1 首页模块

1.1 根目录请求

请求URL

http://localhost:8080/community/

请求方式

GET

请求参数

无

返回参数

无

返回说明

此接口会重定向到首页（即/index路径），不返回任何数据。

例子

请求URL：

http://localhost:8080/community/

跳转后的URL：

http://localhost:8080/community/index

返回视图

```
return "forward:/index"
```

1.2 获取首页数据

请求URL

http://localhost:8080/community/index

请求方式

GET

请求参数

参数名称	是否必须	参数类型	示例值	参数说明
orderMode	否	int	0	排序方式，默认为0

返回参数

参数名称	参数类型	参数说明
discussPosts	List<Map<String, Object>>	讨论帖列表
orderMode	int	排序方式

返回说明

discussPosts

每个Map包含以下键值对：

- "post": DiscussPost对象，代表一个讨论帖。DiscussPost对象包含讨论帖的所有信息，如id、标题、内容、创建时间等。
- "user": User对象，代表发帖的用户。User对象包含用户的所有信息，如id、用户名、密码、邮箱等。
- "likeCount": long类型，代表该讨论帖的点赞数量。

orderMode

排序方式，与请求参数中的orderMode相同。

例子

请求URL：

http://localhost:8080/community/index?orderMode=1

返回值：

```
{
    "discussPosts": [
        {
            "post": {
                "id": 1,
                "title": "Hello, world!",
                "content": "Welcome to our community!",
                "createTime": "2021-01-01T00:00:00"
            },
            "user": {
                "id": 1,
                "username": "admin",
                "password": "123456",
                "email": "admin@community.com"
            },
            "likeCount": 100
```

```
        },
        ...
    ],
    "orderMode": 1
}
```

返回视图

```
return "/index"
```

1.3 获取错误页面

请求URL

http://localhost:8080/community/error

请求方式

GET

请求参数

无

返回参数

无

返回说明

此接口会返回一个错误页面，页面地址为 "/error/500"，不返回任何数据。

例子

请求URL：

http://localhost:8080/community/error

返回：

会跳转到 "/error/500" 页面。

返回视图

```
return "/error/500"
```

1.4 获取访问拒绝页面

返回视图

请求URL

http://localhost:8080/community/denied

请求方式

GET

请求参数

无

返回参数

无

返回说明

此接口会返回一个访问被拒绝的页面，页面地址为 "/error/404"，不返回任何数据。

例子

请求URL：

http://localhost:8080/community/denied

返回：

会跳转到 "/error/404" 页面。

返回视图

```
return "/error/404"
```

2 登录模块

2.1 用户注册

请求URL

http://localhost:8080/community/register

请求方式

POST

请求参数

参数名：user

类型：对象

说明：用户对象，包含用户的各种信息，如用户名、密码、邮箱等。

返回参数

1. 当注册成功时：

参数名：msg

类型：字符串

说明：注册成功的提示信息。

参数名：target

类型：字符串

说明：成功后跳转的页面地址。

2. 当注册失败时：

参数名：usernameMsg

类型：字符串

说明：用户名错误的提示信息。

参数名：passwordMsg

类型：字符串

说明：密码错误的提示信息。

参数名：emailMsg

类型：字符串

说明：邮箱错误的提示信息。

返回说明

- 如果注册成功，将返回一个提示信息（"注册成功,我们已经向您的邮箱发送了一封激活邮件,请尽快激活!"），并跳转到首页（"/index"）。
- 如果注册失败，将返回错误信息，并停留在注册页面（"/site/register"）。

例子

请求URL：

http://localhost:8080/community/register

请求参数：

```
{
  "user": {
    "username": "test",
    "password": "123456",
    "email": "test@example.com"
  }
}
```

返回：

1. 成功：

```
{
  "msg": "注册成功,我们已经向您的邮箱发送了一封激活邮件,请尽快激活!",
  "target": "/index"
}
```

2. 失败：

```
{
  "usernameMsg": "用户名已存在",
  "passwordMsg": "密码长度不足",
  "emailMsg": "邮箱格式错误"
}
```

返回视图

```
return "/site/operate-result"
return "/site/register"
```

2.2 用户账号激活

请求URL

http://localhost:8080/community/activation/{userId}/{code}

请求方式

GET

请求参数

路径参数：

参数名：userId

类型：int

说明：用户ID

参数名：code

类型：字符串

说明：激活码

返回参数

参数名：msg

类型：字符串

说明：激活结果的提示信息

参数名：target

类型：字符串

说明：操作完成后跳转的页面地址

返回说明

- 如果激活成功，返回提示信息为"激活成功,您的账号已经可以正常使用了!"，并跳转到登陆页面（"/login"）。
- 如果该账号已经激活过，返回提示信息为"无效操作,该账号已经激活过了!"，并跳转到首页（"/index"）。
- 如果激活失败，返回提示信息为"激活失败,您提供的激活码不正确!"，并跳转到首页（"/index"）。

