# 8.10易宝面试总结

问题焦点：测试用例方法、jmeter、fiddler

主要问题有：介绍下项目、在项目里的职责

测试方法：工作中用到了哪些测试方法

jmeter提问：怎么使用其他线程组的参数

对正则表达式的了解

高并发、高频率是否接触

正则表达式

fiddler：在工作中用fiddler里做了些啥？

就composer讨论了一下（面试官看法：可以作为测试手段，但不建议）

selenium：看简历有selenium，是有什么使用么？

（就获取投标列表爬虫脚本、还有unittest框架稍微说了一下）

Mysql：连接两个表（left join、right join、嵌套子查询）、数据死锁、数据备份

http协议：请求头内容有哪些

为什么要离职？

有什么想问的么？

# 8.11中软/润和面试

问题涉及：测试用例设计方法、fiddler、linux、数据库

Linux：切换用户、查看日志、关键字查找

大数据：对大数据的了解（不涉及、未接触，后期可以了解）

fiddler：命令（bpu、bpafter、bps、start、stop、quit）

测试知识：测试流程

# 1、正则表达式

正则表达式（Regular Expression）是一种文本模式，包括普通字符（例如，a到z之间的字母）和特殊字符（称为“元字符”）。

正则表达式使用单个字符串来描述、匹配一系列匹配某个句法规则的字符串

1. 示例

^[0-9]+abc$

^为匹配输入字符串的开始位置；

[0-9]+匹配多个数字，[0-9]匹配单个数字，+匹配一个或者多个；

abc匹配字母abc并以abc结尾，$为匹配输入字符串的结束位置；

^[a-z0-9\_-]{3-15}$

{3-15}：3-15个字符的长度

1. 总结

a+b ，可以匹配ab，aooob等，+表示前面的字符必须至少出现一次（1次或多次）

runoo\*b，可以匹配runob，runoob，runoooob等，\*号代表前面的字符可以不出现，也可以出现一次或者多次（0次、或1次、或多次）

colou?r 可以匹配color或者colour，？问好代表前面的字符最多只可以出现一次（0次，或1次）

$ 匹配字符串结尾位置

() 标记一个子表达式的开始和结束位置

\* 匹配前面的子表达式零次或多次

+ 匹配前面的子表达式一次或多次

? 匹配前面的子表达式零次或一次

. 匹配除换行符\n之外的任何单字符

[ 标记一个中括号表达式的开始

\ 将下一个字符标记为或特殊字符、或原义字符、或向后引用、或八进制转义符

^ 匹配输入字符串的开始位置，除非在方括号表达式中使用，当该符号在方括号表达式中使用时，表示不接受该方括号表达式中的字符集合。

{ 标记限定符表达式的开始

| 指明两项之间的一个选择

# fiddler工作中的使用

命令：

bpafter：bpafter后边跟一个字符串，表示中断所有包含该字符串的会话，

bpu：跟bpafter类似，区别：bpu是在发起请求时中断，而bpafter是在收到响应后中断。

bps：bps后面跟的是HTTP方法，表示中断所有为该方法的会话。

cls或clear：清除当前的所有会话。

dump：将所有的会话打包成.zip压缩包的形式保存到C盘根目录下。

g或go：放行所有中断下来的会话。

hide：将Fiddler隐藏

show：将fiddler恢复

start：fiddler开始工作

stop：fiddler停止工作

quit：关闭fiddler

1、发现bug

通过抓包，看http请求的响应状态码

例如：密码未加密的问题

2、定位前后端的bug

例如：某按钮没有反应也没有任何提示，或者前端数据显示错误

首先打开F12看前端有没有报错信息。有报错信息但是没有给服务器发起网络请求，那就是前端问题。

如果没有报错信息，也向服务器发起了请求，然后要看前端请求的地址和传参是否正确，如果正确可能是后端问题，如果不正确就是前端接口调用有问题。

3、composer定位前后端的bug（修改请求）

绕过前端向后台发请求（验证后台有没有做对应的限制）

4、mock数据

1）mock是什么？

mock测试就是在测试过程中，对于某些不容易构造或者不容易获取的对象，用一个虚拟的对象来创建以便测试的测试方法。

2）具体情景Mock应用

Mock可以模拟服务器返回多种情况的展示，而不需要实际调用多少次。可能遇到的情景：

* + - 返回结果状态多达20种，但是对方公司要求接口请求次数不能超过5次/小时；
    - 会员只能请求100次；
    - 对方服务器挂掉，需要我们自己mock数据；
    - 网络超时，服务器连接失败

