DRF框架

类名	说明
APIView	1）继承自View，封装了Django 本身的HttpRequest对象为Request对象。2）统一的异常处理。 3）认证&权限&限流。
GenericAPIView	1）继承自APIView，提供了操作序列化器和数据库数据的方法，通常和Mixin扩展类配合使用。2）过滤&分页。

1. Mixin扩展类

DRF提供了5个扩展类，封装了5个通用的操作流程。

类名	说明
ListModelMixin	提供了一个list方法，封装了返回模型数据列表信息的通用流程。
CreateModelMixin	提供了一个create方法，封装了创建一个模型对象数据信息的通用流程。
RetrieveModelMixin	提供了一个retrieve方法，封装了获取一个模型对象数据信息的通用流程。
UpdateModelMixin	提供了一个update方法，封装了更新一个模型对象数据信息的通用流程。
DestroyModelMixin	提供了一个destroy方法，封装了删除一个模型对象数据信息的通用流程。

2. 子类视图

为了方便我们开发RestAPI，DRF框架除了提供APIView和GenericAPIView视图类之外，还提供了一些子类视图类，这些子类视图类同时继承了`GenericAPIView`和对应的Mixin扩展类，并且提供了对应的请求方法。

类名	说明
ListAPIView	1）继承自ListModelMixin和GenericAPIView。2）如果想定义一个视图只提供`列出模型列表`信息的接口，继承此视图类是最快的方式。
CreateAPIView	1）继承自CreateModelMixin和GenericAPIView。2）如果想定义一个视图只提供`创建一个模型信息`的接口，继承此视图类是最快的方式。
RetrieveAPIView	1）继承自RetrieveModelMixin和GenericAPIView。2）如果想定义一个视图只提供`获取一个模型信息`的接口，继承此视图类是最快的方式。
UpdateAPIView	1）继承自UpdateModelMixin和GenericAPIView。2）如果只想定义一个视图只提供`更新一个模型信息`的接口，继承此视图类是最快的方式。
DestroyAPIView	1）继承自DestroyModelMixin和GenericAPIView。2）如果只想定义一个视图只提供`删除一个模型信息`的接口，继承此视图类是最快的方式。
ListCreateAPIView	1）继承自ListModelMixin，CreateModelMixin和GenericAPIView。2）如果只想定义一个视图提供`列出模型列表`和`创建一个模型信息`的接口，继承此视图类是最快的方式。
RetrieveUpdateAPIView	1）继承自RetrieveModelMixin，UpdateModelMixin和GenericAPIView。2）如果只想定义一个视图提供`获取一个模型信息`和`更新一个模型信息`的接口，继承此视图类是最快的方式。
RetrieveDestroyAPIView	1）继承自RetrieveModelMixin，DestroyModelMixin和GenericAPIView。2）如果只想定义一个视图提供`获取一个模型信息`和`删除一个模型信息`的接口，继承此视图类是最快的方式。
RetrieveUpdateDestoryAPIView	1）继承自RetrieveModelMixin，UpdateModelMixin，DestroyModelMixin和GenericAPIView。2）如果只想定义一个视图提供`获取一个模型信息`和`更新一个模型信息`和`删除一个模型信息`的接口，继承此视图类是最快的方式。

示例1：

```
# 需求1：写一个视图，提供一个接口
    1. 获取一组图书数据 GET /books/
    class BookListView(ListAPIView):
        queryset = BookInfo.objects.all()
        serializer_class = BookInfoSerializer
```

```
# 需求2：写一个视图，提供一个接口
    1. 获取指定的图书数据 GET /books/(?P<pk>\d+)/
    class BookDetailView(RetrieveAPIView):
        queryset = BookInfo.objects.all()
        serializer_class = BookInfoSerializer

# 需求3：写一个视图，提供两个接口
    1. 获取指定的图书数据 GET /books/(?P<pk>\d+)/
    2. 更新指定的图书数据 PUT /books/(?P<pk>\d+)/
    class BookDetailView(RetrieveUpdateAPIView):
        queryset = BookInfo.objects.all()
        serializer_class = BookInfoSerializer
```

3. 视图集类

将操作同一资源的处理方法放在同一个类中(视图集)，处理方法不要以请求方式命名，而是以对应的action命名，

list: 提供一组数据

create: 创建一条新数据

retrieve: 获取指定的数据

update: 更新指定的数据

destroy: 删除指定的数据

进行url配置时需要指明请求方法和处理函数之间的对应关系。

类名	说明
ViewSet	1）继承自ViewSetMixin和APIView。2）如果使用视图集时不涉及数据库和序列化器的操作，可以直接继承此类。
GenericViewSet	1）继承自ViewSetMixin和GenericAPIView。2）如果使用视图集涉及数据库和序列化器的操作，可以直接继承此类。
ModelViewSet	1）继承自5个Mixin扩展类和GenericViewSet。2）如果使用视图集想一次提供通用的5种操作，继承这个类是最快的。
ReadOnlyModelViewSet	1）继承自ListModelMixin，RetrieveModelMixin和GenericViewSet。2）如果使用视图集想一次提供list操作和retrieve操作，继承这个类是最快的。

示例1：

```
# 需求1：写一个视图集，提供以下两种操作
    1. 获取一组图书信息(list) GET /books/
    2. 新建一本图书信息(create) POST /books/
```

```
    class BookInfoViewSet(ListModelMixin, CreateModelMixin,
GenericViewSet):
        queryset = BookInfo.objects.all()
        serializer_class = BookInfoSerializer

# 需求2：写一个视图集，提供以下两种操作
    1. 获取一组图书信息(list) GET /books/
    2. 获取指定图书信息(retrieve) GET /books/(?P<pk>\d+)/
    class BookInfoViewSet(ReadOnlyModelViewSet):
        queryset = BookInfo.objects.all()
        serializer_class = BookInfoSerializer

# 需求3：写一个视图集，提供以下三种操作
    1. 获取一组图书信息(list) GET /books/
    2. 获取指定图书信息(retrieve) GET /books/(?P<pk>\d+)/
    3. 更新指定图书信息(update) PUT /books/(?P<pk>\d+)/
    class BookInfoViewSet(UpdateModelMixin, ReadOnlyModelViewSet):
        queryset = BookInfo.objects.all()
        serializer_class = BookInfoSerializer
```

注: 除了常见的5种基本操作之外，如果想给一个视图集中添加其他处理方法，直接在视图集中定义即可。

4. 路由Router

1）路由Router是专门配合视图集来使用的，可以使用Router自动生成视图集中相应处理函数对应的URL配置项。

2）使用Router自动生成视图集中相应处理函数对应的URL配置项时，除了常见的5种基本操作之外，如果视图集中有添加的其他处理方法，则需要给这些方法加上action装饰器之后，才会动态生成其对应的URL配置项。

其他功能

认证&权限