1. 模型管理类

Django给每一个类都生成一个objects对象，是Manger的一个实例对象。自定义管理器之后django不再帮我们生成默认的objects管理器。

使用方法：

1. 自定义一个模型类的管理器类继承models.Manger，然后在模型类的管理器类中重写，原本objects的各种方法或者添加自己的方法。
2. 在具体模型类中创建一个该模型类管理器的实例对象

如：objects=xxxxxManager()

1. 在使用管理器类中重写的方法或者自定义的方法时：类名.objects.方法名(参数)，即可调用。

自定义管理器类的应用场景：

1. 改变查询的结果集

比如调用BookInfo.books.all()返回的是没有删除的图书的数据。

class BookInfoManger(models.Manger):

‘’’图书模型管理器类’’’

#1.改变查询的结果集

def all(self):

#1.调用父类的all方法获取所有数据

books=super.alll() #返回一个查询集，Queryset

#2.对数据进行过滤

books=books.filter(isDelete=False)

return books

重写一个管理器类中的all方法使得不再返回所有数据，而是返回delete=0的数据。

即在类中继承父类的all方法然后重写该方法，返回查询集。

1. 封装函数，添加额外的方法，操作模型类对应的数据表，如增删改查。

管理器类中定义一个方法帮我们操作模型类对应的数据表。

使用self.models()就可以创建一个跟自定义管理器类对应的模型类对象。

例如：在模型类类里面添加方法使得，可以快速往对应的数据表中添加数据

@classmethod #表示是一个类方法，在函数中直接代表类的名称，可以直接使用

def create\_book(cls/self,bititle,bpub\_date):

#1.创建一个图书对象，cls是类名可以创建图书对象

obj=cls()/model\_class = sel.model book = model\_class()

obj.btitle=btitle

obj.bpub\_date=bpub\_date

#2.保存数据库

obj.save()

#3.返回obj

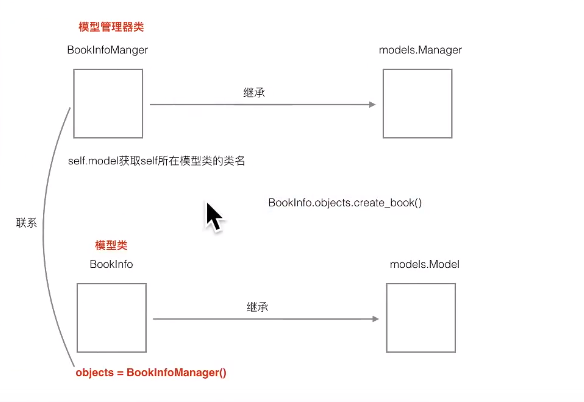
return obj

注意每一个模型管理器类Manger类的对象都有一个属性，【self.models】可以获得对应具体模型类的名称

Manger类中有原本的creat方法，如要使用如下：

Book.objects.create(btitle=’test’,bpub\_date=’1990-10-10’)

总结：



2.元选项

Django默认生成的表名：

应用名\_模型类名小写，如果应用名字改变则对应的表的名称没变的话就会报错，所以要把表的名字和应用的名字分开，也即指定表名。

原选项：

需要在模型类中定义一个元类Meta，在里面定义一个类属性db\_table就可以指定表名

class Meta:

db\_table = ‘bookinfo’指定模型类对应的表名