manage.py inspectdb**

**7-How to View All items in The Model Using Django QuerySet?**

Django Command to View All Items in A Model:

**Users.objects.all ()**

where “User” is a model name.

**8-How to Filter Items in The Model Using Django Query Set?**

Django Command to Filter Items in A Model:

**Users.objects.filter(name="Nitin")**

where “User” is a model name.

**9-How to Get a Particular Item in The Model Using Django Query Set?**

Django Command to Get a Particular Item in A Model:

**Users.objects.get(id=25)**

where “User” is a model name.

**10-How to Delete/Insert/Update an Object Us Query Set in Django?**

QuerySet To Delete an Object:

**Users.objects.filter(id= 54).delete()**

**11-How to Delete/Insert/Update an Object Using Query Set in Django?**

QuerySet To Update an Object:

**user\_to\_be\_modify = User.objects.get(pk = 3)**

**user\_to\_be\_modify.city = "Pune"**

**user\_to\_be\_modify.save()**

QuerySet To Insert/Add an Object:

**new\_user = User(name = "Nitin Mangotra", city=""Gurgaon")**

**new\_user.save()**

**12- How Can You Combine Multiple Query Sets in A View?**

Let's suppose the following two models in Django.

**class Blog(models.Model):**

**title = models.CharField(max\_length=255)**

**content = models.TextField(blank=True)**

**class Email(models.Model):**

**title = models.charField(max\_length=255)**

**content = models. TextField(blank=True)**

Let's suppose the following three querysets generated from above models, that you want to combine.

>>> **query\_set\_1 =Blog.objects.filter(id\_gte=3)**

>>> **query\_set\_2 = Email.objects.filter(id\_\_\_Ite)**

>>> **query\_set\_3 = Blog.objects.filter(id\_\_\_Ite=2)**

**1. Using Union Operator:**

If both querysets belong to the same model, such as query\_set\_1 & query\_set\_3 above, then you can use union operator "|" to easily combine those querysets.

**query\_set\_result = query\_set\_1| query\_set\_3**

You can use the union operator to combine two or more querysets

**combined\_result = query\_set\_1| query\_set\_3| query\_set\_4**

**2. Using Itertools:**

If both querysets belong to the different model, such as query\_set\_1 & query\_set\_2 above, then you can use itertools combine those querysets.

**from itertools import chain**

**combined \_list = list (chain(query\_set\_1, query\_set\_2))**

You just need to mention the querysets you want to combine in a comma-separated manner in chain function. You can also use it to combine more than two querysets.

**combined \_list = list (chain (query\_set\_1, query\_set\_2, query\_set\_3))**

**Mostly Asked Questions.**

**1-Explain Django Architecture? Also Explain Model, Template and Views.**

Django follows a software design pattern called as MVT (Model View Template) Architecture.

It is based on the Model View Controller architecture (MVC). But is slightly different from the MVC pattern as it maintains its own conventions, so, the controller is handled by the framework itself.

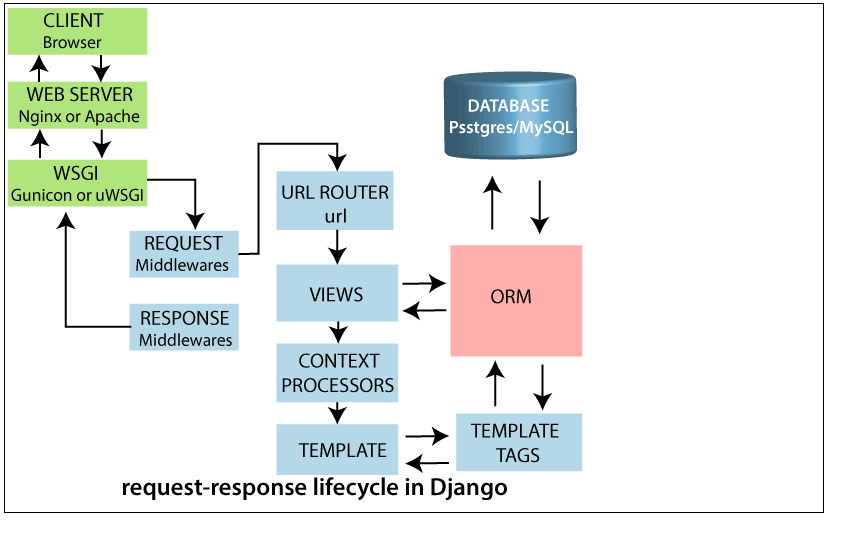
**Model:** It helps in handling the database (Models). They provide the options to create, edit and query data records in the database.

**View:** The View is used to execute the business logic and interact with a model to carry data and renders a template.

**Template:** The template is a presentation layer. It defines the structure of file layout to represent data in a web page. It is an HTML file mixed with Django Template Language (DTL).



**2-Explain how a request is processed in Django?**



**3-What is the difference between a project and an app in Django?**

A Project is the entire Django application, and an App is a module inside the project that deals with one specific use case.

For Example: ~ payment system(app) in the eCommerce app (Project).

**App** is basically a web Application that is created to perform a specific task.

**Project**, on the other hand, is a collection of these apps.

Therefore, a single project can consist of ‘n’ number of apps and a single app can be in multiple projects.

* Command To Create a Project:

**django-admin startproject myproject**

* Command To Create an App:

**python manage.py startapp course**

**4- Which is the default database in the settings file in Django?**

Answer: **SQLite**

**5- Why is Django called a loosely coupled framework?**

Django is called a loosely coupled framework because of its MVT architecture, which is a variant of the MVC architecture.

MVT helps in separating the server code from the client-related code.

Django’s Models and Views are present on the client machine and only templates return to the client, which are essentially HTML, CSS code and contains the required data from the models.

These components are totally independent of each other and therefore, front-end developers and backend developers can work simultaneously on the project as these two parts changing will have little to no effect on each other when changed.

**6-Which is the default port for the Django development server?**

Default Port for Django Development Server: **8000**

By default, the server runs on port 8000 on the IP address **127.0. 0.1**

**7-Explain the Migration in Django.**

migration refers to the process of changing the database schema or data using a set of files called migrations. Migrations are used to manage changes to the database schema over time as the application evolves, allowing developers to keep the database in sync with the codebase.

**makemigrations**: This command generates new migration files based on the changes made to the models.

**python manage.py makemigrations**

**migrate:** This command applies the pending migrations to the database.

**Python manage.py migrate.**

**8-What is Django ORM?**

ORM stands for Object-relational Mapper.

This ORM enables us to interact with databases in a more pythonic way like we can avoid writing raw queries.

It is possible to retrieve, save, delete, and perform other operations over the database without ever writing any SQL query.

It helps us with working with data in a more object-oriented way.

**9-Explain how you can set up the Database in Django?**

To set up a database in Django, you can find its configurations in setting.py file that representing Django settings.

By default, Django uses SQLite database. It is easy for Django users because it doesn’t require any other type of installation.

DATABASES = {

‘default': {

'ENGINE': 'django.db.backends.sqlite3',

; 'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),

}

}

**10-What do you mean by the CSRF Token?**

CSRF (Cross-Site Request Forgery) token is a security feature in Django that protects against unauthorized requests to the server. It is a random string of characters that is generated by the server and included in the HTML form when it is rendered to the user.

When the user submits the form, the CSRF token is also submitted with it. The server then checks the submitted token against the one generated for the user session. If the tokens match, the request is considered valid and processed. If they don't match, the request is rejected.

<h1>

WORD COUNT

</h1>

<a href="/home/">My Home</a>

**{% csrf\_token %}** //csrf token inserted here.

<form action="count">

<textarea cols="50" rows="10" name="FullText"></textarea>

<br/>

<input type="submit" value="CountMe"/>

</form>

**11- What is a Query Set in Django?**

QuerySet is a collection of SQL queries.

A QuerySet in Django is basically a collection of objects from our database.

QuerySets are used by the Django ORM. When we use our models to get a single record or a group of records from the database, they are returned as QuerySets.

It is comparable to a database select operation.

**users.objects.all()**

**users.objects.filter(name="nitin”)**

**users.objects.get(id=3)**

**12-Difference between select\_related and prefetch\_related in Django?**

**select\_related:**

* a Returns a QuerySet that will “follow” foreign-key relationships, selecting additional related-object data when it executes its query.
* This is a performance booster which results in a single more complex query but means later use of foreign-key relationships won't require database queries.

**prefetch\_related:**

* Weuse prefetch\_related when we're going to get a set of things.
* That means forward ManyToMany and backward ManyToMany, ForeignKey.
* A prefetch\_related does a separate lookup for each relationship and performs the “joining” in Python.

**13-Difference between Emp.object.filter(), Emp.object.get() and Emp.objects.all() in Django Queryset?**

* **Emp.objects.all():**

To view all the items from your database, you can make use of the ‘all ()’ function as

mentioned below:

* **Users.objects.all()**

where Users is some class that you have created in your models.

**Emp.object.filter() & Emp.object.get():**

‘To filter out some element from the database, you either use the get() method or the filter() method as follows:

**Users.objects.filter(name="Nitin")**

**Users.objects.get(name="Nitin")**

Basically use **get()** when you want to get a single unique object, & filter() when you want to get all objects that match your lookup parameters

get() throws an error if there’s no object matching the query.

filter() will return an empty queryset.

