

demo后端默认启动在8000端口,所以tomcat默认的8080需要修改

用户

登录

GET /users/login

需要参数:

- username
 - 含义:用户名
 - 类型:string
- password
 - 含义:密码
 - 类型:string

效果以及返回值:

Demo / Login

GET http://localhost:8000/users/login?username=lehr&password=123

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION
username	lehr	
password	123	
Key	Value	Description

Body Cookies Headers (3) Test Results Status: 200 OK Time: 10 ms Size: 182 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "Code": 200,
3   "Msg": "success",
4   "Success": true,
5   "Data": "成功"
6 }
```

当用户名为空或者密码为空的时候则需要判定为不合法输入,返回效果应该是下面这样(以密码不正确为例):

```
1 {
2   "Code": 400,
3   "Msg": "密码不正确",
4   "Success": false,
5   "Data": null
6 }
```

注册

GET /users/register

需要参数:

- username
 - 含义:用户名
 - 类型:string
- password
 - 含义:密码
 - 类型:string

效果以及返回值:

The screenshot shows a REST client interface with a demo titled "Register". The request is a GET to `http://localhost:8000/users/register?username=var&password=123`. The "Query Params" section shows two parameters: `username` with value `var` and `password` with value `123`. The "Body" tab is selected, showing a JSON response in "Pretty" format:

```
1 {
2   "Code": 200,
3   "Msg": "success",
4   "Success": true,
5   "Data": "注册成功"
6 }
```

The status bar at the bottom indicates a successful response: Status: 200 OK, Time: 5 ms, Size: 188 B.

当然,遇到用户名被占用或者各种不良情况的时候也需要做对应的回复,例子如下,可以继续细化msg的内容

```
1  {
2      "Code": 400,
3      "Msg": "输入不合法",
4      "Success": false,
5      "Data": null
6  }
```

todolist

什么是todoList

- 可以理解为一个记事本系统里你要做的事情,很简单的一条提示和一个打钩完成的条目
- 每个人只能看到自己的todo list
- 完成了可以调用接口,标记完成任务

获取某用户的所有todolist

GET /todolist

需要参数:

- username
 - 含义:获取谁的todolist
 - 类型:string

效果以及返回值:

GET http://localhost:8000/todolist?username=lehr

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
<input checked="" type="checkbox"/> username	lehr
Key	Value

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "Code": 200,
3   "Msg": "success",
4   "Success": true,
5   "Data": [
6     {
7       "TodoId": "0",
8       "Title": "微光小练习",
9       "StartTime": "2021-04-11",
10      "Done": false,
11      "Username": "lehr"
12    }
13  ]
14 }
```

todolist是每个人的小任务列表,每个人看到的应该是自己的内容,所以你需要以username进行请求,从而让对应用户拿到自己的todolist内容,后期这里可以考虑用session或者jwt部分优化这个 用户会话追踪 的过程

todolist返回的是个数组,但是上图的示范例子中,只不过是一个只有一个元素的数组而已

另外,因为这个后端是go开发的,所以各个参数名字都是大写开头,java按理说都应该是小写开头,不过为了让前端能适配,所以java选手开发这里的时候可能需要稍稍违反一下命名规则,返回值的字段名称采用大写开头

