**What are the features of Django?**

* SEO Optimized
* Extremely fast
* Fully loaded [framework](https://www.edureka.co/blog/python-frameworks/) that comes along with authentications, content administrations, RSS feeds, etc
* Very secure thereby helping developers avoid common security mistakes such as cross-site request forgery (csrf), clickjacking, cross-site scripting, etc
* It is exceptionally scalable which in turn helps meet the heaviest traffic demands
* Immensely versatile which allows you to develop any kind of websites

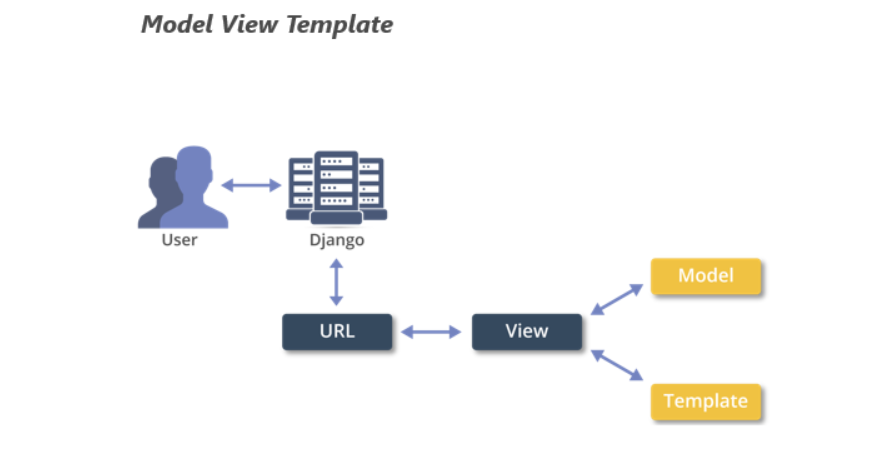
### **advantages of Django?**

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.

## ****Django architecture.****

Django follows the MVT or [Model View Template architecture](https://www.edureka.co/blog/django-tutorial/#architecture) whcih is based on the MVC or Model View Controller architecture. The main difference between these two is that Django itself takes care of the controller part.



the ‘view’ basically describes the data presented to the user. It *does not deal with how* *the data looks* but rather *what the data* *actually is*. Views are basically callback functions for the specified URL’s and these callback functions describe which data is presented.

The ‘templates’ on the other hand deal with the presentation of data, thereby, separating the content from its presentation. In Django, views delegate to the templates to present the data.

The ‘controller’ here is Django itself which sends the request to the appropriate view in accordance with the specified URL. This is why Django is referred to as MTV rather than MVC architecture.

**connect your Django project to the database?**

Django comes with a default [database](https://www.edureka.co/blog/category/databases/) which is SQLite. To connect your project to this database, use the following commands:

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

**Explain the file structure of a typical Django project**

When you create a project using the startproject command, the following files will be created:

A [***typical Django project***](https://data-flair.training/blogs/django-project-layout/) consists of these four files:

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

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

|  |  |
| --- | --- |
| File Name | Description |
| 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 | Consists of the settings for the current project |
| urls.py | Contains the URL’s for the current project |
| wsgi.py | This is an entry-point for the web servers to serve the project you have created |

## ****What are ‘Models’?****

Models are a single and definitive source for information about your data. It consists of all the essential fields and behaviors of the data you have stored. Often, each model will map to a single specific database table.

In Django, models serve as the abstraction layer that is used for structuring and manipulating your data. Django models are a subclass of the django.db.models.Model class and the attributes in the models represent database fields.

## ****Q12. What are ‘views’?****

Django views serve the purpose of encapsulation. They encapsulate the logic liable for processing a user’s request and for returning  
the response back to the user. Views in Django either return an HttpResponse or raise an exception such as Http404. HttpResponse contains the objects that consist of the content that is to be rendered to the user. Views can also be used to perform tasks such as read records from the database, delegate to the templates, generate a PDF file, etc.

