Web 项目1-day04-课堂笔记

用户模块-用户中心功能

1.1 【理解掌握】用户个人信息获取接口实现

API 接口设计：

```
================================================================================
API: GET /user/
================================================================================
请求参数：
    无
================================================================================
响应数据：
    {
      "code": "响应码",
      "message": "提示信息",
      "user": {
        "username":"用户名",
        "mobile": "手机号",
        "email": "邮箱",
        "email_active": "邮箱是否激活"
      }
    }
响应举例：
    {
      "code": 0,
      "message": "OK",
      "user": {
        "username":"smart",
        "mobile": "13155667788",
        "email": "smart@itcast.cn",
```

```
        "email_active": true
      }
    }
==========================================================================
```

email_active 字段添加：

```
class User(AbstractUser):
    """用户模型类"""
    ...

    # 增加 email_active 字段
    # 用于记录邮箱是否激活，默认为 False：未激活
    email_active = models.BooleanField(default=False,
                                       verbose_name='邮箱验证状态')

    class Meta:
        # 表名
        db_table = 'tb_users'
        verbose_name = '用户'
```

用户登录验证：

用户个人信息获取的API，应该只允许已经登录的用户进行访问，我们上面并没有进行判断，那怎么去判断一个用户是否进行了登录呢？？？

这就涉及到了 `request.user` 这个属性：

1）当用户已经登录时，request.user 就是登录的 User 用户对象

2）当用户还未登录时，request.user 就是一个匿名 AnonymousUser 用户对象

对于上面这两种情况，我们可以通过 `request.user.is_authenticated` 进行区分

```
if request.user.is_authenticated:
    # 说明用户已登录
    # request.user：就是登录的 User 用户对象
else:
    # 说明用户未登录
    # request.user：就是一个匿名 AnonymousUser 用户对象
```

登录验证 Mixin 扩展类：

```
from django.http import JsonResponse


def login_required(view_func):
    """登录验证装饰器函数"""
    def wrapper(request, *args, **kwargs):
        if request.user.is_authenticated:
```

```
            # 如果用户已登录，调用对应是 View 视图
            return view_func(request, *args, **kwargs)
        else:
            # 如果用户未登录，直接返回未登录提示
            return JsonResponse({'code': 400,
                                 'message': '用户未登录'})
    return wrapper


class LoginRequiredMixin(object):
    """登录验证 Mixin 扩展类"""
    @classmethod
    def as_view(cls, **init_kwargs):
        view = super().as_view(**init_kwargs)
        # 调用登录验证装饰器函数
        return login_required(view)
```

1.2 【理解掌握】用户个人邮箱设置接口实现

API 接口设计：

```
================================================================================
API: PUT /user/email/
================================================================================
请求参数：
    {
      "email": "用户邮箱"
    }
参数举例：
    {
      "email": "smartli_it@163.com"
    }
================================================================================
响应数据：
    {
      "code": "响应码",
      "message": "提示信息"
    }
响应举例：
    {
      "code": 0,
      "message": "OK"
    }
================================================================================
```

1.3 【理解掌握】Django 框架邮件发送功能实现

SMTP 服务配置：

SMTP（Simple Mail Transfer Protocol）即简单邮件传输协议，客户端发送邮件需要借助于 SMTP 服务。对于 SMTP 服务的来源，有以下两种途径：

1）自己搭建 SMTP 服务：比较麻烦

2）使用免费的 SMTP 服务：如网易163、腾讯qq邮箱等，都提供了免费的 SMTP 服务，设置之后可以直接使用

具体使用参考讲义过程

邮件发送函数：

```
from django.core.mail import send_mail

# 邮件发送函数
send_mail(subject, message, from_email, recipient_list, html_message=None)
# 参数说明：
# subject：邮件标题
# message：普通邮件正文，普通字符串
# from_email：发件人
# recipient_list：收件人列表
# html_message：多媒体邮件正文，可以是html内容
```

Celery 异步发送验证邮件：

发送邮箱验证邮件是耗时的操作，不能阻塞美多商城的响应，此处我们选择使用 `Celery 来实现邮箱验证邮件的异步发送`。

邮箱验证链接的生成：

在给每个登录用户的邮箱发送邮箱验证邮件时，邮件正文中需要包含一个对应用户的邮箱验证链接，当每个用户点击这个邮箱验证链接时，需要将数据库中对应用户的邮箱验证标记设置为已验证，即 email_active 设置为 True。那么如何通过邮箱验证链接区别不同的用户呢？？？

