**1. What is Django?**

Django is an open-source web development framework based on the [Python language](https://hackr.io/blog/python-programming-language), allowing Python developers to create database-driven websites. It was introduced in 2003 at the Lawrence journal-world newspaper.

This popular framework is maintained by a non-profit organization named Django Software Foundation (DSF). Later, in 2005, Django was made available publicly under the 3-clause BSD license.

Let’s see how we can write a Hello World Program using Django.

**Step 1: Create a project**

django-admin startproject hello\_world\_project

**Step 2: Change directory to hello\_world\_project**

cd hello\_world\_project

**Step 3: Create a new file views.py and write the function view for hello world**

**#views.py [new]**

from django.shortcuts import HttpResponse

#hello world view

def hello\_world(request):

**return** HttpResponse("Hello World")

**Step 4: Edit the urls.py file to route the url for hello world view**

**#urls.py**

from django.contrib import admin

from django.urls import path

from .views import hello\_world

urlpatterns = [

path('admin/', admin.site.urls),

path('', hello\_world, name="hello-world")

]

**Step 4: Run the server using python manage.py runserver command**

python manage.py runserver

The above steps are the simplest way to write a hello world program in Django.

Django is monolithic, which is especially designed for big projects, so even to write a small program like Hello World we have to go through many processes.

**2. What are the advantages of Django?**

Most developers prefer Django due to its dominance in the market, as it shows high computational and statistical capabilities. The following are some of the advantages of Django:

* Django follows Python’s ‘batteries included’ trait. Python is generally referred to as ‘batteries included’ because it has a unified standard library, helping developers use readymade packages to add features to a web project.

* Most of the packages in Python’s standard library are open-source. With these packages, you can even implement authentication, admin interfacing, session management, and other advanced functionalities.

* Python and Django are core technologies in IT giants, the Internet of Things (IoT), and blue chip companies. Therefore, learning it would help you establish a lucrative career.

* Security is yet another advantage of using the Django framework. Applications developed on Django are protected against SQL injection, clickjacking, XSS and CSRF attacks, etc.

* The built-in template language of Django promotes the process of building applications.

* Django enables developers to create applications and configure frameworks on the go. Also, it offers support for external libraries and packages. Django focuses on explicit programming, allowing the developers to create applications that require frequent changes.

* It helps in faster development and is thus used by most companies. If the configuration is correct, then you can use Django for optimizing web applications.

* Django has a REST framework, a python library, that helps develop APIs for many applications. Using these APIs, you can add advanced functionalities to your applications.

* Django comes with ML capabilities and libraries such as PyTorch, NumPy, etc.

**3. What are the disadvantages of Django?**

Despite many advantages, developers still find it challenging to make the transition. The following are some disadvantages of using the Django framework:

* Django lacks conventions that can be followed by developers for web development.
* It is not ideal for smaller projects having fewer requirements because it has a vast structure and heavy functionalities
* The Django framework is monolithic, i.e., developers have to work on the given patterns

**4. What are the applications of Django?**

Below are some types of projects that you can create using the Django framework:

* Financial platforms with various features, such as analyzing and calculating output based on the data
* Customized CRM system for the internal data
* B2B CRM system, helping you handle the communications within an organization
* Shopping websites
* Real-estate property evaluation system
* System for managing documents
* Creating an emailing system for sending notifications
* Verification system based on photos
* Platforms for managing investment funds

**5. What are the features of Django?**

The following are the best features of the Django framework:

* Compared to other Python frameworks, Django has the best documentation available in the market
* It is a Python-based framework that offers all Python libraries for use in an application to add extra functionality
* Django primarily maintains a website using URLs rather than its IP address on the server, making it easier for SEO engineers to add the website to the servers
* It is highly versatile and scalable
* Django is highly secure and thoroughly tested
* It also ensures rapid development

**6. Describe the Django architecture.**

Django follows the Model-View-Template(MVT) architecture based on a popular Model-View-Controller(MVC) architectural pattern, which is followed by popular web-frameworks like Ruby on Rails, Laravel etc.

Django’s Model-View-Template architecture divides the complete application into three major logical components:

1. Model
2. View
3. Template

All these three components are responsible for handling the different aspects of the web application.

**Model:**The models handle the database schema for the web applications. It maintains and represents the complete application data into the database. The default relational database used by the model is SQLite which is generally used in development, but in production we can use the MySQL and Postgres.

**View:**The view component manages all the logic of the application that we want to render on the user’s browser. In Django, the view acts as a bridge between the models and the templates. In the views, we can fetch the data from the models and render it on the template.

**Template:**The template component is the collection of static parts of the application such as the HTML, CSS, JavaScript, and Image files. The view uses the template as the base on which the data should be presented, because at the end, the web-application uses the static files to represent the content on the user browser.

Django architecture

**7. What is the project directory structure in Django?**

The following is the project directory structure in Django:

* **manage.py:** It is a command-line utility, allowing users to interact with their Django project
* **\_\_init\_\_.py:** An empty file specifying Python to consider the current directory as a Python package
* **settings.py:** It contains the necessary configurations of the project, such as DB connections
* **urls.py:** It holds all the URLs of a project
* **wsgi.py:** It is an application’s entry point used by web servers to serve the project

**8. What are models in Django?**

In Django, a model is a class that maps to a database table or database collection. Models class contains attributes representing a database field, defined within the app/models.py file. These models act as the abstraction layer, helping in structuring and manipulating data. The models are the subclass of the django.db.models.Model class.

**Example**

Let’s create a Product table with name, price, and descriptions attributes.

**#models.py**

from django.db import models

#product model

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

**name**= **models**.**CharField**(**max\_length**=30)

**price**= **models**.**models**.**IntegerField**()

**description** = **models**.**TextField**()

**def** **\_\_str\_\_**(**self**):

**return** **f**'{**self**.name}'

**9. What are the components of the Django architecture?**

The architecture of Django consists of the following components:

* **Models:** It specifies the database schema and the data structure
* **Views:** It controls what a user will see, the view used to retrieve the data from appropriate models, implement calculation on the data, and pass it to the desired template
* **Templates:** It specifies how the user sees it. It describes how the data received from the views should be formatted to display on a page
* **Controller:** The Django framework and URL parsing

**10. Explain the admin interface in Django.**

The admin interface is the interface already available in Django. It helps web developers save their time by eliminating the need to create another admin panel. Django admin is an application that can be easily imported from the package named django.contrib. It comes with user authentication and other features, such as management of various models and CMS, among other things.

The Django admin interface has two default models, Groups and Users, which are included in the admin interface after migrating the project and creating a new superuser.

To login to the default admin space, we need to route the URL to the admin panel, which is <http://127.0.0.1:8000/admin/> by default.



To get the login credentials, we can create a super user using the command ***python manage.py createsuperuser***.

After login to the admin panel, we have two default models: Groups and Users.



* **Groups:** It is the table for the groups that define the permissions allowed to the users in particular groups.
* **Users:** In the Users table all the details of the users are stored.

The Groups and the Users models come under the Authentication and Authorization module.

