

09 | □关联和断言：一动一静，核心都是在取数据

2020-01-03 高楼

性能测试实战30讲

[进入课程 >](#)



讲述：高楼

时长 15:27 大小 14.16M



对每一个性能测试工具来说，关联和断言都是应该具备的基本功能。

但是有很多新手对关联的逻辑并不是十分理解，甚至有人觉得关联和参数化是一样的，因为它们用的都是动态的数据，并且关联过来的数据也可以用到参数化中，但不一样的点是，关联的数据后续脚本中会用到，参数化则不会。断言倒是比较容易理解，就是做判断。

那么到底该怎样理解关联和断言呢？下面我们通过两个例子来思考一下。

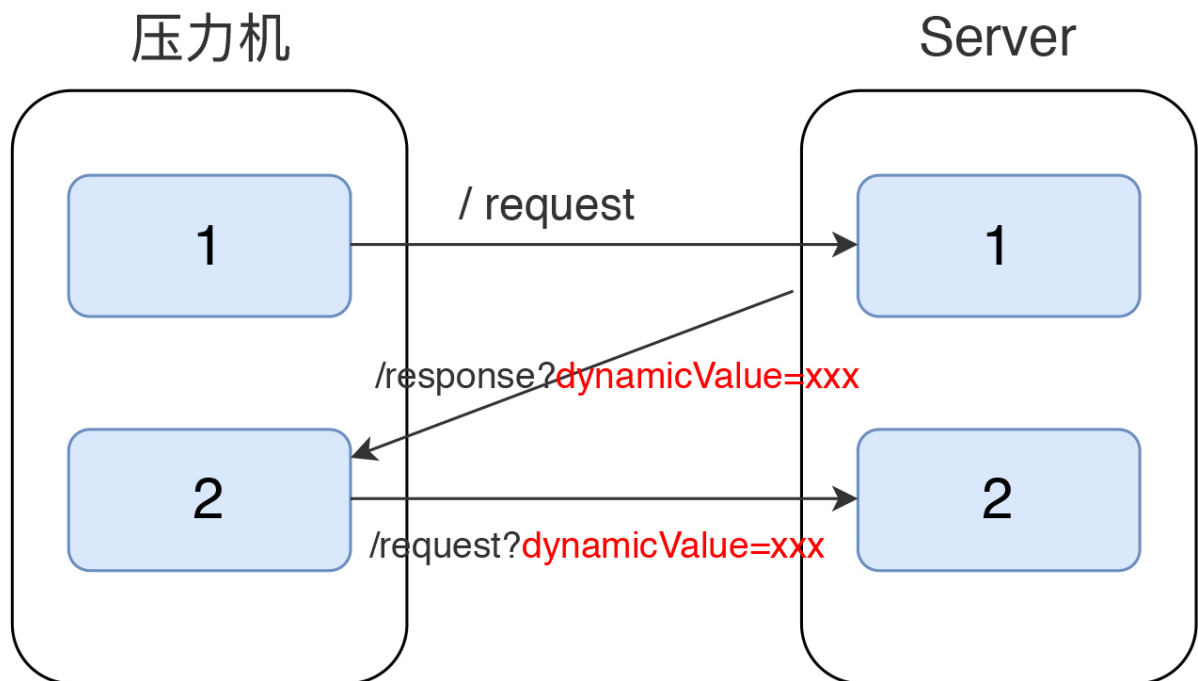
关联

