Table of Contents

接口测试课程

课程目标

- 能够根据接口API文档编写接口测试用例
- 能够使用Postman工具进行接口测试，并能够对大量接口用例进行管理、对接口响应结果进行断言、处理多接口的依赖及生成测试报告
- 能够使用Python+Requests封装的接口测试框架，实现接口对象封装、测试用例编写、测试数据管理及生成测试报告

课程大纲

章节	知识点	
第1章 接口测试基础	1.接口及接口测试概念 2.HTTP协议 3.接口规范	4.接口测试流程 5.项目环境说明
第2章 Postman实现接口测试	1.Postman介绍和安装 2.Postman基本用法 3.Postman高级用法	4.Postman测试报告 5.项目实战
第3章 数据库操作	1. 数据库介绍 2. 数据库基本操作	3. 数据库事务操作
第4章 代码实现接口测试	1. Requests库 2. 集成UnitTest	3. 接口测试框架开发 4. 项目实战
第5章 持续集成	1. 持续集成介绍 2. Git与Git代码托管平台 3. Jenkins	4. 持续集成之Postman 5. 持续集成之代码
第6章 接口测试扩展	1. 接口Mock测试	2. 接口测试总结

接口测试基础

目标

1. 理解接口及接口测试的相关概念
2. 熟悉HTTP协议和接口规范
3. 掌握接口测试流程
4. 熟练掌握如何解析接口文档
5. 熟悉掌握如何编写接口测试用例

接口及接口测试概念

目标

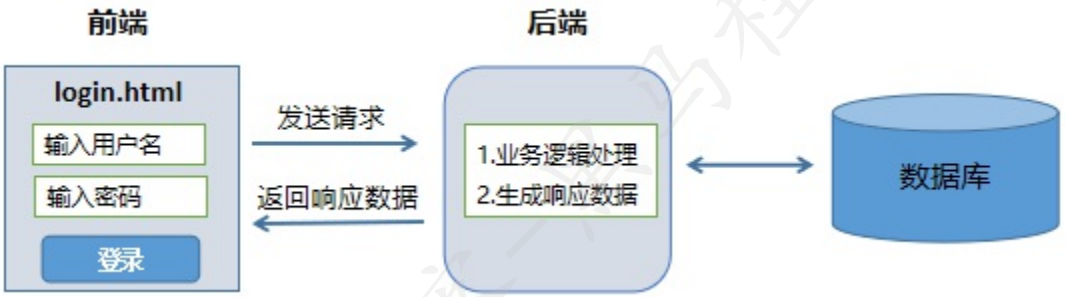
1. 了解什么是接口
2. 理解接口测试的概念
3. 掌握接口测试的原理
4. 理解什么是接口自动化测试

---

1. 接口

> 接口可分为：硬件接口和软件接口；我们这里只关注软件层面的接口。

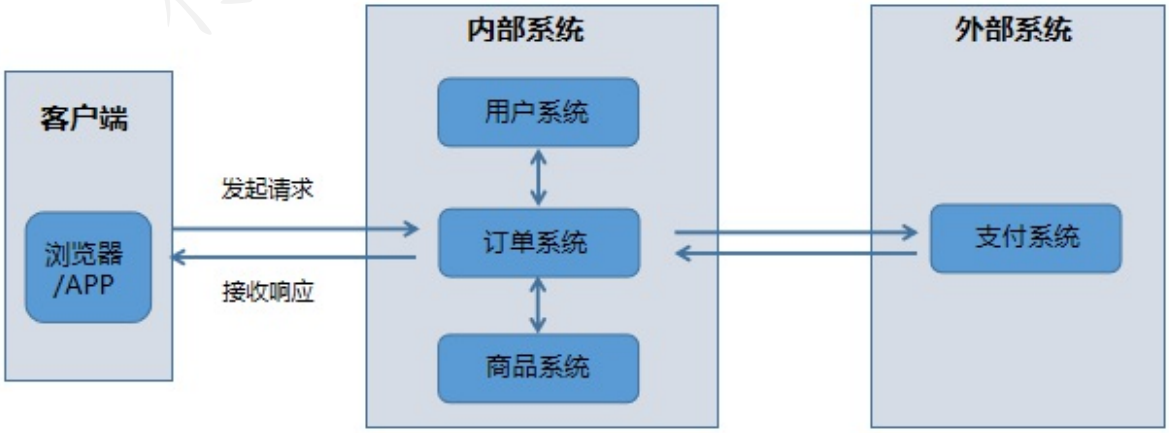
接口：是指系统或组件之间的交互点，通过这些交互点可以实现数据的交互。（数据交互的通道）



1.1 接口的类型

接口测试分类有许多种，按照范围划分：系统之间的接口和程序内部的接口

- 系统之间的接口：多个内部系统之间的交互，内部系统与外部系统之间的交互
- 程序内部的接口：方法与方法之间，模块与模块之间的交互



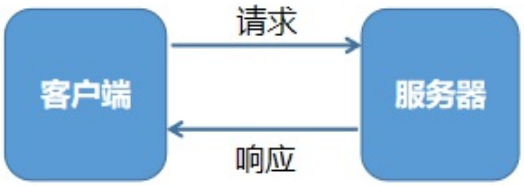
---

2. 接口测试

接口测试：是对系统或组件之间的接口进行测试，主要是校验数据的交换、传递和控制管理过程，以及相互逻辑依赖关系。

2.1 接口测试原理

模拟客户端向服务器发送请求，服务器接收请求后进行相应的业务处理，并向客户端返回响应数据，检查响应数据是否符合预期。



2.2 接口测试的特点

- 测试可以提前介入，提早发现Bug，符合质量控制前移的理念
- 可以发现一些页面操作发现不了的问题
- 接口测试低成本高效益（底层的一个Bug能够引发上层8个左右Bug，接口测试可以实现自动化）
- 不同于传统的单元测试，接口测试是从用户的角度对系统进行全面的检测

2.3 接口测试的实现方式

- 使用接口测试工具来实现（比如：JMeter、Postman）
- 通过编写代码来实现（比如：Python + Requests）

---

3. 接口自动化测试

3.1 概念

> 自动化测试：是把以人为驱动的测试行为转化为机器执行的一种过程。

接口自动化测试：是让程序或工具代替人工自动的完成对接口进行测试的一种过程。

HTTP协议

目标

1. 了解HTTP协议的特点
2. 掌握URL的组成部分
3. 掌握HTTP请求包含的内容
4. 掌握HTTP响应包含的内容
5. 了解常见的响应状态码

---

1. HTTP协议介绍

HTTP：（HyperText Transfer Protocol）超文本传输协议，是一个基于请求与响应模式的、应用层的协议，也是互联网上应用最为广泛的一种网络协议。

1.1 HTTP协议的特点

1. 支持客户端/服务器模式
2. 简单快速
3. 灵活
4. 无连接
5. 无状态

---

2. URL

URL：（Uniform Resource Locator）统一资源定位符，是互联网上标准资源的地址。HTTP使用URL来建立连接和传输数据。

URL格式

```
http://www.itcast.cn:8080/news/index.html?uid=123&page=1
```

- 协议部分：“http”，常见的协议有HTTP，HTTPS、FTP等
- 域名部分：“www.itcast.cn”，也可以使用IP地址作为域名使用
- 端口部分：“8080”，端口可以省略，默认端口（HTTP:80，HTTPS:443，FTP:21）
- 资源路径部分：“/news/index.html”
- 查询参数部分：“uid=123&page=1”，可以允许有多个参数，多个之间用“&”作为分隔符

示例

查询天气信息：http://www.weather.com.cn/data/sk/101010100.html

---

3. HTTP请求

http请求由三部分组成，分别是：请求行、请求头、请求体

```
POST http://demo.zentao.net/user-login.html HTTP/1.1
Host: demo.zentao.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://demo.zentao.net/user-login.html
Content-Type: application/x-www-form-urlencoded
Content-Length: 54
Connection: keep-alive
Upgrade-Insecure-Requests: 1
                                    // 该空行表示请求头数据已经结束
account=demo&password=efc4a3b32e48054865e5a8321cfda3e4
```

请求行

请求行用来说明请求方法、要访问的资源以及所使用的协议版本

常用请求方法：

- GET：从服务器获取资源（一项或多项）
- POST：在服务器新建一个资源
- PUT：在服务器更新资源（客户端提供改变后的完整资源）
- DELETE：从服务器删除资源

其他请求方法（了解）：

- HEAD：请求获取由Request-URI所标识的资源的响应消息报头
- TRACE：请求服务器回送收到的请求信息，主要用于测试或诊断
- CONNECT：保留将来使用
- OPTIONS：请求查询服务器的性能，或者查询与资源相关的选项和需求

请求头

请求头紧接着请求行，请求头部由键值对组成，每行一对。 请求头部通知服务器有关于客户端请求的信息，典型的请求头有：

- User-Agent：产生请求的浏览器类型
- Accept：客户端可识别的内容类型列表
- Content-Type：请求体数据的类型，常见的类型有：
  - text/html： HTML格式
  - text/plain：纯文本格式
  - image/jpeg：jpg图片格式
  - application/json： JSON数据格式
  - application/x-www-form-urlencoded： form表单数据被编码为key/value格式发送到服务器（表单默认的提交数据格式）
  - multipart/form-data： 在表单中进行文件上传时使用

请求体

- 请求体不在GET方法中使用，经常在POST、PUT方法中使用
- 请求体的数据可以是：表单数据、文本、XML、JSON
- 与请求数据相关的最常使用的请求头是Content-Type和Content-Length

---

4. HTTP响应

HTTP响应也由三个部分组成，分别是：状态行、响应头、响应体

```
HTTP/1.1 200 OK
Date: Fri, 22 May 2009 06:07:21 GMT
Content-Type: text/html; charset=UTF-8

<html>
      <head></head>
      <body>...</body>
</html>
```

状态行

状态行由协议版本号、状态码、状态消息三部分组成

状态码有三位数字组成，第一个数字定义了响应的类别：

- 1xx：指示信息--表示请求已接收，继续处理
- 2xx：成功--表示请求已被成功接收、理解、接受
- 3xx：重定向--要完成请求必须进行更进一步的操作
- 4xx：客户端错误--请求有语法错误或请求无法实现
- 5xx：服务器端错误--服务器未能实现合法的请求

> 详细信息见本节后面的列表

响应头

响应头用于描述服务器的基本信息，以及数据的描述，服务器通过这些数据的描述信息，可以通知客户端如何处理响应数据

响应体

响应体就是响应的消息体，数据可以是普通文本、XML、JSON、HTML源码

---

5. 状态码（Status Codes）[科普]

服务器向用户返回的状态码和提示信息，常见的有：

状态码	状态消息	动词	说明
200	OK	[GET]	服务器成功返回用户请求的数据

201	CREATED	[POST/PUT]	用户新建或修改数据成功
202	Accepted	[*]	表示一个请求已经进入后台排队（异步任务）
204	NO CONTENT	[DELETE]	用户删除数据成功
400	Bad Request	[POST/PUT]	客户端请求有语法错误，不能被服务器所理解
401	Unauthorized	[*]	表示用户没有权限（令牌、用户名、密码错误）
403	Forbidden	[*]	表示用户得到授权（与401错误相对），但是访问是被禁止的
404	Not Found	[*]	请求资源不存在，eg：输入了错误的URL
406	Not Acceptable	[GET]	用户请求的格式不可得（比如用户请求JSON格式，但是只有XML格式）
410	Gone	[GET]	用户请求的资源被永久删除，且不会再得到的
500	INTERNAL SERVER ERROR	[*]	服务器发生错误，用户将无法判断发出的请求是否成功
503	Server Unavailable	[*]	服务器当前不能处理客户端的请求，一段时间后可能恢复正常

接口规范

目标

1. 了解传统的接口风格
2. 理解RESTful接口规范

> 思考：
> 如何让前端开发与后台接口开发人员更好的配合，提高工作效率？
> 无规矩不成方圆，制定接口规范

1. 传统接口风格

对用户进行操作的相关接口，包括增删改查

操作	请求方式	URL	成功状态码
查询某个用户	GET/POST	http://127.0.0.1:8080/myweb/user/getUser?id=1 http://127.0.0.1:8080/myweb/user/getById?id=1 http://127.0.0.1:8080/myweb/getUserById?id=1	200
查询所有用户	GET/POST	http://127.0.0.1:8080/myweb/user/getUserList http://127.0.0.1:8080/myweb/user/getUsers	200
添加用户	POST	http://127.0.0.1:8080/myweb/user/addUser	200
修改用户	POST	http://127.0.0.1:8080/myweb/user/updateUser	200
删除用户	GET/POST	http://127.0.0.1:8080/myweb/user/deleteUser?id=1	200

2. RESTful

2.1 定义

一种软件架构风格、设计风格，而不是标准，只是提供了一组设计原则和约束条件。

REST：即(Representational State Transfer)的缩写。词组的翻译是"表现层状态转化"。如果一个架构符合REST原则，就称它为RESTful架构。

2.2 RESTful接口风格

对用户进行操作的相关接口，包括增删改查

操作	请求方式	URL	成功状态码
查询某个用户	GET	http://127.0.0.1:8080/myweb/users/1	200

查询所有用户	GET	http://127.0.0.1:8080/myweb/users	200
添加用户	POST	http://127.0.0.1:8080/myweb/users	201
修改用户	PUT	http://127.0.0.1:8080/myweb/users/1	201
删除用户	DELETE	http://127.0.0.1:8080/myweb/users/1	204

2.3 RESTful架构特点

1. 每一个URL代表一种资源；
2. 客户端和服务器之间，传递这种资源的某种表现层；
3. 客户端通过四个HTTP动词，对服务器端资源进行操作，实现"表现层状态转化"；
4. 接口之间传递的数据最常用格式为JSON。

常用的HTTP动词有下面四个：

- GET：从服务器获取资源（一项或多项）
- POST：在服务器新建一个资源
- PUT：在服务器更新资源（客户端提供改变后的完整资源）
- DELETE：从服务器删除资源

接口测试流程

目标

1. 掌握接口的测试流程

1. 接口测试流程

1. 需求分析
   - 主要依据需求文档
2. 接口文档解析
   - 一般是由开发人员编写接口文档（API文档）
3. 设计测试用例
4. 执行测试
   - 使用接口测试工具实现
   - 通过编写代码实现
5. 接口缺陷管理与跟踪
6. 生成测试报告
7. 接口自动化持续集成（可选）

项目环境说明

目标

1. 熟悉项目功能
2. 了解项目架构

---

1. 项目介绍

IHRM 人力资源管理系统

功能模块

---

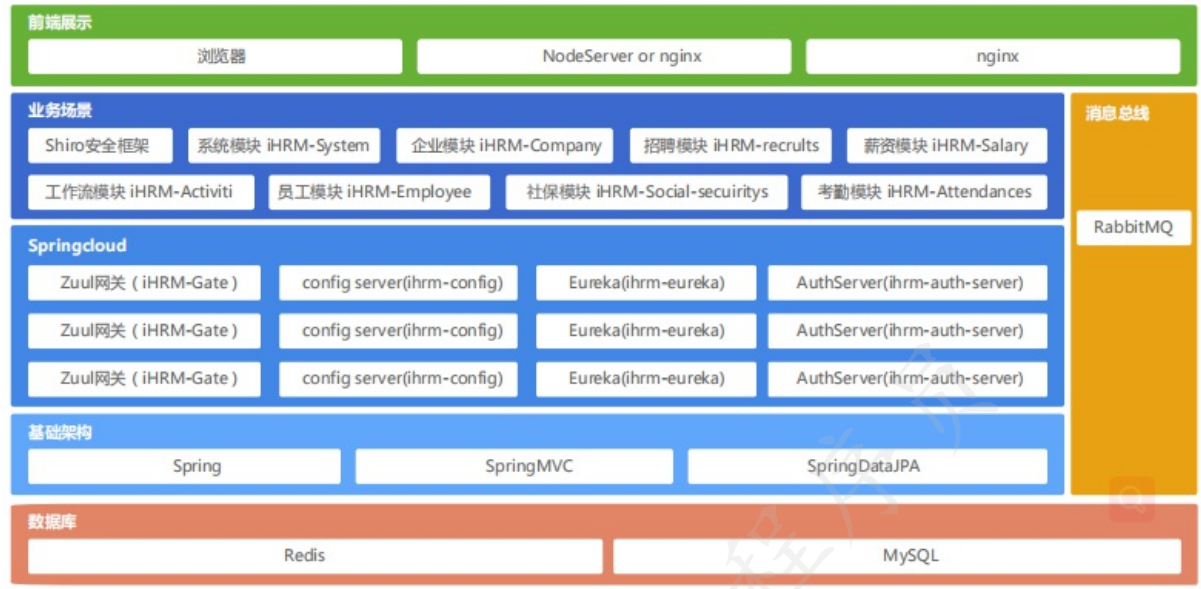
2. 技术架构

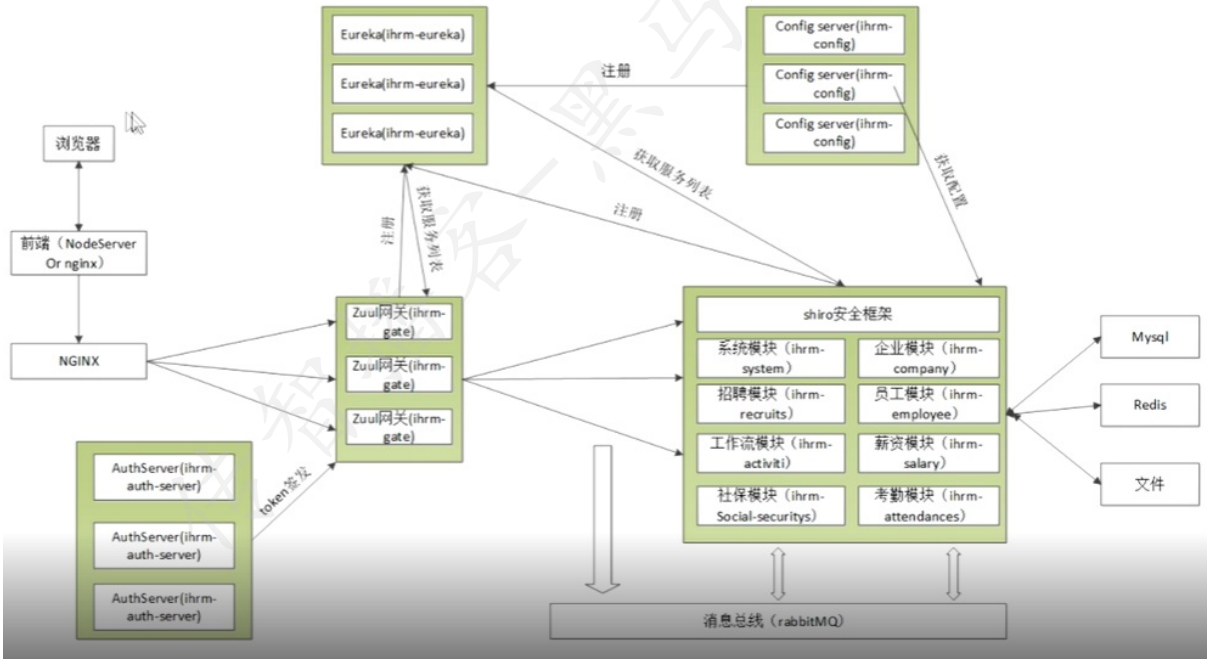
2.1 技术栈

- 前端：以Node.js为核心的Vue.js前端技术生态架构
- 后端：
  - SpringBoot+SpringCloud+SpringMVC+SpringData (Spring全家桶)

- MySQL + Redis + RabbitMQ

2.2 技术架构图





接口文档解析

目标

1. 知道什么是接口文档
2. 掌握接口文档包含的内容
3. 掌握如何解析接口文档

---

1. 接口文档介绍

1.1 什么是接口文档？

接口文档：又称为API文档，一般是由开发人员所编写的，用来描述系统所提供接口信息的文档。 大家都根据这个接口文档进行开发，并需要一直维护和遵守。

1.2 为什么要写接口文档？

1. 能够让前端开发与后台开发人员更好的配合，提高工作效率。（有一个统一参考的文件）
2. 项目迭代或者项目人员更迭时，方便后期人员查看和维护
3. 方便测试人员进行接口测试

---

2. 接口文档内容

一个规范的接口文档，要包含以下信息：

- 基本信息
  - 接口名称、请求方法、请求路径、接口描述
- 请求参数
  - 请求头
  - 请求体（包含具体的请求参数名称、参数类型、是否必须、示例、备注）
- 返回数据
  - 不同情况的响应状态码
  - 响应数据（包含具体的响应数据名称、类型、是否必须、默认值、示例、备注）

2.1 接口文档示例

用户认证（登录注册）

基本信息

Path： /app/v1_0/authorizations

Method： POST

接口描述：

```
1. 线上地址
    http://ttapi.research.itcast.cn/app/v1_0/authorizations
2.  返回HTTP状态码
    1. 201 OK
    2. 400 请求参数错误(包括：参数缺失、手机号格式不正确、验证码失效等)
    3. 507 服务器数据库异常
3. token说明
    token用于访问需要身份认证的普通接口，有效期2小时
```

请求参数

Headers

参数名称	参数值	是否必须	示例	备注
Content-Type	application/json	是		

Body

名称	类型	是否必须	默认值	备注	其他信息
mobile	string	必须		手机号	
code	string	必须		短信验证码	

返回数据

名称	类型	是否必须	默认值	备注	其他信息
message	string	必须		提示信息	
data	object	非必须		数据	
├─token	string	必须		用户token令牌	

3. 接口文档解析案例

查看人力资源管理系统的接口文档，解析以下接口：

1. 登录
2. 添加员工
3. 查询员工
4. 修改员工
5. 删除员工

接口用例设计

目标

1. 熟练掌握如何编写接口测试用例文档

1. 接口测试的测试点

2. 接口用例设计的方法与思路

> 本课程主要关注接口的功能测试

功能测试：验证接口功能是否按照接口文档实现（输入+处理+输出）

- 单接口测试
  - 正向功能：(通过性测试)
    - 仅必填参数
    - 全部参数
    - 参数组合
  - 反向测试：(异常测试)
    - 参数异常：无参、少参、多参、错误参数
    - 数据异常：数据为空、长度不符、类型不符、错误数据
    - 业务数据异常：结合业务功能考虑输出的各种异常返回情况
- 多接口测试：业务场景功能测试（站在用户角度考虑常用的使用场景）
  - 接口之间数据依赖

3. 单接口测试

针对人力资源管理系统的登录接口进行测试

ID	模块	用例名称	接口名称	请求URL	请求类型	请求头	请求参数类型	请求参数	预期结果	测试结果
001	登录	登录成功	登录	/api/sys/login	POST	{"Content-Type": "application/json"}	json	{"mobile":"13800000002", "password":"123456"}	登录成功，状态码：200，返回数据：{"success":true,"code":10000,"message":"操作成功！","data":"xxx"}	
002	登录	用户名不存在	登录	/api/sys/login	POST	{"Content-Type": "application/json"}	json	{"mobile":"13888889999", "password":"123456"}	登录失败，状态码：200，返回数据：{"success":false,"code":20001,"message":"用户名或密码错误","data":null}	
003	登录	密码错误	登录	/api/sys/login	POST	{"Content-Type": "application/json"}	json	{"mobile":"13800000002", "password":"error"}	登录失败，状态码：200，返回数据：{"success":false,"code":20001,"message":"用户名或密码错误","data":null}	
004	登录	请求参数为空	登录	/api/sys/login	POST	{"Content-Type": "application/json"}			登录失败，状态码：200，返回数据：{"success":false,"code":99999,"message":"抱歉，系统繁忙，请稍后重试！","data":null}	
005	登录	用户名为空	登录	/api/sys/login	POST	{"Content-Type": "application/json"}	json	{"mobile":"", "password":"123456"}	登录失败，状态码：200，返回数据：{"success":false,"code":20001,"message":"用户名或密码错误","data":null}	
006	登录	密码为空	登录	/api/sys/login	POST	{"Content-Type": "application/json"}	json	{"mobile":"13800000002", "password":""}	登录失败，状态码：200，返回数据：{"success":false,"code":20001,"message":"用户名或密码错误","data":null}	

4. 场景测试

登录系统后，对员工进行增删改查的操作

ID	模块	用例名称	接口名称	前置条件	请求URL	请求类型	请求头	请求参数类型	请求参数	预期结果	测试结果
001	员工	员工信息管理	添加员工	登录成功	/api/sys/user	POST	{"Content-Type": "application/json", "Authorization":"Bearer xxx"}	json	{"username": "tom", "mobile": "13012340001", "timeOfEntry": "2019-07-01", "formOfEmployment": 1, "workNumber": "1322131", "departmentName": "开发部", "departmentId": "1066240656856453120", "correctionTime": "2019-11-30"}	添加成功，状态码：200，返回数据：{"success":true,"code":10000,"message":"操作成功！", "data":{"id":"113749504"}}	
			查询员工	登录成功	/api/sys/user/:target	GET	{"Content-Type": "application/json", "Authorization":"Bearer xxx"}			查询成功，状态码：200，返回数据：{"success":true,"code":10000,"message":"操作成功！","data":{}}	
			修改员工	登录成功	/api/sys/user/:target	PUT	{"Content-Type": "application/json", "Authorization":"Bearer xxx"}	json	{"username":"tom-new"}	修改成功，状态码：200，返回数据：{"success":true,"code":10000,"message":"操作成功！","data":"113749504"}	
			删除员工	登录成功	/api/sys/user/:target	DELETE	{"Content-Type": "application/json", "Authorization":"Bearer xxx"}			删除成功，状态码：200，返回数据：{"success":true,"code":10000,"message":"操作成功！","data":null}	