



Django Models

Last Updated : 11 Sep, 2024

A **Django model** is the built-in feature that Django uses to create tables, their fields, and various constraints. In short, Django Models is the SQL Database one uses with Django. SQL (Structured Query Language) is complex and involves a lot of different queries for creating, deleting, updating, or any other stuff related to a database. Django models simplify the tasks and organize tables into models. Generally, each model maps to a single database table.

Django Models

Create a model in Django to store data in the database conveniently. Moreover, we can use the admin panel of Django to create, update, delete, or retrieve fields of a model and various similar operations. Django models provide simplicity, consistency, version control, and advanced metadata handling. The basics of a model include –

- Each model is a Python class that subclasses `django.db.models.Model`.
- Each attribute of the model represents a database field.
- With all of this, Django gives you an automatically generated database-access API; see [Making queries](#).

Example

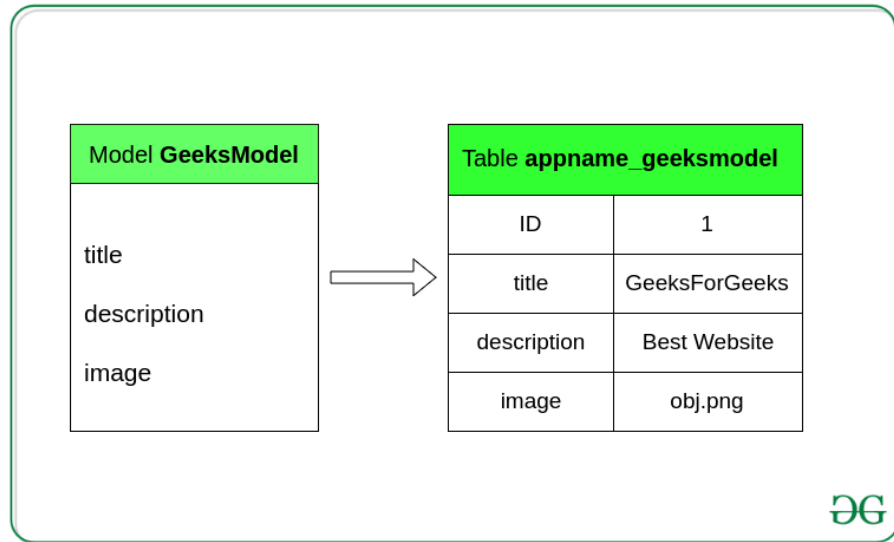
Python



```
1 from django.db import models
2
3 # Create your models here.
4 class GeeksModel(models.Model):
5     title = models.CharField(max_length = 200)
6     description = models.TextField()
```

This program defines a Django model called “GeeksModel” which has two fields: “title” and “description”. “title” is a character field with a maximum length of 200 characters and “description” is a text field with no maximum length. This model can be used to create and interact with a corresponding database table for storing “GeeksModel” objects. Without any additional code or context, running this program will not produce any visible output.

Django maps the fields defined in Django models into table fields of the database as shown below.



Model in Django

To create Django Models, one needs to have a project and an app working in it. After you start an app you can create models in `app/models.py`. Before starting to use a model let's check how to start a project and create an app named `geeks.py`

Refer to the following articles to check how to create a project and an app in Django.

- [How to Create a Basic Project using MVT in Django?](#)
- [How to Create an App in Django ?](#)

Create Model in Django

Syntax

```
from django.db import models

class ModelName(models.Model):
    field_name = models.Field(**options)
```

To create a model, in geeks/models.py Enter the code,

Python



```
1 # import the standard Django Model
2 # from built-in library
3 from django.db import models
4
5 # declare a new model with a name &quot;GeeksModel&quot;;
6
7
8 class GeeksModel(models.Model):
9     # fields of the model
10     title = models.CharField(max_length=200)
11     description = models.TextField()
12     last_modified = models.DateTimeField(auto_now_add=True)
13     img = models.ImageField(upload_to=&quot;
14                             images/&quot;
15                             )
16
17     # renames the instances of the model
18     # with their title name
19     def __str__(self):
20         return self.title
```

This code defines a new Django model called “GeeksModel” which has four fields: “title” (a character field with a maximum length of 200), “description” (a text field), “last_modified” (a date and time field that automatically sets the date and time of creation), and “img” (an image field that will be uploaded to a directory called “images”). The __str__ method is also defined to return the title of the instance of the model when the model is printed.