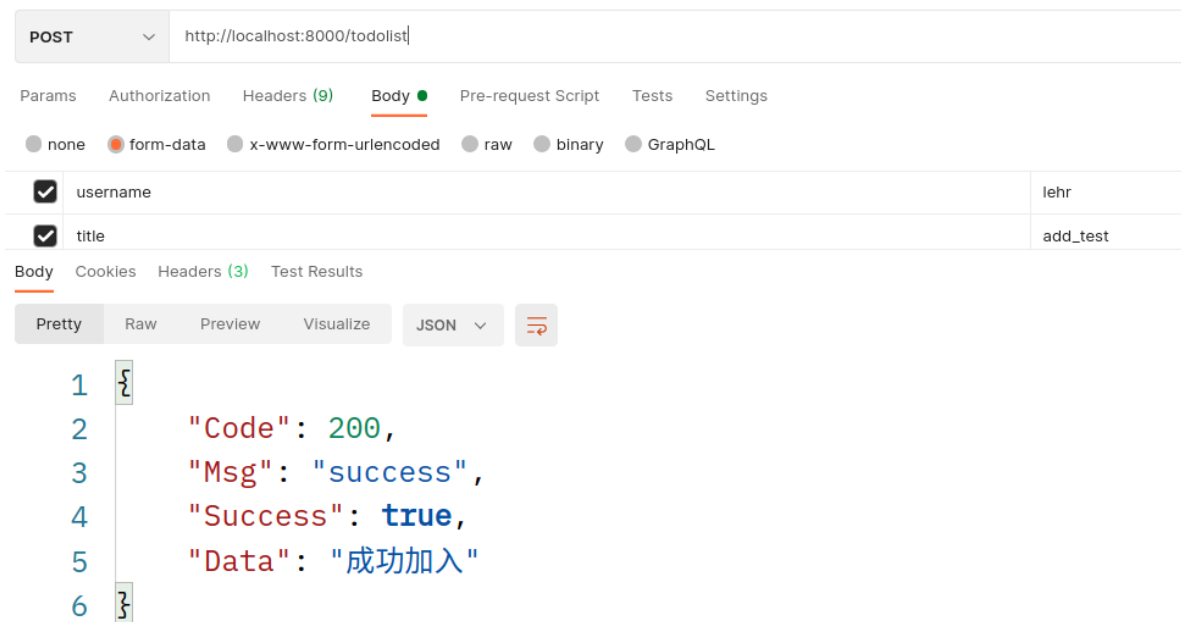
添加一个todolist

POST /todolist

需要参数(PostForm传参!如果不懂看我下图postman传参写法):

- username
 - 含义:谁添加的
 - 类型:string
- title
 - 含义:todolist的内容是什么
 - 类型:string

效果以及返回值:



添加成功之后使用GET方法去获取就能看到效果

当然,对于添加,以及各种其他接口,前端后端都是需要检查参数的合法性的

给一个todolist打上完成标记

PUT /todolist/:todold

需要参数:

- 无,注意其中todold是通过url的方式完成的传参

效果以及返回值:

PUT

http://localhost:8000/todolist/0

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Query Params

	KEY
	Key

Body

Cookies

Headers (3)

Test Results

Pretty

Raw

Preview

Visualize

JSON

```
1  {}
2
3      "Code": 200,
4      "Msg": "success",
5      "Success": true,
6      "Data": "任务完成"
```

情况演示:

对0号todo打上完成标记

操作之前:

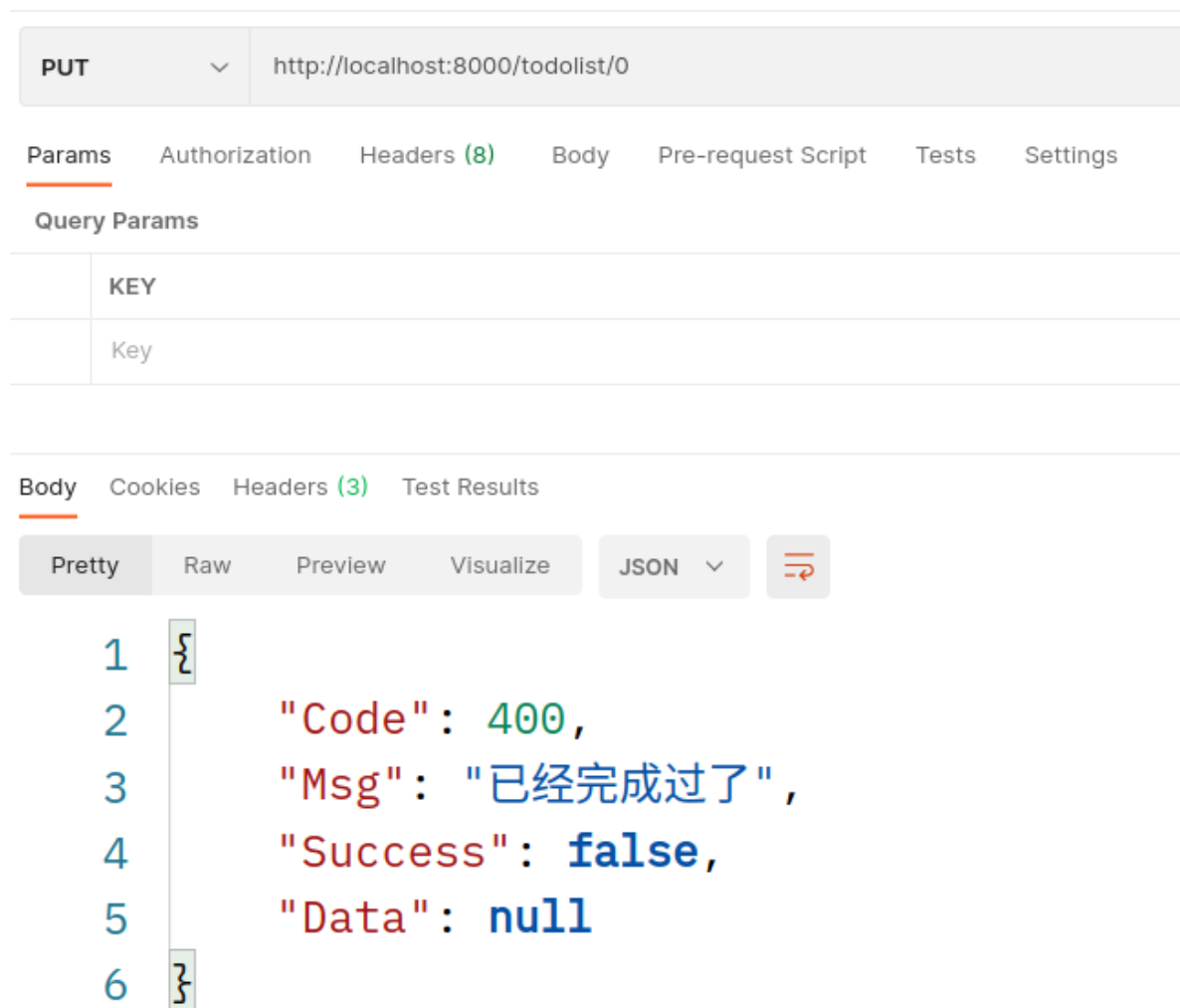
```
"Data": [  
  {  
    "TodoId": "0",  
    "Title": "微光小练习",  
    "StartTime": "2021-04-11",  
    "Done": false,  
    "Username": "lehr"  
  }  
]
```

操作之后:

```
"Data": [  
  {  
    "TodoId": "0",  
    "Title": "微光小练习",  
    "StartTime": "2021-04-11",  
    "Done": true,  
    "Username": "lehr"  
  }  
]
```

Done变为true

当然,如果你对已完成的todo重复操作,需要报错:



PUT http://localhost:8000/todolist/0

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

KEY
Key

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize JSON

```
1 {  
2   "Code": 400,  
3   "Msg": "已经完成过了",  
4   "Success": false,  
5   "Data": null  
6 }
```

删除一个todolist

DELETE /todolist/:todold

需要参数:

- 无,注意其中todold是通过url的方式完成的传参

效果以及返回值:

DELETE

▼

http://localhost:8000/todolist/0

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

Query Params

	KEY
	Key

Body

Cookies

Headers (3)

Test Results

Pretty

Raw

Preview

Visualize

JSON ▼

↻

```
1  {
2      "Code": 200,
3      "Msg": "success",
4      "Success": true,
5      "Data": "删除成功"
6  }
```

删除之后,再用Get查询,目标就没有了

GET

▼

http://localhost:8000/todolist?username=lehr|

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

Query Params

	KEY
<input checked="" type="checkbox"/>	username
	Key

Body

Cookies

Headers (3)

Test Results

Pretty

Raw

Preview

Visualize

JSON ▼

≡

```
1 {
2   "Code": 200,
3   "Msg": "success",
4   "Success": true,
5   "Data": []
6 }
```

Info

什么是Info?