3）Mock有什么用

独立部署的接口和服务，如何保证研发、测试时自己依赖的服务随时都是可用的？

1. mock对测试和开发的好处
2. mock单元测试
3. 前后端分离mock server的使用
4. mock运用

有的项目拥有mock服务器平台，直接把接口文档用excel上传就行了，然后分配服务器端口号就能调用了。

5、mock数据的实际应用场景

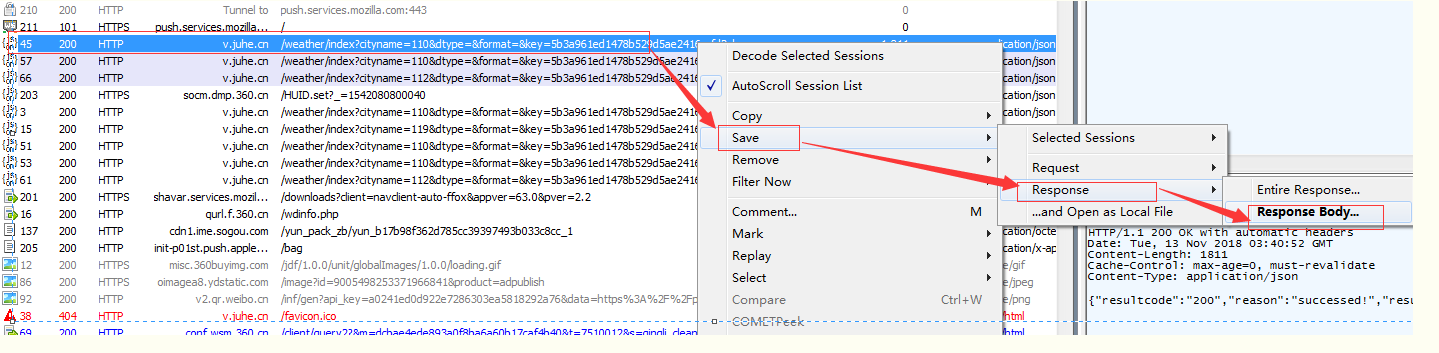
**场景描述1：**

聚合数据的获取全国天气的接口：http://v.juhe.cn/weather/indexcityname=110&dtype=&format=&key=5b3a961ed1478b529d5ae2416ccfd2eb

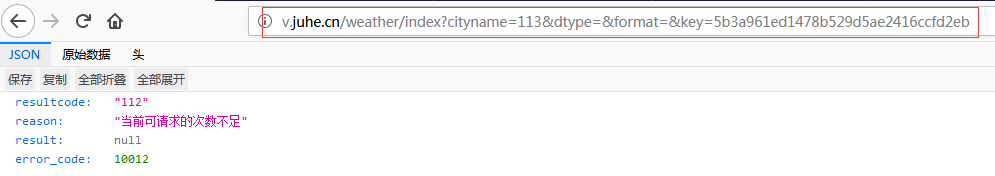
这个接口可以免费调用500次，超过500次，每调用一次收取1分钱，为了免费使用，就要fiddler来mock数据。