现在做性能测试的，有很多都是单纯的接口级测试，这样一来，关联就用得更少了。因为接口级的测试是一发一收就结束了，不需要将数据保存下来再发送出去。

那么什么样的数据需要关联呢？满足如下条件的数据都是需要关联的：

1. 数据是由服务器端生成的；
2. 数据在每一次请求时都是动态变化的；
3. 数据在后续的请求中需要再发送出去。

示意图如下：



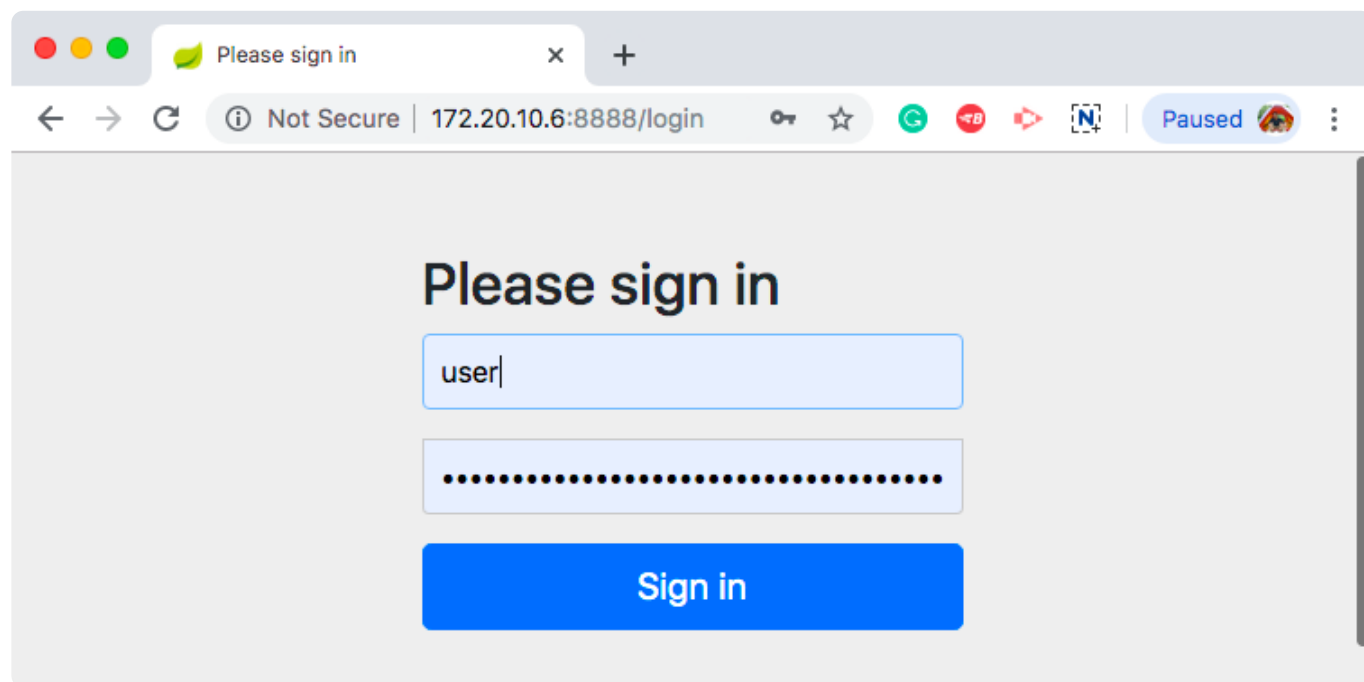
其实我们可以把关联的功能理解为取服务端返回的某个值。在这样的前提之下，我们可以把它用在很多场景之下。

举个例子，我们常见的 Session ID 就是一个典型的需要关联的数据。它需要在交互过程中标识一个客户端身份，这个身份要在后续的交互中一直存在，否则服务端就不认识这个客户端了。