This code does not produce any output. It is defining a model class which can be used to create database tables and store data in Django.

Whenever we create a Model, Delete a Model, or update anything in any of models.py of our project. We need to run two commands makemigrations and migrate. makemigrations basically generates the SQL commands for preinstalled apps (which can be viewed in installed apps in settings.py) and your newly created app's model which you add in installed apps whereas migrate executes those SQL commands in the database file.

So when we run,

Python manage.py makemigrations

SQL Query to create above Model as a Table is created and

Python manage.py migrate

creates the table in the database.

Now we have created a model we can perform various operations such as creating a Row for the table or in terms of Django Creating an instance of Model. To know more visit – [Django Basic App Model – Makemigrations and Migrate](#)

Render a Model in Django Admin Interface

To render a model in Django admin, we need to modify app/admin.py. Go to admin.py in geeks app and enter the following code. Import the corresponding model from models.py and register it to the admin interface.

Python

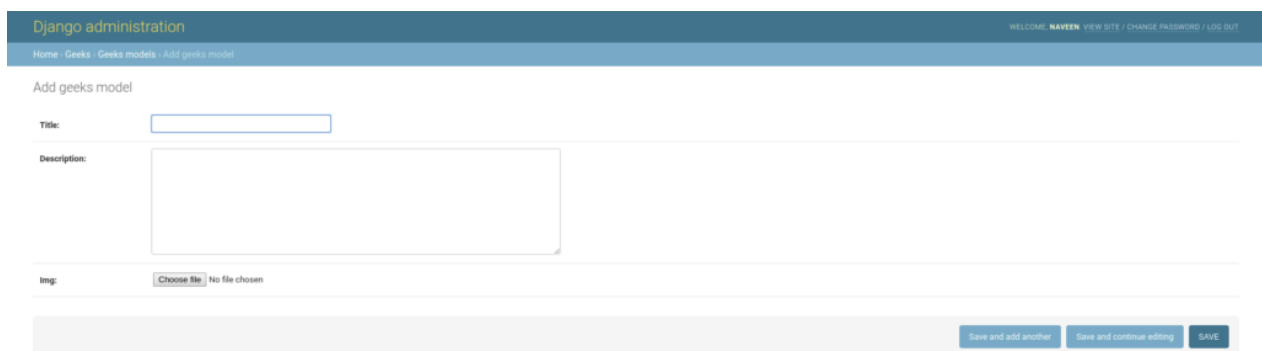
```
1 from django.contrib import admin
2
3 # Register your models here.
4 from .models import GeeksModel
5
6 admin.site.register(GeeksModel)
```

This code imports the admin module from the django.contrib package and the GeeksModel class from the models module in the current directory. It then

registers the GeeksModel class with the Django admin site, which allows the model to be managed through the Django admin interface. This means you can perform CRUD(Create, Read, Update, Delete) operations on the GeeksModel using the Django admin interface.

This code does not produce any output, it simply registers the GeeksModel with the admin site, so that it can be managed via the Django admin interface.

Now we can check whether the model has been rendered in Django Admin. Django Admin Interface can be used to graphically implement CRUD (Create, Retrieve, Update, Delete).