1. 首先，免费使用的过程中，用fiddler获取该接口请求和响应，将响应body保存下来

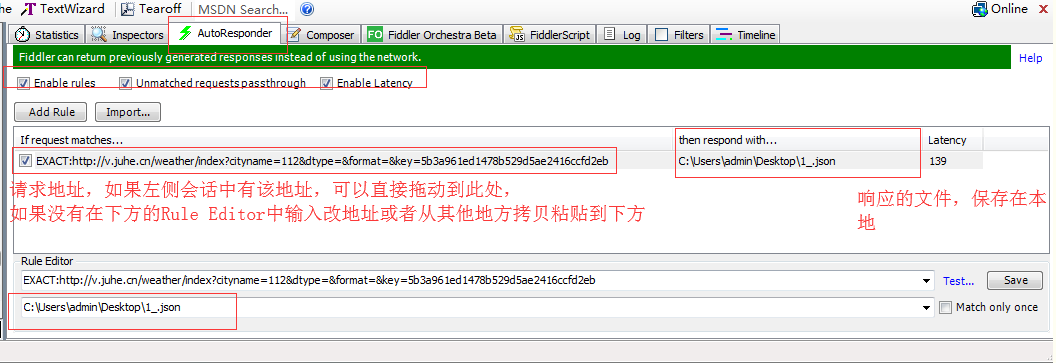
右键点击fiddler左侧的会话框中该接口请求会话-Save-Response-Response Body，将该请求的响应内容保存到桌面（该接口请求的资源是什么类型，保存下来就是什么类型）



1. 超过500次后浏览器请求该接口提示：

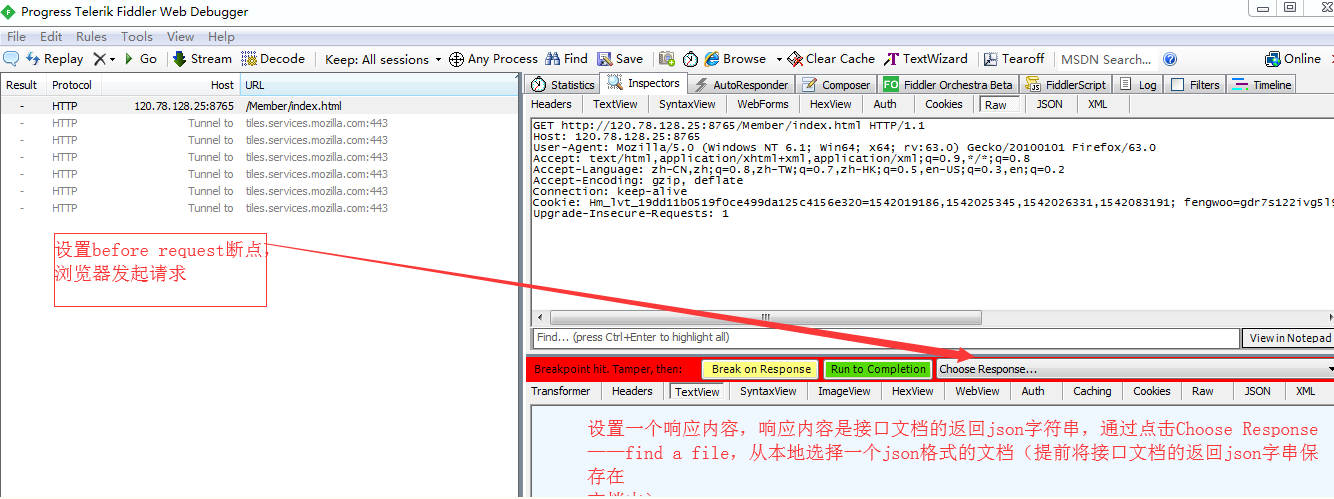


1. 使用fiddler的autoresponder，添加规则，当请求地址和下面的规则中的地址一致时，返回存储在本地（上一步周保存在桌面的文件）的响应内容。



1. 再次浏览器请求该接口，就不去从服务器获取响应，而是按照fiddler设置的规则重定向到本地的文件。这样前端开发或测试可以使用该方法去调用接口，线上请求，返回本地的内容，可以随意修改本地内容，再次请求，可以查看前端页面显示了。

PS：上述第3个步骤，也可以使用before Request设置断点，在截取浏览器的请求后，设置一个响应内容返回（可以是json、css、html、图片、txt等各种形式），如下图



**场景描述2：**

后端接口还未开发好或者不稳定时，前端开发人员就没办法调试代码，此时可以用fiddler的mock功能，只要知道接口返回的json串，请求接口时，重定向返回本地保存的json内容！不依赖于后端接口。