例子

请求URL：

http://localhost:8080/community/activation/1001/abcd1234

返回：

1. 激活成功：

```
{
  "msg": "激活成功,您的账号已经可以正常使用了!",
  "target": "/login"
}
```

2. 账号已经激活过：

```
{
  "msg": "无效操作,该账号已经激活过了!",
  "target": "/index"
}
```

3. 激活失败：

```
{
  "msg": "激活失败,您提供的激活码不正确!",
  "target": "/index"
}
```

返回视图

```
return "/site/operate-result"
```

2.3 获取验证码

请求URL

http://localhost:8080/community/kaptcha

请求方式

GET

请求参数

无

返回参数

无

返回说明

该接口不返回任何数据，而是生成一张验证码图片，并将图片输出给浏览器。同时，将验证码的文本信息存入Redis，并将归属的验证码ID以Cookie的形式返回给客户端。

特别说明

生成的验证码将以Cookie的形式返回给客户端，Cookie的名称为"kaptchaOwner"，有效期为60秒。同时，验证码的文本信息将存入Redis，键为"kaptchaOwner"，有效期也为60秒。

例子

请求URL：

http://localhost:8080/community/kaptcha

返回：

返回一张验证码图片，并在客户端设置一个名为"kaptchaOwner"的Cookie。同时，服务器会在Redis中存入一个键为"kaptchaOwner"的数据，其值为验证码的文本信息。

2.4 用户登录

请求URL

http://localhost:8080/community/login

请求方式

POST

请求参数

参数名：username

类型：字符串

说明：用户名

参数名：password

类型：字符串

说明：密码

参数名：code

类型：字符串

说明：验证码

参数名：rememberme

类型：布尔值

说明：是否记住我，如果选择记住我，登录状态会持续更长时间

返回参数

1. 当登录成功时：

无返回参数，会在客户端设置一个名为"ticket"的Cookie，用于记录登录状态。

2. 当登录失败时：

参数名：codeMsg

类型：字符串

说明：验证码错误的提示信息

参数名：usernameMsg

类型：字符串

说明：用户名错误的提示信息

参数名：passwordMsg

类型：字符串

说明：密码错误的提示信息

返回说明

- 如果验证码不正确，返回提示信息为"验证码不正确!"，并停留在登录页面（"/site/login"）。
- 如果登录成功，将在客户端设置一个名为"ticket"的Cookie，用于记录登录状态，并跳转到首页（"redirect:/index"）。
- 如果登录失败，将返回错误信息，并停留在登录页面（"/site/login"）。

例子

请求URL：

http://localhost:8080/community/login

请求参数：

```
{
  "username": "test",
  "password": "123456",
  "code": "abcd",
  "rememberme": true
}
```

返回：

1. 登录成功，设置Cookie：

```
{
  "ticket": "abcd1234"
}
```

2. 登录失败：

```
{
  "codeMsg": "验证码不正确!",
  "usernameMsg": "用户名不存在",
  "passwordMsg": "密码错误"
}
```