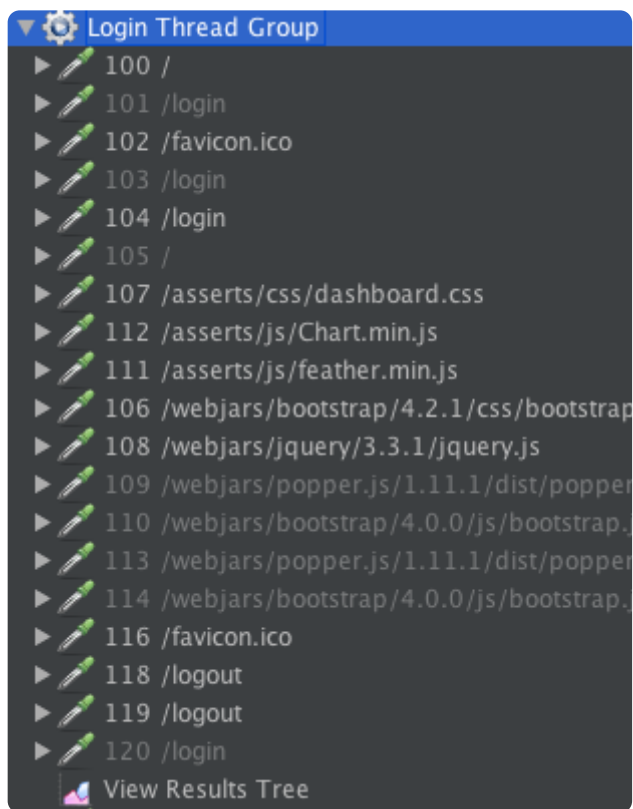
再比如，我们现在用微服务已经非常多了，在 Spring Boot 中有一个 spring-boot-starter-security，默认会提供一个基于 HTTP Basic 认证的安全防护策略。它在登录时会产生一个 CSRF（Cross-Site Request Forgery）值，这个值典型地处于动态变化中。