11. How is the code reusability of Django better than other frameworks?

Django offers better reusability of code than other available Python-based web frameworks. It is a collection of various applications, such as login applications, signup, etc. You can copy such applications from one directory to another by making required changes to the settings.py file. Thus, there is no need to write the entire code for the signup application from scratch. This is why Django helps in the rapid development of web applications.

Django is a batteries-included web framework, which means it comes with some built-in common components such as login, signup, session, and authentication features.

We can easily use those prewritten components code in our web applications by importing them into the script.

**Example**

Let’s create an inbuilt login form in Django using the Django account module.

**Step 1: Include the auth app into the project urls.py file.**

**#urls.py**

from django.contrib import admin

from django.urls import path, **include** # add

urlpatterns = [

path('admin/', admin.site.urls),

path('accounts/', **include**('django.contrib.auth.urls')), # add

]

**Step 2: Create a login.html file inside the templates/registrations directory in the project directory.**

<h2>LogIN</h2>

<form method="post">

{% csrf\_token %}

{{ form.as\_p }}

<button type="submit">Log In</button>

</form>

**Step 3: Configure the settings.py file**

TEMPLATES = [

{

....

'DIRS': [BASE\_DIR / 'templates'],

....

},

]

LOGIN\_REDIRECT\_URL = '/'

 Now we can [go to the login page](https://www.google.com/url?q=http://127.0.0.1:8000/accounts/login/&sa=D&source=editors&ust=1641513048389000&usg=AOvVaw2rmPWz64DOkBrEH3Toy-h_).



12. What happens when the Django website receives a request?

Whenever a user enters the URL in the browser, the Django server receives the request. The server looks for the URL in its URL-config. If the server finds the match there, it will return the corresponding view function.

Then the request is made to the model of an application to get the data. If there is any data to be passed, pass it to the corresponding template. After, the template renders in the browser. If the process does not work as expected, the user will get a “404” error page.

13. Can you customize Django’s admin interface? If yes, then how?

Yes, you can customize Django’s admin interface. Django’s admin is another entirely customizable application. It enables you to download another third-party application for a different view. You can create your own admin application to have complete control over it. Also, for customizing the Django admin site, you can change the settings of the admin site object.

Also, you can make the desired changes to your models and then apply them in the Django admin for adding specific applications, such as the search bar. You can customize even a smaller detail of your Django admin interface. Still, it is advisable to create a new admin rather than making so many changes at a lower level.

14. Why is Django considered a loosely coupled framework?

Django is considered a loosely coupled framework, as it is based on the MVT architecture, a variant of the MVC architecture. The MVT architecture is useful because it entirely separates the server code from the client’s machine.

Models and views are available on a client machine. However, the client only receives the template — the HTML and CSS code — along with the data from the models.

Since these components are different, the front-end and back-end developers can work together on the same project. Making changes to a project by both the teams will not impact each other, thus making Django a loosely coupled framework.

15. Explain the Django REST framework.

With the help of Django's REST framework, you can create RESTful APIs quickly and efficiently. This framework gets funding from various big shot companies and is popular due to its multiple commendable features such as serialization, authentication policies, etc.

RESTful APIs are well-suited for creating web applications since they use low bandwidth and can communicate over the internet via GET, POST and PUT methods.

16. What is the settings.py file, and what does it contain?

Whenever you start the Django server, initially, it looks for the settings.py file, containing the main settings regarding a web application. Also, it contains everything related to your web application, such as databases, a backend engine, templating engines, static file addresses, servers, security keys, middlewares, URL configs, and other essential data.

So, when you start the Django server, it will first execute the settings.py file and, later, load the required engines and databases.

The settings.py file resides in the main project directory.

project\_name

│ db.sqlite3

│ manage.py

│

├───project\_name

│ │ asgi.py

│ │ settings.py

│ │ urls.py

│ │ views.py

│ │ wsgi.py

│ │ \_\_init\_\_.py

17. Why are regular expressions used for defining the URLs?

Django has a powerful way of storing the URLs that are regular expressions. You can easily use the regular expression format for string searching algorithms, making the search process faster.

However, after the release of Django 2.2 and later versions there is no need to use the regular expression for defining the URL. Instead, you can use normal strings. The regular expression is used whenever you want to pass some data from the user via the URL. But whatever you use, the Django server needs to match them.

**Example**

urlpatterns = [

path(products/', views.all\_products),

path('articles/<int:id>/', views.product\_detail),

]

Sometimes the normal string patterns are not enough for the url pattern. There we can use the **re\_path()** function for regular expression urls.

urlpatterns = [

re\_path(r'^product/(?P<year>[0-9]{4})/$', views.product\_year\_archive),

]

18. Explain ORM in Django.

ORM stands for Object-relational mapper, a special feature tool of Django. This tool helps developers to interact with the database in a more Python-esque way. It acts as the abstraction between the models and the database, where the main data is stored.

Using ORM, you can retrieve, save, and delete the data from a database without the need to write any SQL code for it. This tool will help eliminate many loopholes since it lets you maintain control over your code, and is developed in Python.

It does not matter whether the Database is a SQLite, MySQL, Postgre or Oracle the ORM makes sure that the developer writes the same code for all databases.

Django uses the ORM known as Django ORM, it uses classes inherited from models.Modle, to create tables under any database.

The only thing we need to tweak is the database settings in the setting.py file.

Django setting for PostgreSQL connection

DATABASES = {

'default': {

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

'NAME': 'DB\_NAME',

'USER': 'DB\_USER',

'PASSWORD': 'DB\_PASSWORD',

'HOST': 'localhost',

'PORT': '5432',

}

}

Django setting for MySQL connection

DATABASES = {

    'default': {

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

        'NAME': 'DB\_NAME',

        'USER': 'DB\_USER',

        'PASSWORD': 'DB\_PASSWORD',

        'HOST': 'localhost',   # Or an IP Address that your DB is hosted on

        'PORT': '3306',

    }

}

19. How does templating work in Django?

Templates are the reason behind Django’s ability to create dynamic web pages. These templates are HTML code returned as an HTTP response. Furthermore, Django has a templating engine capable of handling templating. You can use some of the template syntaxes while declaring the variables, control logic, and comments.

Once you provide all the template syntax within the HTML structure, the web page is requested and called by the view function. Later, the Django template engine will get the HTML structure with variables and the data to replace these variables. The templating engine will replace these while executing the control logic and generating filters. It will then render the required HTML and send it to the browser.

20. What are view functions? Can you directly import a function within the URL?

A view is a middle layer between a model and a template, and it will take the data from the model and pass it to a template. Every application in Django has the view.py file that stores view functions, and these functions take the argument and return the browser-renderable format.

You can easily import view functions in the URL file. To do so, you need to import the view function in the urls.py file and add the desired path required by the browser to call that function.

Views import function

In the above example, you can see that we have imported all functions from our view module. Later, we added the URL within the urlpatterns list (red box). The ‘ index ‘ function will be called when the ‘dataflair/’ will be searched (yellow box).

21. What is the django.shortcuts.render function?

When the webpage is returned as the HttpResponse instead of the simple string by a view function, use the render() function. This function will allow the developers to pass the data dictionary using a template. Then this function will use the templating engine for combining the template along with the data dictionary.

After that, this function will return the HttpResponse with the rendered text that is being returned by the models. In this way, this function will save a lot of time for developers and allow them to use different templating engines.

The basic render Syntax:

|  |
| --- |
| render(request, template\_name, context=None, content\_type=None, status=None, using=None) |

Where the request is the parameter that will generate the response, the template\_name will specify the value where the template is stored, and the template name will be used for passing the dictionary. For more control, you can mention the content type, data status.

22. How can you add view functions to the urls.py file?

The following are the two methods for adding view functions to the urls.py file:

* By adding the function view: Using this method, you need to import the view as a function. You import the function from the specific view and then add the URL to the urlpattern list.
* By adding the class-based view: This is an object-oriented approach where you import the class from the views.py file and later add the URL to the urlpattern lists. For this, you will require an inbuilt method for calling the class as a view.

23. What does the urls-config file contain?

This config file stores all the URL lists and mapping to their respective view functions. These URLs can be mapped to view functions, class-based views, and url-config of another application.

The default URL list name is urlpatterns, and it contains all the path() or re\_path() URL patterns. The project URL comes with the root urls.py file, and with every application, we can also make an isolated urls.py file for that application.

Example

#urls.py

from django.contrib import admin

from django.urls import path, **include**

urlpatterns = [

path('admin/', admin.site.urls),

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

]

24. What does it mean to say that Django is monolithic?

Django is based on the MVT architecture, and since Django is the controller of the architecture, it has defined some rules that all developers need to follow to execute appropriate files at the right time.

In Django, you get great customizability with implementation. But you are not allowed to make changes to the file names, the predefined lists, and variable names. You have to create the new ones, but you can’t make changes to the predefined variables.

The monolithic behavior helps the developers to understand the project easily. Even if the company changes, the project layout will remain the same. Therefore, developers take less time to understand the code, increasing productivity.

26. What is Jinja templating?

Django comes with the support for many popular templating engines, and by default, it comes with one very powerful templating engine, called Jinja Templating. The latest version is Jinja 2.

Below are some features of the Jinja templating, making it a better option than another templating engine available.

* Sandbox Execution: It is a protected framework useful for automating the testing process
* HTML Escaping: Jinja 2 comes with an automatic HTML Escaping, as <, >, & characters that have special values in templates. If you use it as regular text, these symbols can lead to XSS Attacks, handled by Jinja automatically.
* It shows template inheritance and generates HTML templates much faster than the default engine
* It is easier to debug with Jina compared to the default engine

27. What is user authentication in Django?

Django has a built-in user authentication system capable of handling different objects, such as users, groups, user-permissions, and some cookie-based user sessions.

Django’s User authentication not only helps in authenticating, but also in authorizing a user and checking what permissions that user has.

The system operates on the following objects:

* Users
* Permissions
* Groups
* Password Hashing System
* Forms Validation
* A pluggable backend system

Third-party web applications can be used instead of the default system as you have much more control over user authentication and many other features.

28. What is the purpose of middleware in Django?

In Django, middleware is the component that works on request and transfers it to the view, and before it passes it to the template engine, it starts operating on a response.

Django middleware list

The above image displays the list of middleware installed by default within your Django framework.

It serves several purposes, including session management and user authentication.

29. What is the use of the djangopackages.org website?

In Django, the packages website is the place where all the third-party applications are uploaded. You can install them in your system.

Django interview questions - Django Packages

30. What popular websites use Django?

Django is a compelling framework and is used by various renowned organizations. Some of the highly trafficked websites using Django are:

* Instagram
* Pinterest
* Disqus
* Mozilla
* Bitbucket
* YouTube
* Spotify
* NASA
* Eventbrite

31. How can you set up the database in Django?

You can edit mysite/setting.py, a module representing Django settings. By default, Django uses SQLite, which is easy to use and does not require any type of installation.

If your database choice is different, you have to do the following keys in the DATABASE ‘default’ item for matching your database connection settings:

* Engines: You can make the changes to the database using ‘django.db.backends.sqlite3’, ‘django.db.backeneds.mysql’, ‘django.db.backends.postgresql\_psycopg2’, ‘django.db.backends.oracle’, and so on.
* Name: If you use SQLite as your database, the database file will be on your computer. The name should be a full absolute path, including the file name of that file.

But, if you do not choose SQLite, you need to add the settings such as Password, Host, User, etc.

32. How can you set up static files in Django?

For setting up the static files in Django, you need to consider the below steps:

* Firstly, set the STATIC\_ROOT in the settings.py file
* Run the manage.py collectsatic file
* Finally, set up a Static Files entry on the PythonAnywhere web tab

33. What is the session framework in Django?

Django comes with the session framework helping to store and retrieve the arbitrary data on a per-site-visitor basis. It saves all data on the server-side and abstracts the receiving and sending of cookies. You can implement the session via middleware.

34. How many types of inheritance styles are in Django?

Django has three inheritance styles, as mentioned below.

* Abstract base classes: You can use this style when you want a parent’s class to only store the information you don’t want to use for each child model
* Multi-table Inheritance: You can use this style If you are subclassing an existing model and need each model to have its database table.
* Proxy models: You can use this model if you only want to change the Python level behavior of the model without the need to change the model’s fields

35. What are the applications of middleware in Django?

The following are some applications of middleware in Django:

* Session management
* User authentication
* Cross-site request forgery protection
* Content gzipping

36. What are signals in Django?

Signals are pieces of code that hold information regarding what is happening. You can use a dispatcher for sending the signals and listening to those signals.

The signals become very useful when we want to do something with the data, before or after a certain event occurs.

Here is the list of events on which we can use the Django signals:

* pre\_save() trigger before save().
* post\_save() trigger after save().
* pre\_delete() trigger before delete().
* post\_delete() trigger after delete().
* m2m\_changed() triggers when there is a change in ManyToMany Field.
* request\_started() trigger when the django starts the HTTP request.
* request\_finished() trigger when the django finishes the HTTP request.

Example

Send an email when the user creates a new post.

**from** django.db.models.signals **import** post\_save

**from** .models **import** Blog

**from** django.core.mail **import** send\_mail

@receiver(post\_save, sender=Blog)

**def** **create\_profile**(sender, instance, created, \*\*kwargs):

#if created for the first time

**if** created:

blog\_title = instance.title

send\_mail("Subject", "Message", "from\_email", ['toemail@.com'])

37. What are some important parameters of signals?

The following are the two important parameters of signals:

* Receiver: It specifies the callback function connected to the signal
* Sender: It specifies a particular sender from where a signal is received

38. What is mixin in Django?

Mixin is a type of multiple inheritance that combines the behaviors and attributes of more than one parent class. It provides an excellent way of reusing the code from multiple classes.

For example, generic class-based views have a mixin called TemplateResponseMixin. This mixin is used for defining the render\_to\_response() method. When you combine it with a class present in the View, it results in a TemplateView class.

The only drawback of mixin is that it becomes difficult to analyze what a child class is doing and which methods to override if its code scatters between multiple classes.

39. What are caching strategies in Django?

Caching implies storing the output of calculations to avoid performing the same calculations repetitively. Django comes with a robust cache system helping to create dynamic pages. Therefore, it eliminates the need to evaluate pages repeatedly for every request. The following table highlights some of the significant caching strategies:

|  |  |
| --- | --- |
| Strategy | Description |
| Memcached | It is a memory-based cache server. |
| Filesystem caching | This caching strategy helps in caching the values stored as separate files in a serialized order. |
| Local-memory caching | It is the default cache, and it is used if you have not specified any other. It is a per-process and thread-safe cache. |
| Database caching | The database stores the cache data. |

40. What is the manage.py file in Django?

Whenever you create a project, the manage.py file is automatically created. This is a command-line utility, helping you to interact with your Django project. It performs the same work as Django-admin and sets the DJANGO\_SETTINGS\_MODULE environment variable to point to your project’s settings. It is better if you use manage.py instead of Django-admin if you are working on a single project.

41. How is the “migrate” command used in Django?

In Django, migrations are used for propagating the changes made to models. You can use the migrate command to apply and remove the migration changes made to models.

This command helps synchronize the current set of models and migrations with the database state. You can also use this command with or without passing the parameters. If you do not specify any parameters, all apps will have all their migrations running.

Command:

python manage.py migrate

Output

Migrating the project for the first time will show a similar result.



42. What is the response cycle in Django?

Whenever a user requests a web page, Django will create an HttpRequest object containing the important metadata about that request. After that, Django will load a particular view, passing the HttpRequest as its first argument to the view function. Each view then returns an HttpResponse object.



The following are the steps that take place when a request is received by Django:

* Firstly, the settings.py file, which contains various middleware classes, is loaded
* All the middleware classes get executed in the same order in which they are mentioned
* Now, the request will be moved to the URL Router. The URL Router gets the URL path from the request and later tries to map with the given URL paths within the urls.py.
* After mapping, it calls the equivalent view function, from where the corresponding response is generated.
* The response now passes through the response middleware and is sent back to the client/browser

43. What is the difference between select\_related and prefetch\_related?

|  |  |
| --- | --- |
| select\_related() | prefetch\_related() |
| The select\_related() is lookup for a queryset, it adds the additional forward foreignkey data to the return queryset. | The prefetch\_related() is another lookup for a queryset, it add the additional forward ForeignKey, OneToOne and backward OneToOne data to the return queryset. |
| It reduces the background SQL query using JOIN statements. | It uses the SQL joins and SELECT command to reduce the complex query set. |
| It works when we are selecting single objects. | It is used to select a set of multiple objects. |
| Example  queryset = Blog.objects.select\_related('Author').all() | Example    queryset = Blog.objects.prefetch\_related('Categories').all() |

Example:

**from** django.db **import** models

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

country\_name = models.CharField(max\_length=5)

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

state\_name = models.CharField(max\_length=5)

country = model.ForeignKey(Country)

>> states = State.objects.select\_related('country').all()

>> **for** state **in** states:

... print(state.state\_name)

```Query Executed

SELECT state\_id, state\_name, country\_name FROM State INNER JOIN Country ON (State.country\_id = Country.id)

```

>> country = Country.objects.prefetch\_related('state').get(id=1)

>> **for** state **in** country.state.all():