- 可以理解为系统里的公告
- 每个人都可以看见所有的公告
- 大家也都能进行修改
- 用户可以把某个公告的信息加入到自己的todoList里去

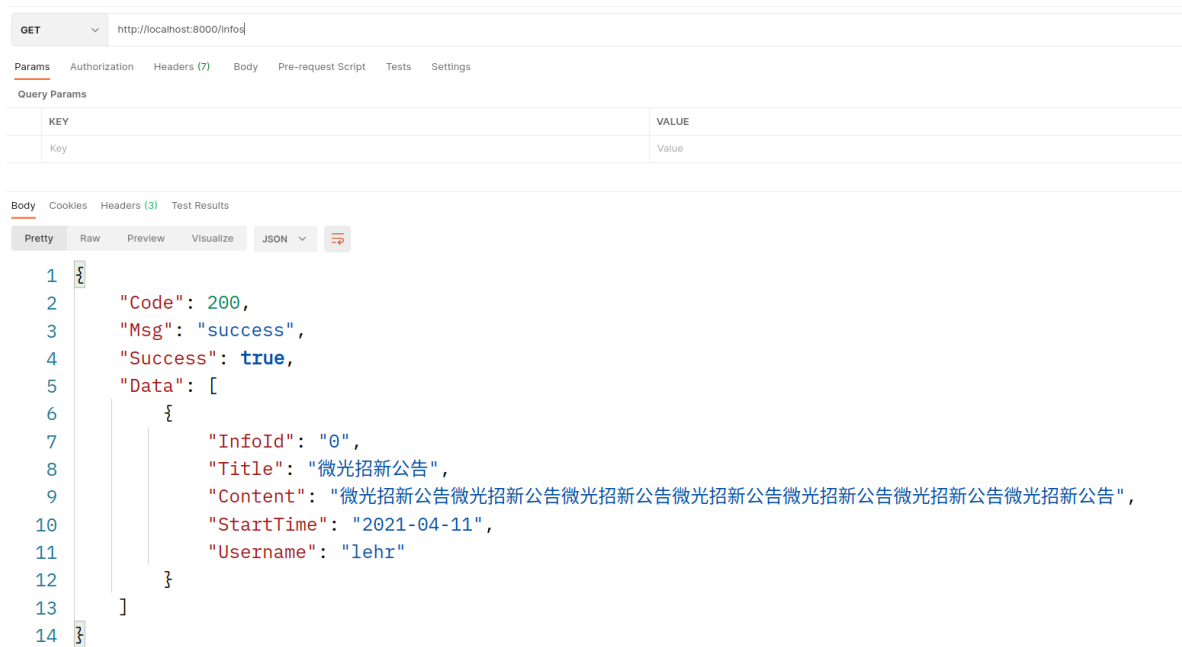
获取所有info

GET /infos

需要参数:

- 无

效果以及返回值:



也是一个数组返回值,但是只不过上图是只有一个元素的数组而已

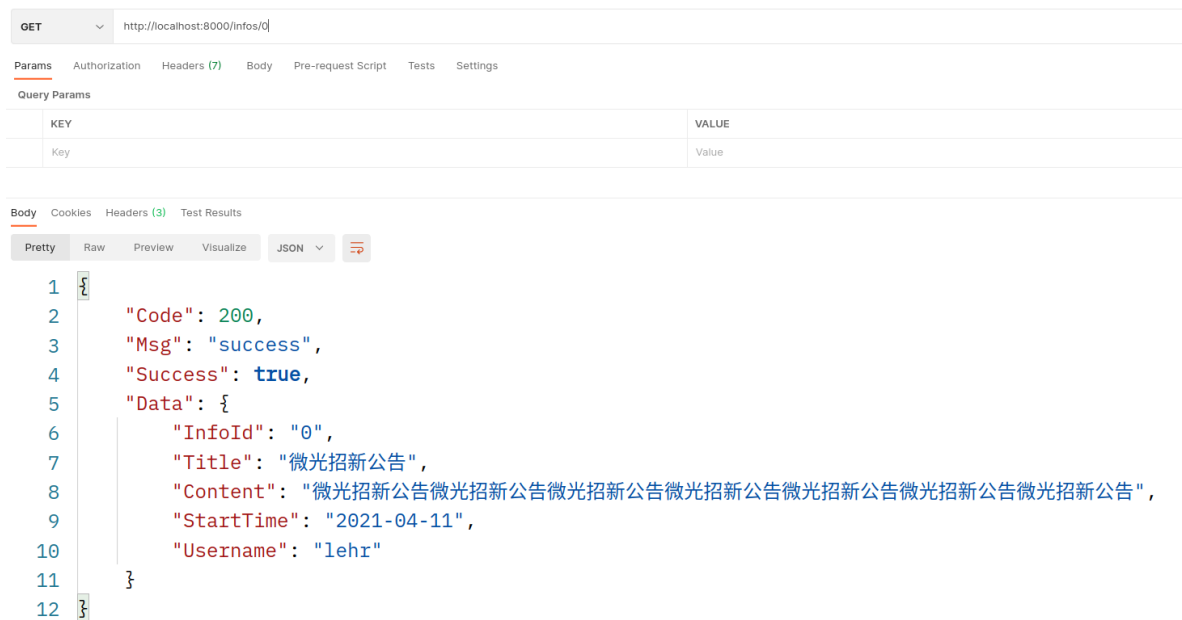
获取某条info

GET /infos/:infoId

需要参数:

- 无,注意其中todoId是通过url的方式完成的传参

效果以及返回值:



返回的是单个对象

添加一条info

POST /infos

需要参数(PostForm传参!如果不懂看我下图postman传参写法):

- title
 - 含义:info的标题
 - 类型:string
- content
 - 含义:info的正文内容
 - 类型:string
- username
 - 含义:用户名,是谁添加的?
 - 类型:string

效果以及返回值:

POST http://localhost:8000/infos?title=addTest&content=im-adding-a-lot-of-things&username=lehr

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

<input checked="" type="checkbox"/> title	addTest
<input checked="" type="checkbox"/> content	im-adding-a-lot-of-things
<input checked="" type="checkbox"/> username	lehr

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize JSON

```

1 {
2   "Code": 200,
3   "Msg": "success",
4   "Success": true,
5   "Data": "添加成功"
6 }
```

用Get查看结果:

```

{
  "Code": 200,
  "Msg": "success",
  "Success": true,
  "Data": [
    {
      "InfoId": "0",
      "Title": "微光招新公告",
      "Content": "微光招新公告微光招新公告微光招新公告微光招新公告微光招新公告微光招新公告微光招新公告",
      "StartTime": "2021-04-11",
      "Username": "lehr"
    },
    {
      "InfoId": "2",
      "Title": "addTest",
      "Content": "im-adding-a-lot-of-things",
      "StartTime": "2021-04-13",
      "Username": "lehr"
    }
  ]
}
```

修改一条info

PUT /infos/:infoId

需要参数:

- 注意其中infoId是通过url的方式完成的传参
- title
 - 含义:info的标题
 - 类型:string
- content
 - 含义:info的正文内容
 - 类型:string
- username
 - 含义:用户名,是谁修改的?
 - 类型:string

效果以及返回值:

The screenshot shows a REST client interface with a PUT request to `http://localhost:8000/infos/2?username=lehr&title=addTest&content=changed!`. The request parameters are listed in a table:

Key	Value
username	lehr
title	addTest
content	changed!

The response body is shown in JSON format:

```

1 {
2   "Code": 200,
3   "Msg": "success",
4   "Success": true,
5   "Data": "修改成功"
6 }
```

可以看到,之前我们在添加接口示范时的那个info被修改了内容:

```

"Data": [
  {
    "InfoId": "0",
    "Title": "微光招新公告",
    "Content": "微光招新公告微光招新公告微光招新公告微光招新公告微光招新公告微光招新公告微光招新公告",
    "StartTime": "2021-04-11",
    "Username": "lehr"
  },
  {
    "InfoId": "2",
    "Title": "addTest",
    "Content": "changed!",
    "StartTime": "2021-04-13",
    "Username": "lehr"
  }
]
```

删除一条info

DELETE /infos/:infoId

需要参数:

- 无,注意其中infoId是通过url的方式完成的传参

效果以及返回值:

The screenshot shows a REST client interface. At the top, a dropdown menu is set to 'DELETE' and the URL is 'http://localhost:8000/infos/1'. Below this, there are tabs for 'Params', 'Authorization', 'Headers (9)', 'Body', 'Pre-request Script', and 'Test'. The 'Params' tab is selected, showing a table with two columns: 'KEY' and 'Value'. The 'Body' tab is also visible, showing a JSON response in 'Pretty' format. The response is a JSON object with the following fields: 'Code' (200), 'Msg' ('success'), 'Success' (true), and 'Data' ('删除成功').

KEY	Value
Key	

```
{
  "Code": 200,
  "Msg": "success",
  "Success": true,
  "Data": "删除成功"
}
```

把info添加到todolist

POST /infos/:infoId/move

需要参数,PostForm传参!:

- 注意其中infoId是通过url的方式完成的传参
- username
 - 含义:目标用户的username
 - 类型:string

效果以及返回值:

POST http://localhost:8000/infos/2/move

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

Key	Value
username	lehr

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize JSON ↻

```
1 {
2   "Code": 200,
3   "Msg": "success",
4   "Success": true,
5   "Data": "成功加入"
6 }
```

然后去对应的user todoist里就可以看见效果了

会多出一个todo,其中title为info的title,日期为当前的日期