**get()** raises **MultipleObjectsReturned** if more than one object was found. The Multiple Objects Returned exception is an attribute of the model class.

get() raises a DoesNotExist exception if an object wasn't found for the given parameters. This exception is also an attribute of the model class.

**14-Which Companies Use Django?**

Instagram, Mozilla, Spotify, Pinterest, Disqus, Bitbucket, Eventbrite, Prezi, Dropbox, Youtube, National Geographic.

**15.How Static Files are defined in Django? Explain its Configuration and uses.**

Ensure that django.contrib.staticfiles is added to your INSTALLED\_APPS

In your settings file. define STATIC\_URL for ex.

STATIC\_URL = **'/static/'**

In your Django templates, use the static template tag to create the URL for the given:

relative path using the configured STATICFILES\_ STORAGE.

**{% load static %}**

<img src="{% static 'my\_app/example.jpg' %}" alt=""My image">

Store your static files in a folder called static in your app. For example

my\_app/static/my\_app/example.jpg QZ

**15.2. How can you set up static files in Django?**

There are three main things required to set up static files in Django:

1) Set the STATIC\_ROOT setting to the directory from which you'd like to serve these files, e.g:

STATIC\_ROOT = "/var/www/example.com/static/"

2) Run the collectstatic management command:

python manage.py collectstatic

This will copy all files from your static folders into the STATIC\_ROOT directory.

3) set up a Static Files entry on the Python Anywhere web tab.

**16.What is the difference between Flask, Pyramid, and Django?**

**Common Questions**

**1-Give a brief about the Django admin.**

Django Admin is a built-in feature of the Django web framework that provides an automatically - generated user interface for managing a website's content. It is essentially a web-based interface that allows authorized users to add, edit, and delete database records, as well as manage site settings and user accounts.

The Django Admin interface is highly customizable and can be tailored to suit the specific needs of a website. It is designed to be secure, scalable, and efficient, making it a powerful tool for website administrators and content managers.

**2- What databases are supported by Django?**

PostgreSQL MySQL SQLite Oracle

Apart from these, Django also supports databases such as ODBC, Microsoft SQL Server, IBM DB2, SAP SQL Anywhere, and Firebird using third-party packages.

Among these the best-suited database is PostgreSQL.

Note: Officially Django doesn’t support any No-SQL databases.

**3- What are the advantages/disadvantages of using Django?**

* **Advantages of using Django:**
* **Rapid development**: Django is designed to make web development faster and easier by providing a lot of built-in functionality and a clear structure for organizing code.
* **Security:** Django has built-in security features such as cross-site scripting (XSS) and cross-site request forgery (CSRF) protection, as well as support for secure password storage.
* **Scalability:** Django is designed to scale easily, from small websites to large-scale applications with millions of users.
* **Versatility:** Django can be used for a wide range of web applications, from simple blogs to complex e-commerce sites and social networks.
* **Community:** Django has a large and active community of developers who contribute to its ongoing development.
* **Disadvantages of Django:**
* Django is Monolithic. You must know the full system to work with it.
* Django's monolithic size makes it unsuitable for smaller projects.
* Everything must be explicitly defined due to a lack of convention.
* Django's modules are bulky.
* Django is completely based on Django ORM.
* Components are deployed together.

**4-What is the Django shortcut method to render an HTML response more easily?**

Django shortcut method is:

**"'render\_to\_response"**

**5-What is the difference between Authentication and Authorization in Django?**

**Authentication** - Who Are You?

**Authorization** - What Permissions Do You Have?

**Authentication** is the process of verifying who someone is, whereas **Authorization** is the process of verifying what specific applications, files, and data a user has access to.

Authentication is the process or action of verifying the identity of a user or process.

**6-What Is Token Based Authentication System?**

A token-based authentication system is a security system that authenticates the users who attempt to log in to a server, a network, or some other secure system, using a security token provided by the server.

**7-What is django.shortcuts.render function?**

The **render ()** function is a shortcut method in Django that simplifies the process of rendering a template into an HTTP response object. It takes an HTTP request object, a template name, and an optional dictionary of context data, and returns an HTTP response object with the rendered template.

from django.shortcuts import render

def my\_view(request):

context = {'name': 'John Doe'}

return **render (request**, 'my\_template.html', context)

**8-Explain Q objects in Django ORM?**

Q object django.db.models.Q is an object to encapsulate a collection of keyword

arguments specified as FIELD LOOKUPS.

Q objects are used to write complex queries, as in filter () functions just "AND" the conditions while if you want to "OR" the conditions you can use Q objects.

**Let’s see an example:**

from django.db import models

from django.db.models import Q.

