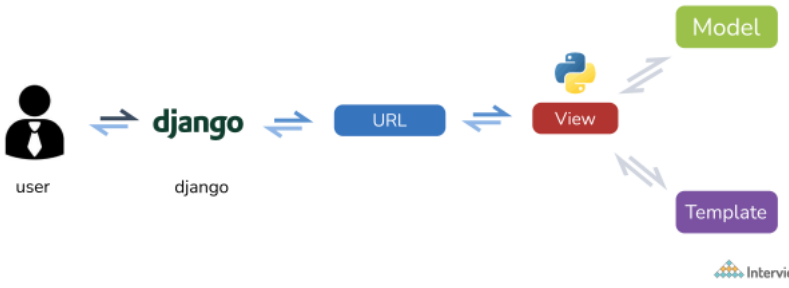
Django

1. Explain Django Architecture?

Django follows the MVT (Model View Template) pattern which is based on the Model View Controller architecture. It's slightly different from the MVC pattern as it maintains its own conventions, so, the controller is handled by the framework itself. The template is a presentation layer. It is an HTML file mixed with Django Template Language (DTL). The developer provides the model, the view, and the template then maps it to a URL, and finally, Django serves it to the user.



2. Explain the django project directory structure?

- manage.py - A command-line utility that allows you to interact with your Django project
- __init__.py - An empty file that tells Python that the current directory should be considered as a Python package
- settings.py - Comprises the configurations of the current project like DB connections.
- urls.py - All the URLs of the project are present here
- wsgi.py - This is an entry point for your application which is used by the web servers to serve the project you have created.

3. What are models in Django?

A model in Django refers to a class that maps to a database table or database collection. Each attribute of the Django model class represents a database field. They are defined in app/models.py

4. What are templates in Django or Django template language?

Templates are an integral part of the Django MVT architecture. They generally comprise HTML, CSS, and js in which dynamic variables and information are embedded with the help of views. Some constructs are recognized and interpreted by the template engine. The main ones are variables and tags.

5. What are views in Django?

A view function, or "view" for short, is simply a Python function that takes a web request and returns a web response. This response can be HTML contents of a web page, or a redirect, or a 404 error, or an XML document, or an image, etc.

6. What is Django ORM?

This ORM (an acronym for Object Relational Mapper) 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 works as an abstraction layer between the models and the database.

7. Define static files and explain their uses?

Websites generally need to serve additional files such as images. Javascript or CSS. In Django, these files are referred to as "static files", Apart from that Django provides `django.contrib.staticfiles` to manage these static files.

8. What is Django Rest Framework(DRF)?

Django Rest Framework is an open-source framework based upon Django which lets you create RESTful APIs rapidly.

9. What is django-admin and manage.py and explain its commands?

django-admin is Django's command-line utility for administrative tasks. In addition to this, a manage.py file is also automatically created in each Django project. Not only does it perform the same purpose as the django-admin but it also sets the `DJANGO_SETTINGS_MODULE` environment variable to point to the project's settings.py file.

10. What is Jinja templating?

Jinja Templating is a very popular templating engine for Python, the latest version is Jinja2.

11. What is the difference between a project and an app in Django?

In simple words Project is the entire Django application and an app is a module inside the project that deals with one specific use case.
For eg, payment system(app) in the eCommerce app(Project).

12. What are Django Signals?

Whenever there is a modification in a model, we may need to trigger some actions.
Django provides an elegant way to handle these in the form of signals. The signals are the utilities that allow us to associate events with actions.
We can implement these by developing a function that will run when a signal calls it.

13. Explain user authentication in Django?

Django comes with a built-in user authentication system, which handles objects like users, groups, user-permissions, and few cookie-based user sessions. Django User authentication not only authenticates the user but also authorizes him.

The system consists and operates on these objects:

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

14. What databases are supported by Django?

PostgreSQL and MySQL, SQLite and 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. Note: Officially Django doesn't support any no-SQL databases.