下面我们来看一下这个值如何处理。

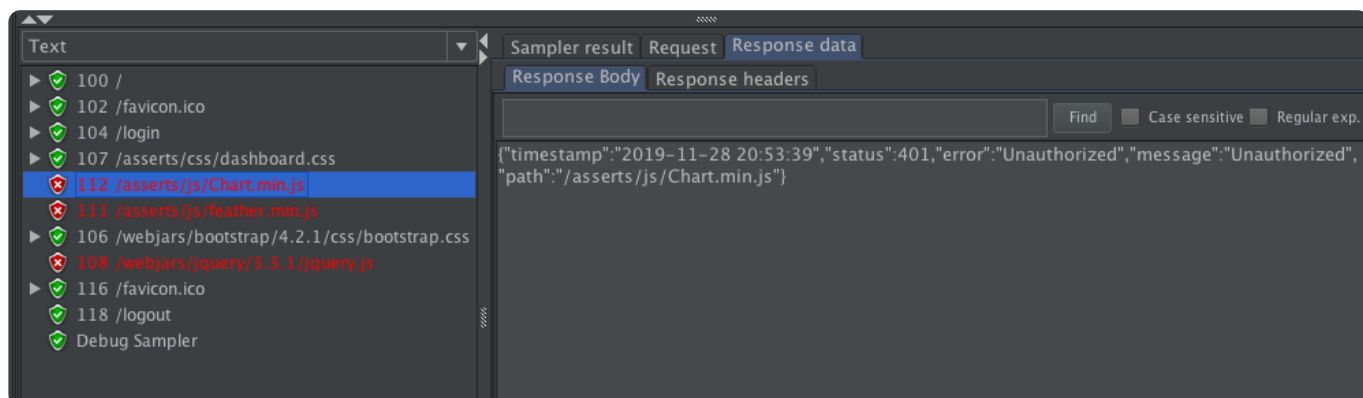
首先，录制登录、退出的脚本。操作如下：



录出的脚本如下所示：

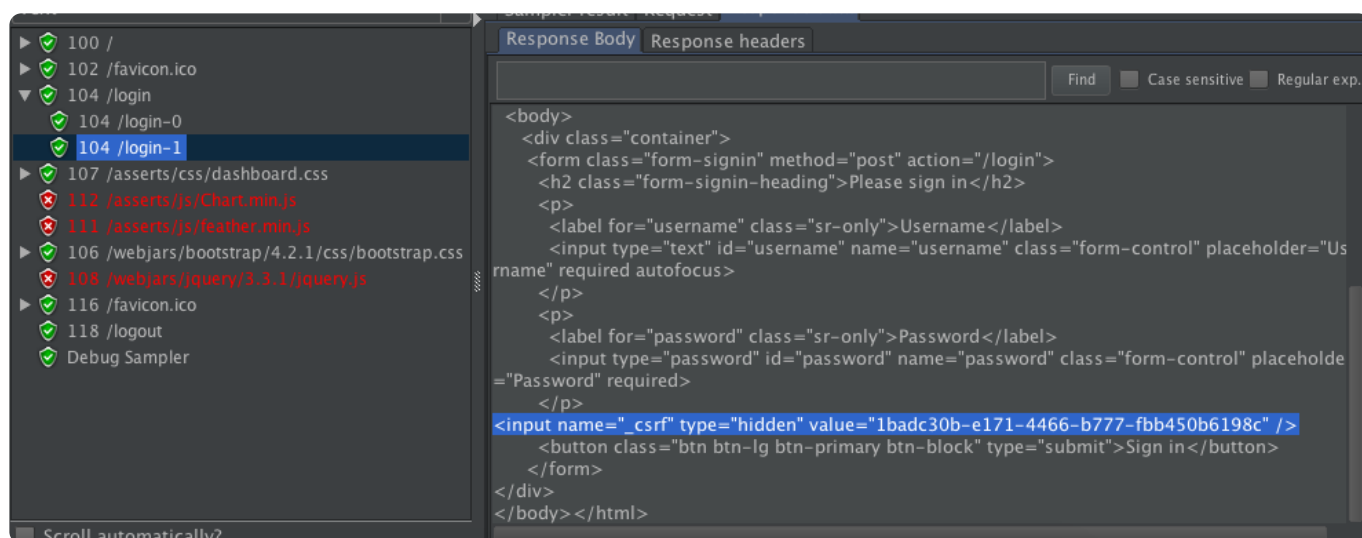


这时直接回放会得到如下结果：



这回你会看到提示了，Unauthorized，没权限。

在回放的脚本中，我们看到了如下的登录返回信息。

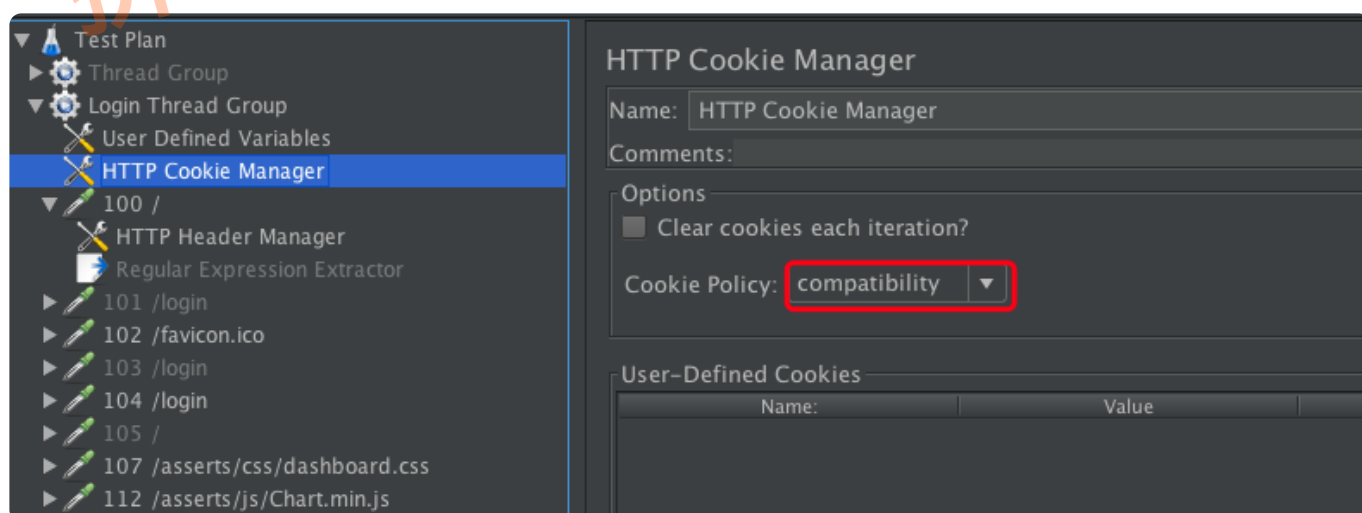


同时，在脚本中，我们可以看到登录时会使用到这个值。

Name:	Value	URL Encode?	Content-Type	Include Equals?
username	user	<input type="checkbox"/>	text/plain	<input checked="" type="checkbox"/>
password	a8d1b953-10fa-4c1a-b83f-55801d11e3...	<input type="checkbox"/>	text/plain	<input checked="" type="checkbox"/>
_csrf	26d990-b89b-4d43-ae5e-32f1388cc683	<input type="checkbox"/>	text/plain	<input checked="" type="checkbox"/>

下面我们就把它关联了。

首先添加 Cookies Manager。JMeter 在处理 CSRF 时，需要添加一个 Cookies manager。如下：



这里的 Cookie Policy 一定要选择 compatibility，以兼容不同的 cookie 策略。

然后取动态值，在返回 CSRF 值的地方加一个正则表达式提取器来做关联。当然还有更多的提取器，我将在后面提及。

Regular Expression Extractor

Name: Regular Expression Extractor

Comments:

Apply to:

☐ Main sample and sub-samples
 ☒ Main sample only
 ☐ Sub-samples only
 ☐ JMeter Variable Name to use

Field to check

☒ Body
 ☐ Body (unescaped)
 ☐ Body as a Document
 ☐ Response Headers
 ☐ Request Headers
 ☐ URL
 ☐ Response Code
 ☐ Response Mes:

Name of created variable: csrfNumber

Regular Expression: <input name="_csrf" type="hidden" value="(.*?)" />

