Django--ORM

2020年9月23日 7:40

Django ORM

```
"""
ORM. 对象关系映射
作用:能够让一个不用sql语句的小白也能够通过python 面向对象的代码简单快捷的操作数据库
不足之处:封装程度太高 有时候sql语句的效率偏低 需要你自己写SQL语句

类                          表

对象                        记录

对象属性                    记录某个字段对应的值

应用下面的models.py文件
"""

# 1 先去models.py中书写一个类
  class User(models.Model):
    # id int primary_key auto_increment
    id = models.AutoField(primary_key=True)
    # username varchar(32)
    username = models.CharField(max_length=32)
    # password int
    password = models.IntegerField()
```

```
*************************# 2 数据库迁移命令*************************
python3 manage.py makemigrations 将操作记录记录到小本本上(migrations文件夹)

python3 manage.py migrate  将操作真正的同步到数据库中
# 只要你修改了models.py中跟数据库相关的代码 就必须重新执行上述的两条命令
```

```
class User(models.Model):
    # id int primary_key auto_increment
    id = models.AutoField(primary_key=True,verbose_name='主键')
    # username varchar(32)
    username = models.CharField(max_length=32,verbose_name='用户名')
    """
    CharField必须要指定max_length参数 不指定会直接报错
    verbose_name该参数是所有字段都有的 就是用来对字段的解释
    """
    # password int
    password = models.IntegerField(verbose_name='密码')


class Author(models.Model):
    # 由于一张表中必须要有一个主键字段 并且一般情况下都叫id字段
    # 所以orm当你不定义主键字段的时候 orm会自动帮你创建一个名为id主键字段
    # 也就意味着 后续我们在创建模型表的时候如果主键字段名没有额外的叫法 那么主键字段
```

```
    # 所以orm当你不定义主键字段的时候 orm会自动帮你创建一个名为id主键字段
    # 也就意味着 后续我们在创建模型表的时候如果主键字段名没有额外的叫法 那么主键字段
可以省略不写
    # username varchar(32)
    username = models.CharField(max_length=32)
    # password int
    password = models.IntegerField()
```

字段的增删改查

```
# 字段的增加
  1.可以在终端内直接给出默认值
  2.该字段可以为空
    info = models.CharField(max_length=32,verbose_name='个人简
介',null=True)
  3.直接给字段设置默认值
    hobby = models.CharField(max_length=32,verbose_name='兴趣爱
好',default='study')

# 字段的修改
  直接修改代码然后执行数据库迁移的两条命令即可!

# 字段的删
  直接注释对应的字段然后执行数据库迁移的两条命令即可!
  执行完毕之后字段对应的数据也都没有了
"""
在操作models.py的时候一定要细心
  千万不要注释一些字段
  执行迁移命令之前最好先检查一下自己写的代码
"""
```

数据的增删改查

```
# 今天只会介绍一点点 后面会详细的介绍

# 查
res = models.User.objects.filter(username=username)
"""
返回值你先看成是列表套数据对象的格式
它也支持索引取值 切片操作 但是不支持负数索引
它也不推荐你使用索引的方式取值
user_obj = models.User.objects.filter(username=username).first()
"""
filter括号内可以携带多个参数 参数与参数之间默认是and关系
你可以把filter联想成where记忆
```

```python
# 增
from app01 import models
res = models.User.objects.create(username=username,password=password)
# 返回值就是当前被创建的对象本身

# 第二种增加
user_obj = models.User(username=username,password=password)
user_obj.save()  # 保存数据
```

```python
# 删除功能
  """
  跟编辑功能逻辑类似
  def delete_user(request):
    # 获取用户想要删除的数据id值
    delete_id = request.GET.get('user_id')
    # 直接去数据库中找到对应的数据删除即可
    models.User.objects.filter(id=delete_id).delete()
    """
        批量删除
    """
    # 跳转到展示页面

    return redirect('/userlist/')

  """
# 真正的删除功能应该需要二次确认 我们这里先不做后面会讲
# 删除数据内部其实并不是真正的删除 我们会给数据添加一个标识字段用来表示当前数据是否被删除了，如果数据被删了仅仅只是讲字段修改一个状态
  username password is_delete
  jason    123      0
  egon     123      1
```

```python
# 创建表关系  先将基表创建出来 然后再添加外键字段
class Book(models.Model):
    title = models.CharField(max_length=32)
    price = models.DecimalField(max_digits=8,decimal_places=2)
    # 总共八位 小数点后面占两位
    """
    图书和出版社是一对多 并且书是多的一方 所以外键字段放在书表里面
    """
    publish = models.ForeignKey(to='Publish')  # 默认就是与出版社表的主键字段做外键关联
    """
    如果字段对应的是ForeignKey 那么会orm会自动在字段的后面加_id
    如果你自作聪明的加了_id那么orm还是会在后面继续加_id

    后面在定义ForeignKey的时候就不要自己加_id
    """
```

```
    """
    图书和作者是多对多的关系 外键字段建在任意一方均可 但是推荐你建在查询频率较高的一方
    """
    authors = models.ManyToManyField(to='Author')
    """
    authors是一个虚拟字段 主要是用来告诉orm 书籍表和作者表是多对多关系
    让orm自动帮你创建第三张关系表
    """
class Author(models.Model):
    name = models.CharField(max_length=32)
    age = models.IntegerField()
    """
    作者与作者详情是一对一的关系 外键字段建在任意一方都可以 但是推荐你建在查询频率较高的表中
    """
    author_detail = models.OneToOneField(to='AuthorDetail')
    """
    OneToOneField也会自动给字段加_id后缀
    所以你也不要自作聪明的自己加_id
    """

class AuthorDetail(models.Model):
    phone = models.BigIntegerField()  # 或者直接字符类型
    addr = models.CharField(max_length=32)
```

```
"""
  orm中如何定义三种关系
    publish = models.ForeignKey(to='Publish')  # 默认就是与出版社表的主键字段做外键关联

    authors = models.ManyToManyField(to='Author')

    author_detail = models.OneToOneField(to='AuthorDetail')



    ForeignKey
    OneToOneField
      会自动在字段后面加_id后缀
"""
```

```
# 在django1.X版本中外键默认都是级联更新删除的
# 多对多的表关系可以有好几种创建方式 这里暂且先介绍一种
# 针对外键字段里面的其他参数 暂时不要考虑 如果感兴趣自己可以百度试试看
```