某些模块没有实现，但是又希望对一些已经实现的模块进行测试，提升测试效率，所以我们需要模拟未实现的模块。

比如前端开发人员已经将注册页面写好了，但是后端注册接口还没写完，测试想要测试前端的注册功能，就要用fiddler的autoresponsder功能或者before request断点调试功能模拟注册接口的返回结果。

设置before request  ，修改服务器返回为接口文档中该接口返回的字符串（有10种返回结果，就要测试10种返回结果），就会不经过服务器，直接将该字串内容显示到前端页面了，这种只是测试前端页面跳转、数据展示、提示语显示等是否正常。

**场景描述3：**

Fiddler mock后端接口的返回值

作用：拦截后端接口的返回，并修改

Fiddler 重定向—利用线上的环境来测试你的代码，但又不对线上产生影响—修改js，图片，CSS，HTML

# 3、MYSQL数据备份

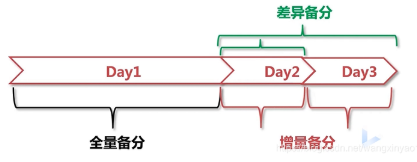
## 1）在之前的工作中是如何对数据库进行备份的？

### a、备份方式：逻辑备份和物理备份



注：我们常用的MariaDB存储引擎使用物理备份，只能对其表结构进行备份，因为其数据是存在内存中。

全量备份和增量备份以及差异备份：



### b、常用的备份工具

|  |  |
| --- | --- |
| 名称 | 特点 |
| mysqldump | 最常用的逻辑备份工具，支持全量备份及条件备份 |
| mysqlpump | 多线程逻辑备份工具，mysqldump的增强版本 |
| xtrabackup | Innodb在线物理备份工具，支持多线程和增量备份 |

2）如何对MySQL进行增量备份和恢复？

3）如何对binlog进行备份？

# 4、数据库锁常见的面试题

事务：就是被绑定在一起作为一个逻辑工作单元的SQL语句分组，如果任何一个语句操作失败那么整个操作就被失败，以后操作就会回滚到操作钱状态，或者是上有个节点。为了确保执行，要么不执行，就可以使用事务。要将有组语句作为事务考虑，就需要通过ACID测试，即原子性，一致性，隔离性和持久性。

锁：在所有的DBMS中，锁是实现事务的关键，锁可以保证事务的完整性和并发性。与现实生活中锁一样，它可以使某些数据的拥有者，在某段时间内不能使用某些数据或数据结构。当然锁还分级别的。

**1.数据库锁有哪些**

行锁 ，表锁 ，页锁， 乐观锁 ，悲观锁

关于锁这篇博客写的不错:https://blog.csdn.net/Michaeles/article/details/86484385

**2.行锁和表锁的区别**

行锁：开销大，加锁慢；会出现死锁；锁定粒度最小，发生锁冲突的概率最低，并发度最高

表锁：开销小，加锁快；不会出现死锁；锁定粒度大，发生锁冲突的概率最高，并发度最低

**3.页锁是什么**

开销和加锁时间界于表锁和行锁之间；会出现死锁；锁定粒度界于表锁和行锁之间，并发度一般。

**4.悲观锁和乐观锁的区别**

1. 悲观锁：  
   就是考虑问题很悲观，每次去拿数据时都认为别人会修改， 所以每次拿数据的时候都会上锁；

实现：sql语句后边加上**for update** 例如：select id，name from biao for update

1. 乐观锁：  
   就是考虑问题很乐观，每次去拿数据的时候都认为别人不会修改， 所以不会上锁;

# 5、数据库死锁

1）什么是死锁？

死锁是指两个或多个事务在同一资源上互相占用，并请求加锁时，而导致的恶性循环现象。当多个事务以不同顺序试图加锁同一资源时，就会产生死锁。

2）锁等待？

mysql数据库中，不同session在更新同行数据时，会出现锁等待。

3）如何优化这类问题，通过数据库哪些表可以监控？

a、尽可能让所有的数据检索都通过索引来完成。

b、合理设计索引。不经常使用的列最好不加锁。

c、尽可能减少基于范围的数据检索过滤条件。

4）通过数据库哪些表可以监控？

重要的三张锁的监控表innodb\_trx，innodb\_locks，innodb\_lock\_waits