Template (\$i\$ where i is capturing group number, starts at 1): \$1\$

Match No. (0 for Random):

Default Value: ☐ Use empty default value

这里的<input name="_csrf" type="hidden" value="(.*?)" />, 就是要取出这个动态的变化值, 保存到变量 csrfNumber 中去。

然后, 发送动态值出去, 将发送时的 CSRF 值替换成变量。

Send Parameters with the Request.

Name:	Value	URL Encode?	Content-Type	Include Equals?
username	user	<input type="checkbox"/>	text/plain	<input checked="" type="checkbox"/>
password	a8d1b953-10fa-4c1a-b83f-55801d11e3...	<input type="checkbox"/>	text/plain	<input checked="" type="checkbox"/>
_csrf	\${csrfNumber}	<input type="checkbox"/>	text/plain	<input checked="" type="checkbox"/>

最后, 再回放, 就会得到如下结果。

Text

100 /

102 /favicon.ico

104 /login

107 /asserts/css/dashboard.css

112 /asserts/js/Chart.min.js

111 /asserts/js/feather.min.js

106 /webjars/bootstrap/4.2.1/css/bootstrap.css

108 /webjars/jquery/3.3.1/jquery.js

116 /favicon.ico

118 /logout

Debug Sampler

Sampler result Request Response data

Response Body Response headers

body {
font-size: .875rem;
}

.feather {
width: 16px;
height: 16px;
vertical-align: text-bottom;
}

/*
* Sidebar

这样我们就能看到可以正常访问了。

这就是一个典型的关联过程。

上面是用的正则提取器, 在 JMeter 中, 还有其他的提取器, 如下图所示:

```
CSS Selector Extractor
JSON Extractor
JSON JMESPath Extractor
Boundary Extractor
Regular Expression Extractor
```

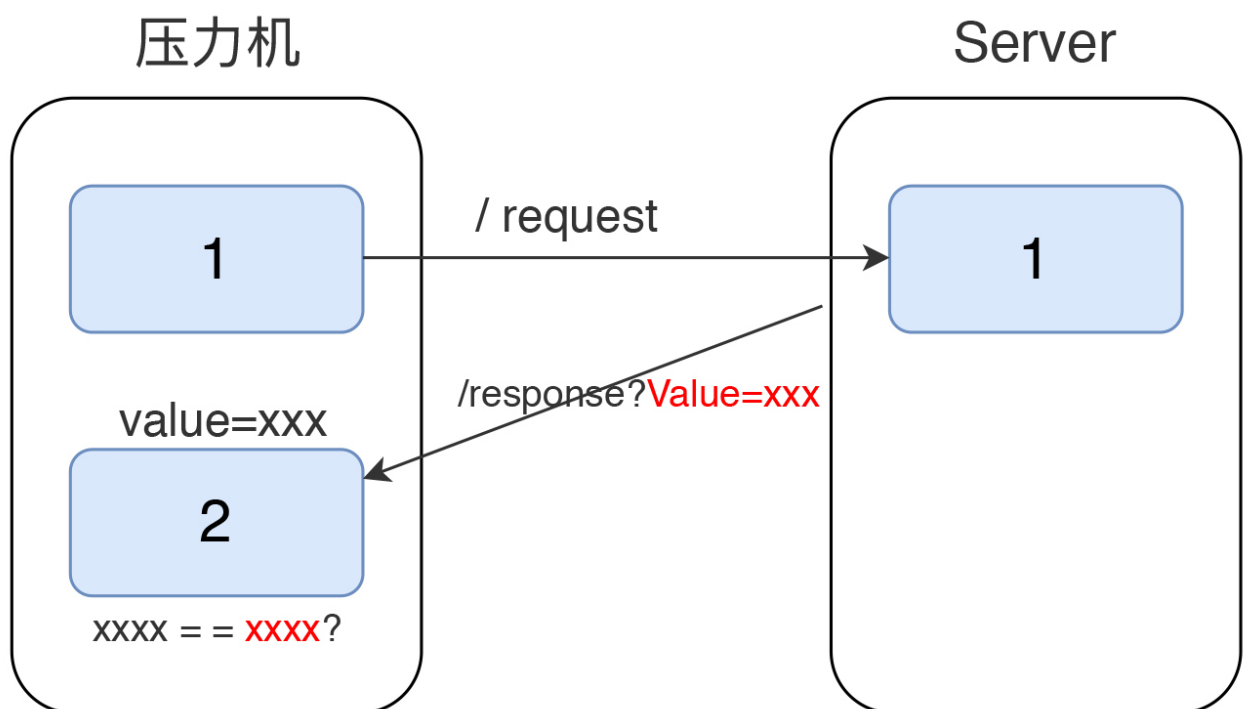
使用什么样的提取器取决于业务的需要，比如说如果你返回的是 JSON 格式，就可以使用上图中的 JSON Extractor。

我们在很多的业务中，都可以看到大量的动态数据。所以做关联一定要有耐心，不然就会找得很混乱。

断言

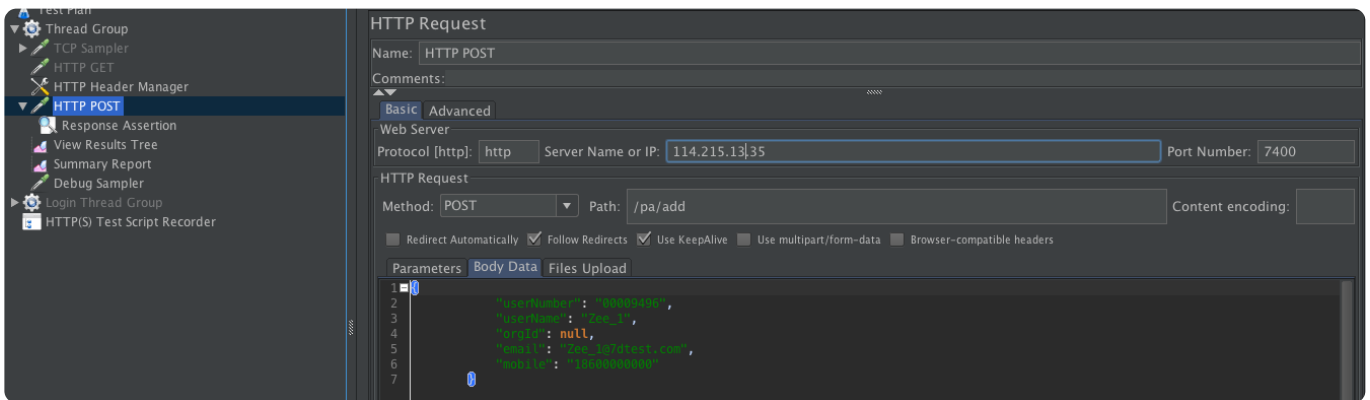
在第 8 篇文章中，我们讲到手工编写脚本，有一个添加断言的动作。断言就是判断服务端的返回是不是正确的。