返回视图

```
return "/site/login"
return "redirect:/index"
return "/site/login"
```

2.5 用户登出

请求URL

http://localhost:8080/community/logout

请求方式

GET

请求参数

参数名：ticket

类型：字符串

说明：用户登录时获取的凭证，存储在Cookie中

返回参数

无

返回说明

用户登出后，将清除用户的登录状态，并跳转到登录页面。

例子

请求URL：

http://localhost:8080/community/logout

返回：

无返回，直接跳转到登录页面。

返回视图

```
return "redirect:/login"
```

3 帖子模块

3.1 添加帖子

@ResponseBody

请求URL

http://localhost:8080/community/discuss/add

请求方式

POST

请求参数

参数名：title

类型：字符串

说明：帖子的标题

参数名：content

类型：字符串

说明：帖子的内容

返回参数

参数名：code

类型：整数

说明：状态码，0表示发布成功，403表示未登录

参数名：msg

类型：字符串

说明：提示信息

返回说明

如果用户未登录，返回状态码403和提示信息"你还没有登录哦!"。

如果发布成功，返回状态码0和提示信息"发布成功!"。

发布帖子后，会触发发帖事件，并在Redis中存储帖子的分数信息。

例子

请求URL：

http://localhost:8080/community/discuss/add

请求参数：

```
{
  "title": "测试标题",
  "content": "测试内容"
}
```

返回：

1. 发布成功：

```
{
  "code": 0,
  "msg": "发布成功!"
}
```

2. 未登录：

```
{
  "code": 403,
  "msg": "你还没有登录哦!"
}
```

3.2 获取论帖子详情

请求URL

http://localhost:8080/community/discuss/detail/{discussPostId}

请求方式

GET

请求参数

参数名：discussPostId

类型：整数

说明：讨论帖子的ID

返回参数

参数名：post

类型：对象

说明：帖子的详细信息

参数名：user

类型：对象

说明：帖子作者的用户信息

参数名：likeCount

类型：整数

说明：帖子的点赞数量

参数名：likeStatus

类型：整数

说明：当前用户对帖子的点赞状态，0表示未点赞，1表示已点赞

参数名：comments

类型：列表

说明：帖子的评论列表，每个评论包括评论信息、评论作者信息、评论的点赞数量、评论的点赞状态、评论的回复列表（每个回复包括回复信息、回复作者信息、回复目标用户信息、回复的点赞数量、回复的点赞状态）和回复数量

返回说明

返回讨论帖子的详细信息，包括帖子信息、作者信息、点赞数量、点赞状态以及评论列表。

每个评论包括评论信息、评论作者信息、评论的点赞数量、评论的点赞状态、评论的回复列表和回复数量。

每个回复包括回复信息、回复作者信息、回复目标用户信息、回复的点赞数量和回复的点赞状态。

例子

请求URL：

http://localhost:8080/community/discuss/detail/123

帖子信息

- post
  - id: 123
  - title: "这是一个测试帖子"
  - content: "这是一个测试帖子的内容"
  - createTime: "2022-01-01 00:00:00"

作者信息

- user
  - id: 1
  - username: "测试用户"
  - avatar: "用户头像的URL"

点赞信息

- likeCount: 100
- likeStatus: 1

评论列表

- comments
  - 第一个评论
    - comment: {id: 1, content: "这是第一个评论", createTime: "2022-01-01 00:00:01"}
    - user: {id: 2, username: "评论用户1", avatar: "用户头像的URL"}
    - likeCount: 10
    - likeStatus: 0
    - replys
      - 第一个回复: {reply: {id: 1, content: "这是第一个回复", createTime: "2022-01-01 00:00:02"}, user: {id: 3, username: "回复用户1", avatar: "用户头像的URL"}, target: null, likeCount: 5, likeStatus: 1}
    - replyCount: 1
  - 第二个评论
    - comment: {id: 2, content: "这是第二个评论", createTime: "2022-01-01 00:00:03"}
    - user: {id: 4, username: "评论用户2", avatar: "用户头像的URL"}
    - likeCount: 20
    - likeStatus: 1
    - replys

        - 第一个回复: {reply: {id: 2, content: "这是第二个回复", createTime: "2022-01-01 00:00:04"}, user: {id: 5, username: "回复用户2", avatar: "用户头像的URL"}, target: {id: 3, username: "回复用户1", avatar: "用户头像的URL"}, likeCount: 10, likeStatus: 0}
        - 第二个回复: {reply: {id: 3, content: "这是第三个回复", createTime: "2022-01-01 00:00:05"}, user: {id: 6, username: "回复用户3", avatar: "用户头像的URL"}, target: {id: 2, username: "评论用户1", avatar: "用户头像的URL"}, likeCount: 15, likeStatus: 1}
    - replyCount: 2

