Django model data types and fields list

Prerequisiste – Django ORM – Inserting, Updating & Deleting Data

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. Be careful not to choose field names that conflict with the models API like clean, save, or delete.

Example:

```
from django.db import models

class Musician(models.Model):
    first_name = models.CharField(max_length=200)
    last_name = models.CharField(max_length=200)
    instrument = models.CharField(max_length=200)

class Album(models.Model):
    artist = models.ForeignKey(Musician, on_delete=models.CASCADE)
    name = models.CharField(max_length=100)
    release_date = models.DateField()
    num_stars = models.IntegerField()
```

Setting a field for storing any type of data is like deciding a data type in C/C++ for storing a particular integer, char etc. Fields in Django are the data types to store a particular type of data. For example, to store an integer, IntegerField would be used. These fields have in-built validation for a particular data type, that is you can not store “abc” in an IntegerField. Similarly, for other fields. This post revolves about major fields one can use in Django Models.
Here are some key attributes one should be aware of before starting to use Django Fields.

Field types

Each field in the model should be an instance of the appropriate Field class. Django uses field class types to determine a few things:

- The column type, which tells the database what kind of data to store (e.g. INTEGER, VARCHAR, TEXT).
- The default HTML widget to use when rendering a form field (e.g. <input type=”text”>, <select>).
- The minimal validation requirements, used in Django’s admin and in automatically-generated forms.

Django ships with dozens of built-in field types which can be used to save any type of data from number to entire HTML file too. Here is a list of all Field types used in Django.

Basic model data types and fields list

FIELD NAME	CLASS	DESCRIPTION
AutoField	class AutoField(**options)	It An IntegerField that automatically increments.
BigAutoField	class BigAutoField(**options)	It is a 64-bit integer, much like an AutoField except that it is guaranteed to fit numbers from 1 to 9223372036854775807.
BigIntegerField	class BigIntegerField(**options)	It is a 64-bit integer, much like an IntegerField except that it is guaranteed

		to fit numbers from -9223372036854775808 to 9223372036854775807.
BinaryField	class BinaryField(**options)	A field to store raw binary data.
BooleanField	class BooleanField(**options)	A true/false field. The default form widget for this field is a CheckboxInput.
CharField	class DateField(auto_now=False, auto_now_add=False, **options)	It is a date, represented in Python by a datetime.date instance.
DateField	class DateField(auto_now=False, auto_now_add=False, **options)	A date, represented in Python by a datetime.date instance
DateTimeField	class DateTimeField(auto_now=False, auto_now_add=False, **options)	It is used for date and time, represented in Python by a datetime.datetime instance.
DecimalField	class DecimalField(max_digits=None, decimal_places=None, **options)	It is a fixed-precision decimal number, represented in Python by a Decimal instance.
DurationField	class DurationField(**options)	A field for storing periods of time.
EmailField	class EmailField(max_length=254, **options)	It is a CharField that checks that the value is a valid email address.
FileField	class FileField(upload_to=None, max_length=100, **options)	It is a file-upload field.
FloatField	class FloatField(**options)	It is a floating-point number represented in Python by a float instance.
ImageField	class ImageField(upload_to=None, height_field=None, width_field=None, max_length=100, **options)	It inherits all attributes and methods from FileField, but also validates that the uploaded object is a valid image.
IntegerField	class IntegerField(**options)	It is an integer field. Values from -2147483648 to 2147483647 are safe in all databases supported by Django.
GenericIPAddressField	class GenericIPAddressField(protocol='both', unpack_ipv4=False, **options)	An IPv4 or IPv6 address, in string format (e.g. 192.0.2.30 or 2a02:42fe::4).
NullBooleanField	class NullBooleanField(**options)	Like a BooleanField, but allows NULL as one of the options.
PositiveIntegerField	class PositiveIntegerField(**options)	Like an IntegerField, but must be either positive or zero (0).

PositiveSmallIntegerField	class PositiveSmallIntegerField(**options)	Like a PositiveIntegerField, but only allows values under a certain (database-dependent) point.
SlugField	class SlugField(max_length=50, **options)	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	class SmallIntegerField(**options)	It is like an IntegerField, but only allows values under a certain (database-dependent) point.
TextField	class TextField(**options)	A large text field. The default form widget for this field is a Textarea.
TimeField	class TimeField(auto_now=False, auto_now_add=False, **options)	A time, represented in Python by a datetime.time instance.
URLField	class URLField(max_length=200, **options)	A CharField for a URL, validated by URLValidator.
UUIDField	class UUIDField(**options)	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	CLASS	DESCRIPTION
ForeignKey	class ForeignKey(to, on_delete, **options)	A many-to-one relationship. Requires two positional arguments: the class to which the model is related and the on_delete option.
ManyToManyField	class ManyToManyField(to, **options)	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.
OneToOneField	class OneToOneField(to, on_delete, parent_link=False, **options)	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.

Recommended Posts:

Intermediate fields in Django | Python

Python | Relational fields in Django models

Render Django Form Fields Manually

Python | Convert mixed data types tuple list to string list

Django Basic App Model - Makemigrations and Migrate

Render Model in Django Admin Interface

Add the slug field inside Django Model

Django App Model - Python manage.py makemigrations command

Data Classes in Python | Set 3 (dataclass fields)

Data Visualization Using Chartjs and Django

Django ORM - Inserting, Updating & Deleting Data

Initial form data - Django Forms

Using Google Cloud Function to generate data for Machine Learning model

List View - Function based Views Django

{{ form.as_ul }} - Render Django Forms as list

NaveenArora
Check out this Author's contributed articles.

If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please Improve this article if you find anything incorrect by clicking on the "Improve Article" button below.

Article Tags : Python Django-models Python Django

1

0

No votes yet.

To-do Done

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments