RESTful风格API开发

使用SpringBoot框架 + Postman，完成图书管理API开发，API开发完成后，使用React完成前端功能。

统一响应格式:

`{"code": "SUCCESS", "message": null, "data": null}`

`code`为`SUCCESS`代表请求成功，`data`中存放具体的业务数据。

`{"code": "ERROR", "message": "失败原因", "data": null}`

`code`不为`SUCCESS`代表请求失败，`message`中是失败原因。

接收参数

- `Query Params`
- `application/x-www-form-urlencoded`
- `application/json`
- `URI`中的参数

新增图书信息

API	**/books**
请求方法	`POST`
Content-Type	`application/x-www-form-urlencoded`
请求参数(Body):	
name	图书名
describe	图书描述
请求结果:	
失败	`{"code": "ERROR", "message": "失败原因", "data": null}`

`成功`返回的数据如下:

```
{
  "code": "SUCCESS",
  "message": null,
  "data": {
    "id": 1,
    "name": "书名1",
    "describe": "图书描述",
    "createdAt": "2020-02-17 14:39:07"
  }
}
```

三、图书列表

API地址	**/books**
请求方法	`GET`
请求参数(Query Params):	例如 `/books?name=java&page=2&limit=5`
name	搜索指定的图书名(模糊匹配)[可选参数]
page	当前页码 [可选参数 如果不传默认值为1]
limit	每页多少条 [可选参数 如果不传默认值为20]
请求结果:	
失败	`{"code": "ERROR", "message": "失败原因", "data": null}`

`成功` 返回的数据如下

```
{
  "code": "SUCCESS",
  "message": null,
  "data": {
    "pagination": {         //页码信息
      "total": 2,           //共有多少条数据
      "page": 1,            //当前是第几页
      "limit": 5,           //每页多少条数据
    },
    "books": [              //图书数据
      {
        "id": 1,
        "name": "书名1",
        "createdAt": "2020-02-17 14:39:07"
      },
```

```
      {
        "id": 2,
        "name": "书名2",
        "createdAt": "2020-02-17 14:39:07"
      }
    ]
  }
}
```

四、图书详情

API	/books/{bookId}
请求方式	`GET`
请求参数:	`注意不是QueryString，参数是URI的一部份`
bookId	图书ID 例如 `/books/1` 表示查询图书ID为1的数据
请求结果:	
失败	`{"code": "ERROR", "message": "失败原因", "data": null}`

`成功` 返回的数据如下:

```
{
  "code": "SUCCESS",
  "message": null,
  "data": {
    "id": 1,
    "name": "书名1",
    "describe": "图书描述",
    "createdAt": "2020-02-17 14:39:07"
  }
}
```

四、修改图书信息

API	/books/{bookId}
请求方式	`POST`
Content-Type	`application/json`
请求参数(URI)：	
bookId	图书ID 例如`/books/1` 表示更新图书ID为1的数据
请求正文参数(Body)：	
name	书名
describe	图书描述
请求结果：	
成功	`{"code": "SUCCESS", "message": "更新成功", "data": null}`
失败	`{"code": "ERROR", "message": "失败原因", "data": null}`

五、删除图书

API	/books/{bookId}
请求方式	`DELETE`
请求参数：	
bookId	图书ID 例如`/books/1` 表示删除图书ID为1的数据
请求结果：	
成功	`{"code": "SUCCESS", "message": "删除成功", "data": null}`
失败	`{"code": "ERROR", "message": "失败原因", "data": null}`