django 操作数据库

Django模型是与数据库相关的，与数据库相关的代码一般写在models.py中，Django 支持sqlite3，MySQL，PostgreSQL等数据库，只需要在settings.py中配置即可，不用更改models.py中的代码，丰富的API极大的方便了使用.在本例中我们就使用默认的sqlite3数据库.

配置django支持mysql数据库
1.安装必备的软件包

```
yum -y install mysql-devel
```

2.修改配置文件

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'mydatabase',  ##数据库需要在mysql中先创建好
        'USER': 'mydatabaseuser',
        'PASSWORD': 'mypassword',
        'HOST': '127.0.0.1',
        'PORT': '3306',
    }
}
```

NAME：指定的数据库名
USER：数据库登录的用户名，mysql一般都是root
PASSWORD：登录数据库的密码，必须是USER用户所对应的密码
HOST：由于一般的数据库都是C/S结构的，所以得指定数据库服务器的位置，我们一般数据库服务器和客户端都是在一台主机上面，所以一般默认都填127.0.0.1.
PORT：数据库服务器端口，mysql默认为3306.

实验准备：

1. 创建django APP 名字为login，并将其注册至配置文件中.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'login',  #将APP注册
]
```

2. 创建Models

```
    from django.db import models
    class Person(models.Model):
        name = models.CharField(max_length=30)
        age = models.IntegerField()

        def __str__(self):
            return self.name
```

2.1 开始创建数据库

```
    (webapps) [root@harbor-a webapps]# python manage.py  makemigrations
    Migrations for 'login':
      login/migrations/0001_initial.py
        - Create model Person

    (webapps) [root@harbor-a webapps]# python manage.py  migrate
    Operations to perform:
      Apply all migrations: admin, auth, contenttypes, login, sessions
    Running migrations:
      Applying login.0001_initial... OK

##注意报错：
    python3.*报“ImportError: No module named ‘MySQLdb'”
    报错：
    [root@bbs s12bbs]# python3 manage.py --help
    Traceback (most recent call last):
      File "/usr/lib/python3.4/site-packages/django/db/backends/mysql/base.py", line 25, in <module>
        import MySQLdb as Database
    ImportError: No module named ‘MySQLdb'

    解决：

    vim /usr/lib/python3.4/site-packages/django/db/backends/mysql/base.py

    导入
    在from django.utils.safestring import SafeBytest,SafeTest下添加下面两条
    import pymysql
    pymysql.install_as_MySQLdb()
```

2.2 查看我们刚刚所创建的数据库及数据表.

```
    (webapps) [root@harbor-a webapps]# sqlite3  db.sqlite3(db.sqlite3是数据库文件)

    SQLite version 3.7.17 2013-05-20 00:56:22
    Enter ".help" for instructions
    Enter SQL statements terminated with a ";"
    sqlite> .table
    auth_group                  django_admin_log
    auth_group_permissions      django_content_type
    auth_permission             django_migrations
    auth_user                   django_session
    auth_user_groups            login_person #刚创建的数据表
    auth_user_user_permissions
    sqlite>
```

3. 开始操作数据库，进入django shell

```
    (webapps) [root@harbor-a webapps]# python manage.py  shell
```

3.1 向表中写入数据

```
    方法1
    In [1]: from login.models import Person
    In [2]: Person.objects.create(name="liushuo", age=20)
    Out[2]: <Person: My name is: liushuo>

    方法2
    In [3]: p = Person(name='zhanglong', age=21)
    In [4]: p.save()

    方法3
    In [5]: p = Person(name='lisi')
    In [6]: p.age=28                                                    In [7]: p.save()
```

3.2 返回一个元组，新创建时返回true，已存在时返回false.

```
    In [8]: Person.objects.get_or_create(name='zhangsan', age=21)
    Out[8]: (<Person: My name is: zhangsan>, True)
```

```
    In [11]: Person.objects.get_or_create(name='liushuo', age=20)
    Out[11]: (<Person: My name is: liushuo>, False)


3.3 获取对象
    In [12]: Person.objects.all()
    Out[12]: <QuerySet [<Person: My name is: liushuo>, <Person: My name is: zhanglong>, <Person: My name is: lisi>, <Person: My name is: zhangsan>, <Person: My name
is: lisi>, <Person: My name is: wangwu>]>

3.4 筛选结果
    In [13]:  Person.objects.filter(name__contains='zhang')
    Out[13]: <QuerySet [<Person: My name is: zhanglong>, <Person: My name is: zhangsan>]
```

```
3.5 删除数据
models.UserInfo.objects.filter(email=input_em).delete()
```

```
3.6 修改数据
models.UserInfo.objects.filter(email=input_em).update(pwd='nihao')
```

In [ ]: