**模型和字段**

一个模型（model）就是一个单独的、确定的数据的信息源，包含了数据的字段和操作方法。通常，每个模型映射为一张数据库中的表。

基本的原则如下：

* 每个模型在Django中的存在形式为一个Python类
* 每个模型都是django.db.models.Model的子类
* 模型的每个字段（属性）代表数据表的某一列
* Django将自动为你生成数据库访问API

**简单示例：**

下面的模型定义了一个“人”，它具有first\_name和last\_name字段：

**from** django.db **import** models

**class** **Person**(models**.**Model):

first\_name **=** models**.**CharField(max\_length**=**30)

last\_name **=** models**.**CharField(max\_length**=**30)

每一个字段都是一个类属性，每个类属性表示数据表中的一个列。

上面的代码，相当于下面的原生SQL语句：

**CREATE** **TABLE** myapp\_person (

"id" serial **NOT** **NULL** **PRIMARY** **KEY**,

"first\_name" varchar(30) **NOT** **NULL**,

"last\_name" varchar(30) **NOT** **NULL**

);

注意：

* 表名myapp\_person由Django自动生成，默认格式为“项目名称+下划线+小写类名”，你可以重写这个规则。
* Django默认自动创建自增主键id，当然，你也可以自己指定主键。
* 上面的SQL语句基于PostgreSQL语法。

通常，我们会将模型编写在其所属app下的models.py文件中，没有特别需求时，请坚持这个原则，不要自己给自己添加麻烦。

创建了模型之后，在使用它之前，你需要先在settings文件中的INSTALLED\_APPS 处，注册models.py文件所在的myapp。看清楚了，是注册app，不是模型，也不是models.py。如果你以前写过模型，可能已经做过这一步工作，可跳过。

INSTALLED\_APPS = [

#...

'myapp',

#...

]

**当你每次对模型进行增、删、修改时，请务必执行命令python manage.py migrate，让操作实际应用到数据库上。这里可以选择在执行migrate之前，先执行python manage.py makemigrations让修改动作保存到记录文件中，方便github等工具的使用。**

**模型字段fields**

字段是模型中最重要的内容之一，也是唯一必须的部分。字段在Python中表现为一个类属性，体现了数据表中的一个列。请不要使用clean、save、delete等Django内置的模型API名字，防止命名冲突。下面是一个展示，注意字段的写法：

**from** django.db **import** models

**class** **Musician**(models**.**Model):

first\_name **=** models**.**CharField(max\_length**=**50)

last\_name **=** models**.**CharField(max\_length**=**50)

instrument **=** models**.**CharField(max\_length**=**100)

**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()

**字段命名约束：**

Django不允许下面两种字段名：

* 与Python关键字冲突。这会导致语法错误。例如：

class Example(models.Model): pass = models.IntegerField() # 'pass'是Python保留字！

* 字段名中不能有两个以上下划线在一起，因为两个下划线是Django的查询语法。例如：

class Example(models.Model): foo\_\_bar = models.IntegerField() # 'foo\_\_bar' 有两个下划线在一起!

由于你可以自定义表名、列名，上面的规则可能被绕开，但是请养成良好的习惯，一定不要那么起名。

SQL语言的join、where和select等保留字可以作为字段名，因为Django对它们都进行了转义。

**常用字段类型**

字段类型的作用：

* 决定数据库中对应列的数据类型(例如：INTEGER, VARCHAR, TEXT)
* HTML中对应的表单标签的类型，例如<input type=“text” />
* 在admin后台和自动生成的表单中最小的数据验证需求

Django内置了许多字段类型，它们都位于django.db.models中，例如models.CharField。这些类型基本满足需求，如果还不够，你也可以自定义字段。

下表列出了所有Django内置的字段类型，但不包括关系字段类型（字段名采用驼峰命名法，初学者请一定要注意）：

| **类型** | **说明** |
| --- | --- |
| AutoField | 一个自动增加的整数类型字段。通常你不需要自己编写它，Django会自动帮你添加字段：id = models.AutoField(primary\_key=True)，这是一个自增字段，从1开始计数。如果你非要自己设置主键，那么请务必将字段设置为primary\_key=True。Django在一个模型中只允许有一个自增字段，并且该字段必须为主键！字段必须为主键！ |
| BigAutoField | (1.10新增)64位整数类型自增字段，数字范围更大，从1到9223372036854775807 |
| BigIntegerField | 64位整数字段（看清楚，非自增），类似IntegerField ，-9223372036854775808 到9223372036854775807。在Django的模板表单里体现为一个textinput标签。 |
| BinaryField | 二进制数据类型。使用受限，少用。 |
| **BooleanField** | 布尔值类型。默认值是None。在HTML表单中体现为CheckboxInput标签。如果要接收null值，请使用NullBooleanField。 |
| **CharField** | 字符串类型。必须接收一个max\_length参数，表示字符串长度不能超过该值。默认的表单标签是input text。最常用的filed，没有之一！ |
| CommaSeparatedIntegerField | 逗号分隔的整数类型。必须接收一个max\_length参数。常用于表示较大的金额数目，例如1,000,000元。 |
| **DateField** | class DateField(auto\_now=False, auto\_now\_add=False, \*\*options)日期类型。一个Python中的datetime.date的实例。在HTML中表现为TextInput标签。在admin后台中，Django会帮你自动添加一个JS的日历表和一个“Today”快捷方式，以及附加的日期合法性验证。两个重要参数：（参数互斥，不能共存）auto\_now:每当对象被保存时将字段设为当前日期，常用于保存最后修改时间。auto\_now\_add：每当对象被创建时，设为当前日期，常用于保存创建日期(注意，它是不可修改的)。设置上面两个参数就相当于给field添加了editable=False和blank=True属性。如果想具有修改属性，请用default参数。例子：pub\_time = models.DateField(auto\_now\_add=True)，自动添加发布时间。 |
| DateTimeField | 日期时间类型。Python的datetime.datetime的实例。与DateField相比就是多了小时、分和秒的显示，其它功能、参数、用法、默认值等等都一样。 |
| DecimalField | 固定精度的十进制小数。相当于Python的Decimal实例，必须提供两个指定的参数！参数max\_digits：最大的位数，必须大于或等于小数点位数 。decimal\_places：小数点位数，精度。 当localize=False时，它在HTML表现为NumberInput标签，否则是text类型。例子：储存最大不超过999，带有2位小数位精度的数，定义如下：models.DecimalField(..., max\_digits=5, decimal\_places=2)。 |
| DurationField | 持续时间类型。存储一定期间的时间长度。类似Python中的timedelta。在不同的数据库实现中有不同的表示方法。常用于进行时间之间的加减运算。但是小心了，这里有坑，PostgreSQL等数据库之间有兼容性问题！ |
| **EmailField** | 邮箱类型，默认max\_length最大长度254位。使用这个字段的好处是，可以使用DJango内置的EmailValidator进行邮箱地址合法性验证。 |
| **FileField** | class FileField(upload\_to=None, max\_length=100, \*\*options)上传文件类型，后面单独介绍。 |
| FilePathField | 文件路径类型，后面单独介绍 |
| FloatField | 浮点数类型，参考整数类型 |
| **ImageField** | 图像类型，后面单独介绍。 |
| **IntegerField** | 整数类型，最常用的字段之一。取值范围-2147483648到2147483647。在HTML中表现为NumberInput标签。 |
| **GenericIPAddressField** | class GenericIPAddressField(protocol='both', unpack\_ipv4=False, \*\*options)[source],IPV4或者IPV6地址，字符串形式，例如192.0.2.30或者2a02:42fe::4在HTML中表现为TextInput标签。参数protocol默认值为‘both’，可选‘IPv4’或者‘IPv6’，表示你的IP地址类型。 |
| NullBooleanField | 类似布尔字段，只不过额外允许NULL作为选项之一。 |
| PositiveIntegerField | 正整数字段，包含0,最大2147483647。 |
| PositiveSmallIntegerField | 较小的正整数字段，从0到32767。 |
| SlugField | slug是一个新闻行业的术语。一个slug就是一个某种东西的简短标签，包含字母、数字、下划线或者连接线，通常用于URLs中。可以设置max\_length参数，默认为50。 |
| SmallIntegerField | 小整数，包含-32768到32767。 |
| **TextField** | 大量文本内容，在HTML中表现为Textarea标签，最常用的字段类型之一！如果你为它设置一个max\_length参数，那么在前端页面中会受到输入字符数量限制，然而在模型和数据库层面却不受影响。只有CharField才能同时作用于两者。 |
| TimeField | 时间字段，Python中datetime.time的实例。接收同DateField一样的参数，只作用于小时、分和秒。 |
| **URLField** | 一个用于保存URL地址的字符串类型，默认最大长度200。 |
| **UUIDField** | 用于保存通用唯一识别码（Universally Unique Identifier）的字段。使用Python的UUID类。在PostgreSQL数据库中保存为uuid类型，其它数据库中为char(32)。这个字段是自增主键的最佳替代品，后面有例子展示。 |

### 这里有如何上传文件和图片的方法：

### 1.****FileField：****

**class** FileField(upload\_to**=**None, max\_length**=**100, **\*\***options)[source]

上传文件字段（不能设置为主键）。默认情况下，该字段在HTML中表现为一个ClearableFileInput标签。在数据库内，我们实际保存的是一个字符串类型，默认最大长度100，可以通过max\_length参数自定义。真实的文件是保存在服务器的文件系统内的。

重要参数upload\_to用于设置上传地址的目录和文件名。如下例所示：

**class** **MyModel**(models**.**Model):

*# 文件被传至`MEDIA\_ROOT/uploads`目录，MEDIA\_ROOT由你在settings文件中设置*

upload **=** models**.**FileField(upload\_to**=**'uploads/')

*# 或者*

*# 被传到`MEDIA\_ROOT/uploads/2015/01/30`目录，增加了一个时间划分*

upload **=** models**.**FileField(upload\_to**=**'uploads/%Y/%m/%d/')

**Django很人性化地帮我们实现了根据日期生成目录的方式！**

**upload\_to参数也可以接收一个回调函数，该函数返回具体的路径字符串**，如下例：

**def** **user\_directory\_path**(instance, filename):

*#文件上传到MEDIA\_ROOT/user\_<id>/<filename>目录中*

**return** 'user\_{0}/{1}'**.**format(instance**.**user**.**id, filename)

**class** **MyModel**(models**.**Model):

upload **=** models**.**FileField(upload\_to**=**user\_directory\_path)

例子中，user\_directory\_path这种回调函数，必须接收两个参数，然后返回一个Unix风格的路径字符串。参数instace代表一个定义了FileField的模型的实例，说白了就是当前数据记录。filename是原本的文件名。

### 2. ****ImageField****

**class** ImageField(upload\_to**=**None, height\_field**=**None, width\_field**=**None, max\_length**=**100, **\*\***options)[source]

用于保存图像文件的字段。其基本用法和特性与FileField一样，只不过多了两个属性height和width。默认情况下，该字段在HTML中表现为一个ClearableFileInput标签。在数据库内，我们实际保存的是一个字符串类型，默认最大长度100，可以通过max\_length参数自定义。真实的图片是保存在服务器的文件系统内的。

height\_field参数：保存有图片高度信息的模型字段名。 width\_field参数：保存有图片宽度信息的模型字段名。

**使用Django的ImageField需要提前安装pillow模块，pip install pillow即可。**

使用FileField或者ImageField字段的步骤：

1. 在settings文件中，配置MEDIA\_ROOT，作为你上传文件在服务器中的基本路径（为了性能考虑，这些文件不会被储存在数据库中）。再配置个MEDIA\_URL，作为公用URL，指向上传文件的基本路径。请确保Web服务器的用户账号对该目录具有写的权限。
2. 添加FileField或者ImageField字段到你的模型中，定义好upload\_to参数，文件最终会放在MEDIA\_ROOT目录的“upload\_to”子目录中。
3. 所有真正被保存在数据库中的，只是指向你上传文件路径的字符串而已。可以通过url属性，在Django的模板中方便的访问这些文件。例如，假设你有一个ImageField字段，名叫mug\_shot，那么在Django模板的HTML文件中，可以使用{{ object.mug\_shot.url }}来获取该文件。其中的object用你具体的对象名称代替。
4. 可以通过name和size属性，获取文件的名称和大小信息。

### 安全建议

无论你如何保存上传的文件，一定要注意他们的内容和格式，避免安全漏洞！务必对所有的上传文件进行安全检查，确保它们不出问题！如果你不加任何检查就盲目的让任何人上传文件到你的服务器文档根目录内，比如上传了一个CGI或者PHP脚本，很可能就会被访问的用户执行，这具有致命的危害。

### 3. ****FilePathField****

**class** FilePathField(path**=**None, match**=**None, recursive**=**False, max\_length**=**100, **\*\***options)[source]

一种用来保存文件路径信息的字段。在数据表内以字符串的形式存在，默认最大长度100，可以通过max\_length参数设置。

它包含有下面的一些参数：

path：必须指定的参数。表示一个系统绝对路径。

match:可选参数，一个正则表达式，用于过滤文件名。只匹配基本文件名，不匹配路径。例如foo.\*\.txt$，只匹配文件名foo23.txt，不匹配bar.txt与foo23.png。

recursive:可选参数，只能是True或者False。默认为False。决定是否包含子目录，也就是是否递归的意思。

allow\_files:可选参数，只能是True或者False。默认为True。决定是否应该将文件名包括在内。它和allow\_folders其中，必须有一个为True。

allow\_folders： 可选参数，只能是True或者False。默认为False。决定是否应该将目录名包括在内。

比如：

FilePathField(path="/home/images", match="foo.\*", recursive=True)

它只匹配/home/images/foo.png，但不匹配/home/images/foo/bar.png，因为默认情况，只匹配文件名，而不管路径是怎么样的。

### 4. ****UUIDField：****

数据库无法自己生成uuid，因此需要如下使用default参数：

**import** uuid *# Python的内置模块*

**from** django.db **import** models

**class** **MyUUIDModel**(models**.**Model):

id **=** models**.**UUIDField(primary\_key**=**True, default**=**uuid**.**uuid4, editable**=**False)

*# 其它字段*