... print(state.state\_name)

```Query Executed

SELECT id, country\_name FROM country WHERE id=1;

SELECT state\_id, state\_name WHERE State WHERE country\_id IN (1);

44. What are the different tasks that can be performed using Django-admin?```

Django-admin is the command-line utility of Django used for carrying out administrative tasks. The below table highlights various tasks performed using Django-admin:

|  |  |
| --- | --- |
| Task | Command |
| Displaying the usage information and the list of commands provided by each application. | django-admin help |
| Displaying the list of available commands | django-admin help –command |
| Getting the description of a specific command and the list of its available options | django-admin help <command> |
| Checking the version of Django | django-admin version |
| Creating new migrations depending on the changes made in the models | django-admin makemigrations |
| Synchronizing a database state with its current set of models and migrations | django-admin migrate |
| Starting the development server | django-admin runserver |
| Sending a test email to confirm if Django is working | django-admin sendtestemail |
| Starting the Python interactive interpreter | django-admin shell |
| Displaying all migrations in your project | django-admin showmigrations |

45. How can you connect the Django project with the database?

By default the Djagno connected to the SQLite database, with the following setting:

DATABASES = {

'default': {

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

'NAME': BASE\_DIR / 'db.sqlite3',

}

}

But we can also connect it to the production level database such as[MySQL and PostgreSQL.](https://hackr.io/blog/postgresql-vs-mysql)

**Connect Django With MySQL**

To connect Dango with mysql we first have to install the mysqlclient as an adapter.

**pip install mysqlclient**

And we also need to change the default DATABASE setting in the setting.py file

DATABASES = {

'default': {

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

'NAME': 'DB\_NAME',

'USER': 'DB\_USER',

'PASSWORD': 'DB\_PASSWORD',

'HOST': 'localhost', # Or an IP Address that your DB is hosted on

'PORT': '3306',

}

}

**Connect Django With PostgreSQL**

To connect Dango with PostgreSQL we first have to install the psycopg2 as an adapter.

**pip install psycopg2**

And we also need to change the default DATABASE setting in the setting.py file

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.postgresql\_psycopg2',

'NAME': 'myproject',

'USER': 'myprojectuser',

'PASSWORD': 'password',

'HOST': 'localhost',

'PORT': '',

}

}

After making changes in the setting.py file and installing the proper adapter, we need to migrate the database so Python ORM can create tables in the newly selected Database.

1. **python manage.py migrate:**This command looks at the INSTALLED\_APPS settings and creates database tables accordingly
2. **python manage.py makemigrations:** This will inform Django that you have created/ changed your models
3. **python manage.py sqlmigrate <name of the app followed by the generated id>:** Sqlmigrate takes migration names and returns their SQL

46. What types of exception classes are there in Django?

The following are the exception classes available in Django:

|  |  |
| --- | --- |
| Exception | Description |
| AppRegistryNotReady | Raised when we try to use models before the app loading process. |
| ObjectDoesNotExist | Base class for DoesNotExist exceptions. |
| EmptyResultSet | Raised if a query does not return any result. |
| FieldDoesNotExist | Raised if the requested field does not exist. |
| MultipleObjectsReturned | A query will raise this exception if only one object is expected, but multiple objects are returned. |
| SuspiciousOperation | Raised when a suspicious operation has been performed by the user. |
| PermissionDenied | Raised when a user does not have an appropriate permission to perform a specified action. |
| ViewDoesNotExist | Raised by django.urls if the requested view does not exist. |
| MiddlewareNotUsed | Raised if a middleware is not used in the server configuration. |
| ImproperlyConfigured | Raised if Django is somehow improperly configured. |
| FieldError | Raised if there is a problem with a model field. |
| ValidationError | Raised if the data validation fails to form or model field validation. |

47. What are the specific Django field class types?

Field class types specify the following:

* The database column type.
* The default HTML widget for availing while rendering a form field.
* The minimal validation requirements used in Django admin that helps in automatically generated forms.

48. What is a model in Django?

A Model is specified as a Python class derived from the Model class. This model class is imported from the django.db.models library. The main concept of Django Models is to create objects for storing data from a user in a user-defined format.

The model class is a pre-defined class with lots of benefits. You can define the field with specific attributes as you can do in SQL, but the same can also be achieved in Python.

This class is parsed by Django ORM or backend engine, and there is no need to do anything related to a database, such as creating tables and defining fields afterward mapping the fields with the attribute of the class.

**Example**

**from** django.db **import** models

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

id = models.IntegerField(max\_length=30)

name = models.CharField(max\_length=30)

dob = models.DateField()

49. What are the different types of views available in Django?

The following are the two different types of views available in Django:

* Function-Based Views: It lets you import a view as a function
* Class-based Views: It is an object-oriented approach

50. Which companies use Django?

The following is a list of some companies that use Django:

* PBS
* Instagram
* Mozilla
* The Washington Times
* Disqus, Bitbucket
* NextDoor
* YouTube
* Pinterest
* DISCUS

## Basic Django Interview Questions and Answers

### 1. What is Django? And why is it used?

Django is a high-level Python web framework that enables the rapid development of secure and maintainable websites. It's free and open source. It takes care of much of the hassle of web development and allows you to focus on writing apps without any need to reinvent the wheel.

The purpose behind developing this framework is to make developers spend time on new application components instead of already developed components.

The reasons why Django is most preferred are:

* The Django framework is fast and flexible.
* Suits for any web app development
* It's secure and Scalable.
* Portable

### 2. Is Django backend or front end?

Django is suitable for both the backend and frontend. It's a collection of Python libraries that allow you to develop useful web apps ideal for backend and frontend purposes.

### 3. What is the latest version of Django? And explain its features.

The latest version of Django is Django 3.1. The new features of it are:

* Supports asynchronous views and middleware
* Provides JSON field support for all database backends
* Admin layout
* Path lib
* Code Reusability

CDN Integration

* SECURE\_REFERRER\_POLICY

|  |
| --- |
| If you want to enrich your career and become a professional in Python Django, then enroll in "[**Python Django Training**](https://mindmajix.com/python-django-training)". This course will help you to achieve excellence in this domain. |

### 4. What is the difference between Python and Django?

Both Python and Django are intertwined but not the same. Python is a programming language used for various application developments: machine learning, artificial intelligence, desktop apps, etc.

Django is a Python web framework used for full-stack app development and server development.

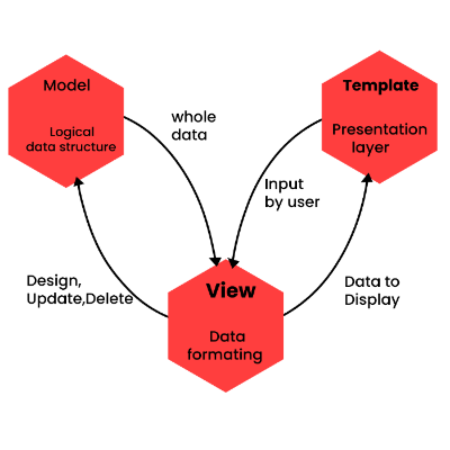
Using core Python, you can build an app from scratch or craft the app with Django using prewritten bits of code



### 5. What architecture does Django use?

Django follows a Model-Template-View (MTV) architecture. It contains three different parts:

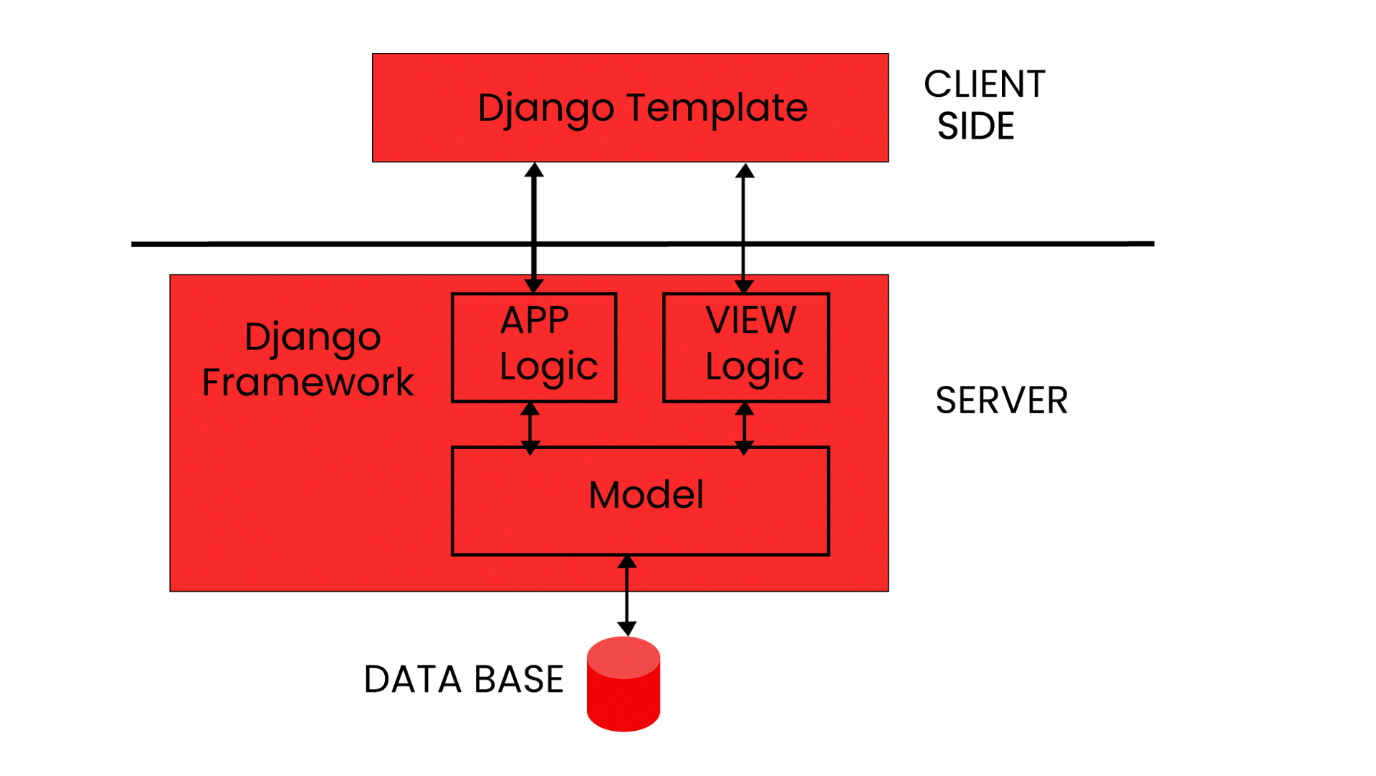
* **Model**: Logical data structure behind the entire app and signified by a database.
* **Template:** Deals with the presentation of data.
* **View:** It's a user interface. What you see when you visit a website is called a user interface. Represented by HTML/CSS/Javascript files.



### 6. Explain Django Architecture

As discussed in the previous question, Django follows MTV architecture - Model, Template, View.

The below diagram depicts the working cycle of Django MTV architecture:



From the diagram, you'll notice Template is on the Client side, and both the Model and View are on the Server side. Django uses request and response objects to communicate between the client and server.

If the website receives the request, it is transmitted from browser to server to manage the view file using a template.

After sending the correct URL request, the app logic and Model initiate the right response to the presented request. After that, a detailed response is sent back to View to check the response and transmit it as an HTTP response or desired user format. Then it again passes to the browser via Templates.

For your clear understanding, let's take a real-life example:

While logging into Django based website, you open the login page. It happens because View will process the request and send it to the login page URL. Then the response is sent from a server to the browser.

After then, you'll enter the credentials in Template, and the data sent back to the View to rectify the request, and then data is presented in the Model. Then the Model verifies the data provided by the user in the connected database.

If the user's data matches, it sends the related data (profile name, image, etc.) to the Views.

Otherwise, the model passes the negative result to the Views.

That's how the Django MTV architecture is working.

### 7. Explain Django's code reusability.

Compared to other frameworks, Django offers more code reusability. As Django is a combination of apps, copying those apps from one directory to another with some tweaks to the settings.py file won't need much time to write new applications from scratch.

That is why Django is the rapid development framework, and this kind of code reusability is not allowed in any other framework.

### 8. Is Django easy to learn?

Yes, Django is an easy-to-learn framework compared to others. Having some knowledge of Python and web-working helps you to start developing with Django.

|  |
| --- |
| → Learn [**Python Tutorial**](https://mindmajix.com/python-tutorial) |

### 9. What are the unique features of Django that make it a better framework?

The best features of Django that make it better compared to others are:

* Compared to other open-source technologies, Django offers excellent documentation in the market.
* It's a
* [web framework](https://mindmajix.com/top-20-python-frameworks-list) and one of the main reasons that people started using it. It's the only one that can solve any kind of operation out there.
* Django is SEO optimized.
* Django is scalable and can flexibly switch from small to large-scale projects.
* Versatile in nature. Django allows you to build applications for various types of domains.
* It has a vast community to connect with and share.
* Provides rapid development

### 10. What are the advantages of Django?

Django has many advantages, but we'll look at major ones that differentiate it from other frameworks.

* Better CDN connectivity and content management
* Designed as batteries included framework
* Supports MVC programming paradigm
* Provides robust security features
* Accelerated custom web app development
* Compatible with major operating systems and databases

### 11. Describe the inheritance styles in Django?

Django offers three inheritance styles:

1. **Abstract base classes:** You use this style when you want the parent class to retain the data you don't want to type out for every child model.
2. **Multi-table inheritance:**You use this style when you want to use a subclass on an existing model and want each model to have its database table.
3. **Proxy models:** You use this style to modify Python-level behaviour with the models without changing the Model's field.

### 12. What are Django Models?

A model is a definitive source of information about data, defined in the “app/models.py”.

Models work as an abstraction layer that structures and manipulates data. Django models are a subclass of the "django.db.models". Model class and the attributes in the models represent database fields.

### 13. Give a brief about the settings.py file.

As the name implies, it's the main settings file of the Django file. Everything inside the Django project, like databases, middlewares, backend engines, templating engines, installed applications, static file addresses, main URL configurations, allowed hosts and servers, and security key stores in this file as a dictionary or list.

So when Django files start, it first executes the settings.py file and then loads the respective databases and engines to quickly serve the request.

### 14. Is Django a CMS?

No, Django is not CMS (Content Management System). It's just a web framework and programming tool that allows you to build websites.

### 15. What are static files in Django? And how can you set them?

In Django, static files are the files that serve the purpose of additional purposes such as images, CSS, or JavaScript files. Static files managed by “django.contrib.staticfiles”. There are three main things to do to set up static files in Django:

1) Set STATIC\_ROOT in settings.py

2) Run manage.py collect static

3) Set up a Static Files entry on the PythonAnywhere web tab

### 16. What is the use of Middlewares in Django?

Middlewares in Django is a lightweight plugin that processes during request and response execution. It performs functions like security, CSRF protection, session, authentication, etc. Django supports various built-in middlewares.

### 17. What is the difference between CharField and TextField in Django?

* TextField is a large text field for large-sized text. In Django, TextField is used to store paragraphs and all other text data. The default form widget for this field is TextArea.
* CharField is a string field used for small- to large-sized strings. It is like a string field in C/C++. In Django, CharField is used to store small strings like first name, last name, etc.

### 18. Describe Django Field Class types?

Every field in a model is an instance of the appropriate field class. In Django, field class types determine:

* The column type describes the database about what kind of data to store (e.g., INTEGER, VARCHAR, TEXT).
* The default HTML widget, while rendering a form field

(e.g. <input type="text">, <select>)

* The minimal validation requirements used in automatically generated forms and Django admin.

### 19. What is the usage of "Django-admin.py" and "manage.py"?

* Django-admin.py - It is a command-line utility for administrative tasks.
* manage.py - It is automatically created in each Django project and controls the Django project on the server or even to begin one. It has the following usage:

1. Manages the project's package on the sys. path.
2. Sets the DJANGO\_SETTINGS\_MODULE environment variable

### 20. What are signals in Django?

Django includes a "signal dispatcher" to notify decoupled applications when some action takes place in the framework. In a nutshell, signals allow specific senders to inform a suite of receivers that some action has occurred. They are instrumental when we use more pieces of code in the same events.

Django provides a set of built-in signals that enable users to get notified of specific actions.

|  |  |
| --- | --- |
| **Signal** | **Description** |
| Django.db.models.signals.pre\_save(or)django.db.models.signals.post\_save | Sent before or after a model’s save() method calls. |
| django.db.models.signals.pre\_delete  (or)django.db.models.signals.post\_delete | Sent before or after a model’s delete() method or query set’s delete() method calls. |
| django.db.models.signals.m2m\_changed | We use this signal when ManyToManyField on a model changes. |
| Django.core.signals.request\_started(or)django.core.signals.request\_finished | We use this signal when Django starts or finishes an HTTP request. |

## Django Interview Questions for Experienced

### 21. What’s the difference between a project and an app in Django?

The app is a module that deals with the dedicated requirements in a project. On the other hand, the project covers an entire app. In Django terms, a project can contain different apps, while an app features in various projects.

### 22. Explain Django URL in brief?

Django allows you to design URL functions however you want. For this, you need to create a Python module informally called URLconf (URL configuration).

This module is purely a Python code and acts as a mapping between URL path expressions and [Python functions](https://mindmajix.com/python-string-functions). Also, this mapping can be as long or short as needed and can also reference other mappings.

The length of this mapping can be as long or short as required and can also reference other mappings. Django also provides a way to translate URLs according to the active language.

### 23. What are Django Exceptions?

An exception is an abnormal event that leads to program failure. Django uses its exception classes and python exceptions as well to deal with such situations.

We define Django core exceptions in "Django.core.exceptions". The following classes are present in this module:

|  |  |
| --- | --- |
| **Exception** | **Description** |
| AppRegistryNotReady | This class raises for using models before loading the app process. |
| ObjectDoesNotExist | It’s a base class for DoesNotExist exceptions. |
| EmptyResultSet | This exception arises when the query fails to return results. |
| FieldDoesNotExist | When the requested file does not exist, this exception arises. |
| MultipleObjectsReturned | It raises by the query multiple objects returned when we expect only one object. |
| SuspiciousOperation | It raises when the user has performed some operation, which is considered suspicious from a security perspective. |
| PermissionDenied | It arises when a user does not have permission to execute a specific action requested. |
| ViewDoesNotExist | When the requested view does not exist, this exception raises. |
| MiddlewareNotUsed | When there is no middleware in server configuration, this exception arises. |
| ImproperlyConfigured | When Django configuration is improper, this exception arises. |
| FieldError | When there is a problem with the model field, this exception arises. |
| ValidationError | It raises when data validation fails. |

### 24. Explain Django session

Django uses the session to keep track of the state between the site and a particular browser. Django supports anonymous sessions. The session framework stores and retrieves data on a per-site-visitor basis. It stores the information on the server side and supports sending and receiving cookies. Cookies store the data of session ID but not the actual data itself.

### 25. What are Django cookies?

A cookie is a piece of information stored in the client's browser. To set and fetch cookies, Django provides built-in methods. We use the set\_cookie() method for setting a cookie and the get() method for getting the cookie.

You can also use the request.COOKIES['key'] array to get cookie values.

### 26. Flask vs. Django: What's the difference between Flask & Django?

Flask and Django are the two most popular Python web frameworks. The following table lists some significant differences between Django and Flask

|  |  |  |
| --- | --- | --- |
| **Comparison Factor** | **Django** | **Flask** |
| created | Django is a web development framework for Python. Its created in 2005 | Flask is a web microframework offering basic features of web apps. Its created in 2010 |
| Project Type | High-level Python web framework for easy and simple projects. | Low-level Python web framework. |
| Features | The best features of Django are open-source, rapid development, robust documentation, great community, and easy to learn. | The best features of Flask are open source, lightweight, and require less code to develop an app. |
| Type of Framework | Full-stack web framework | WSGI (Web Server Gateway Interface ) framework |
| Templates, Admin, and ORM | Built-in | Requires installation |
| Flexibility | Django Web Framework supports a large number of third-party applications. | Flask Web Framework doesn't offer support for third-party applications. |
| Companies using | Instagram, Coursera, Udemy. | Netflix, Reddit, Lyft, MIT |
| Visual Debugging | Django does not support visual debugging. | Flask supports visual debugging. |
| Bootstrapping tool | Builtin | Not available |
| Working style | Offers monolithic working style | Offers diversified working style |
| Project layout | The structure of the project layout is conventional. | The structure of the project layout for the flask is random. |

### 27. How to check the version of Django installed on your system?

To check the version of [Django installed on your system](https://docs.djangoproject.com/en/3.1/topics/install/), open the command prompt and enter the following command:

py -m django --version

You can also try to import Django and use the get\_version() method as follows:

import django

print(django.get\_version())

### 28. Give a brief about Django Admin.

Django Admin is the command-line utility for administrative tasks. It's a preloaded interface to fulfill all web developer's needs and is imported from the "django.contrib packages".

Django Admin interface has its user authentication and offers advanced features like authorizing the access, CMS (Content Management System), managing various models, etc.

You can even perform the following tasks using Django admin as listed out in the table:

|  |  |
| --- | --- |
| **Command** | **Task** |
| django-admin help | Displays the usage of the information and commands list provided by each application. |
| django-admin help –command | Displays available commands |
| django-admin help <command> | Displays the command description and its available options |
| django-admin version | Determines Django’s version |
| django-admin make migrations | Depending on the changes done in the model creates new migrations |
| django-admin migrate | Synchronizes the database state with the present set of models and migrations |
| django-admin runserver | Starts the development server |
| django-admin sendtestemail | A test mail sent to confirm Django email working status |
| django-admin shell | Starts the Python interactive interpreter |
| django-admin showmigrations | Displays all the project’s migrations |

### 29. How do you create a Django project?

To create a Django project, navigate to the directory where you want to do a project and type the following command:

$ django-admin startproject ABC

That will create an "ABC" folder with the following structure −

ABC/

manage.py

myproject/

\_\_init\_\_.py

settings.py

urls.py

wsgi.py

Note: Here, "ABC" is the name of the project. You can mention any name you want.

### 30. Name some companies using Django.

Various companies out there are using Django. Of them, major are Instagram, Pinterest, Udemy, Mozilla Firefox, Reddit, etc.

### 31. How do Django views work?

Django views are the critical component of the framework They serve the purpose of encapsulation. They encapsulate the logic liable to process a user's request and return a response to the user.

Either they return HTTP responses or raise an exception such as 404 in Django. Besides, Views also perform tasks like reading records from a database, generating PDF files, etc.

Every app in Django comes with a views.py file, and this contains the views functions. Views function can be imported directly in the URLs file in Django.

To achieve that, you have to import the view function in the urls.py file first and add the path/URL that the browser should request to call that View function.

|  |
| --- |
| Explore [**Python Interview Questions**](https://mindmajix.com/python-interview-questions) that help you grab high-paying jobs. |

### 32. Give a brief about Django Template?

Django Templates generate dynamic web pages. Using templates, you can show the static data and the data from various databases connected to the app through a context dictionary. You can create any number of templates based on project requirements. Even it's OK to have none of them.

Django template engine handles the templating in the Django web framework. Some template syntaxes declare variables, filters, control logic, and comments.

Django ships built-in backends for its template system called the Django template language (DTL).

### 33. Describe Django ORM.

In Django, the most notable feature is Object-Relational Mapper (ORM), which allows you to interact with app data from various relational databases such as SQLite, MySQL, and PostgreSQL.

Django ORM is the abstraction between web application data structure (models) and the database where the data is stored. Without writing any code, you can retrieve, delete, save, and perform other operations over the database.

The main advantage of ORMs is rapid development. ORMs make projects more portable. It's easier to change the database with Django ORM.

### 34. When to use iterators in Django ORM?

Iterators are containers in Python containing several elements. Every object in the iterator implements two methods that are \_\_init\_\_() and the \_\_next\_\_() methods.

In Django, the fair use of an iterator is when you process results that take up a large amount of memory space. For this, you can use the iterator() method, which evaluates the QuerySet and returns the corresponding iterator over the results.

### 35. What is Django caching? And explain the strategies used to implement it.

Caching is the process of saving expensive calculation output to avoid performing the same calculation again.

Django supports a robust cache system to save web pages such that they don't have to be evaluated repeatedly for each request.

They are few strategies to implement caching in Django, and the following table lists them:

|  |  |
| --- | --- |
| **Strategy** | **Description** |
| Memcached | The most efficient and faster memory-based cache server |
| Filesystem caching | Cache files store in serial order in separate files. |
| Local-memory caching | If you have not specified any other, this is the default cache. It’s per-process and threads safe as well. |
| Database caching | Cache data will be stored in the database and works OK if you have a well-indexed database server. |

### 36. How does Django process a request?

Whenever the Django Server receives a request, the system follows an algorithm to determine which Python code needs execution. Here are the steps that sum up the algorithm:

* Django checks the root URL configuration.
* Next, Django looks at all the variable URL patterns in the URLconf for the match of the requested URL
* If the URL matches, it returns the associated view function.
* It will then request the data from the Model of that app for any data requirement and pass it to the corresponding Template rendered by the browser.
* Django sends an error-handling view if none of the URLs match the requested URL.

### 37. Which Python version should be used with Django?

Python 3 is the most recommended version for Django. Because it's faster, has more features, and is better supported.

### 38. Explain the file structure of a typical Django project.

A typical Django project consists of these four files:

* manage.py
* settings.py
* \_\_init\_\_.py
* urls.py
* wsgi.py

The final four files are inside a directory, which is at the same level as manage.py.

* manage.py is the command-line utility of your Django project and controls the Django project on the server.
* settings.py file includes information on all the apps installed in the project.
* The urls.py file acts as a map for the whole web project.
* The \_\_init\_\_.py file is an empty file that makes the python interpreter understand that the directory consisting of settings.py is a module/ package.
* The wsgi.py file is for the server format WSGI

### 39. Why is Django called a loosely coupled framework?

Django is known as a loosely coupled framework beca+use of its MTV architecture.

Django's architecture is a variant of MVC architecture, and MTV is beneficial because it completely discards server code from the client's machine. Models and Views are present on the client machine, and templates only return to the client.

All the architecture components are different from each other. Both frontend and backend developers can work simultaneously on the projects as they won't affect each other when changed.

### 40. What is the Django REST framework (DRF)?

Django REST framework is a flexible and powerful toolkit for building Web APIs rapidly.

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

* Web browsable API
* Authentication policies
* [Serialization](https://mindmajix.com/python/serialization)
* Extensive documentation and excellent community support.
* Perfect for web apps since they have low bandwidth.
* Global companies like Red Hat, Mozilla, Heroku, Eventbrite, etc., trust this framework.

## Advanced Django Interview Questions

### 41. Is Django too monolithic? Explain this statement.

The Django framework is monolithic, which is valid to some extent. As Django's architecture is MTV-based, it requires some rules that developers need to follow to execute the appropriate files at the right time.

With Django, you get significant customizations with implementations. Through this, you cannot change file names, variable names, and predefined lists.

Django's file structure is a logical workflow. Thus the monolithic behavior of Django helps developers to understand the project efficiently.

### 42. Explain user authentication in Django

Django comes with a built-in user authentication system to handle objects such as users, groups, permissions, etc. It not only performs authentication but authorization as well.

Following are the system objects:

* users
* Groups
* Password Hashing System
* Permissions
* A pluggable backend system
* Forms Validation

Apart from this, there are various third-party web apps that we can use instead of the default system to provide more user authentication with more features.

### 43. What is the "django.shortcuts.render" function?

When a View function returns a web page as HttpResponse instead of a simple string, we use the render function.

Render is a shortcut for passing a data dictionary with a template. This function uses a templating engine to combine templates with a data dictionary.

Finally, the render() returns the HttpResponse with the rendered text, the models' data.

Syntax:

render(request, template\_name, context=None, content\_type=None, status=None, using=None)

The request generates a response.

The template name and other parameters pass the dictionary.

For more control, specify the content type, the data status you passed, and the render you are returning.

### 44. What is the use of forms in Django?

Forms serve the purpose of receiving user inputs and using that data for logical operations on databases. Django supports form class to create HTML forms. It defines a form and how it works and appears.

Django's forms handle the following parts:

* Prepares and restructures data to make it ready for rendering
* Creates HTML forms for the data
* Processes submitted forms and data from the client.

### 45. Can you explain how to add View functions to the urls.py file?

There are two ways to add the view function to the main URLs config:

1. Adding a function View

In this method, import the particular View's function and add the specific URL to the URL patterns list.

2. Adding a Class-based view

This one is a more class-based approach. For this, import the class from the views.py and then add the URL to the URL patterns. An inbuilt method is needed to call the class as a view.

Write the name of the function on the previous method as shown below:

class\_name.as\_view()

This will pass your view class as a view function.

Both function-based and class-based have their advantages and disadvantages. Depending on the situation, you can use them to get the right results.

### 46. Explain Django Security.

Protecting user's data is an essential part of any website design. Django implements various sufficient protections against several common threats. The following are Django's security features:

* Cross-site scripting (XSS) protection
* SQL injection protection
* Cross-site request forgery (CSRF) protection
* Enforcing SSL/HTTPS
* Session security
* Clickjacking protection
* Host header validation

### 47. What is Ajax in Django?

AJAX (Asynchronous JavaScript And XML) allows web pages to update asynchronously to and from the server by exchanging data in Django. That means without reloading a complete webpage you can update parts of the web page.

It involves a combination of a browser built-in XMLHttpRequest object, HTML DOM, and JavaScript.

### 48. How to handle Ajax requests in Django?

To handle Ajax requests in the Django web framework, perform the following:

* Initialize Project
* Create models
* Create views
* Write URLs
* Carry out requests with Jquery Ajax.
* Register models to admin

### 49. What are Django generic views?

Writing views is a heavy task. Django offers an easy way to set Views called Generic Views. They are classes but not functions and stored in "django.views.generic".

Generic views act as a shortcut for common usage patterns. They take some common idioms and patterns in view development and abstract them to write common views of data without repeating yourself quickly.

### 50. What is the correct way to make a variable available to all your templates?

In case all your templates need the same objects, use "RequestContext." This method takes HttpRequest as the first parameter and populates the context with a few variables simultaneously as per the engine's context\_processors configuration option.

## Django FAQ

### 51. Tell me how to use a file-based session?

For this, we have to set the SESSION\_ENGINE settings to “Django.contrib.sessions.backends.file.”

### 52. What command-line loads data in Django?

“Django-admin.py load data” loads data in Django. This command line performs data searching and loads the contents of the named fixtures into the database.

### 53. What is CRUD?

CRUS is an acronym for Create, Read, Update, and Delete. It’s a mnemonic framework used for constructing models when building application programming interfaces (APIs).

### 54. Explain Django’s Request/Response Cycle.

When a process starts, the Django server receives a request and checks for a matching URL in the project-defined URL patterns. If the URL matches, it executes the associated code in the view file with the URL and sends a response. If the server can’t find a matching URL, it invokes a 404-status code.

### 55. What do you use middleware for in Django?

For the following functions, you can use Middleware in Django:

* Cross-site request forgery protection
* Content Gzipping
* User authentication
* Session management

### 56. Does Django support multiple-column primary keys?

Django does not support multiple-column primary keys. It only supports single-column primary keys.

### 57. What is a QuerySet?

In the context of Django, QuerySet is a set of SQL queries. To see the SQL query from the Django filter call, type the command print(b.query).

### 58. How to check the raw SQL queries running in Django??

Make sure that the DEBUG setting is set to True, and type the following commands:

* from Django.db import connection
* connection.queries

### 59. Are Django signals asynchronous?

No, Django signals are synchronous. There is no background thread or asynchronous jobs to execute them. When we use signals in applications, they allow you to maintain the code to understand application behavior and solve issues faster and better.

### 60. What are the Django disadvantages?

Not suitable for small projects due to its monolithic size

* Everything connects on Django’s ORM.
* Everything must be defined explicitly due to a lack of convention.
* Django web framework has a steep learning curve.