对于上面这个问题，为了区分是哪个用户要进行邮箱验证，需要在邮箱验证链接中包含每个用户的个人信息，比如包含用户的 id 和 email，依此来区分不同的用户。但是考虑到用户信息的安全性，用户的这些信息不要直接放到邮箱验证链接中，而是先将用户的这些信息进行加密，然后再添加到邮箱验证链接中。这里就涉及到信息的加密和解密操作，可以使用之前的 itsdangerous 进行操作。

1.4 【理解掌握】用户个人邮箱验证功能接口实现

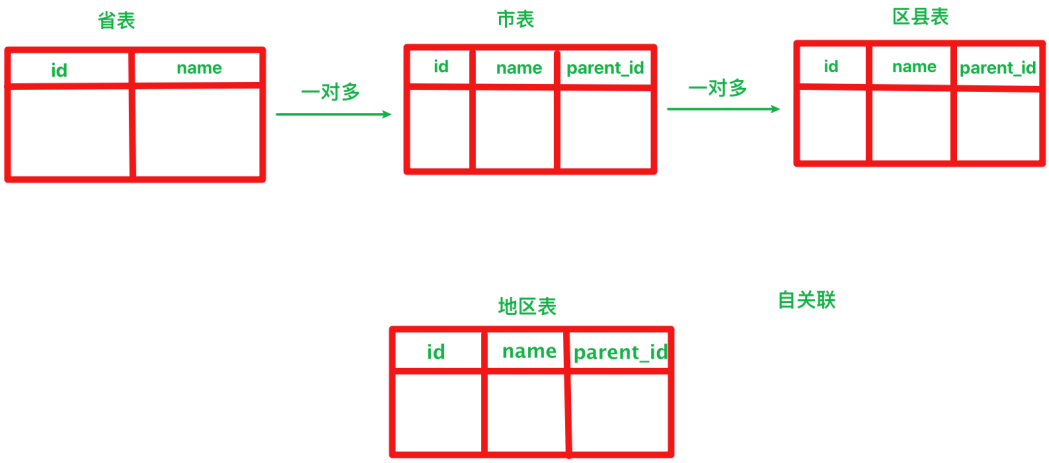
```
================================================================================
API: PUT /emails/verification/?token=<加密的用户个人信息>
================================================================================
请求参数：
    通过查询字符串参数 token 携带加密之后的用户个人信息
参数举例：
    /emails/verification/?
token=eyJhbGciOiJIUzUxMiIsImlhdCI6MTYwMzI3MDQ0NywiZXhwIjoxNjAzMjc3NjQ3fQ.eyJ1c
2VyX2lkIjoxLCJlbWFpbCI6InNtYXJ0bGlfaXRAMTYzLmNvbSJ9.mT3MAAjCK9p-
quNaTZVwABUqdV5SJ0NJZwB83N9VHkz3B45MnUOMFLTiD0YqQzpxWWZiGoD7CcXIRCfRaNLLXA
```

```
==========================================================================
响应数据：
    {
      "code": "响应码",
      "message": "提示信息"
    }
响应举例：
    {
      "code": 0,
      "message": "OK"
    }
==========================================================================
```

用户模块-收货地址管理

2.1 【理解掌握】省市县三级联动功能

自关联表：



对于省市县地区数据的存储，我们需要在数据库中生成一个地区表，然后将省市县的数据导入到表中，具体表结构如下：

id	name	parent_id
20001	江苏省	NULL
20010	南京市	20001
20011	玄武区	20010
30001	上海市	NULL
30010	徐汇区	30001
30011	长宁市	30001

对于上面这张表，其实就是一个自连接的表结构。

地区模型类：

```
from django.db import models


class Area(models.Model):
    """地区模型类"""
    # 创建 name 字段，用户保存名称
    name = models.CharField(max_length=20,
                            verbose_name='地区名称')
    # 自关联字段：parent
    # 第一个参数是 self：指自己和自己关联
    # on_delete=models.SET_NULL：如果父级地区被删除，下级地区的 parent_id 置为NULL
    parent = models.ForeignKey('self',
                               on_delete=models.SET_NULL,
                               related_name='subs',
                               null=True,
                               blank=True,
                               verbose_name='父级地区')

    class Meta:
        db_table = 'tb_areas'
        verbose_name = '地区'
        verbose_name_plural = '地区'

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

```
# 情况一：未指定 related_name 参数
# 根据 id 获取指定的地区
area = Area.objects.get(id='20010')
# 获取该地区的上级地区数据
area.parent
# 获取该地区的下级地区数据
area.area_set.all()


