

Python考核 7

学习任务

时间:2022.3.15-2022.4.1

基本任务：RealWorld程序后端接口编写

考核方式：提交学习笔记、提交作业的源代码。文件发送至training@sizhu.tech, 邮件主题 python方向 -

姓名-第xx次学习任务

推荐资料：

阅读《FLASK WEB开发：基于PYTHON的WEB应用开发》（“狗书”）第一部分的第1、2、5、7章

注：“狗书”中包含的模板的部分，在学习中可以跳过

项目github地址：<https://github.com/gothinkster/realworld>

注：应为github在外网，所以如果有同学打不开的，可以查看我们导出的html版本，或者和我们联系解决详情说明：

1. 阅读导出的github中的RealWorld项目的接口文档（重点查看接口的请求方式、响应信息、以及接口规范（REST风格））

2. 在本地创建flask项目，并将本地创建的项目与远程的gitee或github仓库连接。

3. 连接本地的数据库，对相应的接口所需要的数据进行归类、划分，然后建表。并于项目连接

4. 编写满足接口文档规范的接口函数，并使用postman或apiPost等接口调试工具调试自己所写的接口

（注意：在这个过程中重点查看自己所写的接口是否满足接口文档的规范，请求后获得的信息格式是否与文档相同）

5. 将代码使用git命令上传到远程仓库中。并将自己的仓库地址填写在提交的学习笔记中。

6. 上次任务没有写“前后端分离项目”和“Rest风格”的理解的同学请在这次的任务中将该内容写在学习笔记中。

详情请前往 [Introduction | RealWorld \(realworld-docs.netlify.app\)](https://realworld-docs.netlify.app)查看

Specs	▼
Backend Specs	▼
Introduction	
Endpoints	请求体相关说明
API Response format	响应相关说明
Error Handling	
CORS	
Postman	

这个博客系统相对来说较为简单，主要模型用户模型、文章模型、评论模型、标签模型

RealWorld example appsDocumentationGitHub

IntroductionImplementation CreationIntroductionExpectationsFeaturesSpecsBackend SpecsIntroductionEndpointsAPI Response formatError HandlingCORSPostmanTestsFrontend SpecsMobile SpecsCommunity

Endpoints

Authentication Header:

You can read the authentication header from the headers of the request

Authorization: Token jwt.token.here

Authentication:

POST /api/users/login

Example request body:

```
{  "user": {    "email": "jake@jake.jake",    "password": "jakejake"  }}
```

No authentication required, returns a User

Required fields: email, password

Authentication Header:

Authentication:

Registration:

Get Current UserUpdate UserGet ProfileFollow userUnfollow user

List ArticlesFeed ArticlesGet ArticleCreate ArticleUpdate ArticleDelete Article

Add Comments to an ArticleGet Comments from an ArticleDelete CommentFavorite ArticleUnfavorite Article

Get Tags

用户

文章

评论

标签

对于api响应

将响应统一封装成

例如,


```
{  
  
  "code": 10000,  
  
  "data": data,  
  
  "message": "错误信息放在此处"  
  
}
```

code是状态码，可以定义自己的状态码，并配置详细的说明文档，让前端了解即可

比如：你可以定义 code:10000为正常响应，10001为异常响应

message一般用来存放错误消息

例如：

 RealWorld example apps [Documentation](#)

Introduction

Implementation Creation

Introduction

Expectations

Features

Specs

Backend Specs

Introduction

Endpoints

API Response format

Error Handling

CORS

Postman

Tests

API Response format

JSON Objects returned by API:

Make sure the right content type like `Content-Type: application/json; charset=utf-8` is correctly returned.

Users (for authentication)

```
{  
  "user": {  
    "email": "jake@jake.jake",  
    "token": "jwt.token.here",  
    "username": "jake",  
    "bio": "I work at statefarm",  
    "image": null  
  }  
}
```

Copy

将接口的响应改成：

```
{  
  "code": 10000,  
  "data": {  
    "user": {  
      "email": "jake@jake.jake",  
      "token": "jwt.token.here",  
      "username": "jake",  
      "bio": "I work at statefarm",  
      "image": null  
    }  
  },  
  "message": null  
}
```

后面的接口同理。