## ****What are ‘templates’?****

Django’s template layer renders the information to be presented to the user in a designer-friendly format. Using templates, you can generate [HTML](https://www.edureka.co/blog/what-is-html/) dynamically. The HTML consists of both static as well as dynamic parts of the content. You can have any number of templates depending on the requirement of your project. It is also fine to have none of them.

Django has its own template system called the Django template language (DTL). Regardless of the backend, you can also load and render templates using Django’s standard admin.

## ****What is the difference between a Project and an App?****

An app is basically a Web Application that is created to do something for example, a database of employee records. A project, on the other hand, is a collection of apps of some particular website. Therefore, a single project can consist of ‘n’ number of apps and a single app can be in multiple projects.

**how a request is processed in Django?**

In case some user requests a page from some Django powered site, the system follows an algorithm that determines which Python code needs to be executed. Here are the steps that sum up the algorithm:

1. Django first determines which root URLconf or URL configuration module is to be used
2. Then, that particular Python module is loaded and then Django looks for the variable urlpatterns
3. These URL patterns are then run by Django, and it stops at the first match of the requested URL
4. Once that is done, the Django then imports and calls the given view
5. In case none of the URLs match the requested URL, Django invokes an error-handling view

## ****exception classes present in Django.****

Django uses its own exceptions as well as those present in Python. Django core exceptions are present in django.core.exceptions class some of which are mentioned in the table below:

|  |  |
| --- | --- |
| Exception | Description |
| AppRegistryNotReady | Raised when you try to use your models before the app loading process (initializes the ORM) is completed. |
| ObjectDoesNotExist | This is the base class for DoesNotExist exceptions |
| EmptyResultSet | This exception may be raised if a query won’t return any result |
| FieldDoesNotExist | This exception is raised by a model’s \_meta.get\_field() function in case the requested field does not exist |
| MultipleObjectsReturned | This is raised by a query if multiple objects are returned and only one object was expected |

**What happens when a typical Django website gets a request? Explain.**

**Ans.**When a user enters a URL in the browser the same request is received by the Django Server. The server then looks for the match of the requested URL in its URL-config and if the URL matches, it returns the corresponding view function. It will then request the data from the Model of that application, if any data is required and pass it to the corresponding template which is then rendered in the browser, otherwise, a 404 error is returned.

**Why is Django called loosely coupled framework?**

**Ans.**Django is called a loosely coupled framework because of the MTV architecture it’s based on. Django’s architecture is a variant of MVC architecture and MTV is useful because it completely separates server code from the client’s machine.

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

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

Therefore, Django is called a loosely coupled framework.

**Explain the importance of settings.py file and what data/ settings it contains.**

**Ans.**When Django server starts, it first looks for settings.py. As the name settings, it is the main settings file of your web application. Everything inside your Django project like databases, backend engines, middlewares, installed applications, main URL configurations, static file addresses, templating engines, allowed hosts and servers and security key stores in this file as a list or dictionary.

So, when your Django server starts it executes settings.py file and then loads particular engines and databases so that when a request is given it can serve the same quickly.

**What is Django ORM?**

**Ans.**Django ORM is one of the special feature-rich tools in Django. ORM is an acronym for Object-Relational Mapper. This ORM enables a developer to interact with a database in a pythonic way.

Django ORM is the abstraction between models (web application data-structure) and the database where the data is stored. It makes possible to retrieve, save, delete and perform other operations over the database without ever writing any SQL code.

It also covers many loopholes and takes all the field attributes and gives you more control over your code in Python rather than any database language.

**How does Django Templating Work?**

**Ans.**Django Templates are the Django’s answer to generate dynamic web pages. Templates, in general, are the HTML or the formats which can return as an Http response.

Django templating engine handles templating in the Django framework. There are some template syntaxes which declares variables, control logic, filters, and comments. After putting these inside the HTML structure, when the web page is requested and called upon by the view function, the Django Template engine gets two things, the HTML structure with variables in place and the data to replace with those variables. It replaces the variables with data while also executing the control logic and generating filters. It renders the required HTML and sends it to the browser when all the work gets complete.

**What are View functions? Can we directly import a function in URL?**

**Ans.** The View is the middle component in Django that receives data from the Django models and pass the same to the Templates.

Every application in Django comes with views.py file, this file contains the View functions.

The View functions are functions which receive an argument and they return a browser-renderable format or a redirect.

### **What is a QuerySet in Django?**

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

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

No, Django is not a Content Management System (CMS). Instead, it is a Web framework and a programming tool that helps you in building elegant websites.

### How can you set up static files in Django?

Basically, we require three main things 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

### List the database backends supported by Django Framework?

Django officially supports four database backends, they are

* PostgreSQL
* MySQL
* SQLite
* Oracle

### How to set/unset session in Django?

**Setting Session in Django**

request.session['key'] = 'value'

Unset Session in Django

del request.session['key']

### How to get current page URI in Django template.

You can use **{{ request.path }}** and **{{ request.get\_full\_path }}** to get current page URI in Django template.

**Explain the migration in Django and how you can do in SQL?**

Migration in Django is to make changes to your models like deleting a model, adding a field, etc. into your database schema.  There are several commands you use to interact with migrations.

* Migrate
* Makemigrations
* Sqlmigrate

To do the migration in SQL, you have to print the SQL statement for resetting sequences for a given app name.

**django-admin.py sqlsequencreset**

Use this command to generate SQL that will fix cases where a sequence is out sync with its automatically incremented field data.

1. **How to install django ?**

To install Django, we need to download and install Python as required depending on the operating system in use. t’s recommended  to create a virtual environment. Run the command **pip install “django>=2.2,<3”** on the terminal and wait until it’s successfully installed.

1. **How to check the latest version of Django?**

To check the latest version of Django, visit their homepage <https://www.djangoproject.com/> and on the right of the page you’ll see a “Download latest release” button with the latest version on it.

1. **How can you setup Database in Django?**

Create the initial Django project skeleton by using the **django-admin startproject** command.

Open the settings from the **settings.py**file to confirm all settings such as your timezone are suitable for your project, else make changes or add new settings where necessary.

Install MySQL Database Connector.

Create the database.

Add the MySQL Database Connection to your Application.

Test MySQL Connection to Application.

1. **at does Django templates contain ?**

Django templates contains the static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted.

1. **How to create super user in django ?**

To create a super user,,

* Create project using the **django-admin startproject** command.
* Move into the project location and run **python manage.py makemigrations && python manage.py migrate && python manage.py createsuperuser**

1. **What is ORM ? Advantages of ORM ?**

ORM (Object-relational mapping) is a programming technique for converting data between incompatible type systems using object-oriented programming languages.

Advantages include:

* Concurrency support
* Cache management

1. **How to create a model in django ?**

Add the model object in the **models.py**file, updated settings for the newly created app by adding it to the **INSTALLED\_APPS**section in **settings.py**, make migrations, and verify the database schema.

1. **What is migration in django ?**

Migrations are a way of propagating changes made in the model into the database schema (adding a field, deleting a model, etc.)

1. **How to do migrations in django ?**

To do migrations , create or update a model and in the app directory, run the command **./manage.py makemigrations <app name> && ./manage.py migrate <app name>**

1. **How a request is process in Django ?**

When the user makes a request of your application, a WSGI handler is instantiated, which:

* imports your settings.py file and Django’s exception classes.
* loads all the middleware classes it finds in the MIDDLEWARE\_CLASSES or MIDDLEWARES(depending on Django version) tuple located in settings.py
* builds four lists of methods which handle processing of request, view, response, and exception.
* loops through the request methods, running them in order
* resolves the requested URL
* loops through each of the view processing methods
* calls the view function (usually rendering a template)
* processes any exception methods
* loops through each of the response methods, (from the inside out, reverse order from request middlewares)
* finally builds a return value and calls the callback function to the web server

1. **How to make Foreign Key relation in django ?**

To make a foreign key relation , create a many-to-one relationship by calling the **ForeignKey()**function on the model with a value of the key’s name.