它的判断逻辑是这样的：

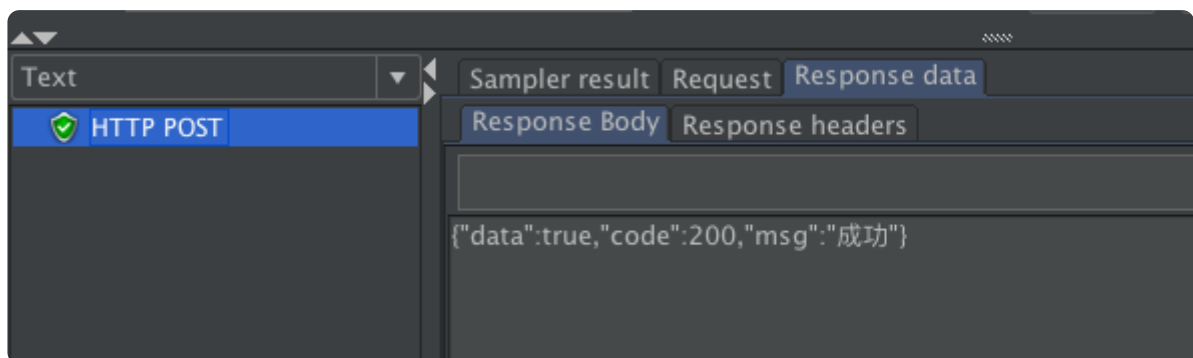


在压力工具中，我们已经知道要比对的值是什么了，接下来就看服务端返回的对不对了。下面我们来详细说一下这个逻辑。

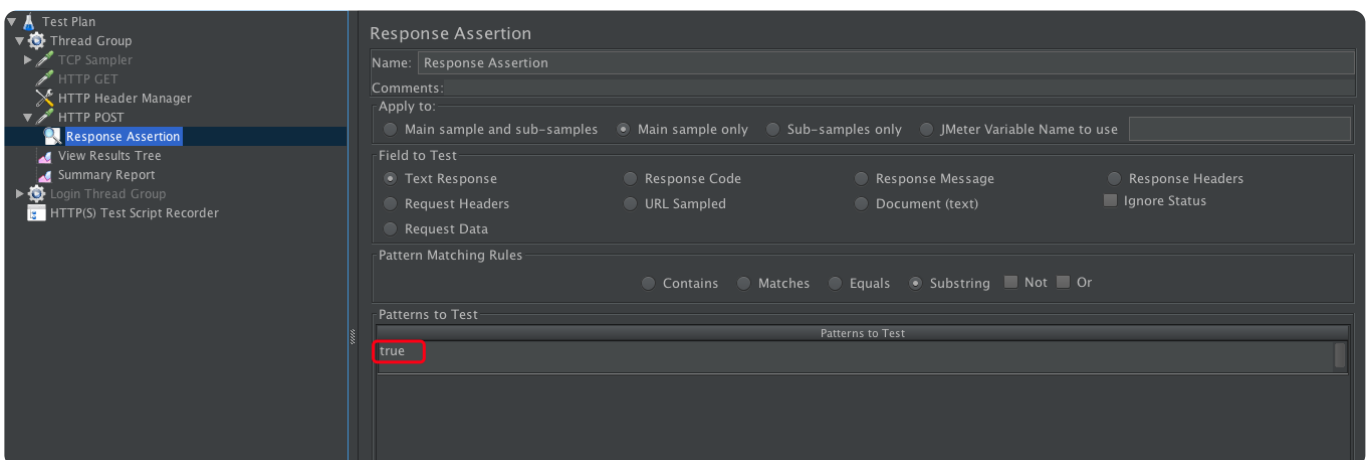
先写一个 POST 接口脚本。



执行下，看到如下结果：




添加断言。



关键点来了，我们知道图片中的这个“true”服务端返回的，可是它到底是从服务端的什么地方产生的呢？

下面我们来看一下服务端的代码。处理我们的 add 请求的，是这样的代码段：

 复制代码


```
1  @PostMapping("/add")
2  public ResultVO<Boolean> add(@RequestBody User user) {
3      Boolean result = paService.add(user);
4      return ResultVO.<Boolean>builder().success(result).build();
5  }
```

我们 post 出去的数据是：

 复制代码

```
1  {
2      "userNumber": "00009496",
3      "userName": "Zee_2",
4      "orgId": null,
5      "email": "Zee_2@7dtest.com",
6      "mobile": "18600000000"
7  }
```

代码中对应的是：

 复制代码

```
1  @Override
2  public String toString() {
3      return "User{" +
