Django 模板 Django 表单

Django 模型

Django 对各种数据库提供了很好的支持，包括：PostgreSQL、MySQL、SQLite、Oracle。

Django 为这些数据库提供了统一的调用API。 我们可以根据自己业务需求选择不同的数据库。

MySQL 是 Web 应用中最常用的数据库。本章节我们将以 Mysql 作为实例进行介绍。你可以通过本站的 MySQL 教程 了解更多Mysql的基础知识。

如果你没安装 mysql 驱动，可以执行以下命令安装：

```
sudo pip install mysqlclient
```

数据库配置

我们在项目的 settings.py 文件中找到 DATABASES 配置项，将其信息修改为：

HelloWorld/HelloWorld/settings.py: 文件代码：

```
DATABASES = {
'default': {
'ENGINE': 'django.db.backends.mysql', # 或者使用 mysql.connector.django
'NAME': 'test',
'USER': 'test',
'PASSWORD': 'test123',
'HOST':'localhost',
'PORT':'3306',
}
}
```

这里添加了中文注释，所以你需要在 HelloWorld/settings.py 文件头部添加 **# -*- coding: UTF-8 -*-**。

上面包含数据库名称和用户的信息，它们与 MySQL 中对应数据库和用户的设置相同。Django 根据这一设置，与 MySQL 中相应的数据库和用户连接起来。

定义模型

创建 APP

Django规定，如果要使用模型，必须要创建一个app。我们使用以下命令创建一个 TestModel 的 app:

```
django-admin startapp TestModel
```

目录结构如下：

```
HelloWorld
|-- TestModel
```

```
|   |-- __init__.py
|   |-- admin.py
|   |-- models.py
|   |-- tests.py
|   `-- views.py
```

我们修改 TestModel/models.py 文件，代码如下：

HelloWorld/TestModel/models.py: 文件代码：

```
# models.py
from django.db import models
class Test(models.Model):
name = models.CharField(max_length=20)
```

以上的类名代表了数据库表名，且继承了models.Model，类里面的字段代表数据表中的字段(name)，数据类型则由CharField（相当于varchar）、DateField（相当于datetime）， max_length 参数限定长度。

接下来在settings.py中找到INSTALLED_APPS这一项，如下：

```
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'TestModel',               # 添加此项
)
```

在命令行中运行：

```
$ python manage.py migrate   # 创建表结构

$ python manage.py makemigrations TestModel  # 让 Django 知道我们在我们的模型有一些变更
$ python manage.py migrate TestModel   # 创建表结构
```

看到几行 "Creating table…" 的字样，你的数据表就创建好了。

```
Creating tables ...
……
Creating table TestModel_test  #我们自定义的表
……
```

表名组成结构为：应用名_类名（如：TestModel_test）。

注意：尽管我们没有在models给表设置主键，但是Django会自动添加一个id作为主键。

数据库操作

接下来我们在 HelloWorld 目录中添加 testdb.py 文件（下面介绍），并修改 urls.py：

HelloWorld/HelloWorld/urls.py: 文件代码：

```
from django.conf.urls import *
from . import view,testdb
urlpatterns = [
url(r'^hello$', view.hello),
url(r'^testdb$', testdb.testdb),
]
```

