Django Interview Questions

A list of top frequently asked **Django interview questions** and answers are given below.

1) Explain Django.

Django is a free and open source web application framework, written in Python. It is a server-side web framework that provides rapid development of secure and maintainable websites.

---

2) What does Django mean?

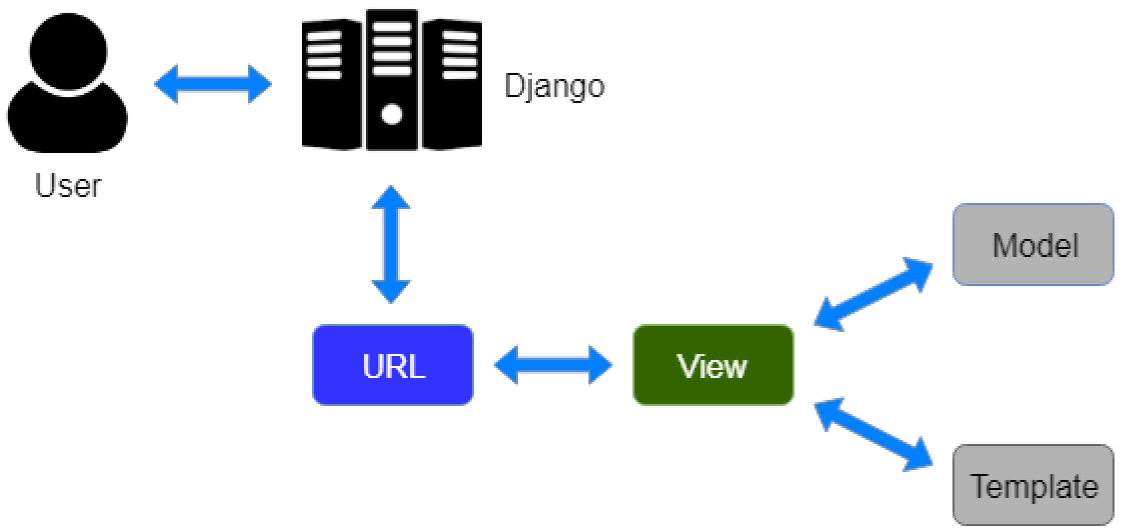
Django is named after **Django Reinhardt**, a **gypsy jazz guitarist** from the **1930s** to early **1950s** who is known as one of the best **guitarists** of all time.

---

3) Which architectural pattern does Django follow?

Django follows **Model-View-Template (MVT)** architectural pattern.

See the following graph that shows the **MVT** based control flow.



Here, a user **requests** for a resource to the Django, Django works as a controller and check to the available resource in URL.

If URL maps, **a view is called** that interact with model and template, it renders a template.

Django responds back to the user and sends a template as a **response**.

---

4) Explain Django architecture.

Django follows **MVT (Model View Template)** pattern. It is slightly different from **MVC**.

Model: It is the data access layer. It contains everything about the data, i.e., how to access it, how to validate it, its behaviors and the relationships between the data.Let's see an example. We are creating a model **Employee** who has two fields **first_name** and **last_name**.

```
from django.db import models


class Employee(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
```

View: It is the business logic layer. This layer contains the logic that accesses the model and defers to the appropriate template. It is like a bridge between the model and the template.

```
import datetime
# Create your views here.
from django.http import HttpResponse
def index(request):
    now = datetime.datetime.now()
    html = "<html><body><h3>Now time is %s.</h3></body></html>" % now
    return HttpResponse(html)    # rendering the template in HttpResponse
```

Template: It is a presentation layer. This layer contains presentation-related decisions, i.e., how something should be displayed on a Web page or other type of document.

To configure the template system, we have to provide some entries in **settings.py** file.

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR,'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

---

5) Is Django a high-level web framework or low-level framework?

Django is a high-level Python's web framework which was designed for rapid development and clean, realistic design.

---

6) How would you pronounce Django?

Django is **pronounced JANG-oh**. Here D is silent.

---

7) How does Django work?

Django can be broken into many components:

Models.py file: This file defines your data model by extending your single line of code into full database tables and add a pre-built administration section to manage content.

Urls.py file: It uses a regular expression to capture URL patterns for processing.

Views.py file: It is the main part of Django. The actual processing happens in view.

When a visitor lands on Django page, first Django checks the URLs pattern you have created and used the information to retrieve the view. After that view processes the request, querying your database if necessary, and passes the requested information to a template.

After that, the template renders the data in a layout you have created and displayed the page.

---

8) Which foundation manages the Django web framework?

Django web framework is managed and maintained by an independent and non-profit organization named Django Software Foundation (DSF). The primary foundation goal is to promote, support, and advance the Django Web framework.

---

9) Is Django stable?

Yes, Django is quite stable. Many companies like Disqus, Instagram, Pinterest, and Mozilla have been using Django for many years.

---

10) What are the features available in Django web framework?

Features available in Django web framework are:

- Admin Interface (CRUD)
- Templating
- Form handling
- Internationalization
- A Session, user management, role-based permissions
- Object-relational mapping (ORM)
- Testing Framework
- Fantastic Documentation

---

11) Explain the advantages of Django?

Advantages of Django:

- Django is a Python's framework which is easy to learn.
- It is clear and readable.
- It is versatile.
- It is fast to write.
- No loopholes in design.
- It is secure.
- It is scalable.
- It is versatile.

---

12) What are the disadvantages of Django?

Following is the list of disadvantages of Django:

- Django' modules are bulky.
- It is completely based on Django ORM.
- Components are deployed together.
- You must know the full system to work with it.

---

13) What are the inheritance styles in Django?

There are three possible inheritance styles in Django:

1) Abstract base classes: This style is used when you only want parent's class to hold information that you don't want to type out for each child model.

2) Multi-table Inheritance: This style is used if you are sub-classing an existing model and need each model to have its database table.

3) Proxy models: This style is used, if you only want to modify the Python level behavior of the model, without changing the model's fields.

---

14) Is Django a content management system (CMS)?

No, Django is not a CMS. Instead, it is a Web framework and a programming tool that makes you able to build websites.

---

15) How can you set up static files in Django?

There are three main things required 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 some typical usage of middlewares in Django?

Some usage of middlewares in Django is:

- Session management,
- Use authentication
- Cross-site request forgery protection
- Content Gzipping

---

17) What does of Django field class types do?

The Django field class types specify:

- The database column type.
- The default HTML widget to avail while rendering a form field.
- The minimal validation requirements used in Django admin.
- Automatic generated forms.

---

18) What is the usage of Django-admin.py and manage.py?

Django-admin.py: It is a Django's command line utility for administrative tasks.

Manage.py: It is an automatically created file in each Django project. It is a thin wrapper around the Django-admin.py. It has the following usage:

- It puts your project's package on sys.path.
- It sets the DJANGO_SETTING_MODULE environment variable to points to your project's setting.py file.

---

19) What are the signals in Django?

Signals are pieces of code which contain information about what is happening. A dispatcher is used to sending the signals and listen for those signals.

---

20) 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.

---

21) How to handle URLs in Django?

To handle URL, **django.urls** module is used by the Django framework.

Let's open the file **urls.py** of the project and see the what it looks like:

```
// urls.py

from django.contrib import admin
from django.urls import path

urlpatterns = [
    path('admin/', admin.site.urls),
]
```

See, Django already has mentioned a URL here for the admin. The path function takes the first argument as a route of string or regex type.

The view argument is a view function which is used to return a response (template) to the user.

The **django.urls** module contains various functions, **path(route,view,kwargs,name)** is one of those which is used to map the URL and call the specified view.

---

22) What is Django Exception?

An exception is an abnormal event that leads to program failure. To deal with this situation, Django uses its exception classes and supports all core Python exceptions as well. Django core exceptions classes are defined in django.core.exceptions module.

---

23) What are the different types of Django Exception Classes?

The django.core.exceptions module contains the following classes

Exception	Description
AppRegistryNotReady	It is raised when attempting to use models before the app loading process.
ObjectDoesNotExist	The base class for DoesNotExist exceptions.
EmptyResultSet	If a query does not return any result, this exception is raised.

FieldDoesNotExist	It raises when the requested field does not exist.
MultipleObjectsReturned	This exception is raised by a query if only one object is expected, but multiple objects are returned.
SuspiciousOperation	This exception is raised when a user has performed an operation that should be considered suspicious from a security perspective.
PermissionDenied	It is raised when a user does not have permission to perform the action requested.
ViewDoesNotExist	It is raised by django.urls when a requested view does not exist.
MiddlewareNotUsed	It is raised when a middleware is not used in the server configuration.
ImproperlyConfigured	The ImproperlyConfigured exception is raised when Django is somehow improperly configured.
FieldError	It is raised when there is a problem with a model field.
ValidationError	It is raised when data validation fails to form or model field validation.

24) What is Django Session?

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.

25) What is the role of Cookie in Django?

A cookie is a small piece of information which is stored in the client browser. It is used to store user's data in a file permanently (or for the specified time). Cookie has its expiry date and time and removes automatically when gets expire. Django provides built-in methods to set and fetch cookie.

The set_cookie() method is used to set a cookie and get() method is used to get the cookie.

The request.COOKIES['key'] array can also be used to get cookie values.

```
from django.shortcuts import render
from django.http import HttpResponse

def setcookie(request):
    response = HttpResponse("Cookie Set")
    response.set_cookie('java-tutorial', 'javatpoint.com')
    return response
```

```
def getcookie(request):
   tutorial  = request.COOKIES['java-tutorial']
   return HttpResponse("java tutorials @: "+  tutorial);
```