



**FREE eBook**

# LEARNING django-models

Free unaffiliated eBook created from  
**Stack Overflow contributors.**

#django-  
models

# Table of Contents

<b>About</b> .....	<b>1</b>
<b>Chapter 1: Getting started with django-models</b> .....	<b>2</b>
Remarks.....	2
Examples.....	2
Installation or Setup.....	2
What is Django models?.....	3
Django Model example.....	3
<b>Credits</b> .....	<b>5</b>

---

# About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [django-models](#)

It is an unofficial and free django-models ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official django-models.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to [info@zzzprojects.com](mailto:info@zzzprojects.com)

---

# Chapter 1: Getting started with django-models

## Remarks

This section provides an overview of what django-models is, and why a developer might want to use it.

It should also mention any large subjects within django-models, and link out to the related topics. Since the Documentation for django-models is new, you may need to create initial versions of those related topics.

## Examples

### Installation or Setup

Generally, each model maps to a single database table. We to write the field type, limits, size, etc in model.py file of the app. This will create the necessary table and fields in the database.

```
''' models.py '''
from django.db import models

class table_name(models.Model):
    field_name= models.field_type(conditions)
```

Next we need to inform Django in `settings.py` about the app which will be using this model.

```
''' settings.py '''

INSTALLED_APPS = [
    #...
    'app_name',
    #... ]
```

We are almost done. Next we need to migrate this app so that database tables are created. In terminal type the following:

```
python manage.py migrate
```

`migrate` will create the necessary databases by checking the `app_installed` in the `setting.py`

By `makemigrations`, Django will know the changes that are made to the models.

```
python manage.py makemigrations
```

That's it. Your database is created and you can see the schema in the terminal

```
python manage.py sqlmigrate app_name 0001
```

## What is Django models?

A Django model typically refers to a table in the database, attributes of that model becomes the column of that table. In more of a real-world example, you would create a model for any entity in your application, and store its attributes with Django fields which automatically handles data-types conversions for the database you would be using.

One of the great features for Django is its ORM, you don't have to write any database query, and even it's recommended NOT to write one when using Django. ORM converts your Django models and all the operations you do with it to the corresponding database queries. This means that all the manipulation you have to do, it now with the python objects created out from that model, and all the underlying database stuff would be taken care by Django's ORM. There are a bunch of tweaks and customizations you could do with it.

Django's ORM supports all the major database like Postgres, MySQL, sqlite3, and other enterprises database provided with proper drivers. This also means that you don't have to care what underlying database you are using, or even if you want to shift from one database from another, you can do it without changing a single line of your application logic, just change the database string from settings.py, dump the old data, and you should be good to go.

## Django Model example

A simple example would be for a library management application; you would have 2 models, for example, student and book

in models.py:

```
from django.db import models

class student(models.Model):
    roll_no = models.IntegerField(primary_key=True)
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
```

Here we have given roll\_no a primary key to the student model, but even if we don't give a primary key to any attribute, Django would automatically assign an attribute called id, which would be automatically assigned and incremented on the creation of new rows.

Now you can just import this model into your views or in a project and interact with it by simply creating an object of that model.

Django has many inbuilt Fields available, or even you can create your own as well.

Django also supports relationships between models, many-to-many, one-to-one, many-to-one.

Django's detailed doc for [Models](#)

Read [Getting started with django-models](https://riptutorial.com/django-models/topic/7575/getting-started-with-django-models) online: <https://riptutorial.com/django-models/topic/7575/getting-started-with-django-models>

---

# Credits

S. No	Chapters	Contributors
1	Getting started with django-models	<a href="#">Community</a> , <a href="#">Md.Sifatul Islam</a> , <a href="#">Vatsal Parekh</a>