The screenshot shows the Django Admin interface for adding a new 'geeks model'. The page has a blue header with 'Django administration' and a user welcome message. Below the header, there's a breadcrumb trail: 'Home > Geeks > Geeks models > Add geeks model'. The main content area is titled 'Add geeks model' and contains a form with three fields: 'Title' (a text input), 'Description' (a large text area), and 'img' (a file upload button labeled 'Choose file' with 'No file chosen' text). At the bottom right of the form, there are three buttons: 'Save and add another', 'Save and continue editing', and 'SAVE'.

To check more on rendering models in django admin, visit – [Render Model in Django Admin Interface](#)

Django CRUD – Inserting, Updating and Deleting Data

Django lets us interact with its database models, i.e. add, delete, modify and query objects, using a database-abstraction API called ORM(Object Relational Mapper). We can access the Django ORM by running the following command inside our project directory.

```
python manage.py shell
```

Adding Objects

To create an object of model Album and save it into the database, we need to write the following command:

```
>>> a = GeeksModel(  
    title = "GeeksForGeeks",  
    description = "A description here",  
    img = "geeks/abc.png"  
)  
>>> a.save()
```

Retrieving Objects

To retrieve all the objects of a model, we write the following command:

```
>>> GeeksModel.objects.all()  
<QuerySet [<GeeksModel: Divide>, <GeeksModel: Abbey Road>, <GeeksModel:  
Revolver>]>
```

Modifying existing Objects

We can modify an existing object as follows:

```
>>> a = GeeksModel.objects.get(id = 3)  
>>> a.title = "Pop"  
>>> a.save()
```

Deleting Objects

To delete a single object, we need to write the following commands:

```
>>> a = Album.objects.get(id = 2)  
>>> a.delete()
```

To check detailed post of Django's ORM (Object) visit [Django ORM – Inserting, Updating & Deleting Data](https://www.geeksforgeeks.org/django-orm-inserting-updating-deleting-data/)

Validation on Fields in a Model

Built-in Field Validations in Django models are the default validations that come predefined to all Django fields. Every field comes in with built-in validations from Django validators. For example, IntegerField comes with built-

in validation that it can only store integer values and that too in a particular range.

Enter the following code into models.py file of **geeks** app.

Python



```
1 from django.db import models
2 from django.db.models import Model
3 # Create your models here.
4
5 class GeeksModel(Model):
6     geeks_field = models.IntegerField()
7
8     def __str__(self):
9         return self.geeks_field
```

After running makemigrations and migrate on Django and rendering above model, let us try to create an instance using string “GfG is Best”.

Add geeks model

Please correct the error below.

This field is required.

Geeks field:

You can see in the admin interface, one can not enter a string in an IntegerField. Similarly every field has its own validations. To know more about validations visit, [Built-in Field Validations – Django Models](#)

More on Django Models –

- [Change Object Display Name using __str__ function – Django Models](#)

- [Custom Field Validations in Django Models](#)
- [Django python manage.py migrate command](#)
- [Django App Model – Python manage.py makemigrations command](#)
- [Django model data types and fields list](#)
- [How to use Django Field Choices ?](#)
- [Overriding the save method – Django Models](#)

Django Model data types and fields list

The most important part of a model and the only required part of a model is the list of database fields it defines. Fields are specified by class attributes. Here is a list of all Field types used in Django.

Field Name	Description
AutoField	It An IntegerField that automatically increments.
BigAutoField	It is a 64-bit integer, much like an AutoField except that it is guaranteed to fit numbers from 1 to 9223372036854775807.
BigIntegerField	It is a 64-bit integer, much like an IntegerField except that it is guaranteed to fit numbers from -9223372036854775808 to 9223372036854775807.
BinaryField	A field to store raw binary data.
BooleanField	A true/false field. The default form widget for this field is a <code>CheckboxInput</code> .
CharField	It is string filed for small to large-sized input
DateField	A date, represented in Python by a <code>datetime.date</code> instance