这些信息将被渲染到HTML页面中，由于这是一个后端渲染的页面，因此在接口调用后，实际返回的是一个已经渲染好的HTML页面，而不是JSON格式的数据。

返回视图

```
return "/site/discuss-detail"
```

3.3 置顶帖子

@ResponseBody

请求URL

http://localhost:8080/community/discuss/top

请求方式

POST

请求参数

参数名：id

类型：整数

说明：帖子的ID

返回参数

参数名：code

类型：整数

说明：状态码，0表示操作成功

返回说明

如果操作成功，返回状态码0。

置顶帖子后，会触发发帖事件。

例子

请求URL：

http://localhost:8080/community/discuss/top

请求参数：

```
{
  "id": 123
}
```

返回：

```
{
  "code": 0
}
```

3.4 设置帖子为精华帖

@ResponseBody

请求URL

http://localhost:8080/community/discuss/wonderful

请求方式

POST

请求参数

参数名：id

类型：整数

说明：帖子的ID

返回参数

参数名：code

类型：整数

说明：状态码，0表示操作成功

返回说明

如果操作成功，返回状态码0。

设置为精华帖后，会触发发帖事件，并计算帖子分数。

例子

请求URL：

http://localhost:8080/community/discuss/wonderful

请求参数：

```
{
  "id": 123
}
```

返回：

```
{
  "code": 0
}
```

3.5 删除帖子

@ResponseBody

请求URL

http://localhost:8080/community/discuss/delete

请求方式

POST

请求参数

参数名：id

类型：整数

说明：帖子的ID

返回参数

参数名：code

类型：整数

说明：状态码，0表示操作成功

返回说明

如果操作成功，返回状态码0。

删除帖子后，会触发删帖事件。

例子

请求URL：

http://localhost:8080/community/discuss/delete

请求参数：

```
{
  "id": 123
}
```

返回：

```
{
  "code": 0
}
```

4 评论模块

添加评论

请求URL

http://localhost:8080/community/comment/add/{discussPostId}

请求方式

POST

请求参数

路径参数：

参数名：discussPostId

类型：整数

说明：讨论帖的ID

请求体参数：

参数名：userId

类型：整数

说明：评论者的用户ID

参数名：status

类型：整数

说明：评论的状态，0表示正常

参数名：createTime

类型：日期

说明：评论创建的时间

参数名：content

类型：字符串

说明：评论的内容

参数名：entityType

类型：整数

说明：实体类型，可以是帖子或者是评论

参数名：entityId

类型：整数

说明：实体ID，可以是帖子的ID或者是评论的ID

返回说明

添加评论后，会触发评论事件。如果评论的实体是帖子，还会触发发帖事件，同时计算帖子分数。

返回一个重定向的URL，跳转到帖子详情页面。

例子

请求URL：

http://localhost:8080/community/comment/add/123

请求参数：

```
{
  "userId": 1,
  "content": "这是一条评论",
  "entityType": 1,
  "entityId": 123
}
```

返回视图

```
return "redirect:/discuss/detail/" + discussPostId
```

5 消息模块

5.1 获取私信列表

请求URL

http://localhost:8080/community/letter/list

请求方式

GET

返回参数

Model中渲染的数据：

参数名：conversations

类型：List<Map<String, Object>>

说明：私信会话列表，每个Map包含会话信息、私信数量、未读私信数量和对话目标用户信息。

参数名：letterUnreadCount

类型：整数

说明：用户的未读私信数量。

参数名：noticeUnreadCount

类型：整数

说明：用户的未读通知数量。

返回说明

返回私信列表页面。

例子

请求URL：

http://localhost:8080/community/letter/list

返回：

私信列表页面，Model中包含的数据如下：


```
{
  "conversations": [
    {
      "conversation": {
        "id": 1,
        "fromId": 2,
        "toId": 3,
        "content": "你好",
        "createTime": "2022-01-01T00:00:00"
      },
      "letterCount": 1,
      "unreadCount": 0,
      "target": {
        "id": 3,
        "username": "user3",
        "password": "123456",
        "salt": "abc",
        "email": "user3@example.com",
        "headerUrl": "http://localhost:8080/community/user/header/3",
```

```
        "createTime": "2022-01-01T00:00:00",
        "status": 0,
        "activationCode": "123456",
        "type": 0
      }
    },
    // 其他会话...
  ],
  "letterUnreadCount": 0,
  "noticeUnreadCount": 0
}
```

返回视图

```
return "/site/letter"
```

5.2 获取私信详情

请求URL

http://localhost:8080/community/letter/detail/{conversationId}

请求方式

GET

请求参数

路径参数:

参数名：conversationId

类型：字符串

说明：会话的ID

返回参数

Model中渲染的数据:

参数名：letters

类型：List<Map<String, Object>>

说明：私信列表，每个Map包含私信信息和发送者用户信息。

参数名：target

类型：User

说明：私信的目标用户。

返回说明

返回私信详情页面。在请求私信详情的过程中，会将这些私信设置为已读状态。

例子

请求URL：

http://localhost:8080/community/letter/detail/123

返回：

私信详情页面，Model中包含的数据如下：


```
{
  "letters": [
    {
      "letter": {
        "id": 1,
        "fromId": 2,
        "toId": 3,
        "content": "你好",
        "createTime": "2022-01-01T00:00:00"
      },
      "fromUser": {
        "id": 2,
        "username": "user2",
        "password": "123456",
        "salt": "abc",
        "email": "user2@example.com",
        "headerUrl": "http://localhost:8080/community/user/header/2",
        "createTime": "2022-01-01T00:00:00",
        "status": 0,
        "activationCode": "123456",
        "type": 0
      }
    },
    // 其他私信...
  ],
  "target": {
    "id": 3,
    "username": "user3",
    "password": "123456",
    "salt": "abc",
    "email": "user3@example.com",
    "headerUrl": "http://localhost:8080/community/user/header/3",
    "createTime": "2022-01-01T00:00:00",
    "status": 0,
    "activationCode": "123456",
    "type": 0
  }
}
```

返回视图

```
return "/site/letter-detail"
```

5.3 发送私信

@ResponseBody

请求URL

http://localhost:8080/community/letter/send

请求方式

POST

请求参数

请求体参数：

参数名：toName

类型：字符串

说明：私信的目标用户的用户名

参数名：content

类型：字符串

说明：私信的内容

返回参数

返回一个JSON字符串，包含状态码和消息。状态码为0表示发送成功，为1表示发送失败。

返回说明

发送私信。如果目标用户不存在，返回状态码为1和错误消息。

例子

请求URL：

http://localhost:8080/community/letter/send

请求参数：

```
{
  "toName": "user3",
  "content": "你好"
}
```

返回：

```
{
  "code": 0,
  "msg": "发送成功"
}
```

5.4 获取通知列表

请求URL

http://localhost:8080/community/notice/list

请求方式

GET

返回参数

Model中渲染的数据：

参数名：commentNotice

类型：Map<String, Object>

说明：评论类通知的详细信息，包括通知本身、触发通知的用户、被评论的实体类型、被评论的实体ID、评论的帖子ID、评论通知数量和未读评论通知数量。

参数名：likeNotice

类型：Map<String, Object>

说明：点赞类通知的详细信息，包括通知本身、触发通知的用户、被点赞的实体类型、被点赞的实体ID、点赞的帖子ID、点赞通知数量和未读点赞通知数量。

参数名：followNotice

类型：Map<String, Object>

说明：关注类通知的详细信息，包括通知本身、触发通知的用户、被关注的实体类型、被关注的实体ID、关注通知数量和未读关注通知数量。

参数名：letterUnreadCount

类型：整数

说明：用户的未读私信数量。

参数名：noticeUnreadCount

类型：整数

说明：用户的未读通知数量。

返回说明

返回通知列表页面。

例子

请求URL：

http://localhost:8080/community/notice/list

返回：

通知列表页面，Model中包含的数据如下：

```
{
  "commentNotice": {
    "message": {
      "id": 1,
      "fromId": 2,
      "toId": 3,
      "content": "你好",
      "createTime": "2022-01-01T00:00:00"
    },
    "user": {
      "id": 2,
      "username": "user2",
      "password": "123456",
      "salt": "abc",
      "email": "user2@example.com",
      "headerUrl": "http://localhost:8080/community/user/header/2",
      "createTime": "2022-01-01T00:00:00",
      "status": 0,
      "activationCode": "123456",
      "type": 0
    },
    "entityType": "post",
    "entityId": 1,
    "postId": 1,
    "count": 1,
    "unread": 0
  },
  "likeNotice": {
    // 类似于commentNotice...
  },
  "followNotice": {
    // 类似于commentNotice，但没有postId...
  },
  "letterUnreadCount": 0,
  "noticeUnreadCount": 0
}
```

返回视图

```
return "/site/notice"
```

5.5 获取通知详情

请求URL

http://localhost:8080/community/notice/detail/{topic}

请求方式

GET

请求参数

路径参数：

参数名：topic

类型：字符串

说明：通知的主题，可以是"comment"、"like"或"follow"。

返回参数

Model中渲染的数据：

参数名：notices

类型：List<Map<String, Object>>

说明：通知列表，每个Map包含通知本身、通知内容、触发通知的用户、被通知的实体类型、被通知的实体ID、通知的帖子ID（如果有）和通知的作者。

返回说明

返回通知详情页面。在请求通知详情的过程中，会将这些通知设置为已读状态。

例子

请求URL：

http://localhost:8080/community/notice/detail/comment

返回：

通知详情页面，Model中包含的数据如下：


```
{
  "notices": [
    {
      "notice": {
        "id": 1,
        "fromId": 2,
        "toId": 3,
        "content": "你好",
        "createTime": "2022-01-01T00:00:00"
      },
      "user": {
        "id": 2,
        "username": "user2",
```

```
        "password": "123456",
        "salt": "abc",
        "email": "user2@example.com",
        "headerUrl": "http://localhost:8080/community/user/header/2",
        "createTime": "2022-01-01T00:00:00",
        "status": 0,
        "activationCode": "123456",
        "type": 0
      },
      "entityType": "post",
      "entityId": 1,
      "postId": 1,
      "fromUser": {
        "id": 2,
        "username": "user2",
        "password": "123456",
        "salt": "abc",
        "email": "user2@example.com",
        "headerUrl": "http://localhost:8080/community/user/header/2",
        "createTime": "2022-01-01T00:00:00",
        "status": 0,
        "activationCode": "123456",
        "type": 0
      }
    },
    // 其他通知...
  ]
}
```


返回视图

```
return "/site/notice-detail"
```

6 点赞模块

点赞

@ResponseBody

请求URL

http://localhost:8080/community/like

请求方式

POST

请求参数

请求体参数:

参数名：entityType

类型：整数

说明：被点赞的实体的类型。

参数名：entityId

类型：整数

说明：被点赞的实体的ID。

参数名：entityUserId

类型：整数

说明：被点赞的实体所属的用户的ID。

参数名：postId

类型：整数

说明：点赞的帖子的ID。

返回参数

返回一个JSON字符串，包含点赞的数量和点赞的状态。状态为1表示已点赞，为0表示未点赞。

返回说明

点赞。如果点赞成功，会触发一个点赞事件。如果被点赞的是帖子，还会更新帖子的分数。

例子

请求URL：

http://localhost:8080/community/like

请求参数：

```
{
  "entityType": 2,
  "entityId": 1,
  "entityUserId": 3,
  "postId": 1
}
```

返回：

```
{
  "code": 0,
  "msg": "点赞成功",
  "data": {
    "likeCount": 1,
    "likeStatus": 1
  }
}
```

7 关注模块

7.1 关注

@ResponseBody

请求URL

http://localhost:8080/community/follow

请求方式

POST

请求参数

请求体参数：

参数名：entityType

类型：整数

说明：被关注的实体的类型。

参数名：entityId

类型：整数

说明：被关注的实体的ID。

返回参数

返回一个JSON字符串，表示关注操作是否成功。

返回说明

关注。如果关注成功，会触发一个关注事件。

例子

请求URL：

http://localhost:8080/community/follow

请求参数：

```
{
  "entityType": 2,
  "entityId": 1
}
```

返回：

```
{
  "code": 0,
  "msg": "已关注!"
}
```

7.2 取消关注

@ResponseBody

请求URL

http://localhost:8080/community/unfollow

请求方式

POST

请求参数

请求体参数:

参数名：entityType

类型：整数

说明：被取消关注的实体的类型。

参数名：entityId

类型：整数

说明：被取消关注的实体的ID。

返回参数

返回一个JSON字符串，表示取消关注操作是否成功。

返回说明

取消关注。

例子

请求URL:

http://localhost:8080/community/unfollow

请求参数:

```
{
  "entityType": 2,
  "entityId": 1
}
```

返回:

```
{
  "code": 0,
  "msg": "已取消关注!"
}
```

7.3 获取关注列表

请求URL

http://localhost:8080/community/followees/{userId}

请求方式

GET

请求参数

路径参数:

参数名：userId

类型：整数

说明：用户ID。

返回参数

返回一个Model，包含当前用户信息和该用户关注的用户列表。

返回说明

获取关注列表。返回的Model中包含以下键值:

- "user"：当前用户的信息。
- "users"：该用户关注的用户列表，每个元素是一个Map，包含被关注用户的信息和当前用户是否关注了该用户。

例子

请求URL:

http://localhost:8080/community/followees/1

返回:

一个Model，包含以下键值:

- "user"：当前用户的信息。
- "users"：该用户关注的用户列表，每个元素是一个Map，包含被关注用户的信息和当前用户是否关注了该用户。

例如:

```
Model model;
model.addAttribute("user", new User(1, "张三", "123456"));
model.addAttribute("users", Arrays.asList(
    Map.of(
        "user", new User(2, "李四", "123456"),
        "hasFollowed", true
    ),
    Map.of(
        "user", new User(3, "王五", "123456"),
        "hasFollowed", false
    )
));
```

返回视图

```
return "/site/followee"
```

7.4 获取粉丝列表

请求URL

http://localhost:8080/community/followers/{userId}

请求方式

GET

请求参数

路径参数：

参数名：userId

类型：整数

说明：用户ID。

返回参数

返回一个Model，包含当前用户信息和该用户的粉丝列表。

返回说明

获取粉丝列表。返回的Model中包含以下键值：

- "user"：当前用户的信息。
- "users"：该用户的粉丝列表，每个元素是一个Map，包含粉丝的信息和当前用户是否关注了该粉丝。

例子

请求URL：

http://localhost:8080/community/followers/1

返回：

一个Model，包含以下键值：

- "user"：当前用户的信息。
- "users"：该用户的粉丝列表，每个元素是一个Map，包含粉丝的信息和当前用户是否关注了该粉丝。

例如：

```
Model model;
model.addAttribute("user", new User(1, "张三", "123456"));
model.addAttribute("users", Arrays.asList(
    Map.of(
        "user", new User(2, "李四", "123456"),
        "hasFollowed", true
    ),
    Map.of(
        "user", new User(3, "王五", "123456"),
        "hasFollowed", false
    )
));
```

返回视图

```
return "/site/follower"
```

8 用户模块

8.1 获取用户设置页面

请求URL

http://localhost:8080/community/user/setting

请求方式

GET

请求参数

无

返回参数

返回一个Model，包含上传文件的凭证和文件名。

返回说明

获取用户设置页面。返回的Model中包含以下键值：

- "uploadToken"：上传文件的凭证。
- "fileName"：上传文件的名称。

注意

此接口需要用户登录才能访问。

例子

请求URL：

http://localhost:8080/community/user/setting

返回：

一个Model，包含以下键值：

- "uploadToken"：例如，"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9"。
- "fileName"：例如，"abc123"。

返回的页面路径为"/site/setting"。

返回视图

```
return "/site/setting"
```

8.2 更新用户头像

@ResponseBody

请求URL

http://localhost:8080/community/user/header/url

请求方式

POST

请求参数

请求体参数：

参数名：fileName

类型：字符串

说明：上传的文件名称。

返回参数

返回一个JSON字符串，包含操作的状态码和消息。

返回说明

更新用户头像路径。返回的JSON字符串中包含以下键值：

- 状态码：0表示操作成功，1表示操作失败。
- 消息：如果操作成功，消息为空；如果操作失败，消息为"文件名不能为空!"。

例子

请求URL：

http://localhost:8080/community/user/header/url

请求体：

```
{
    "fileName": "abc123"
}
```

返回：

如果操作成功，返回：

```
{
    "code": 0,
    "msg": ""
}
```

如果操作失败，返回：

```
{
    "code": 1,
    "msg": "文件名不能为空!"
}
```

8.3 上传用户头像（废弃）

请求URL

http://localhost:8080/community/user/upload

请求方式

POST

请求参数

请求体参数：

参数名：headerImage

类型：文件

说明：用户上传的头像图片。

返回参数

返回一个Model，包含操作的错误信息（如果有）。

返回说明

上传用户头像。如果上传成功，会更新当前用户的头像路径，并重定向到主页。如果上传失败，返回的Model中会包含错误信息。

- "error"：错误信息。如果上传的文件为空，错误信息为"您还没有选择图片!"；如果文件的格式不正确，错误信息为"文件的格式不正确!"。

注意

此接口需要用户登录才能访问。

例子

请求URL：

http://localhost:8080/community/user/upload

请求体：

文件：用户的头像图片。

返回：

如果上传成功，重定向到"http://localhost:8080/community/index"。

如果上传失败，返回一个Model，包含错误信息，例如"您还没有选择图片!"，并返回到"/site/setting"页面。

返回视图

```
return "redirect:/index"
```

8.4 获取用户头像（废弃）

请求URL

http://localhost:8080/community/user/header/{fileName}

请求方式

GET

请求参数

路径参数：

参数名：fileName

类型：字符串

说明：用户头像的文件名称。

返回参数

返回用户的头像图片。

返回说明

获取用户的头像图片。如果读取头像失败，会记录错误日志。

例子

请求URL：

http://localhost:8080/community/user/header/abc123.png

返回：

用户的头像图片。

8.5 获取用户个人主页

请求URL

http://localhost:8080/community/user/profile/{userId}

请求方式

GET

请求参数

路径参数：

参数名：userId

类型：整型

说明：用户的ID。

返回参数

返回一个Model，包含用户的个人信息。

返回说明

获取用户的个人主页，包括用户的基本信息，点赞数量，关注数量，粉丝数量，以及当前登录用户是否已关注该用户。如果该用户不存在，会抛出一个异常。

返回的Model中包含以下键值：

- "user"：用户对象，包含用户的基本信息。
- "likeCount"：用户获得的点赞数量。
- "followeeCount"：用户关注的人数。
- "followerCount"：用户的粉丝数量。
- "hasFollowed"：当前登录用户是否已关注该用户。如果当前没有登录用户，值为false。

例子

请求URL：

http://localhost:8080/community/user/profile/123

返回：

如果用户存在，返回一个Model，包含用户的个人信息，例如：


```
{
    "user": {
        "id": 123,
        "username": "abc",
        "password": "def",
        "salt": "ghi",
        "email": "abc@def.com",
        "headerUrl": "http://localhost:8080/community/user/header/abc123.png",
        "createTime": "2020-01-01 00:00:00"
    },
    "likeCount": 100,
    "followeeCount": 50,
    "followerCount": 200,
    "hasFollowed": true
}
```


如果用户不存在，抛出一个异常，例如：


```
{
    "message": "该用户不存在!"
}
```


返回视图

```
return "/site/profile"
```

9 搜索模块

搜索帖子

请求URL

http://localhost:8080/community/search?keyword=xxx

请求方式

GET

请求参数

查询参数：

参数名：keyword

类型：字符串

说明：搜索关键词。

返回参数

返回一个Model，包含搜索结果的帖子列表和搜索关键词。

返回说明

搜索帖子。返回的Model中包含以下键值：

- "discussPosts"：搜索结果的帖子列表，每个元素是一个Map，包含帖子的信息、作者的信息和帖子的点赞数量。
- "keyword"：搜索关键词。

例子

请求URL：

http://localhost:8080/community/search?keyword=Java

返回：

一个Model，包含以下键值：

- "discussPosts"：搜索结果的帖子列表，例如：

```
Arrays.asList(
    Map.of(
        "post", new DiscussPost(1, "Java基础", "Java是一种面向对象的编程语言", 1),
        "user", new User(1, "张三", "123456"),
        "likeCount", 10
    ),
    Map.of(
        "post", new DiscussPost(2, "Java进阶", "Java的高级特性", 2),
        "user", new User(2, "李四", "123456"),
        "likeCount", 20
    )
)
```

- "keyword"：搜索关键词，例如："Java"。

返回视图

```
return "/site/search"
```

10 网站数据统计模块

10.1 获取网站数据统计页面

请求URL

http://localhost:8080/community/data

请求方式

GET 或 POST

请求参数

无

返回参数

返回一个页面，路径为"/site/admin/data"。

返回说明

获取网站数据统计页面。返回的是一个页面，路径为"/site/admin/data"。

例子

请求URL：

http://localhost:8080/community/data

返回：

一个页面，路径为"/site/admin/data"。

返回视图

```
return "/site/admin/data"
```

10.2 统计独立访客

请求URL

http://localhost:8080/community/data/uv

请求方式

POST

请求参数

请求体参数：

参数名：start

类型：日期

格式：yyyy-MM-dd

说明：统计开始日期。

参数名：end

类型：日期

格式：yyyy-MM-dd

说明：统计结束日期。

返回参数

返回一个Model，包含统计的UV、统计开始日期和统计结束日期。

返回说明

统计网站UV。返回的Model中包含以下键值：

- "uvResult"：统计的UV。
- "uvStartDate"：统计开始日期。
- "uvEndDate"：统计结束日期。

例子

请求URL：

http://localhost:8080/community/data/uv

请求体：

```
{
    "start": "2021-01-01",
    "end": "2021-12-31"
}
```

返回：

一个Model，包含以下键值：

- "uvResult"：例如，10000。
- "uvStartDate"：例如，"2021-01-01"。
- "uvEndDate"：例如，"2021-12-31"。

返回视图

```
return "forward:/data"
```

10.3 统计活跃用户

接口名称

请求URL

http://localhost:8080/community/data/dau

请求方式

POST

请求参数

请求体参数：

参数名：start

类型：日期

格式：yyyy-MM-dd

说明：统计开始日期。

参数名：end

类型：日期

格式：yyyy-MM-dd

说明：统计结束日期。

返回参数

返回一个Model，包含统计的活跃用户数量、统计开始日期和统计结束日期。

返回说明

统计活跃用户。返回的Model中包含以下键值：

- "dauResult"：统计的活跃用户数量。
- "dauStartDate"：统计开始日期。
- "dauEndDate"：统计结束日期。

例子

请求URL：

http://localhost:8080/community/data/dau

请求体：

```
{
    "start": "2021-01-01",
    "end": "2021-12-31"
}
```

返回：

一个Model，包含以下键值：

- "dauResult"：例如，5000。
- "dauStartDate"：例如，"2021-01-01"。
- "dauEndDate"：例如，"2021-12-31"。

返回视图

```
return "forward:/data"
```

11 分享模块

分享网页为长图

@ResponseBody

请求URL

http://localhost:8080/community/share

请求方式

GET

请求参数

请求参数：

参数名：htmlUrl

类型：字符串

说明：需要分享的网页的URL。

返回参数

返回一个JSON字符串，包含生成长图的访问路径。

返回说明

分享网页为长图。生成的长图的文件名是随机生成的。异步生成长图，并返回长图的访问路径。

返回的JSON字符串中包含以下键值：

- "code"：操作的状态码。0表示成功。
- "msg"：操作的消息。如果成功，值为null。
- "data"：操作的数据。是一个Map，包含以下键值：
    - "shareUrl"：生成长图的访问路径。

例子

请求URL：

http://localhost:8080/community/share?htmlUrl=http://localhost:8080/community/user/profile/123

返回：

如果操作成功，返回一个JSON字符串，例如：

```
{
    "code": 0,
    "msg": null,
    "data": {
        "shareUrl": "http://localhost:8080/community/share/image/abc123.png"
    }
}
```