

实验 06：API 服务测试

一、实验目的

- 1、了解 RESTful API 的基本概念和设计原则。
- 2、理解 API 测试中状态码、请求头、响应体的含义和作用。
- 3、理解 HTTP 协议中各种请求方法的区别和使用场景。
- 4、掌握 Postman 和 Swagger 的安装、配置和基本操作方法。

二、实验学时

2 学时

三、实验类型

综合性

四、实验需求

1、硬件

每人配备计算机 1 台，建议优先使用个人计算机开展实验。

实验基于信息技术学院教学容器化云计算平台开展。

2、软件

IntelliJ IDEA Community。

3、网络

本地主机能够访问互联网和实验中心网络。

4、工具

Postman v10

Navicat Premium 12

五、实验任务

- 1、完成编写测试计划(参考附件 06-1[测试计划])。
- 2、完成编写测试用例(参考附件 06-2[测试用例])。
- 3、完成 Postman 安装，能够掌握基本功能使用。
- 4、完成 Swagger 部署并实现应用。
- 5、使用 postman 进行接口测试。
- 6、使用 Swagger 进行接口测试。
- 7、使用 Postman 实现接口自动化测试。
- 8、编写测试报告(参考附件 06-3[测试报告])。

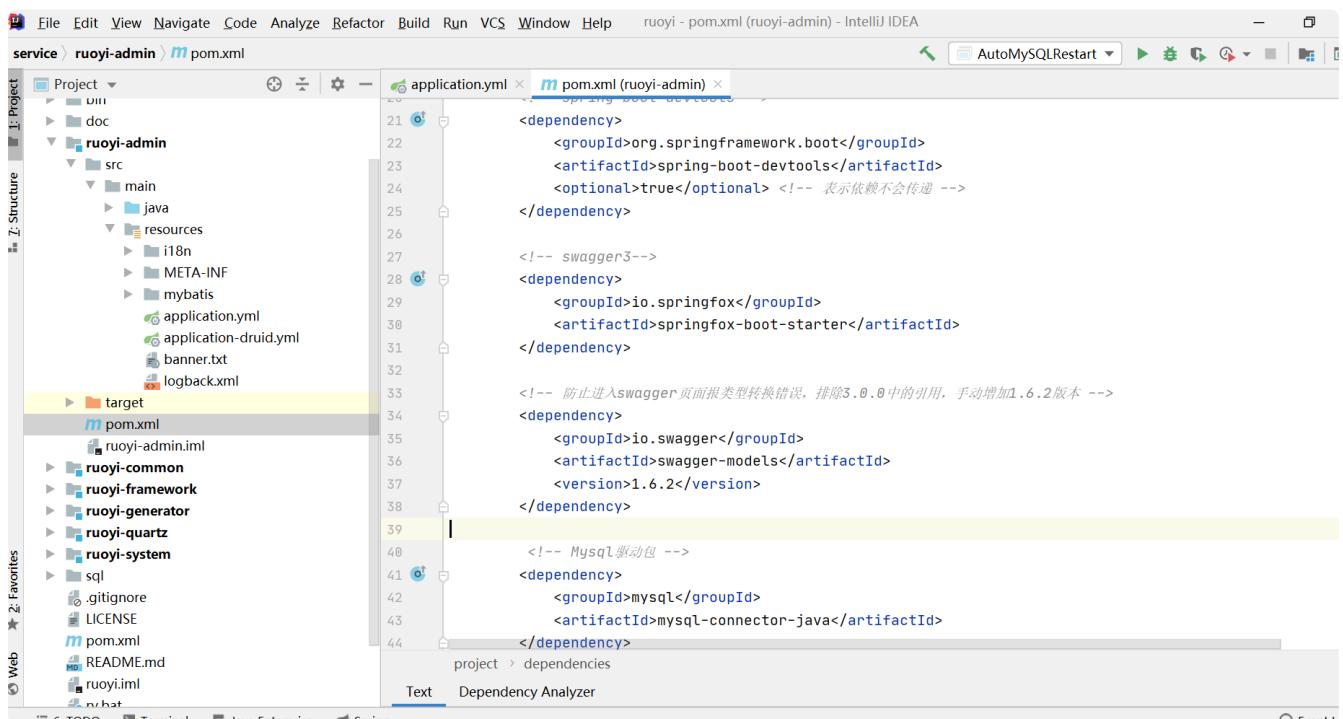
六、实验内容及步骤

1、Postman 安装与使用

- (1) 访问官网(<https://www.postman.com/downloads/>)下载、安装 Postman
- (2) Postman 左侧界面创建测试项目