**Models.objects.get(Q(question\_\_startswith="When'), Q(answer\_\_startswith='On') | Q(answer\_\_startswith="At') )**

[Q Objects can be combined with the help of the | and & operators to get a new Q Object]

This is equivalent to the following SQL WHERE Clause:

SELECT \* FROM Model WHERE question LIKE ‘When%’ And (**answer="0n%" OR answer="At%"**)

**9-What is the significance of the manage.py file in Django?**

The manage.py file in Django is a command-line utility that allows developers to perform various tasks related to the Django project, such as creating a new application, running the development server, managing the database, and more.

Some of the common tasks that can be performed using manage.py include:

* Creating a new Django application: python manage.py startapp <app\_name>
* Running the development server: python manage.py runserver
* Creating database tables: python manage.py migrate
* Creating a superuser for the admin site: python manage.py createsuperuser
* Running tests: python manage.py test

The manage.py file is also responsible for setting up the Django environment by setting the DJANGO\_SETTINGS\_MODULE environment variable to the appropriate settings module for the project.

In summary, the manage.py file is an important tool for managing a Django project and performing various tasks related to it.

**10-What is the use of the include function in the urls.py file in Django?**

The include() function in the urls.py file in Django is used to include other URL patterns from other applications or modules. This allows for a modular approach to building URLs in Django, which can help organize a large codebase.

When you include another URLconf via include(), Django will look for patterns in that file and apply them as if they were defined in the including file. This is especially useful for breaking up URL configurations into smaller parts.

from django.urls import path, include

from import views

urlpatterns = [

path('blog/', include('blog.urls')),

path('about/', views.about),

path('', views.home),

]

**11- What does {% include %} do in Django?**

The {% include %} template tag in Django is used to include the contents of another template within the current template. It allows you to reuse common elements of your website, such as headers, footers, or navigation menus, across multiple pages without having to repeat the same code on each page.

{% include "template\_name.html" %}

**{% include "header.html" %}**

<h1>Welcome to my website</h1>

**{% include "footer.html" %}**

**12-What is Django Rest Framework (DRF)?**

Django Rest Framework (DRF) is a powerful module for building web APIs.

AG Django Rest Framework (DRF) is a framework that helps you quickly create RESTful it is very easy to build model-backed APIs that have authentication policies and are browsable.

RESTful APIs are perfect for web applications since they use low bandwidth and are designed such that they work well with communications over the Internet like GET, POST, PUT, etc.

DRF is especially useful if we have an existing Django web application, and we wish to quickly generate an API for it.

The following are the significant reasons that are making REST framework perfect choice:

* Web browsable API
* Serialization
* Authentication policies
* Extensive documentation and excellent community support.
* Perfect for web apps since they have low bandwidth.

**13-What is a Middleware in Django?**

Middleware in Django is a way to add extra functionality to the request/response processing flow. It sits between the web server and the view and provides hooks for processing requests and responses at various stages.

Some usage of Middleware in Django is:

* Session management,
* Use authentication.
* Cross-site request forgery protection (CSRF)
* Content Gzipping.

**14-What is a session in Django?**

Sessions are fully supported in Django.

Using the session framework, you can easily store and retrieve arbitrary data based on the per-site-visitors.

This framework basically stores data on the server-side and takes care of sending and receiving cookies.

These cookies consist of a session ID but not the actual data itself unless you explicitly use a cookie-based backend.

A session is a mechanism to store information on the server side during the interaction with the web application.

By default, session stores in the database and also allows file-based and cache-based sessions.

**15-What are Django Signals?**

Two important parameters in signals are as follows:

**Receiver:** It specifies the call back function which connected to the signal.

**Sender:** It specifies a particular sender from where a signal is received.

Two key elements the Senders and the receivers are in the signal’s machinery. The sender is responsible to dispatch a signal, and the receiver is the one who receives this signal and then performs something.

**16-What is the context in Django?**

A context in Django is a dictionary, in which keys represent variable names and values represent their values. This dictionary (context) is passed to the template which then uses the variables to output the dynamic content.

A context is a variable name -> variable value mapping that is passed to a template.

Context processors let you specify a number of variables that get set in each context automatically — without you having to specify the variables in each render () call.

**17-What are Django exceptions?**

In Django, exceptions are errors that are raised when something goes wrong in your code. Django provides a number of built-in exceptions that you can use to handle common errors in your application, such as

ValidationError, ObjectDoesNotExist, and PermissionDenied.

from django.shortcuts import get\_object\_or\_404, render

from myapp.models import BlogPost

def delete\_blog\_post(request, post\_id):

post = get\_object\_or\_404(BlogPost, id=post\_id)

if request.user != post.author:

raise PermissionDenied('You do not have permission to delete this post')

post.delete()

return render(request, 'myapp/blog\_post\_deleted.html')

Django also provides a generic **ExceptionMiddleware** class that catches any unhandled exceptions and handles them in a consistent way. By default, Django will show a user-friendly error page that provides information about the exception, including the error message and a stack trace.

**18-Django File Structure?**