# 情况二：指定了 related_name='subs'
# 根据 id 获取指定的地区
area = Area.objects.get(id='20010')
# 获取该地区的上级地区数据
area.parent
# 获取该地区的下级地区数据
area.subs.all()
```

三级联动API：

1）获取所有省级地区的信息

```
================================================================================
API: GET /areas/
================================================================================
请求参数：
    无
================================================================================
响应数据：
    {
      "code": "响应码",
      "message": "提示信息",
      "provinces": [
          {
              "id": "省id",
              "name": "省名称"
          },
          ...
      ]
    }
响应举例：
    {
      "code": 0,
      "message": "OK",
      "provinces": [
          {
              "id": "110000",
              "name": "北京市"
          },
          {
              "id": "120000",
              "name": "天津市"
          },
          ...
      ]
    }
================================================================================
```

2）选择省或市时，获取其下级地区的信息

```
================================================================================
API: GET /areas/(?P<pk>\d+)/
================================================================================
请求参数：
    通过 URL 地址传递指定地区的id
参数举例：
    GET /areas/130000/
================================================================================
```

响应数据：

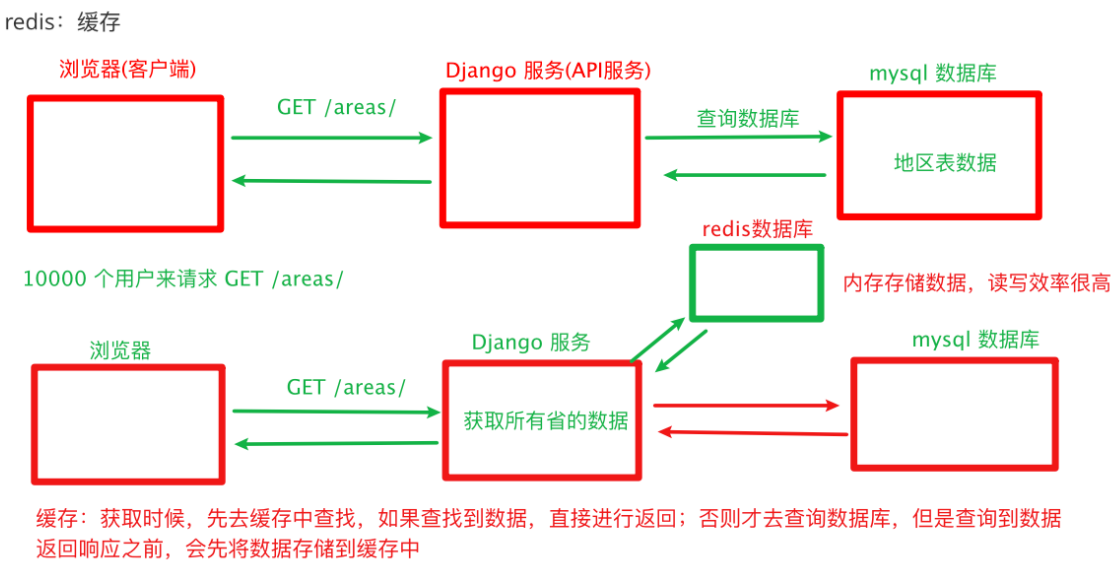
```
{
  "code": "响应码",
  "message": "提示信息",
  "subs": [
      {
          "id": "下级地区id",
          "name": "下级地区名称"
      },
      ...
  ]
}
```

响应举例：

```
{
  "code": 0,
  "message": "OK",
  "subs": [
      {
          "id": "130100",
          "name": "石家庄市"
      },
      {
          "id": "130200",
          "name": "唐山市"
      },
      ...
  ]
}
```

==========

缓存数据的设置：



```python
from django.core.cache import cache

# 存储缓存数据
cache.set('key', '数据内容', '有效期: s')
# 获取缓存数据
cache.get('key')
# 删除缓存数据
cache.delete('key')
```

2.2 【理解掌握】地址数据新增接口实现

```
================================================================================
API: POST /addresses/
================================================================================
请求参数:
    {
      "title": "地址标题",
      "receiver": "收件人",
      "province_id": "省id",
      "city_id": "市id",
      "district_id": "区县id",
      "place": "详细地址",
      "mobile": "手机号",
      "phone": "固定电话",
      "email": "邮箱"
    }
================================================================================
响应数据:
    {
      "code": "响应码",
      "message": "提示信息",
      "address": {
        "id": "地址id",
        "title": "地址标题",
        "receiver": "收件人",
        "province": "省名称",
        "city": "市名称",
        "district": "区县名称",
        "place": "详细地址",
        "mobile": "手机号",
        "phone": "固定电话",
        "email": "邮箱"
      }
    }
================================================================================
```