Field Name	Description
	It is used for date and time, represented in Python by a <code>datetime.datetime</code> instance.
<u>DecimalField</u>	It is a fixed-precision decimal number, represented in Python by a <code>Decimal</code> instance.
<u>DurationField</u>	A field for storing periods of time.
<u>EmailField</u>	It is a <code>CharField</code> that checks that the value is a valid email address.
<u>FileField</u>	It is a file-upload field.
<u>FloatField</u>	It is a floating-point number represented in Python by a <code>float</code> instance.
<u>ImageField</u>	It inherits all attributes and methods from <code>FileField</code> , but also validates that the uploaded object is a valid image.
<u>IntegerField</u>	It is an integer field. Values from -2147483648 to 2147483647 are safe in all databases supported by Django.
<u>GenericIPAddressField</u>	An IPv4 or IPv6 address, in string format (e.g. 192.0.2.30 or 2a02:42fe::4).
<u>NullBooleanField</u>	Like a <code>BooleanField</code> , but allows <code>NULL</code> as one of the options.
<u>PositiveIntegerField</u>	Like an <code>IntegerField</code> , but must be either positive or zero (0).
<u>PositiveSmallIntegerField</u>	Like a <code>PositiveIntegerField</code> , but only allows values under a certain (database-dependent) point.

Field Name	Description
SlugField	Slug is a newspaper term. A slug is a short label for something, containing only letters, numbers, underscores or hyphens. They're generally used in URLs.
SmallIntegerField	It is like an IntegerField, but only allows values under a certain (database-dependent) point.
TextField	A large text field. The default form widget for this field is a Textarea.
TimeField	A time, represented in Python by a datetime.time instance.
URLField	A CharField for a URL, validated by URLValidator.
UUIDField	A field for storing universally unique identifiers. Uses Python's UUID class. When used on PostgreSQL, this stores in a uuid datatype, otherwise in a char(32).

Relationship Fields

Django also defines a set of fields that represent relations.

Field Name	Description
ForeignKey	A many-to-one relationship. Requires two positional arguments: the class to which the model is related and the on_delete option.
ManyToManyField	A many-to-many relationship. Requires a positional argument: the class to which the model is related, which works exactly the same as it does for ForeignKey, including recursive and lazy relationships.

Field Name	Description
OneToOneField	A one-to-one relationship. Conceptually, this is similar to a ForeignKey with unique=True, but the “reverse” side of the relation will directly return a single object.

Field Options

Field Options are the arguments given to each field for applying some constraint or imparting a particular characteristic to a particular Field. For example, adding an argument `null = True` to CharField will enable it to store empty values for that table in relational database.

Here are the field options and attributes that an CharField can use.

Field Options	Description
Null	If True , Django will store empty values as NULL in the database. Default is False .
Blank	If True , the field is allowed to be blank. Default is False .
<code>db_column</code>	The name of the database column to use for this field. If this isn't given, Django will use the field's name.
Default	The default value for the field. This can be a value or a callable object. If callable it will be called every time a new object is created.
help_text	Extra “help” text to be displayed with the form widget. It's useful for documentation even if your field isn't used on a form.
primary_key	If True , this field is the primary key for the model.

Field Options	Description
<u>editable</u>	If False , the field will not be displayed in the admin or any other ModelForm. They are also skipped during model validation. Default is True .
<u>error_messages</u>	The error_messages argument lets you override the default messages that the field will raise. Pass in a dictionary with keys matching the error messages you want to override.
<u>help_text</u>	Extra “help” text to be displayed with the form widget. It’s useful for documentation even if your field isn’t used on a form.
<u>verbose_name</u>	A human-readable name for the field. If the verbose name isn’t given, Django will automatically create it using the field’s attribute name, converting underscores to spaces.
<u>validators</u>	A list of validators to run for this field. See the <u>validators documentation</u> for more information.
<u>Unique</u>	If True, this field must be unique throughout the table.

Are you ready to elevate your web development skills from foundational knowledge to advanced expertise? Explore our [Mastering Django Framework - Beginner to Advanced Course](#) on GeeksforGeeks, designed for aspiring developers and experienced programmers. This comprehensive course covers everything you need to know about Django, from the basics to advanced features. Gain practical experience through **hands-on projects** and real-world applications, mastering essential Django principles and techniques. Whether

you're just starting or looking to refine your skills, this course will empower you to build sophisticated web applications efficiently. Ready to enhance your web development journey? Enroll now and unlock your potential with Django!

N Nave...



Previous Article

Django Static File

Next Article

Django model data types and fields list

Similar Reads

Django Models | Set - 1

Prerequisites : Django Creating apps Models - According to Django Models, A model is the single, definitive source of information about your data. It contains...

2 min read

Django Models | Set - 2

Model Fields - Model fields define the datatype which will be stored in the provided variable. To store price, the integer type is more suitable. To store heig...

2 min read

Python | Relational fields in Django models

Prerequisite: Django models Django models represent real-world entities, and it is rarely the case that real-world entities are entirely independent of each other...

4 min read

BigIntegerField - Django Models

BigIntegerField is a 64-bit integer, much like an IntegerField except that it is guaranteed to fit numbers from -9223372036854775808 to...

4 min read

AutoField - Django Models

According to documentation, An AutoField is an IntegerField that automatically increments according to available IDs. One usually won't need to use this directl...

4 min read

BigAutoField - Django Models

BigAutoField is a 64-bit integer, much like an AutoField except that it is guaranteed to fit numbers from 1 to 9223372036854775807. One can create a...

4 min read

BooleanField - Django Models

BooleanField is a true/false field. It is like a bool field in C/C++. The default form widget for this field is CheckboxInput, or NullBooleanSelect if null=True. The...

4 min read

CharField - Django Models

CharField is a string field, for small- to large-sized strings. It is like a string field in C/C++. CharField is generally used for storing small strings like first name, last...

4 min read

DateField - Django Models

DateField is a field that stores date, represented in Python by a datetime.date instance. As the name suggests, this field is used to store an object of date...

5 min read

DateTimeField - Django Models

DateTimeField is a date and time field which stores date, represented in Python by a datetime.datetime instance. As the name suggests, this field is used to stor...

5 min read

Article Tags : [Python](#) [Django-models](#) [Python Django](#)

Practice Tags : [python](#)

Corporate & Communications Address:-
A-143, 9th Floor, Sovereign Corporate
Tower, Sector- 136, Noida, Uttar Pradesh
(201305) | Registered Address:- K 061,
Tower K, Gulshan Vivante Apartment,
Sector 137, Noida, Gautam Buddh
Nagar, Uttar Pradesh, 201305



Company

About Us
Legal
In Media
Contact Us
Advertise with us
GFG Corporate Solution
Placement Training Program
GeeksforGeeks Community

DSA

Data Structures
Algorithms
DSA for Beginners
Basic DSA Problems
DSA Roadmap
Top 100 DSA Interview Problems
DSA Roadmap by Sandeep Jain
All Cheat Sheets

Web Technologies

HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
Bootstrap
Web Design

Computer Science

Languages

Python
Java
C++
PHP
GoLang
SQL
R Language
Android Tutorial
Tutorials Archive

Data Science & ML

Data Science With Python
Data Science For Beginner
Machine Learning
ML Maths
Data Visualisation
Pandas
NumPy
NLP
Deep Learning

Python Tutorial

Python Programming Examples
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question
Django

DevOps

Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths
Software Development
Software Testing

System Design

High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns
OOAD
System Design Bootcamp
Interview Questions

School Subjects

Mathematics
Physics
Chemistry
Biology
Social Science
English Grammar
Commerce
World GK

Git
Linux
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

Interview Preparation

Competitive Programming
Top DS or Algo for CP
Company-Wise Recruitment Process
Company-Wise Preparation
Aptitude Preparation
Puzzles

GeeksforGeeks Videos

DSA
Python
Java
C++
Web Development
Data Science
CS Subjects

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved