

Table of Contents

接口测试	1.1
第1章-接口测试基础知识	1.2
接口及接口测试	1.2.1
接口测试工具	1.2.2
接口架构规范	1.2.3
案例-项目《学生管理》	1.2.4
思考?	1.2.5
第2章-接口测试工具的使用	1.3
Jmeter是什么? 有何用?	1.3.1
环境搭建 (Jmeter)	1.3.2
Jmeter功能概要	1.3.3
Jmeter 各元件中重点梳理	1.3.4
第3章-Jmeter工具核心知识点	1.4
Jmeter核心-参数化	1.4.1

接口测试

目标

1. 了解项目开发中为什么要采用接口
2. 了解接口测试课程学习大纲

1. 为什么要学习接口测试

1.1 项目开发中为什么要采用接口？

1. 开发效率和质量
2. 方便与第三方交互
3. 维护便捷(后台代码修改，接口无需改变)

1.2 什么是软件接口？

说明：一个数据访问地址，一个规范交互标准，对指定数据进行（增删改查）

1.3 总结：

软件项目开发模式大多数都采用接口了，那我们作为软件测试人员要学接口吗？

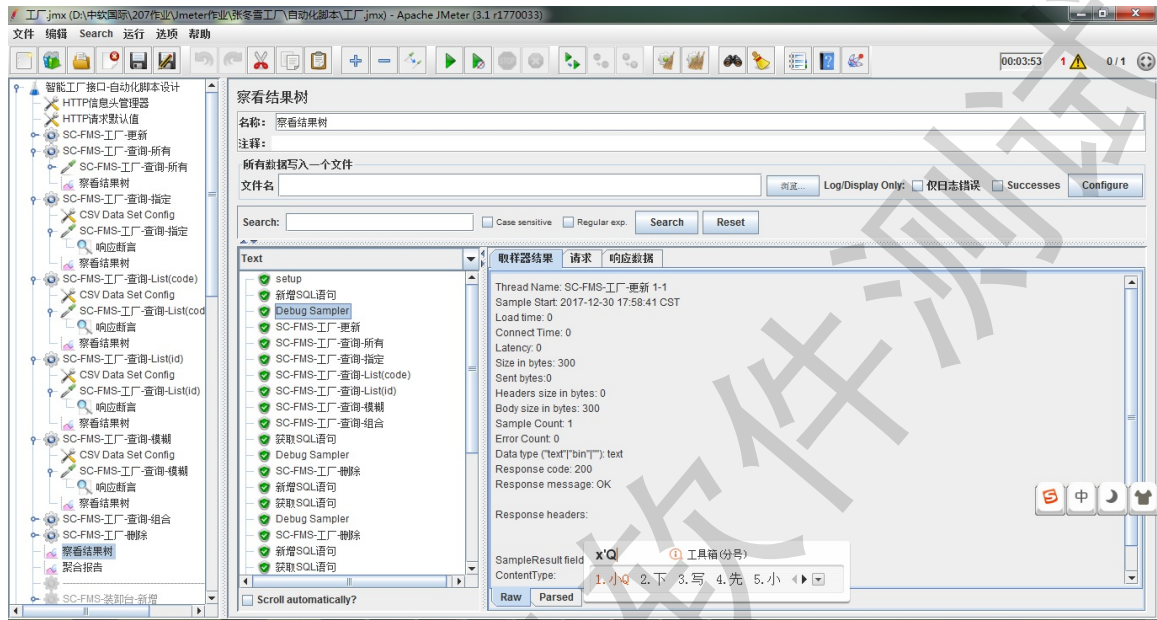
2. 接口测试课程大纲

2.1 安排

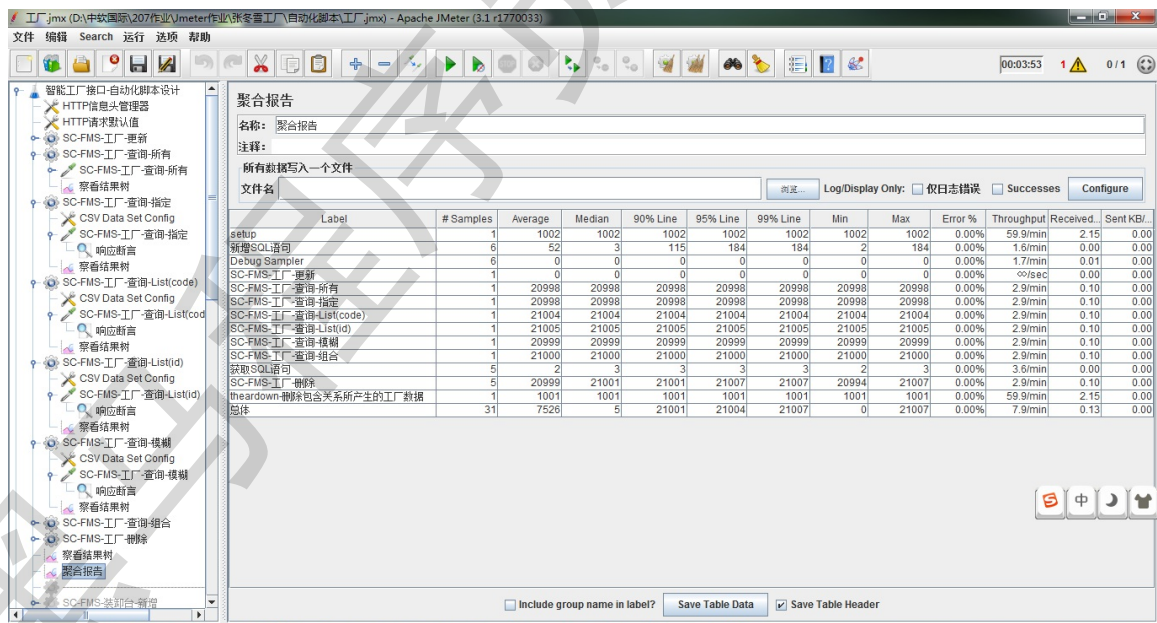
序号	阶段	内容
01	第一阶段	- 理解接口测试
02	第二阶段	- 熟悉Jmeter工具组成
03	第三阶段(核心)	- 参数化、集合点、关联、断言、数据库
04	第四阶段	- 属性管理器及逻辑控制器
05	第五阶段	- 项目实战(接口功能脚本、自动化脚本、性能脚本)

3. 接口学完样品

3.1 Jmeter工具元件使用



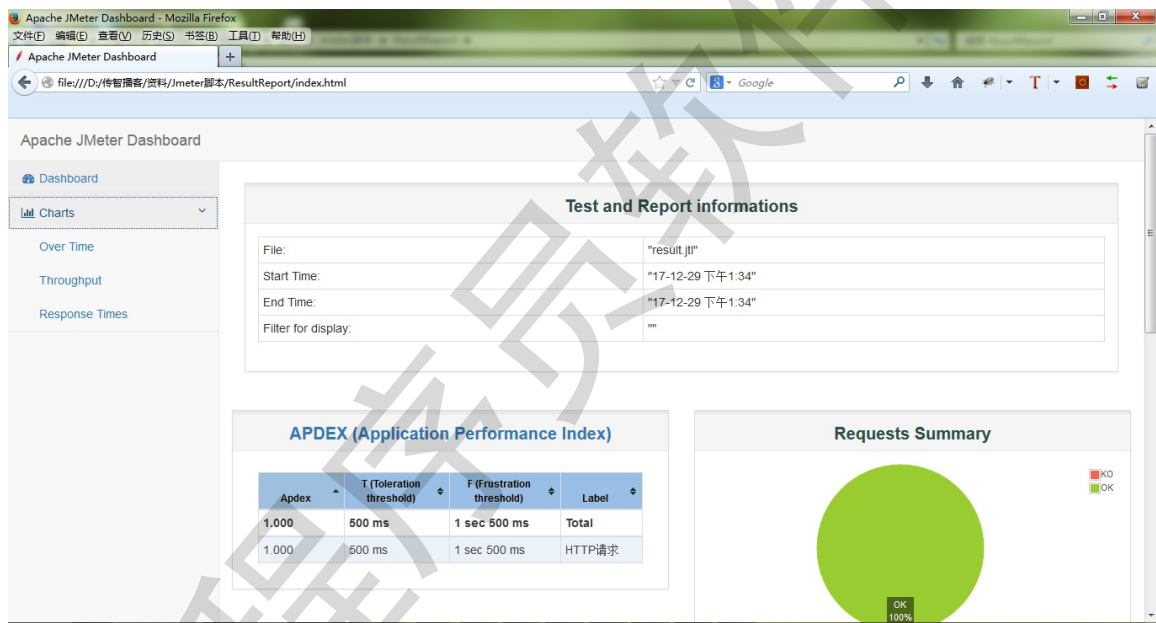
3.2 聚合报告



3.3 用例设计

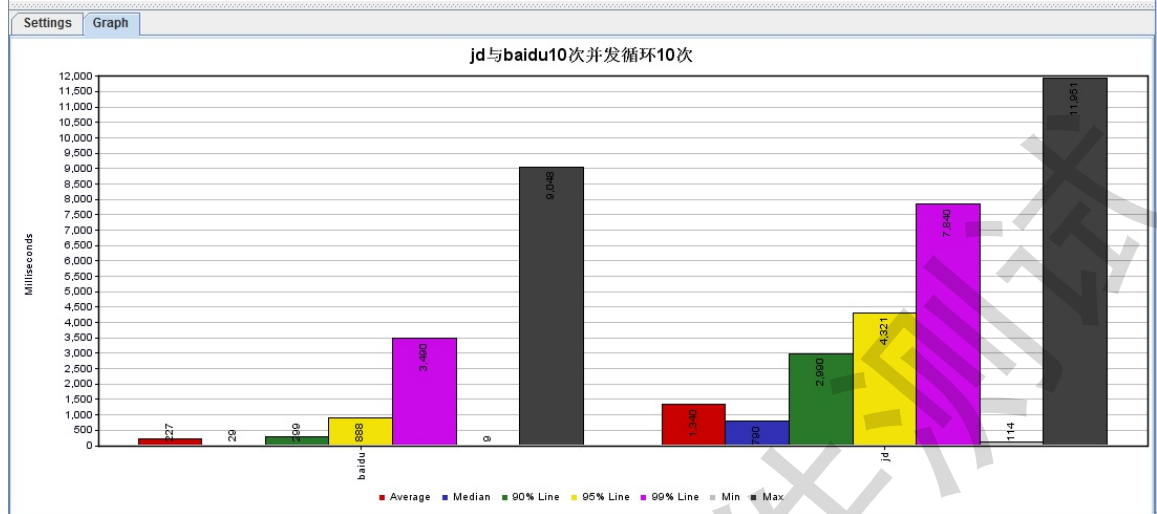
A	B	C
	查询指定工厂下装卸台-指定 http://service-root/FactoryModelService/Factories/{FactoryId}/LoadingDocks/{LoadingDockId} (FactoryId存在且格式正确) (LoadingDockId为格式不正确)	调用成功, 返回为空;
10 SC-FMS-装卸台List集合查		
11 询		
12	前提: HTTP Method为GET 查询指定工厂下装卸台-List集合 http://service-root/FactoryModelService/Factories/{FactoryId}/LoadingDocks?\${codeList}={code1}, {code2} (FactoryId存在且格式正确) (code1, code2存在且格式正确)	调用成功, 返回code1, code2的工厂列表集合信息;
13	http://service-root/FactoryModelService/Factories/{FactoryId}/LoadingDocks?\${idList}={id1}, {id2} (FactoryId存在且格式正确) (id1, id2存在且格式正确)	调用成功, 返回id1, id2工厂列表集合信息;
14	http://service-root/FactoryModelService/Factories/{FactoryId}/LoadingDocks?\${codeList}={code1}, {code2} (FactoryId存在且格式正确) (code1, code2一为正常, 一个为空或异常)	调用成功, 返回code正常的工厂信息
15	http://service-root/FactoryModelService/Factories/{FactoryId}/LoadingDocks?\${idList}={id1}, {id2} (FactoryId存在且格式正确) (id1, id2一为正常, 一个为空或异常)	调用成功, 返回id正常的工厂信息
16 SC-FMS-装卸台List查询-异		
17 常情况	前提: HTTP Method为GET 查询指定工厂下装卸台-List集合 http://service-root/FactoryModelService/Factories/{FactoryId}/LoadingDocks?\${codeList}={code1}, {code2} (FactoryId存在且格式正确) ([code1, code2]为空)	调用成功, 返回FactoryId工厂下全部装卸台信息;
18		
19		

3.4 html性能报告

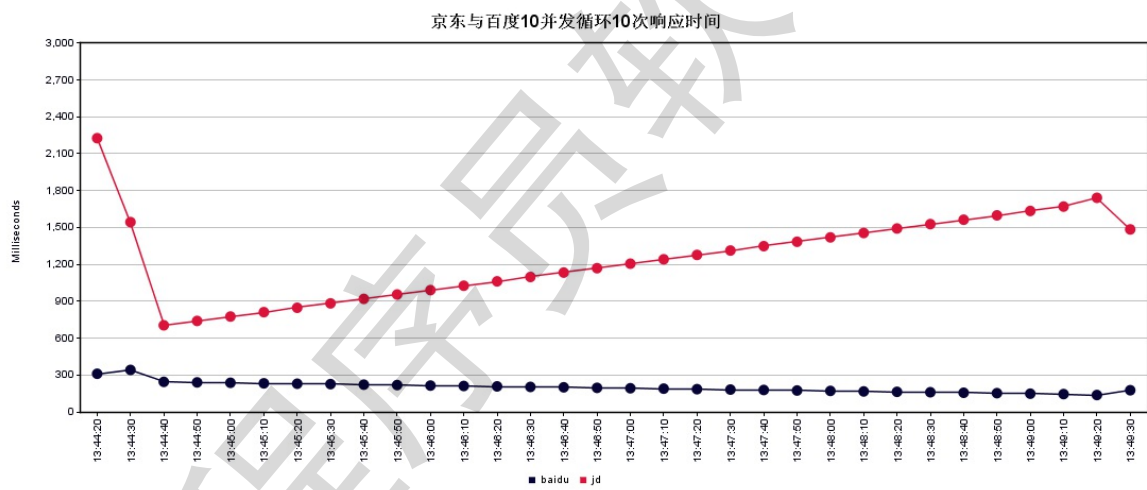


3.5 Aggregate Graph

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/s	Sent KB/sec
baidu	200	227	29	299	888	3490	9	9048	0.00%	38.0/min	1.72	0.11
jd	200	1340	790	2990	4321	7840	114	11951	0.50%	38.0/min	78.34	0.21
总体	400	783	257	2254	3359	6371	9	11951	0.25%	1.3/sec	80.06	0.32



3.6 Response Time



4. 学完这套课程，我学到到什么

1. 熟练使用Jmeter工具；
2. 掌握接口测试规范和RESTful风格；
3. 掌握基于Jmeter完成接口测试（功能、自动化、性能）脚本；
4. 掌握基于Jmeter对Web项目性能压测；

目标

1. 理解什么是接口？
2. 什么是接口测试？
3. 学习并掌握**RESTClient**接口插件工具的使用
4. 理解接口常用架构（**RESTful**风格）

接口与接口测试

目标

- 明白与理解接口

软件接口

```
http://www.sojson.com/open/api/weather/json.shtml?city=北京  
http://www.weather.com.cn/data/sk/101010100.html  
http://www.weather.com.cn/data/cityinfo/101010100.html
```

2、接口测试

2.1 概念

- 接口测试就是代替前端或者第三方验证后台响应数据是否正确

2.2 接口测试原理

- 模拟客户端向服务器发送请求报文，服务器接收请求报文后对相应的报文做处理并向客户端返回应答，客户端接收响应数据后并进行判断的一个过程。
 - 请求：是否正确，默认请求成功是200（GET），如果请求错误也能返回404、500等。
 - 检查：返回数据的正确性与完整性
 - 安全性：接口一般不会暴露在网上任意被调用，需要做一些限制，比如次数限制。

2.3 接口测试分类

- web接口测试
- 模块接口测试

1). web接口测试

- 服务器接口测试
- 外部接口测试

服务器接口测试

- 是测试浏览器与服务器的接口。

外部接口测试

- 就是第三方接口测试
- 举例：支付接口测试/天气预报接口测试

2.4 接口测试-优点

- 能为项目平台带来高效的缺陷监测和质量监督能力；
- 平台越复杂，系统越庞大，接口测试的效果越明显（提高测试效率，提升用户体验，降低研发成本）

常用接口测试插件工具

目标

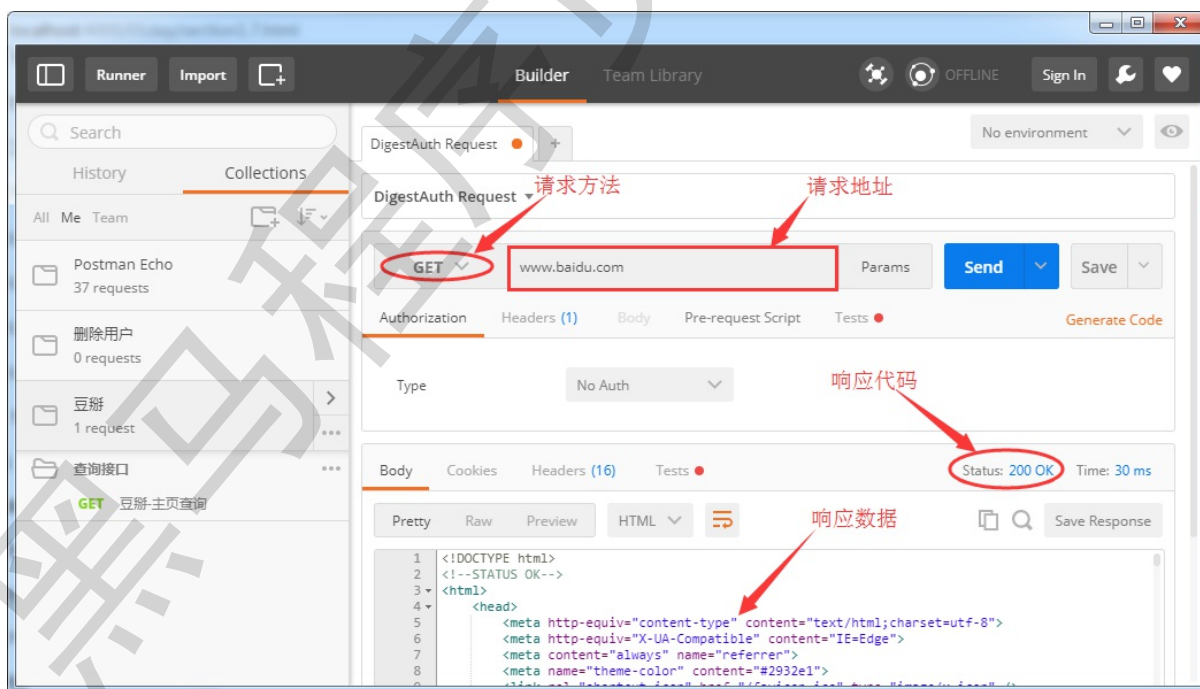
- 熟悉并了解开发常用接口测试插件
- 快速掌握离线及在线安装插件方式
- 体验接口测试

接口测试工具：

- Postman
- RestClient
- Jmeter

1. Postman

：是google开发的一款功能强大的网页调试与发送网页HTTP请求，并能运行测试用例的的Chrome插件



1.1 安装环境

- 安装-谷歌浏览器(65.0.3311.4_chrome)

- 安装-Postman插件(Postman_v4.1.3.zip)
- 安装方式(离线/在线)

1.2 Postman安装步骤

- 1). 安装谷歌浏览器
- 2). 打开浏览器扩展页面
 - chrome://extensions/
 - 自定义及控制->更多工具->程序扩展
- 3). 勾选开发者模式
- 4). 加载已解压的扩展程序

1.3 Postman演示

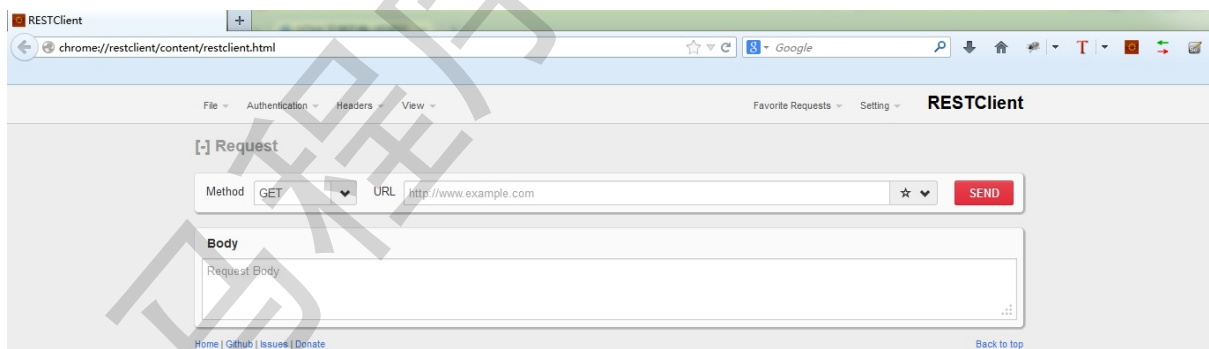
- 请求方法 (GET): 详情请见接口架构风格讲解
- 请求URL: <http://www.sojson.com/open/api/weather/json.shtml?city=北京>

1.4 结果

- 查看响应状态码
- 查看响应数据

2. RESTClient

: FirefoxRESTClient的插件，这款插件由国人开发，功能上支持于返回的数据高亮显示



2.1 安装环境

- 浏览器(火狐35.0)
- 接口插件工具(RESTClient 2.0.5)
- 安装方式(在线)

2.2 RESTClient安装步骤

- 1). 安装火狐浏览器

- 2). 打开浏览器-附加组件
 - about:addons
 - Ctrl+Shift+A
 - 工具菜单->附加组件
- 3). 搜索RESTClient 2.0.5
- 4). 安装

2.3 RESTClient演示

- 请求方法（GET）
- 请求URL: <http://www.weather.com.cn/data/sk/101010100.html>

2.4 结果

- 查看响应状态码
- 查看响应数据

2.5 JSON科普

概念：是一种轻量级的数据交换格式。

语法：JSON由键/值对组合方式，

```
{
  "name": "张三",
  "age": 18,
}

{
  "brand": "奔驰",
  "price": "50万"
}
```

总结

- 接口
- 接口测试
- Postman插件
- RESTClient插件

思考-接口的组成？

<http://www.weather.com.cn/data/sk/101010100.html>

```
http://127.0.0.1:8000/api/departments/  
http://127.0.0.1:8000/api/departments/T02/  
http://127.0.0.1:8000/api/departments/?$dep_id_list=T01,T02,T03
```

黑盒程序员软件测试

接口架构

目标

- 了解接口常用架构-RESTful架构相关知识

1. 定义

RESTful架构是一种接口设计架构风格，而不是标准，只是提供了一组设计原则。

2. 风格

- [http://服务器地址:端口号\[/项目名称/版本\]**资源**集合\[/单个资源\]](#)

- http://: 为我们HTTP协议的访问头标准
- 服务器地址: 为我们项目服务器IP地址
- 端口号: 为我们服务器内项目访问的指定编号
- [/项目名称/版本]: 可选
- 资源: 互联网-图片、音乐、视频、文本、数据

3. RESTful相关知识(科普)

3.3 HTTP请求方法

- GET (SELECT): 从服务器取出资源（一项或多项）。
- POST (CREATE): 在服务器新建一个资源。
- PUT (UPDATE): 在服务器更新资源（客户端提供改变后的完整资源）。
- DELETE (DELETE): 从服务器删除资源。

3.4 响应状态

客户端请求服务后，服务器响应给客户端的状态码。

3.5 状态码集合汇总

序号	状态码	动词	说明
			服务器成功返回用户请求的数据，该操作是幂等的（Idempotent）幂等:无论执

			行操作多少次，结果都会执行 1 次结果相同
02	201 CREATED	[POST/PUT/PATCH]	用户新建或修改数据成功
03	202 Accepted	[*]	表示一个请求已经进入后台排队（异步任务）
04	204 NO CONTENT	[DELETE]	用户删除数据成功
05	400 INVALID REQUEST	[POST/PUT/PATCH]	用户发出的请求有错误，服务器没有进行新建或修改数据的操作，该操作是幂等的
06	401 Unauthorized	[*]	表示用户没有权限（令牌、用户名、密码错误）
07	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
08	404 NOT FOUND	[*]	用户发出的请求针对的是不存在的记录，服务器没有进行操作，该操作是幂等的
09	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）
10	410 Gone	[GET]	用户请求的资源被永久删除，且不会再得到的
11	422 Unprocesable entity	[POST/PUT/PATCH]	当创建一个对象时，发生一个验证错误
12	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

3.6 对结果进行判断

序号	方法	预期结果
01	GET	collection: 代码: 200; 数据: 返回资源对象的列表（数组）
02	GET	collection/resource: 代码: 200; 数据: 返回单个资源对象
03	POST	collection: 代码: 200/201; 数据: 返回新生成的资源对象
04	PUT	collection/resource: 代码: 200/201; 数据: 返回完整的资源对象
05	DELETE	collection/resource: 代码: 204; 数据: 返回为空

4. 总结(RESTful)

--	--	--

序号	名称	值
01	定义	一种软件架构风格、设计风格，而不是标准
02	RESTful 风格	http://服务器地址:端口号/[服务名]/[版本]/资源集合/单个资源
03	请求方法	GET(获取资源);POST(新增资源);PUT(更新资源);DELETE(删除资源)
04	状态码	GET:200;POST:200/201;PUT:200/201;DELETE:204
05	[JSON]	是一种轻量级的数据交换格式;JSON是已键/值对组合方式，键名写在前面并用双引号 "" 包裹，使用冒号 : 分隔，然后紧接着值：1 {"name": "张三","age":18}

练习-**RESTful**风格的小项目来加强下对**RESTful**的认知

工具及资料

- 1.火狐浏览器及RESTClient插件
- 2.项目接口文档说明

案例-《学生信息管理》

目标：

- 验证RESTful风格，加强理解；
 - 通过实践，理解接口测试；
-

1、背景：

学生信息管理系统中接口采用了标准RESTful架构风格，帮助同学们更深切理解RESTful

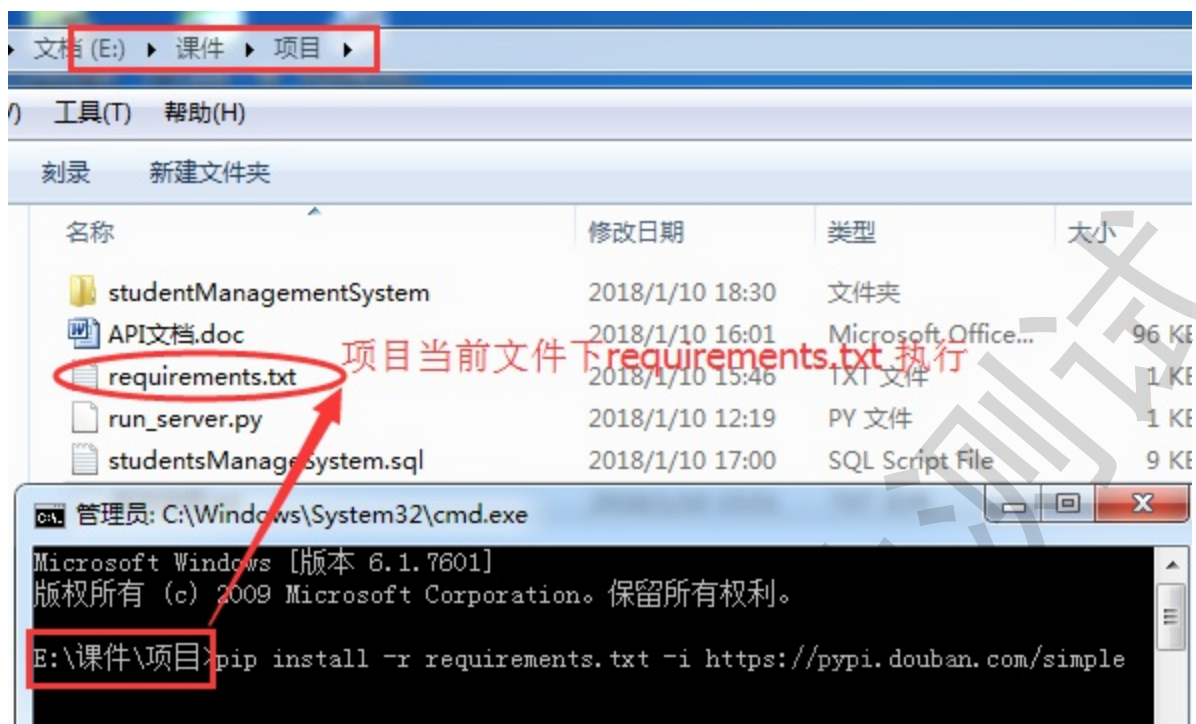
2、简介：

学生信息管理系统的功能是收集学生的个人信息，以便向老师提供每个学生在校或毕业生学籍的情况，还可以让学生用自己的学号去查看自己在校期间的表现。

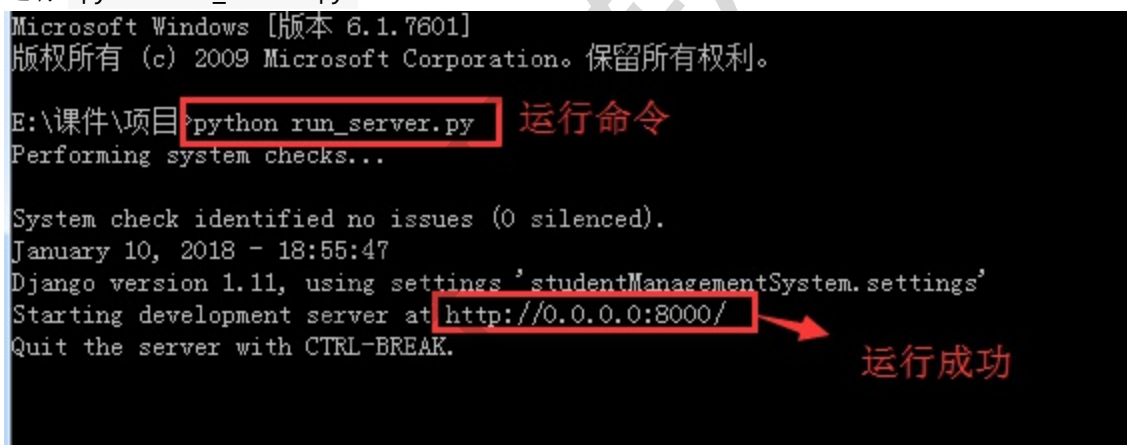
3、环境搭建

- 1). 安装Python3(3.5以上版本)
- 2). 搭建图书管理项目环境（安装依赖模块）
 - 在项目当前文件夹下执行CMD，运行以下命令：

```
pip install -r requirements.txt -i https://pypi.douban.com/simple
```

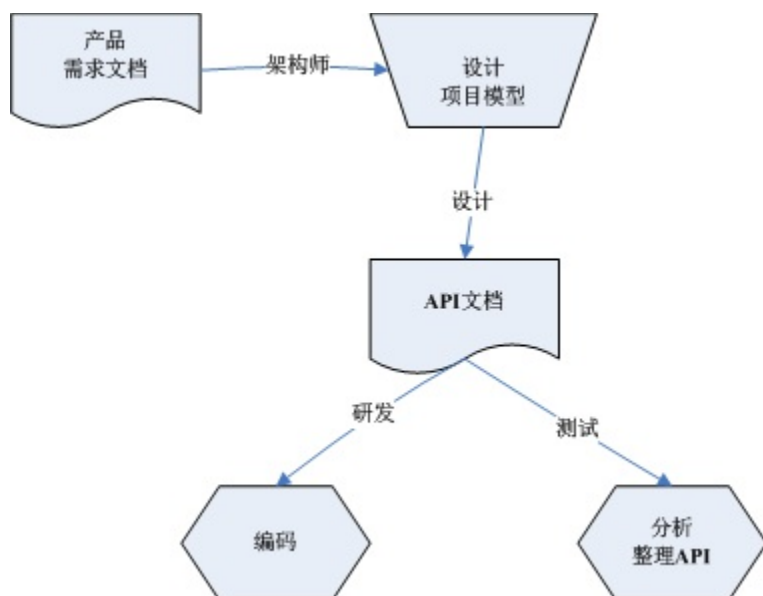
- 运行 `python run_server.py`



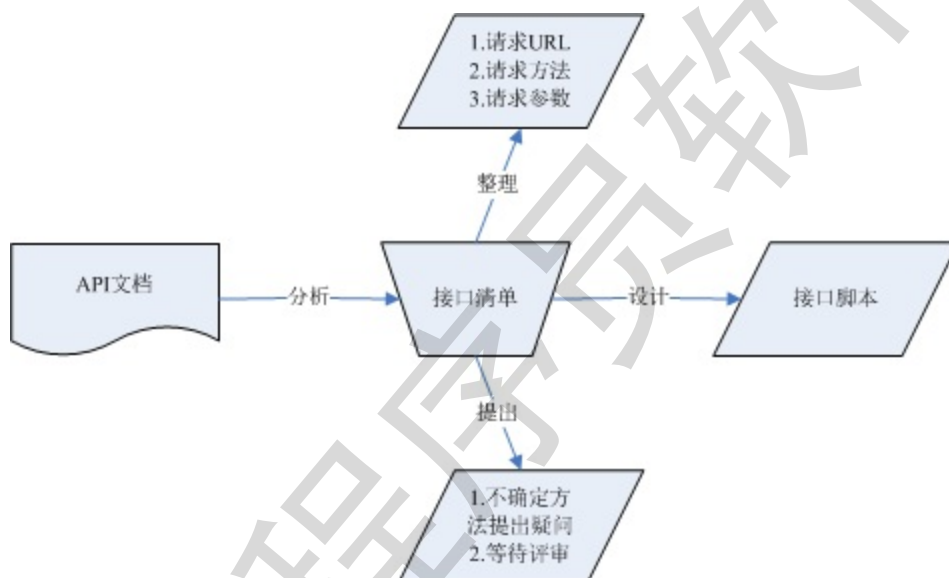
- RESTClient/Postman安装完毕
- 执行API接口清单

4、API接口清单

4.1 API文档由来



4.2 API文档作用



4.3 API接口清单

一、查询

1.1 学院-查询所有

请求方法: GET

请求地址: <http://127.0.0.1:8000/api/departments/>

1.2 学院-查询指定

请求方法: GET

请求地址: <http://127.0.0.1:8000/api/departments/T02/>

(注: T02为学院ID;)

1.3 根据指定参数进行学院-List-\$dep_id_list的相关查询

请求方法: GET

请求地址: [http://127.0.0.1:8000/api/departments/?\\$dep_id_list=T01,T02,T03](http://127.0.0.1:8000/api/departments/?$dep_id_list=T01,T02,T03)

(注: \$dep_id_list: 为参数名称; T01, T02, T03为: 学院ID;)

1.4学院-List-\$master_name_list查询

请求方法: GET

请求地址: [http://127.0.0.1:8000/api/departments/?\\$master_name_list=Java-Master,Test-Master](http://127.0.0.1:8000/api/departments/?$master_name_list=Java-Master,Test-Master)

(注: \$master_name_list: 为参数名称;Java-Master,Test-Master为:院长名称;)

1.5学院-模糊

请求方法: GET

请求地址: http://127.0.0.1:8000/api/departments/?blur=1&dep_name=C

(注:blur: 为开启模糊查询参数1为开启; dep_name: 为参数名称; C: 学院名称包含字符;)

1.6学院-组合

请求方法: GET

请求地址: http://127.0.0.1:8000/api/departments/?slogan=Here is Slogan&master_name=Test-Master&dep_name=Test学院

(注:dep_name: 学院名称; master_name: 为院长名称; slogan: 学院口号; 三个条件可随意组合或单独使用)

二、新增

2.1学院-新增

1) 请求方法: POST

2) 请求地址: <http://127.0.0.1:8000/api/departments/>

3) 请求JSON报文:

4) 调用传入的json串如下 (可新增多条, 之间用, 隔开):

```
{
  "data": [
    {
      "dep_id": "T01",
      "dep_name": "Test学院",
      "master_name": "Test-Master",
      "slogan": "Here is Slogan"
    }
  ]
}
```

5) 新增成功返回报文:

```
{
  "already_exist": {
    "results": [],
    "count": 0
  },
  "create_success": {
    "results": [
      {
        "dep_id": "T02",
        "dep_name": "Java学院",
        "master_name": "Java-Master",
        "slogan": "java"
      }
    ],
    "count": 1
  }
}
```

```

}
6) 新增失败id已存在-返回报文:
{
  "already_exist": {
    "results": [
      {
        "dep_id": "T01",
        "dep_name": "Test学院",
        "master_name": "Test-Master",
        "slogan": "Here is Slogan"
      }
    ],
    "count": 1
  },
  "create_success": {
    "results": [],
    "count": 0
  }
}

```

7) 新增失败json格式错误:

```

{
  "status_code": 400,
  "detail": "请求体参数格式错误。"
}

```

三、更新

3.1学院-更新

1). 请求方法: PUT

2). 请求地址: <http://127.0.0.1:8000/api/departments/T03/>
(注: 1: 为学院ID)

3). 请求JSON报文:

```

{
  "data": [
    {
      "dep_id": "T03",
      "dep_name": "C++/学院",
      "master_name": "C++-Master",
      "slogan": "Here is Slogan"
    }
  ]
}

```

4). 修改成功返回:

```

{
  "dep_id": "T03",
  "dep_name": "C++/学院",
  "master_name": "C++-Master",
  "slogan": "Here is Slogan"
}

```

四、删除

4.1学院-删除单个

请求方法: DELETE

请求地址: <http://127.0.0.1:8000/api/departments/T03/>

(注:10为学院ID)

4.2学院-删除多个

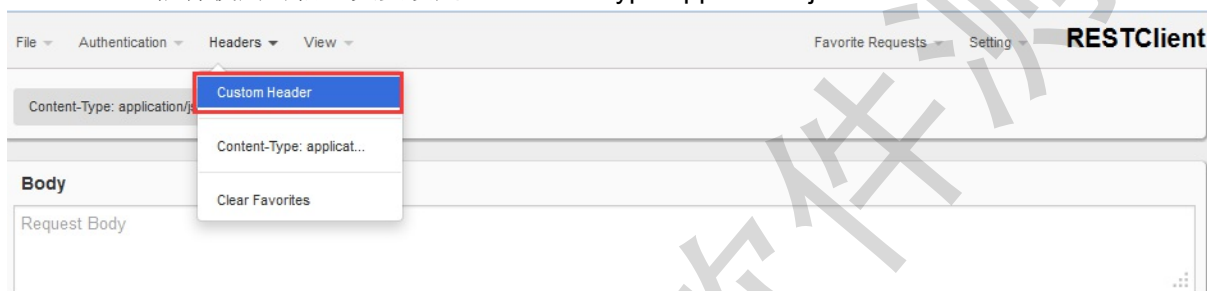
请求方法: DELETE

请求地址: [http://127.0.0.1:8000/api/departments/?\\$dep_id_list=8,9,11](http://127.0.0.1:8000/api/departments/?$dep_id_list=8,9,11)

(注:\$dep_id_list: 为参数名称; 8,9,11: 为学院ID)

注意

RESTClient:插件使用时信息头要设定: Content-Type application/json



思考？

场景

1. 需求：对我们学院接口-查询所有，执行100次，如何去做？
2. 50个用户同时请求如何操作？

目标

- 认识Jmeter并熟悉Jmeter各个元件
- 学会使用Jmeter基于HTTP协议软件录制

Jmeter 介绍

目标

- 了解Jmeter背景
 - 了解Jmeter能做什么？
 - 了解JDK原理
-

1. Jmeter是什么？

1.1 概念

Jmeter: 是Apache公司使用Java平台开发的一款测试工具。

作用:

1. 接口测试
2. 性能测试
3. 压力测试
4. Web自动化测试
5. 数据库测试
6. JAVA程序测试

优点:

1. 开源、免费
2. 支持多协议
3. 小巧
4. 功能强大


Binaries

[apache-jmeter-3.3.tgz md5 sha512 pgp](#)
[apache-jmeter-3.3.zip md5 sha512 pgp](#)

Source

[apache-jmeter-3.3_src.tgz md5 sha512 pgp](#)
[apache-jmeter-3.3_src.zip md5 sha512 pgp](#)

源代码下载

-  **apache-jmeter-3.1.zip** 修改日期: 2017/3/17 17:05 创建日期: 2017/12/27 21:39
WinRAR ZIP 压缩文件 大小: 47.7 MB
-

1.5 缺点

1. 不支持IP欺骗
2. 使用JMeter无法验证JS程序，也无法验证页面UI，所以要须要和Selenium配合来完成Web2.0应用的测试

1.6 总结

1. JMeter概念
2. JMeter作用
3. JMeter优点
4. JMeter缺点

2. 了解：配置JMeter运行环境（JDK、JRE、JVM）

2.1 JDK

JDK概念：java开发工具包，程序员使用。包含JRE和JVM。

下载方式：

1. 官网下载地址：<<http://www.oracle.com/>>
2. 百度搜索“JDK”关键字

2.2 JRE

JRE概念： JAVA程序运行环境，包含JVM和JVM运行时所需要的资源。

下载方式：

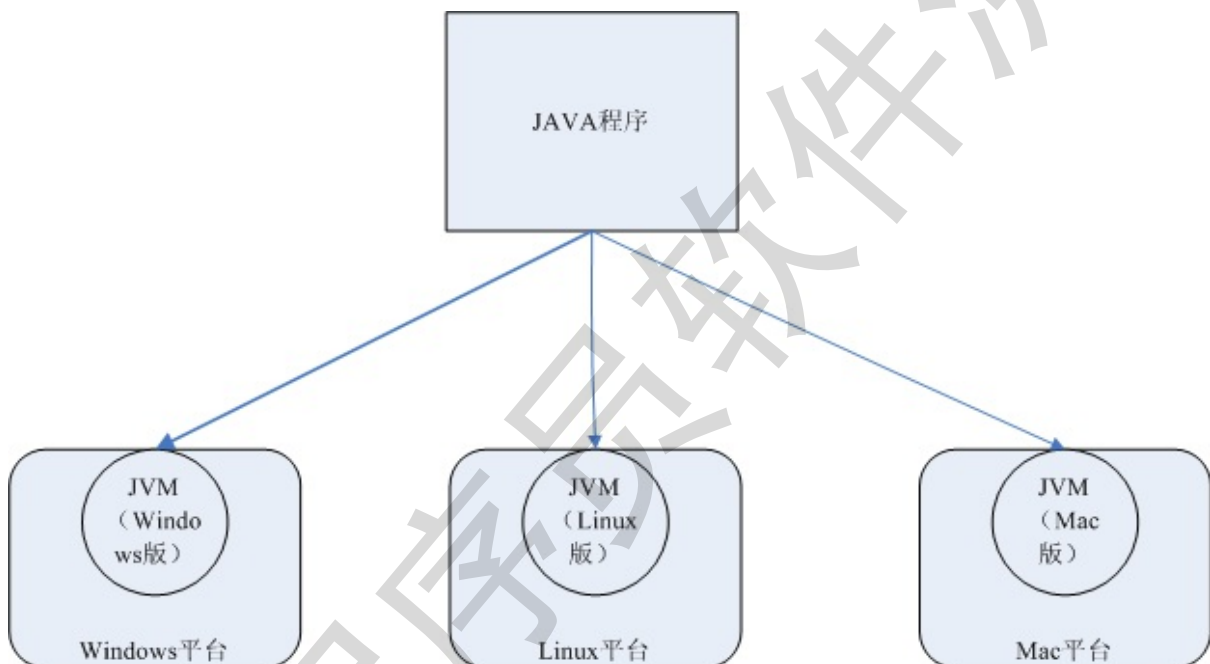
1. 官网下载地址：<<http://www.oracle.com/>>
2. 百度搜索“JRE”关键字

2.3 JVM

JVM概念： Java虚拟机

无需下载，JDK或者JRE包含。

2.4 JAVA跨平台原理【扩展】



总结：对于测试人员，我们要记住，如果使用JMeter，必须要安装JDK或者JRE。

Jmeter工具安装

目标

- 了解Jmeter安装目录结构

2. Jmeter下载与安装

2.1 官网下载地址:

http://jmeter.apache.org/download_jmeter.cgi

下载示意图:



注意:

下载后, 解压文件到任意目录, 避免在一个有空格的路径安装Jmeter, 这将导致远程测试出现问题。

2.2 启动JMeter的两种方式:

- 进入bin目录
 1. 双击 ApacheJMeter.jar文件;
 2. 双击 Jmeter.bat文件;
 - i. 出现Jmeter不是内部或外部命令在环境变量PATH中添加Jmeter路径bin目录, 比如 (E:\测试\Tools\apache-jmeter-3.1\bin)
 - ii. 出现'findstr' 不是内部或外部命令, 在PATH中添加

(%SystemRoot%/system32;%SystemRoot%;

3. 两种打开方式的区别
4. 发送桌面快捷方式

3 Jmeter常用目录文件介绍

3.1 Bin目录

存放可执行文件和配置文件

- Jmeter.bat: windows系统中JMeter的启动文件
- ApacheJMeter.jar Java环境下的JMeter启动文件
- Jmeter.log: 日志文件
- Jmeter.sh: linux系统中JMeter的启动文件
- Jmeter.properties: 系统配置文件
- Jmeter-server.bat: windows分布式测试要用到的服务器配置
- Jmeter-serve: linux分布式测试要用到的服务器配置

3.2 docs目录(了解开源)

docs: 是JMeter的java Doc, 可打开api/index.html页面来查看;

3.3 printable_docs目录

printable_docs的usermanual子目录下的内容是JMeter的用户手册文档, 其中usermanual下component_reference.html是最常用到的核心元件帮助文档。

注意: lib文件夹也是一个常用文件夹, 使用时再讲。

Jmeter 工具功能界面布局

目标

- 了解Jmeter功能界面布局
- 熟悉测试计划面板

1. 主界面布局

JMeter的主界面布局分为标题栏、菜单栏、工具栏、树形标签栏和内容栏

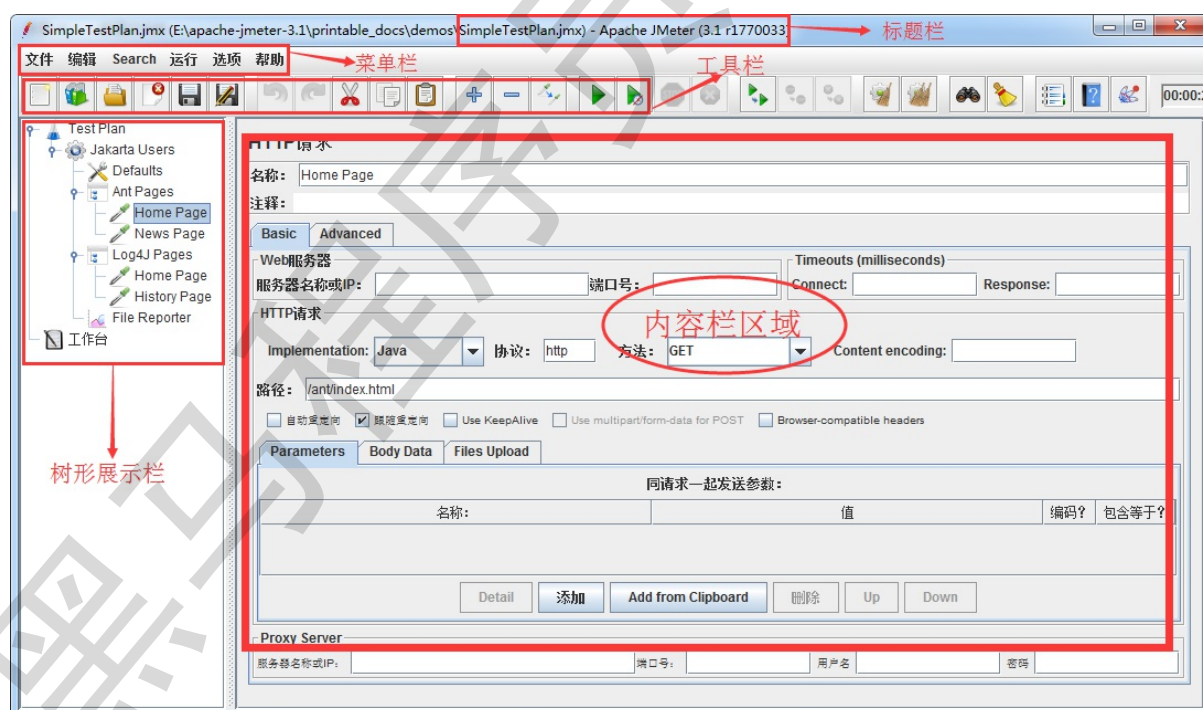
标题栏：主要显示计划信息及JMeter版本。

菜单栏：全部的功能的都包含在菜单栏中。

工具栏：工具栏中的按钮在菜单栏都可以找到，工具栏就相当于菜单栏常用功能的快捷按钮

树形标签栏：树形标签栏通常用来显示测试用例（计划）相关的标签。

内容栏：配合树形标签栏显示，树形标签中点击哪个标签，内容栏中就显示相应的内容和操作。



2. 使用JMeter进行接口测试

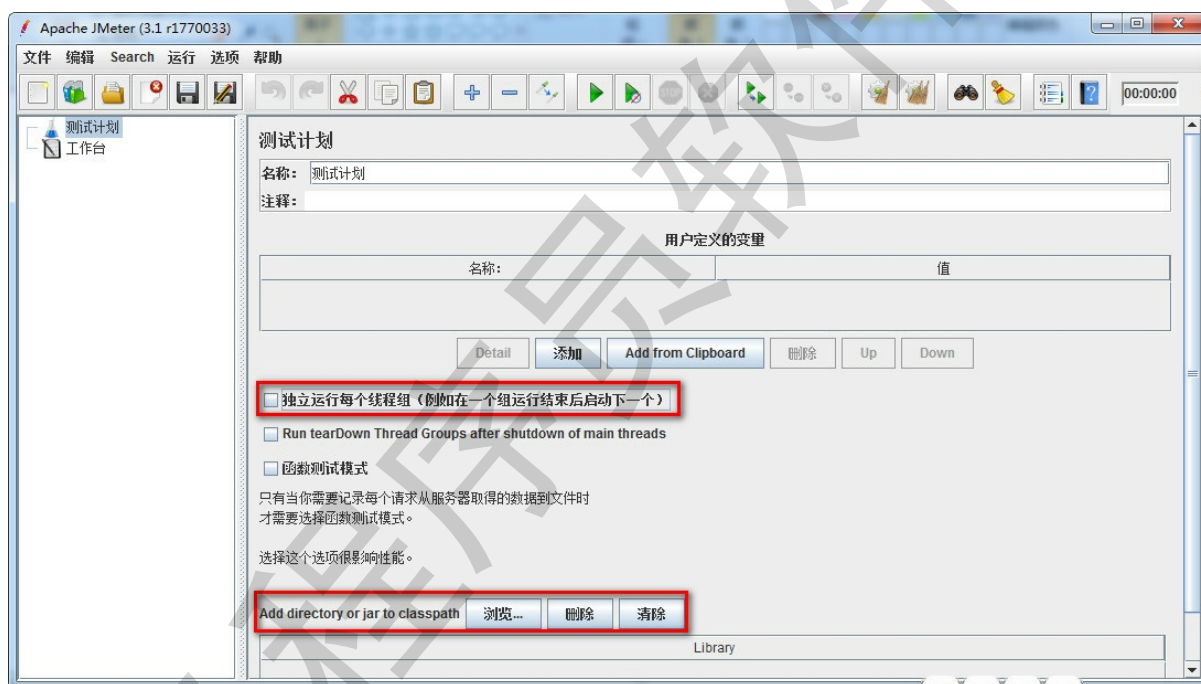
遗留的问题：

1. 需求对我们学院查询执行**100**次，如何去做？
2. **50**个请求同时请求如何操作？

使用JMeter的解决方案

1. 添加【**测试计划**】
2. 基于添加的测试计划添加【**线程组**】，循环次数设置为**100**次
3. 在【取样器】中基于线程组添加**HTTP请求**
4. 在【监听器】基于线程组添加【察看结果树】
5. 在监听器基于线程组添加【聚合报告】

3. Test Plan(测试计划)



作用：

1. 本次测试所需要的【组件】都是基于测试计划添加；
2. 本次测试所有组件的设置与执行都基于测试计划；

组件：完成指定功能代码段的封装；

选项(在这里我们只介绍我们会使用到的选项)

- 独立运行每个线程组：

进程：是每个正在运行的应用程序。

线程：按照进程的指令去执行指定的代码。

线程组（多线程）：多个线程的组合。

线程组（多线程）的执行顺序是并行的。

勾选：让本次测试计划中所有线程组保持从上到下顺序执行

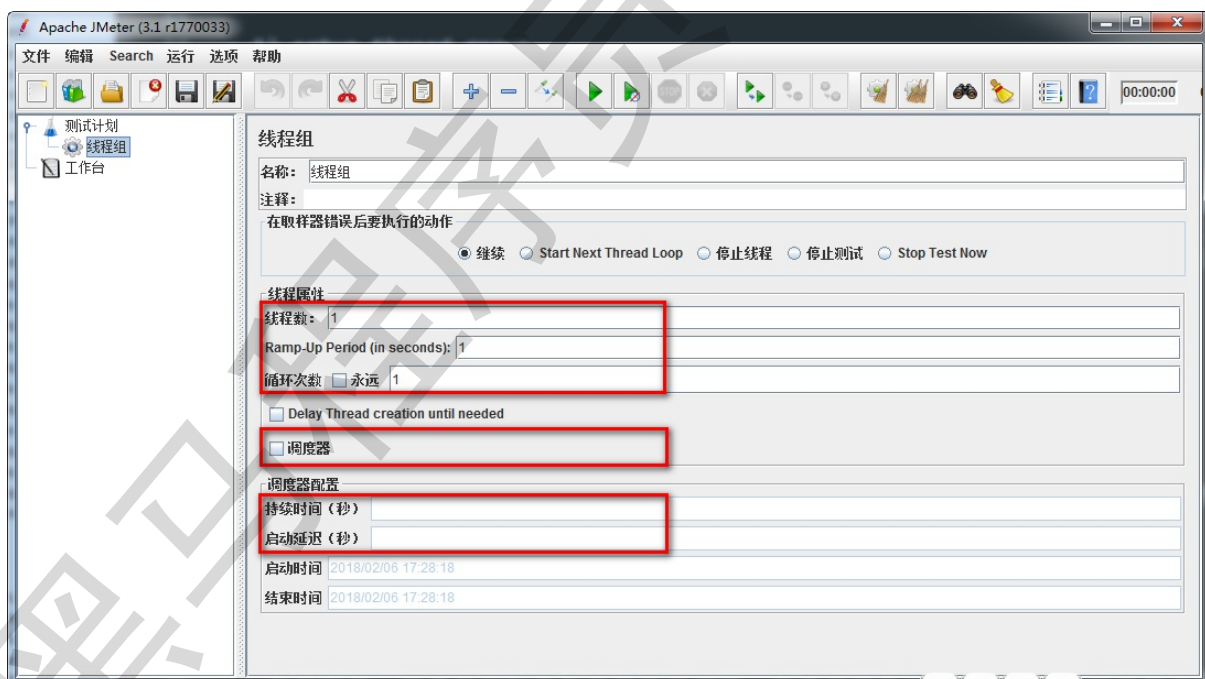
- Add directory or jar to classpath:

加载第三方jar包；比如：测试数据库时使用，加载数据库驱动jar包。

4. Threads(User)线程组 【重点】

- 1) thread group(线程组)
- 2) setup thread group【特殊线程组】
- 3) teardown thread group【特殊线程组】

4.1 thread group(线程组)



作用：

1. 添加测试中使用的大多数组件

线程属性

- 线程数：虚拟用户数
- Ramp-Up Period(in seconds): 启动虚拟全部用户数所需要的时间
- 循环次数：指定次数或勾线永远
- 调度器：勾选后，调度器配置才能使用；

调度器配置

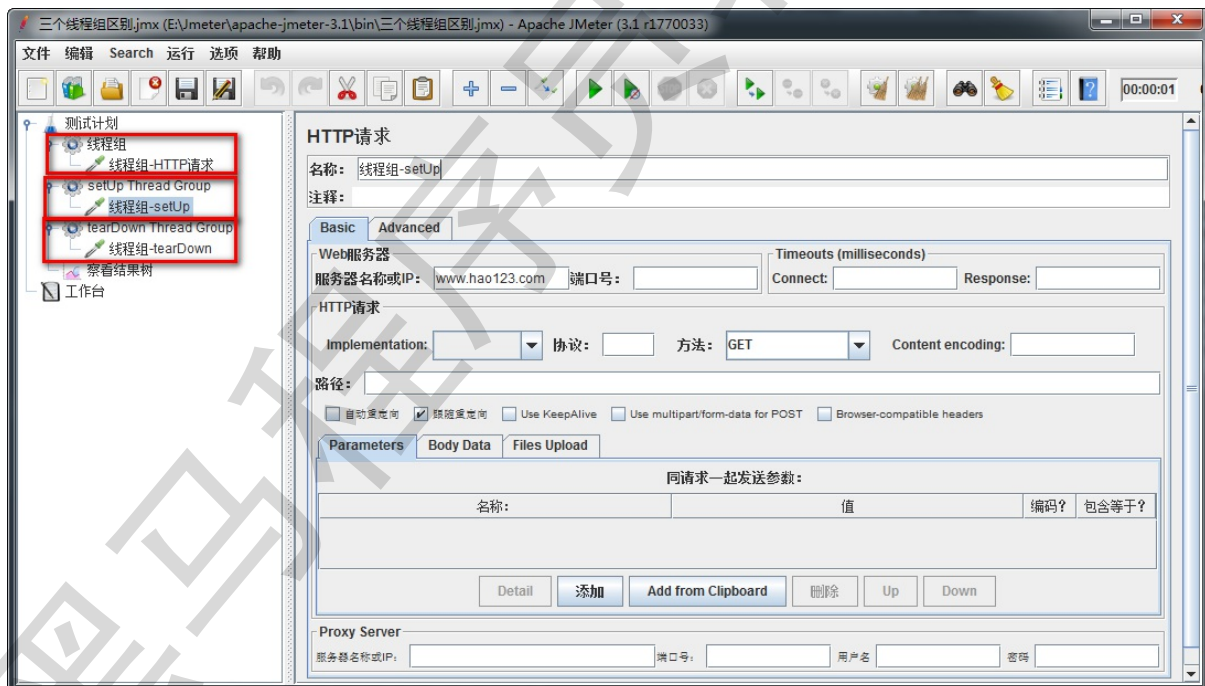
- 持续时间（秒）：设置脚本压测持续时间
- 启动延迟（秒）：启动延迟时间

提示：为了解setup thread、teardown thread两个线程组我们结合案例了解下

4.2 案例

1. 通过thread group、setup thread group、teardown thread group 三个线程组去访问去访问查询所有学院接口

效果图：



需求组件：

1. 基于测试计划添加【线程组】
2. 基于线程组添加【HTTP请求】
3. 基于测试计划添加【setup thread group】线程组
4. 基于setup thread group添加HTTP请求
5. 基于测试计划添加【teardown thread group】线程组

6. 基于teardown thread group添加HTTP请求
7. 基于测试计划添加【察看结果树】

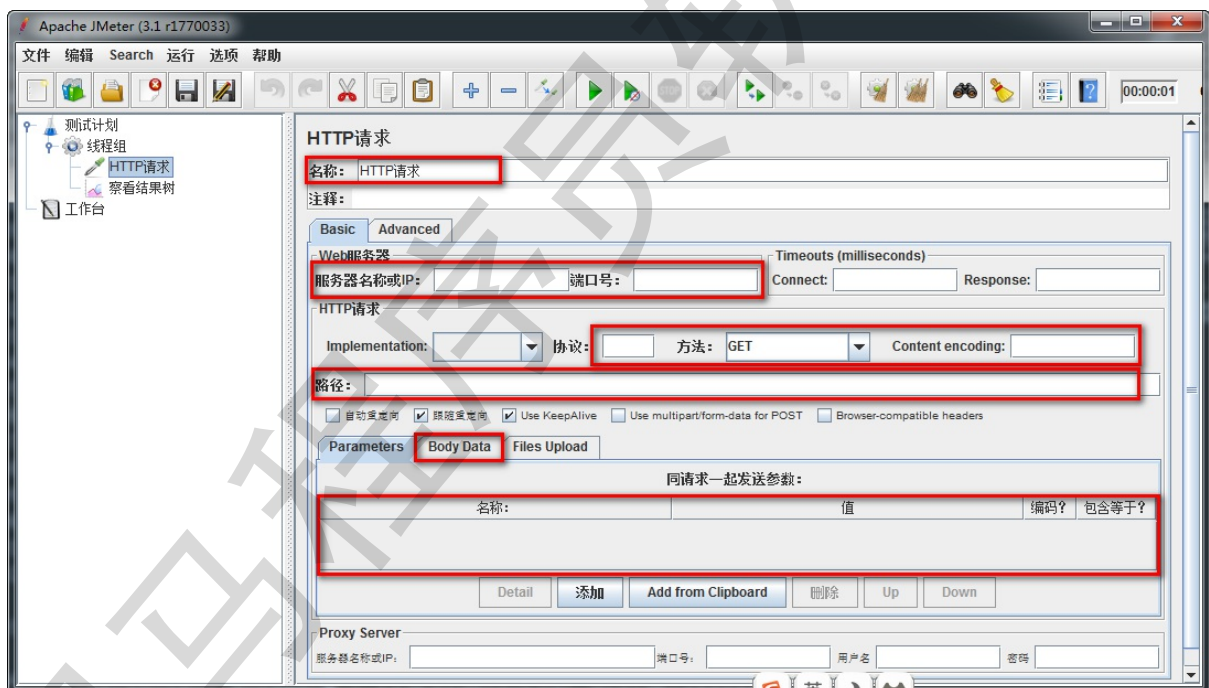
组件设置细节：

1. 测试计划勾选独立运行线程组
2. HTTP请求：服务器地址 `http://127.0.0.1:8000/api/departments/`
3. HTTP请求：修改HTTP请求名称，以做区分

4.3 组件详解

1. HTTP请求
2. 察看结果树

4.3 - HTTP请求



作用：

1. 模拟前端或第三方软件向服务器发送请求；
2. 设置请求时的方法和参数数据；

参数详解：

1. 名称：本属性用于标识一个取样器，建议使用一个有意义的名称。
2. 服务器名称或IP：HTTP请求发送的目标服务器名称或IP地址。
3. 端口号：目标服务器的端口号，默认值为80。
4. 协议：向目标服务器发送HTTP请求时的协议，可以是http或者是https，默认值为http。
5. 方法：发送HTTP请求的方法，可用方法包括GET、POST、PUT、DELETE。
6. Content encoding：内容的编码方式，默认值为iso8859；一般设置【UTF-8】
7. 路径：目标URL路径（不包括服务器地址和端口）
8. 同请求一起发送参数：请求时需要传递参数，如：学院资源list查询

http://127.0.0.1:8000/api/departments/?\$dep_id_list=T01,T02,T03

参数名称：【\$dep_id_list】

参数值：T01,T02,T03

Body Data选项作用：

1. 新增或更新时需要传递JSON报文；如学院新增是的JSON报文填写位置：

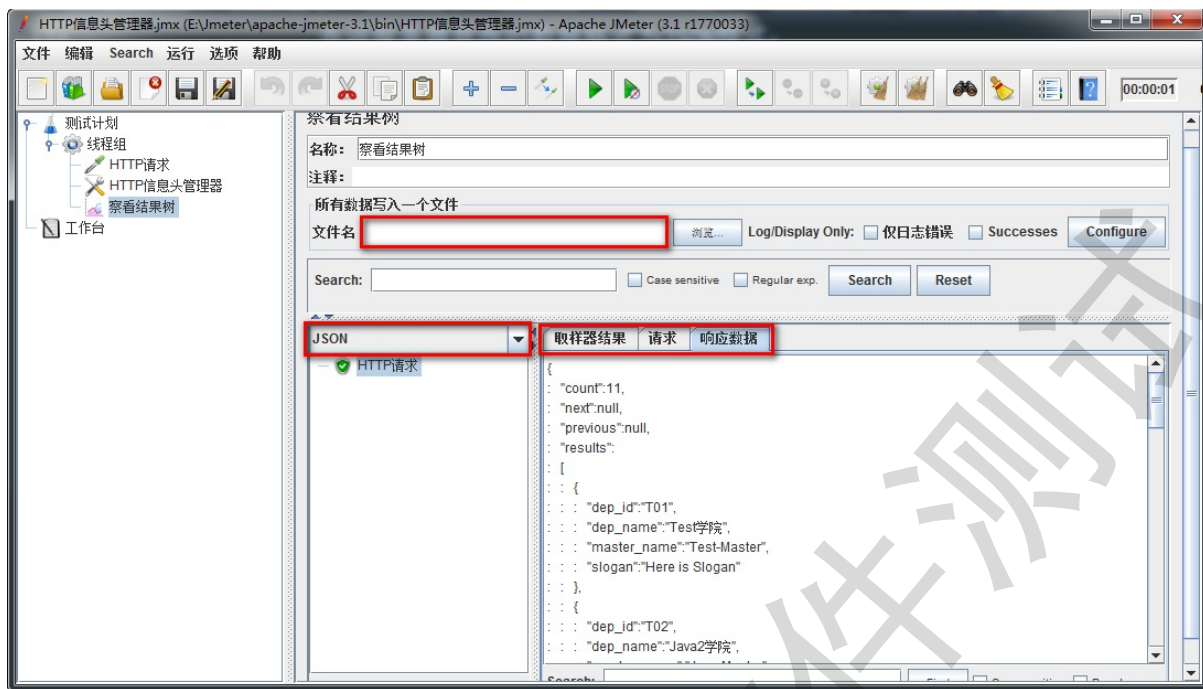
```
{
  "data": [
    {
      "dep_id": "T01",
      "dep_name": "Test学院",
      "master_name": "Test-Master",
      "slogan": "Here is Slogan"
    }
  ]
}
```

2. 【注意】：新增和更新时传入报文也需要设置Content-Type:application/json
告诉服务器我传的数据格式为JSON格式；
设置地点：配置元件-->HTTP信息头管理器（用到的时候我们在讲解）

HTTP请求总结：

1. 接口完整请求地址
2. JSON报文存放地址
3. 设置默认请求数据格式

4.3 - 察看结果树



作用：

1. 查看请求服务器时的请求信息；
2. 查看服务器响应数据；
3. 记录信息到指定文件；

说明：

1. 文件名：存放服务器响应后的状态信息； 如：e:\查询所有response.txt
2. 取样结果：服务器响应的信息头信息；比如：响应代码，响应数据大小
3. 请求：查看向服务器请求时的信息；比如：请求地址、方法、数据等
4. 响应数据：查看服务器响应的数据；比如：获取资源时，返回的JSON数据

察看结果树总结：

1. 查看请求
2. 查看响应
3. 存储请求状态信息

4.4 线程组总结：

setup thread group：一种特殊线程组，测试计划运行之前首先执行，一般做初始化操作
teardown thread group：一种特殊线程组，测试计划运行结束时运行，一般做收尾工作
thread group(线程组)：线程组，我们测试计划中场景创建和实现都是基于此线程组

元件

概念：相同类似功能组件的集合称之为元件

1. 逻辑控制器
2. 配置元件
3. 定时器
4. 前置处理器
5. **Sampler**
6. 后置处理器
7. 断言
8. 监听器

元件结论：

只学重要的、常用的

Jmeter 各元件中需要掌握元件

目标

- 整理出各大元件常用的重点组件

1. 配置元件（**config Element**）

- 1) CSV Data Set Config
- 2) HTTP请求默认值
- 3) HTTP信息头管理器

2. 前置处理器（**Per Processors**）

- 1) 用户参数

3. 定时器（**Timer**）

- 1) Synchronizing Timer

4. 取样器（**sample**）

- 1) HTTP请求
- 2) JDBC Request
- 3) Debug Sampler

5. 后置处理器（**Post Processors**）

- 1) 正则表达式提取器
- 2) XPath Extractor

6. 断言（**Assertions**）

- 1) 响应断言

6. 监听器（Listener）

- 1) 察看结果树
- 2) 聚合报告
- 3) 断言结果

8. 逻辑控制器

- 1) 如果（If）控制器
- 2) ForEach控制器
- 3) 循环控制器

总结：

正常来说，应该开始按照顺序一个组件一个组件的进行讲解。

问题：每个组件都不能独立执行。都需要多个组件进行配合，才能够解决实际问题。

解决方案：按照JMeter主要解决的问题点来讲解组件。

Jmeter 工具核心知识点

目标

- 参数化：数据分离
- 数据库：连接数据库获取数据
- 关联：动态获取数据
- 集合点：掌握如何基于Jmeter并发测试
- 断言：判断自动化脚本执行成功或失败

Jmeter 参数化

目标

- 学习掌握Jmeter中常用参数化方式

1. 为什么要参数化？

1.1 需求新增10条数据

学院-新增

- 1) 请求方法: POST
- 2) 请求地址: `http://127.0.0.1:8000/api/departments/`
- 3) 请求JSON报文:
- 4) 调用传入的json串如下 (可新增多条, 之间用, 隔开):

```
{
  "data": [
    {
      "dep_id": "T01",
      "dep_name": "Test学院",
      "master_name": "Test-Master",
      "slogan": "Here is Slogan"
    }
  ]
}
```

问题:

1. 键所对应的值都是写死的, 只能手动更改
2. 无法解决新增大数量 (1000条) 的问题

2. 什么是参数化?

概念: 根据需求动态获取数据并进行赋值的过程

在Jmeter中参数化常用方式:

CSV Data Set Config

用户参数

用户定义的变量

函数

1. CSV Data Set Config（数据集配置）

概念：一种从外部读取数据功能的组件

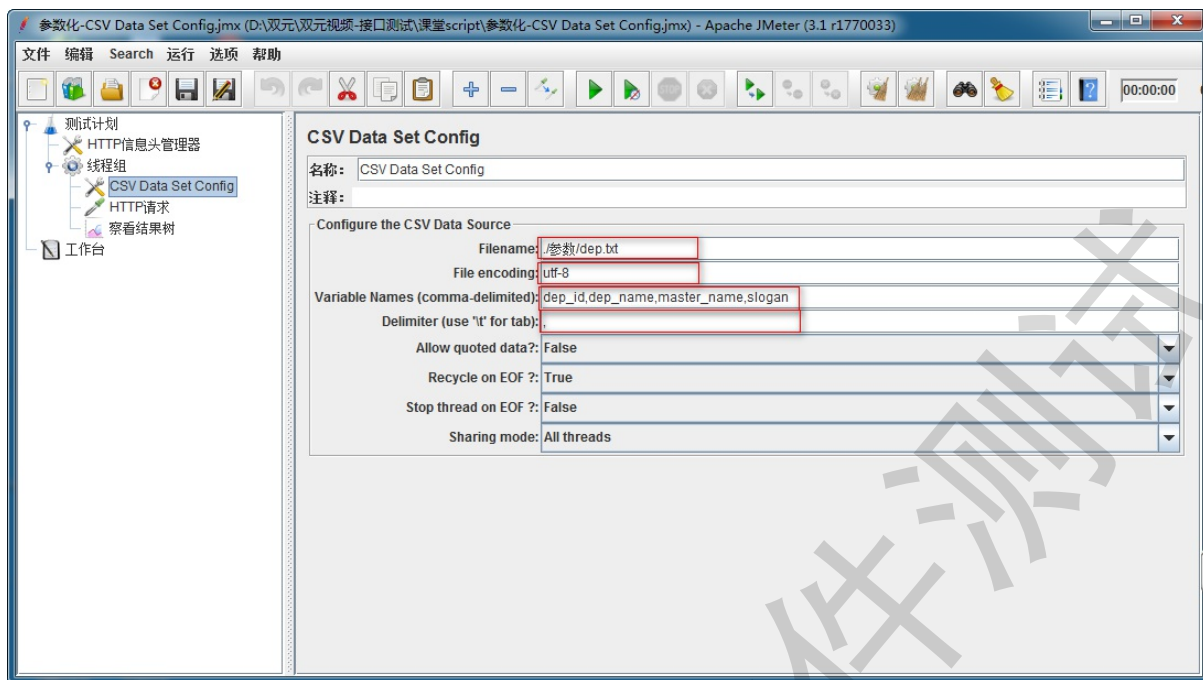
1.1 实施方案分析

1. 基于测试计划->线程组
2. 基于线程组->配置元件->CSV Data Set Config
3. 基于线程组->Sampler->HTTP请求
4. 基于测试计划->HTTP信息头管理器
5. 基于测试计划->监听器->察看结果树

1.2 组件要点分析

1. 线程组：循环次数10
2. CSV Data Set Config 读取变量配置
3. HTTP请求：Body Data填写(JSON报文) 方法(POST)
4. 参数化引用格式：\${参数名} 如：\${dep_id}
5. HTTP信息头管理器：Content-Type:application/json;charset=utf-8

1.3 CSV Data Set Config 参数配置图

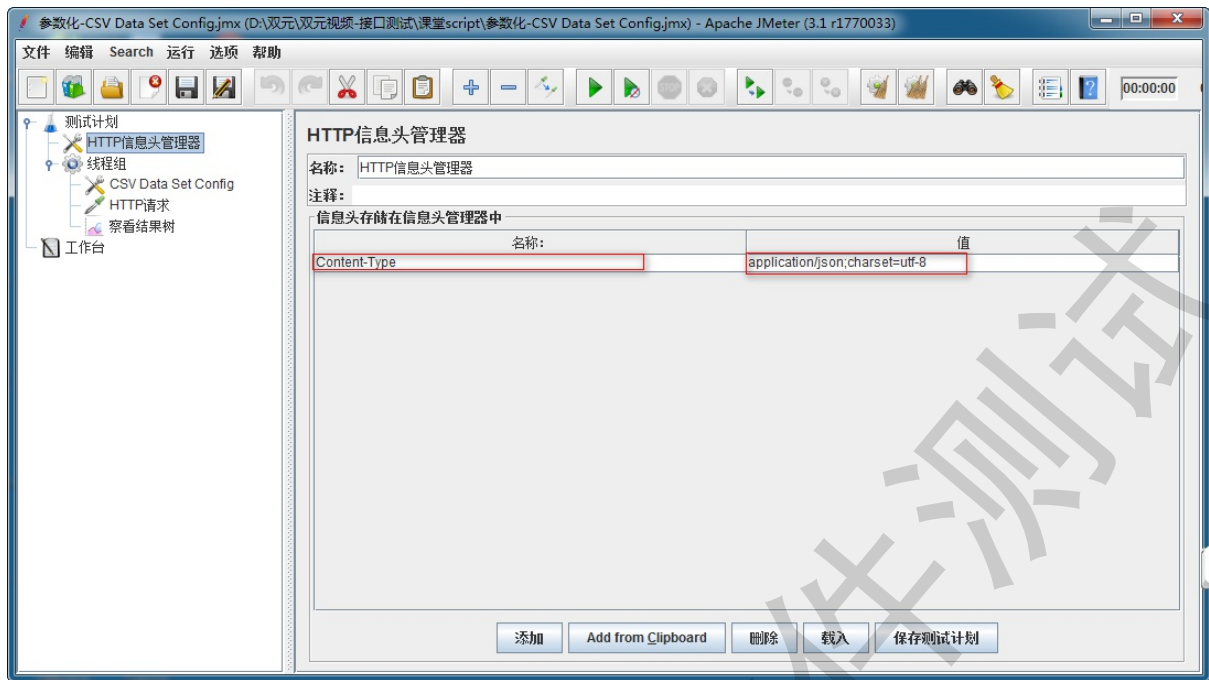


1. **Filename:**文件路径+文件名+后缀名 如: d:/a.txt;
2. **File Encoding:**文件编译字符编码, 一般设置utf-8;
3. **Variable Names:**读取参数后保存的变量名称;
4. **Delimiter:**如文件中使用的是逗号分隔, 则填写逗号; 如使用的是TAB, 则填写\t;

1.4 HTTP信息头管理器 参数配置图

作用

告诉服务器请求的数据格式



1. Content-Type:指定请求信息格式-类型名称
2. application/json:指定请求信息为-JSON格式
3. charset=utf-8:字符编码

1.5 CSV Data Set Config-总结:

1. 参数化概念
2. CSV Data Set Config 配置参数设置
3. 参数化引用格式
4. HTTP请求
5. HTTP信息头管理器作用与设置