添加数据

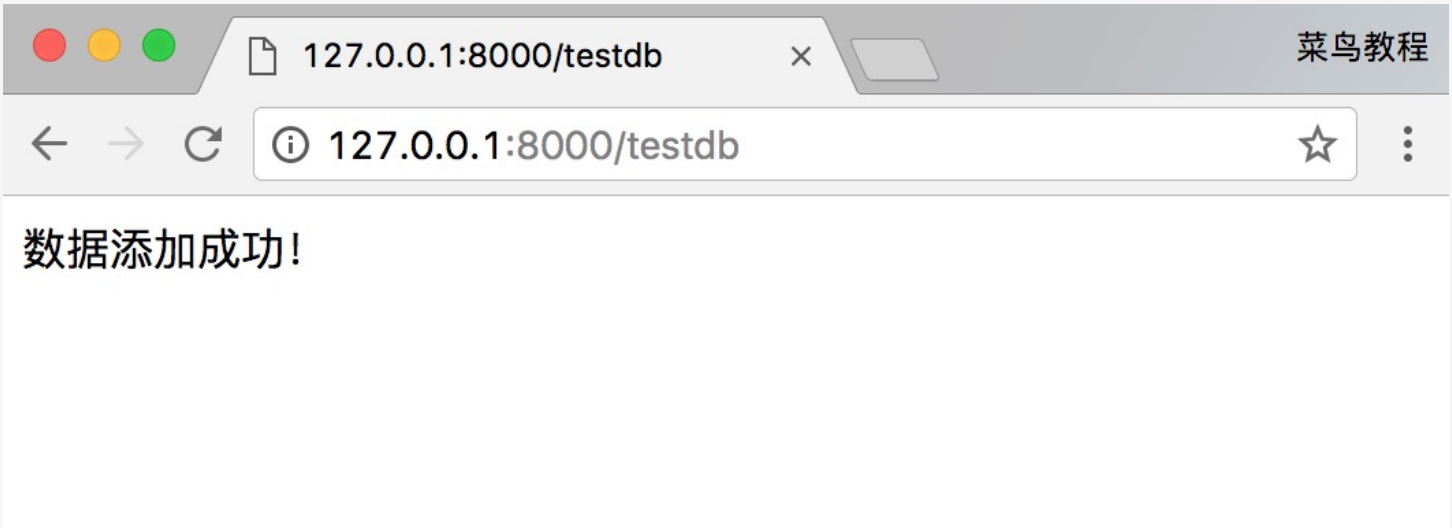
添加数据需要先创建对象，然后再执行 save 函数，相当于SQL中的INSERT：

HelloWorld/HelloWorld/testdb.py: 文件代码：

```
# -*- coding: utf-8 -*-
from django.http import HttpResponse
from TestModel.models import Test
# 数据库操作
def testdb(request):
test1 = Test(name='runoob')
test1.save()
return HttpResponse("<p>数据添加成功! </p>")
```

访问 **http://127.0.0.1:8000/testdb** 就可以看到数据添加成功的提示。

输出结果如下：



获取数据

Django提供了多种方式来获取数据库的内容，如下代码所示：

HelloWorld/HelloWorld/testdb.py: 文件代码：

```
# -*- coding: utf-8 -*-
from django.http import HttpResponse
from TestModel.models import Test
```

```
# 数据库操作
def testdb(request):
# 初始化
response = ""
response1 = ""
# 通过objects这个模型管理器的all()获得所有数据行，相当于SQL中的SELECT * FROM
list = Test.objects.all()
# filter相当于SQL中的WHERE，可设置条件过滤结果
response2 = Test.objects.filter(id=1)
# 获取单个对象
response3 = Test.objects.get(id=1)
# 限制返回的数据 相当于 SQL 中的 OFFSET 0 LIMIT 2;
Test.objects.order_by('name')[0:2]
#数据排序
Test.objects.order_by("id")
# 上面的方法可以连锁使用
Test.objects.filter(name="runoob").order_by("id")
# 输出所有数据
for var in list:
response1 += var.name + " "
response = response1
return HttpResponse("<p>" + response + "</p>")
```

更新数据

修改数据可以使用 save() 或 update():

HelloWorld/HelloWorld/testdb.py: 文件代码：

```
# -*- coding: utf-8 -*-
from django.http import HttpResponse
from TestModel.models import Test
# 数据库操作
def testdb(request):
# 修改其中一个id=1的name字段，再save，相当于SQL中的UPDATE
test1 = Test.objects.get(id=1)
test1.name = 'Google'
test1.save()
# 另外一种方式
#Test.objects.filter(id=1).update(name='Google')
# 修改所有的列
# Test.objects.all().update(name='Google')
return HttpResponse("<p>修改成功</p>")
```

删除数据

删除数据库中的对象只需调用该对象的delete()方法即可：

HelloWorld/HelloWorld/testdb.py: 文件代码：

```
# -*- coding: utf-8 -*-
from django.http import HttpResponse
from TestModel.models import Test
# 数据库操作
def testdb(request):
```

```
# 删除id=1的数据
test1 = Test.objects.get(id=1)
test1.delete()
# 另外一种方式
# Test.objects.filter(id=1).delete()
# 删除所有数据
# Test.objects.all().delete()
return HttpResponse("<p>删除成功</p>")
```

Django 模板 Django 表单