15. What's the use of a session framework?

Using the session framework, you can easily store and retrieve arbitrary data based on the pre-site-visitors. It stores data on the server-side and takes care of the process of sending and receiving cookies. These cookies just consist of a session ID, not the actual data itself unless you explicitly use a cookie-based backend.

16. What's the use of Middleware in Django?

Middleware is something that executes between the request and response. In simple words, you can say it acts as a bridge between the request and response. Similarly In Django when a request is made it moves through middlewares to views and data is passed through middleware as a response.

Or

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 context in the Django?

Context is a dictionary mapping template variable name given to Python objects in Django. This is the general name, but you can give any other name of your choice if you want.

18. What's the significance of the settings.py file?

As the name suggests this file stores the configurations or settings of our Django project, like database configuration, backend engines, middlewares, installed applications, main URL configurations, static file addresses, templating engines, main URL configurations, security keys, allowed hosts, and much more.

19. What is mixin?

Mixin is a type of multiple inheritances wherein you can combine behaviors and attributes of more than one parent class. It provides us with an excellent way to reuse code from multiple classes. One drawback of using these mixins is that it becomes difficult to analyze what a class is doing and which methods to override in case of its code being too scattered between multiple classes.

20. Difference between Django OneToOneField and ForeignKey Field?

Both of them are of the most common types of fields used in Django. The only difference between these two is that ForeignKey field consists of on_delete option along with a model's class because it's used for many-to-one relationships while on the other hand, the OneToOneField, only carries out a one-to-one relationship and requires only the model's class.

21. How can you combine multiple QuerySets in a View?

Initially, Concatenating QuerySets into lists is believed to be the easiest approach. Here's an example of how to do that:

```
from itertools import chain
result_list = list(chain(model1_list, model2_list, model3_list))
```

22. How to get a particular item in the Model?

```
ModelName.objects.get(id="term")
```

23. How to obtain the SQL query from the queryset?

```
print(queryset.query)
```

24. What are the ways to customize the functionality of the Django admin interface?

There are multiple ways to customize the functionality of the Django admin interface. You can piggyback on top of an add/change form that's automatically generated by Django, you can add JavaScript modules using the js parameter. This parameter is basically a list of URLs that point to the JavaScript modules that are to be included in your project within a <script> tag. You can also write views for the admin if you want.

25. Difference between select_related and prefetch_related?

Though both the functions are used to fetch the related fields on a model but their functioning is bit different from each other. In simple words, select_related uses a foreign key relationship, i.e. using join on the query itself while on the prefetch_related there is a separate lookup and the joining on the python side. Let's try to illustrate this via an example:

26. Explain Q objects in Django ORM?

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
>> objects = Models.objects.get(
   Q(tag__startswith='Human'),
   Q(category='Eyes') | Q(category='Nose')
)
```Query Executed
SELECT * FROM Model WHERE tag LIKE 'Human%' AND (category='Eyes' OR category='Nose')
```

27. What are Django exceptions?

In addition to the standard Python exceptions, Django raises of its own exceptions.List of the exceptions by Django (https://docs.djangoproject.com/en/3.1/ref/exceptions/)

28. 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.

30. 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

31. 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.

32. 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:

33. 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.
```

34. 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.

35. 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.

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. Why is Django called a loosely coupled framework?

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

Django's architecture is a variant of MVC architecture, and MVT 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.

38. What is the Django REST framework (DRF)?

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

39. Is Django too monolithic? Explain this statement.

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

40. 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

41. 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.

42. 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

43. 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.

44. 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".

45. 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

46. Does Django support multiple-column primary keys?

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

47. 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).

48. 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

49. 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.

50 What are the two important parameters in signals?

Two important parameters in signals are:

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

51 how to hide methods in class

```
class A(object):
    def someMethodToHide():
        pass
```

52 List comprehension and dictionary comprehension in django.

53 what is decorator in django

In Python, a decorator is a function that takes another function as an argument and adds functionality or augments the function without changing it. Django, as a Python web framework, comes with a large number of built-in decorators. These built-in decorators are used when decorating function-based views.

54. how to write custom middlewere in django?

55. map and filter difference in python

56 What is method overriding and method overloading in Python?

Key characteristics	Map	Filter	Reduce
Syntax	map(function, iterable object)	filter(function, iterable object)	reduce(function, iterable object)
Effect of the function on the iterable	Applies the function evenly to all the items in the iterable object	Function is a boolean condition that rejects all the items in the iterable object that are not true	Breaks down the entire process of applying the function into pair-wise operations
Input to output variables	N to N	N to M where N>=M	N to 1
Use Case	Transformation/Mapping	Splitting the data	Single output operations
Example	List of the square of all numbers in a list	All even numbers from a list	Product of all the items in the list

In the method overloading, methods or functions must have the same name and different signatures. Whereas in the method overriding, methods or functions must have the same name and same signatures.

57 what is __str__ in django.

The __str__() method is called whenever you call str() on an object. Django uses str(obj) in a number of places. Most notably, to display an object in the Django admin site and as the value inserted into a template when it displays an object.

58 what is __repr__ in django.

Python __repr__() function returns the object representation in string format. This method is called when repr() function is invoked on the object. If possible, the string returned should be a valid Python expression that can be used to reconstruct the object again

Function/method	Description
`str()`	Return *printable/human-readable* string representation
`repr()`	Return string representation that's a *valid Python expression* (meaning it can be passed to `eval()`)
`__str__()`	Overload `str()` for printable string representation
`__repr__()`	Overload `repr()` for evaluatable string representation

59 which generic foreignkey in django.

In Django, a GenericForeignKey is a feature that allows a model to be related to any other model in the system, as opposed to a ForeignKey which is related to a specific one. This post is about why GenericForeignKey is usually something you should stay away from.

60 can we pass list/tuple as key in dictionary python?

A tuple containing a list cannot be used as a key in a dictionary.

61 aggregation and anonotation in python.

Aggregate calculates values for the entire queryset. Annotate calculates summary values for each item in the queryset.

62 sql and postgres difference

63 what is Q in django?

Q object encapsulates a SQL expression in a Python object that can be used in database-related operations. Using Q objects we can make complex queries with less and simple code. For example, this Q object filters whether the question starts wiht 'what': from django. db.

64 what is trigger in django?

Django Triggers is a light-weight framework for having one part of an application generate a trigger while another part responds to to it. Triggers are persistent and can be scheduled to be processed at a later time.

65 What is a use of WSGI file in Django?

PostgreSQL	MySQL
This PostgreSQL is released under the PostgreSQL License.	This MySQL is released under the GNU License with other proprietary agreements.
It's free and open-source software. No need to pay anything	This software is owned by Oracle Corporations.
PostgreSQL is fully ACID compliant	MySQL is an ACID-compliant only when using with NDB and InnoDB engines
PostgreSQL is the best performance database tool even it works better with complex queries	MySQL performs well with only OLAP and OLTP applications only when you consider speed.
PostgreSQL works best with business integration applications, It is more suitable for data warehousing and data analysis applications, but they need fast read-write speed	MySQL is a reliable software even it works well with BI but difficult to read

It is used **as an interface between application server to connect with django or any python framework which implements wsgi specified by python community**, taking about usage mostly you will see it is used with gunicorn.

66 what is scrum in django?

Scrum is a project management technique which is used to organize the work and planning around a project, often a software development project. Scrum builds upon the pillars of transparency, inspection, and adaptation to provide a framework for rapid project development.

67 what is CI/CD in django?

It's all about automatically building committed code, so you can use it with Python to avoid breaking your builds. You can create and maintain a stable base, clone and commit frequently, and test rapidly! You instantly get faster deployments thanks to the eliminated pip install bloat.

68. All joins in db and all modifications in lists and dictionary, must have to knowladge about ORM query / filtering.