2、Swagger 部署与应用

- (1) 在 pom.xml 引入 Swagger 依赖并在 application.xml 中配置



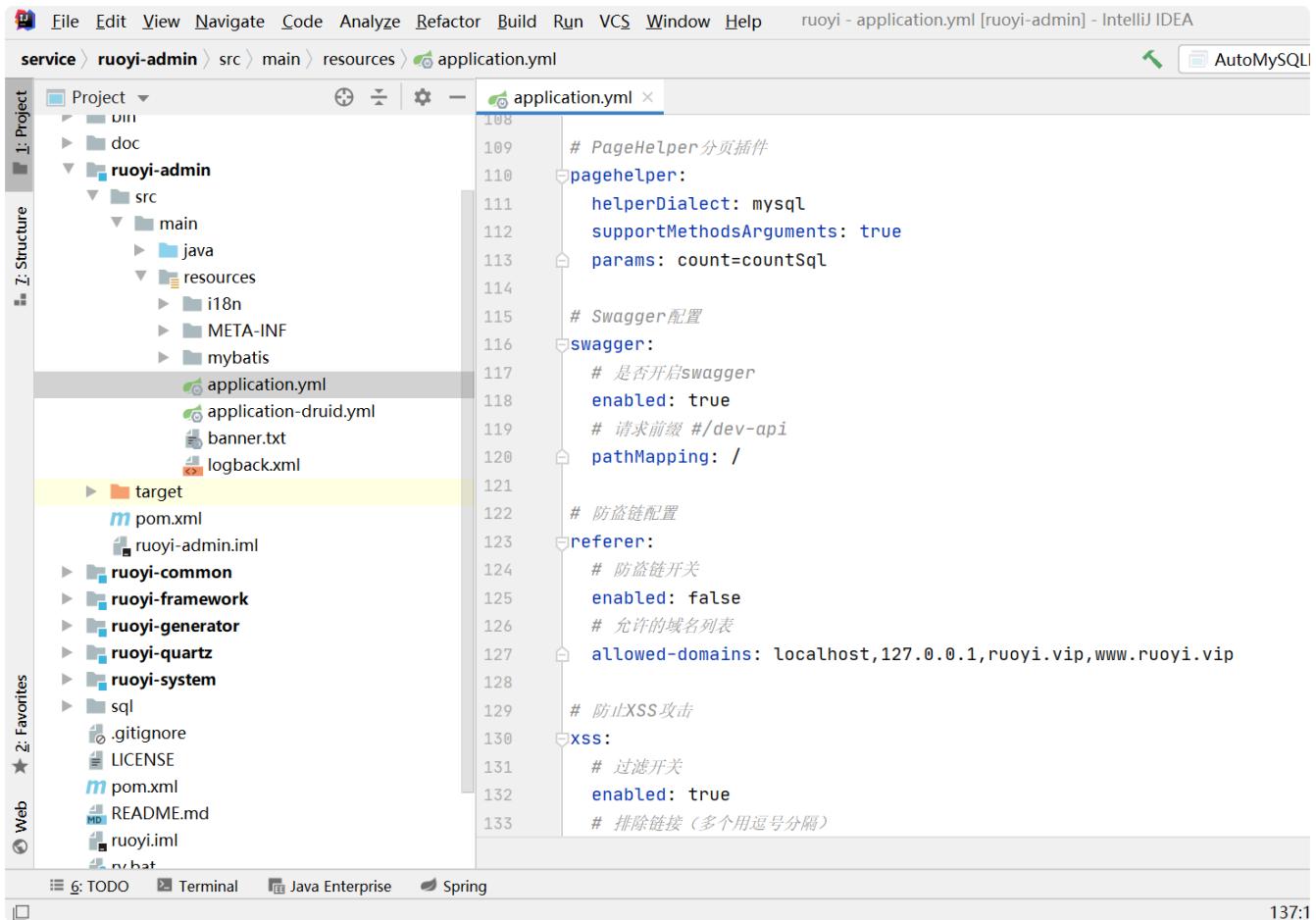
```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <optional>true</optional> <!-- 表示依赖不会传递 -->
</dependency>

<!-- swagger3-->
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-boot-starter</artifactId>
</dependency>

<!-- 防止进入swagger 页面报类型转换错误，排除3.0.0 中的引用，手动增加1.6.2版本 -->
<dependency>
    <groupId>io.swagger</groupId>
    <artifactId>swagger-models</artifactId>
    <version>1.6.2</version>
</dependency>

<!-- Mysql驱动包 -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
</dependency>
```

图 6-2-1-1 在 pom.xml 引入 Swagger 依赖



```

108 # PageHelper分页插件
109 <pagehelper:
110   helperDialect: mysql
111   supportMethodsArguments: true
112   params: count=countSql
113
114 # Swagger配置
115 <swagger:
116   # 是否开启swagger
117   enabled: true
118   # 请求前缀 #/dev-api
119   pathMapping: /
120
121 # 防盗链配置
122 <referer:
123   # 防盗链开关
124   enabled: false
125   # 允许的域名列表
126   allowed-domains: localhost,127.0.0.1,ruoyi.vip,www.ruoyi.vip
127
128 # 防止XSS攻击
129 <xss:
130   # 过滤开关
131   enabled: true
132   # 排除链接 (多个用逗号分隔)
133

```

图 6-2-1-2 swagger 在 application.xml 的配置

3、使用 postman 进行接口测试

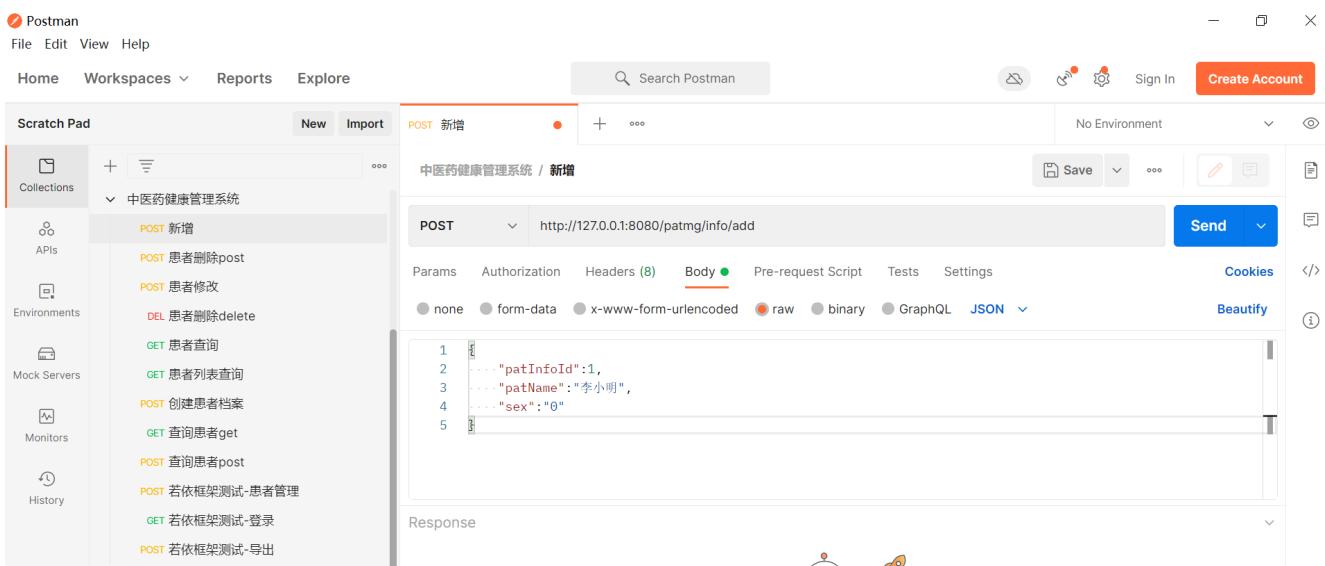
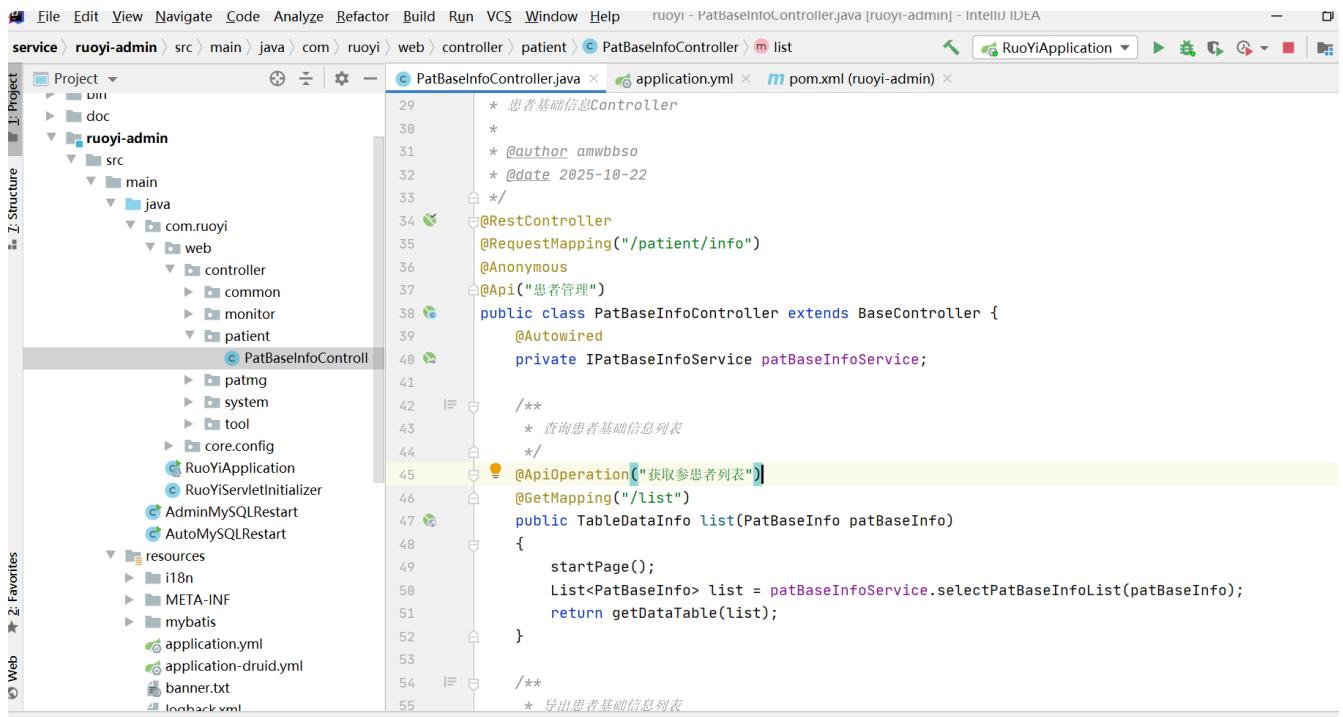


图 6-3-1 使用 postman 进行接口测试

4、使用 Swagger 进行接口测试

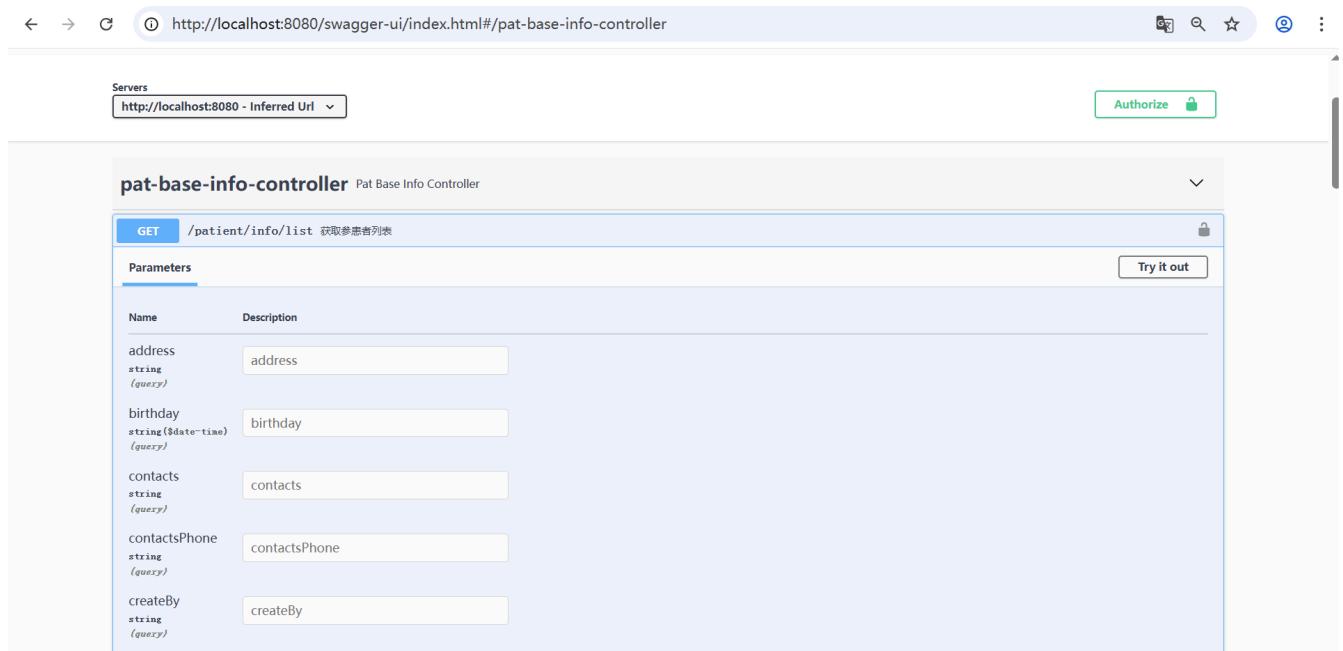
(1) 在 Controller 增加注解@Api 和 @ApiOperation



```
29     * 患者基础信息Controller
30     *
31     * @author amwbbso
32     * @date 2025-10-22
33     */
34     @RestController
35     @RequestMapping("/patient/info")
36     @Anonymous
37     @Api("患者管理")
38     public class PatBaseInfoController extends BaseController {
39         @Autowired
40         private IPatBaseInfoService patBaseInfoService;
41
42         /**
43          * 查询患者基础信息列表
44          */
45         @ApiOperation("获取患者列表")
46         @GetMapping("/list")
47         public TableDataInfo list(PatBaseInfo patBaseInfo) {
48             startPage();
49             List<PatBaseInfo> list = patBaseInfoService.selectPatBaseInfoList(patBaseInfo);
50             return getDataTable(list);
51         }
52
53         /**
54          * 导出患者基础信息列表
55         */
```

图 6-4-1 在 Controller 增加注解@Api 和 @ApiOperation

(2) 使用 swagger 测试接口(<http://localhost:8080/swagger-ui/index.html>)



pat-base-info-controller Pat Base Info Controller

GET /patient/info/list 获得患者列表

Parameters

Name	Description
address	address
birthday	birthday
contacts	contacts
contactsPhone	contactsPhone
createBy	createBy

Try it out

图 6-4-2-1 使用 Swagger 测试接口

Request URL
http://localhost:8080/patient/info/list

Server response

Code Details

200 Response body

```
[{"total": 12, "rows": [{"createBy": "", "createTime": null, "updateBy": null, "updateTime": null, "status": 1, "patInfoId": 4, "patientId": "123", "patName": "张三", "sex": "男", "age": 0, "issueSex": null, "idType": "1", "idNo": "41010419830106001X", "birthday": null, "phone": "13333333333", "contact": null, "contactPhone": null, "nativePlace": "郑州", "nation": "汉", "education": "大学", "maritalStatus": null, "occupation": null, "housekeeperType": null, "residence": null, "other": ""}]}]
```

Download

Response headers

```
connection: keep-alive
content-type: application/json
date: Tue, 28 Oct 2025 05:13:16 GMT
keep-alive: timeout=60
transfer-encoding: chunked
vary: Origin,Access-Control-Request-Method,Access-Control-Request-Headers
x-content-type: application/json
x-frame-options: SAMEORIGIN
x-xss-protection: 1, mode=block
```

图 6-4-2-2 查看返回报文

5、使用 Postman 实现接口自动化测试

(1) 编写 Tests 脚本

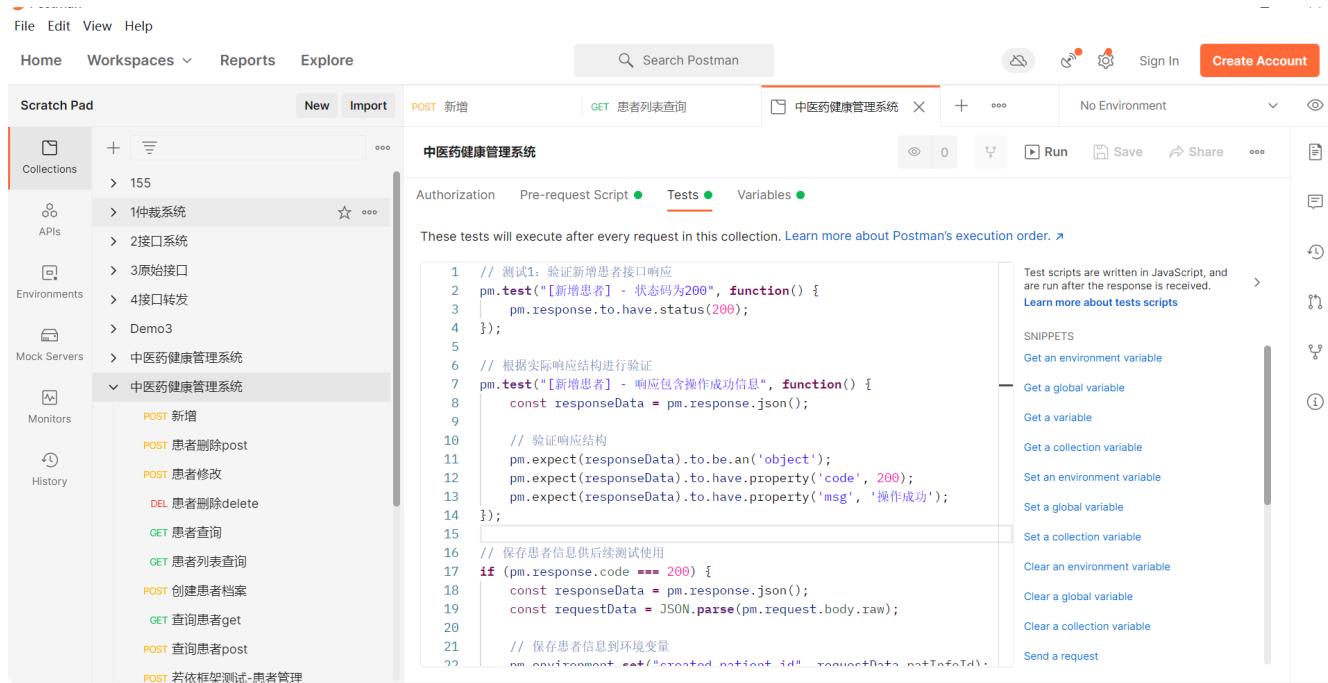
Java

```
1 // 测试：验证新增患者接口响应
2 pm.test("[新增患者] - 状态码为200", function() {
3     pm.response.to.have.status(200);
4 });
5
6 // 根据实际响应结构进行验证
7 pm.test("[新增患者] - 响应包含操作成功信息", function() {
8     const responseData = pm.response.json();
9
10    // 验证响应结构
11    pm.expect(responseData).to.be.an('object');
12    pm.expect(responseData).to.have.property('code', 200);
13    pm.expect(responseData).to.have.property('msg', '操作成功');
14 });
15
16 // 保存患者信息供后续测试使用
17 if (pm.response.code === 200) {
18     const responseData = pm.response.json();
19     const requestData = JSON.parse(pm.request.body.raw);
20
21     // 保存患者信息到环境变量
22     pm.environment.set("created_patient_id", requestData.patInfoId);
23     pm.environment.set("created_patient_name", requestData.patName);
24     pm.environment.set("created_patient_sex", requestData.sex);
25
26     console.log("✅ 新增患者成功");
27     console.log("响应消息: " + responseData.msg);
28     console.log("患者ID: " + pm.environment.get("created_patient_id"));
29     console.log("患者姓名: " + pm.environment.get("created_patient_na
e"));
30     console.log("患者性别: " + pm.environment.get("created_patient_se
x"));
31 }
32
33 // 验证响应时间
34 pm.test("[新增患者] - 响应时间小于3秒", function() {
35     pm.expect(pm.response.responseTime).to.be.below(3000);
36 });
37
38 // 验证响应头
39 pm.test("[新增患者] - 响应内容类型正确", function() {
40     pm.expect(pm.response.headers.get('Content-Type')).to.include('appl
```

```
ication/json');
41});
```

代码 6-5-1-1 新增患者接口 Tests 示例

(2) 测试接口



```

1 // 测试1: 验证新增患者接口响应
2 pm.test("[新增患者] - 状态码为200", function() {
3     pm.response.to.have.status(200);
4 });
5
6 // 根据实际响应结构进行验证
7 pm.test("[新增患者] - 响应包含操作成功信息", function() {
8     const responseData = pm.response.json();
9
10    // 验证响应结构
11    pm.expect(responseData).to.be.an('object');
12    pm.expect(responseData).to.have.property('code', 200);
13    pm.expect(responseData).to.have.property('msg', '操作成功');
14 });
15
16 // 保存患者信息供后续测试使用
17 if (pm.response.code === 200) {
18     const responseData = pm.response.json();
19     const requestData = JSON.parse(pm.request.body.raw);
20
21     // 保存患者信息到环境变量
22     pm.environment.set("created_patient_id", responseData.patientId);
}

```

● 图 6-5-2-1 编写测试脚本示例

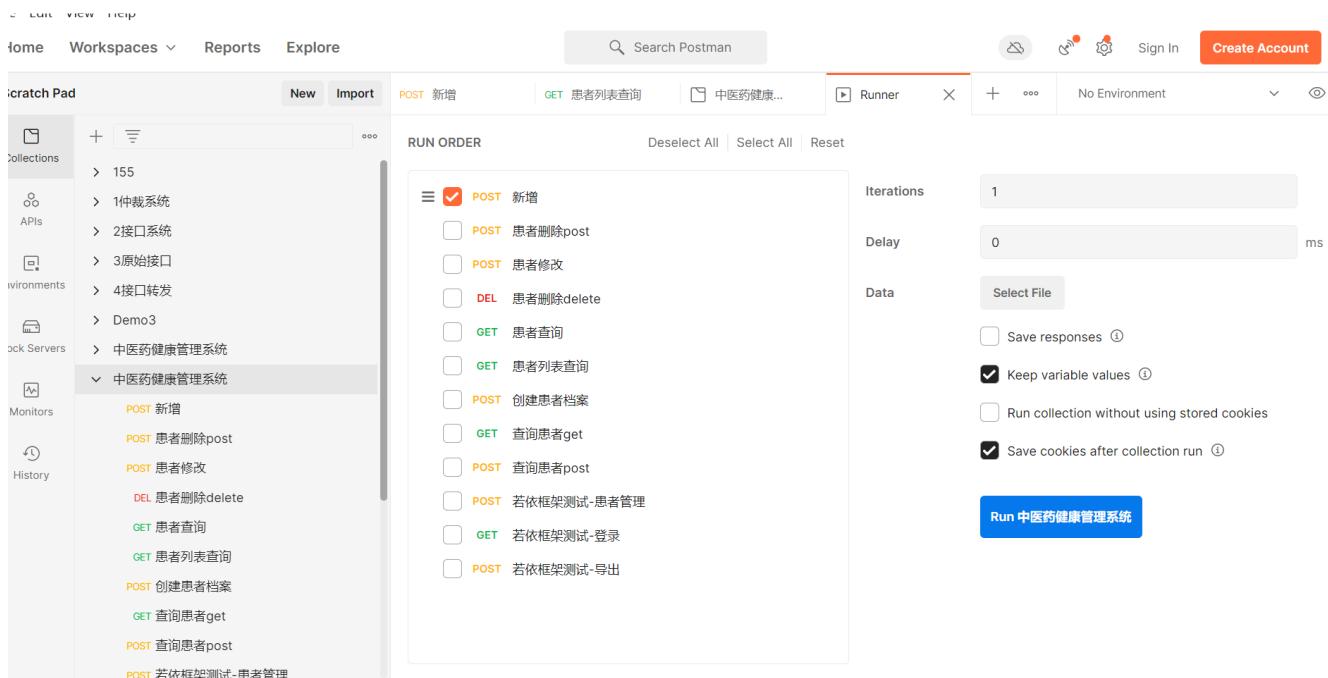


图 6-5-2-2 测试执行新增患者示例

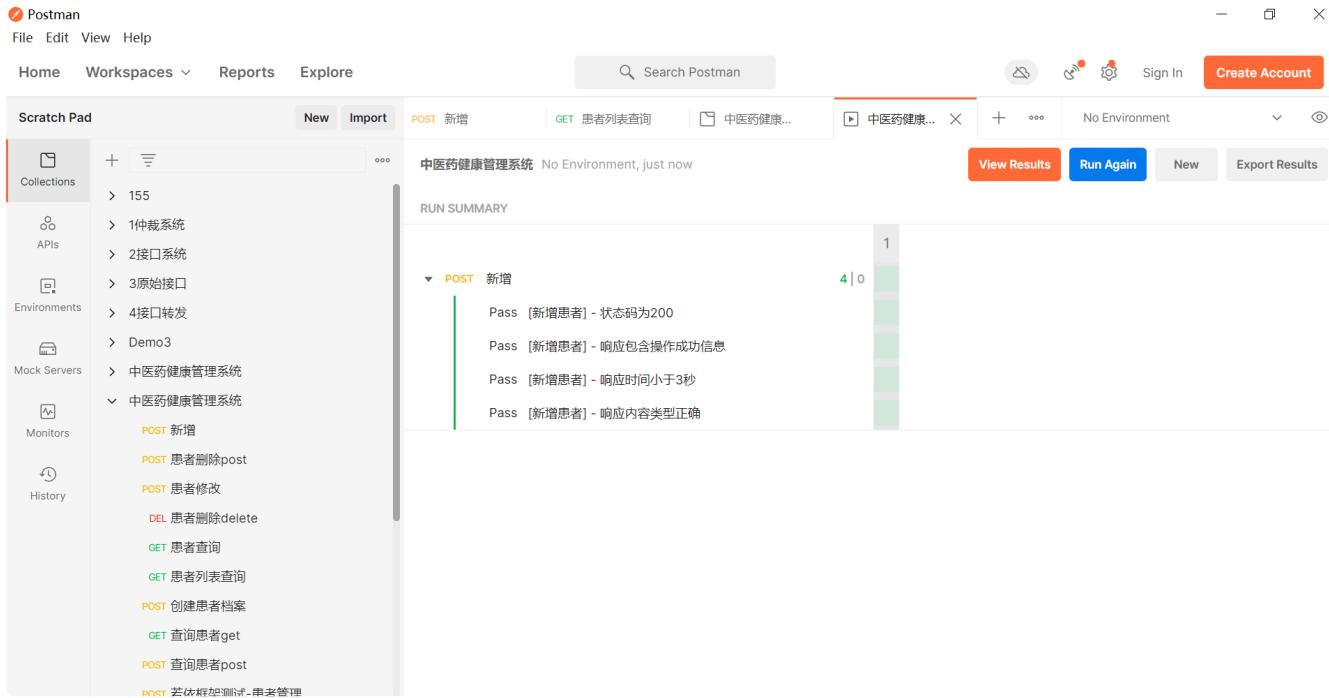


图 6-5-2-3 查看测试结果示例

七、实验考核

1、本课程实验考核方案

本课程实验考核采用【实验智能评】【实验随堂查】方式开展，根据不同的实验内容选择不同的考核方式。

【实验智能评】：实验完成后提交 GitLab，通过自动化代码评审工具进行评分。

【实验随堂查】：在实验课上通过现场演示的方式向实验指导教师进行汇报，并完成现场问答交流。

2、本实验考核要求

本实验考核方式：实验智能评

实验 4-9 作为本课程第 2 次实验考核。

考核要求：

- (1) 学生通过 GitLab 提交实验成果：{此部分说明需要提交的内容}。
- (2) 由 GitLab 根据成果和交流情况综合评分。