4          "id='" + id + '\'' +
5          ", userNumber='" + userNumber + '\'' +
6          ", userName='" + userName + '\'' +
7          ", orgId='" + orgId + '\'' +
8          ", email='" + email + '\'' +
9          ", mobile='" + mobile + '\'' +
10         ", createTime='" + createTime +
11         "'}";
12 }
```

ID 是自增的：


 复制代码

```
1  @Id
```

```
2     @GeneratedValue(strategy = GenerationType.IDENTITY, generator = "select uu'  
3     private String id;
```

然后由 paServer.add 添加到数据库里去：

```
1 Boolean result = paService.add(user);
```

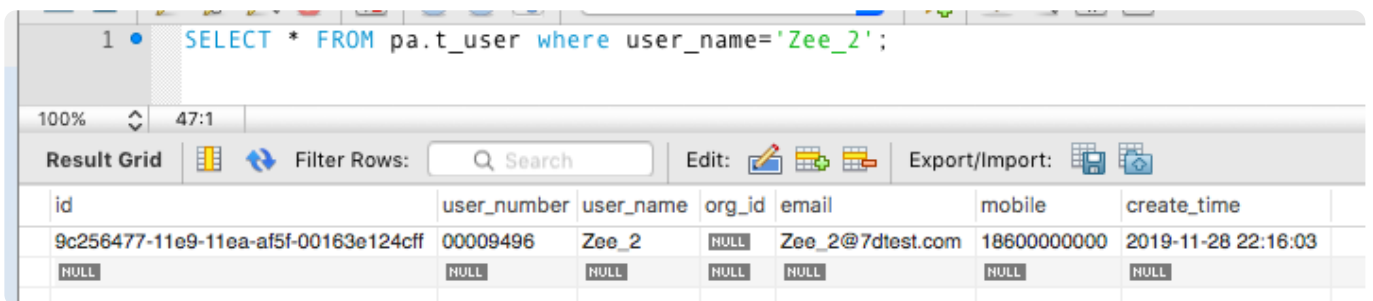
 复制代码

add 的实现是：

```
1     public Boolean add(User user) {  
2         return mapper.insertSelective(user) > 0;  
3     }
```

 复制代码

这就是一个关键了。这里 return 的是 `mapper.insertSelective(user) > 0` 的结果，也就是一个 true，也就是说，这时在数据库中插入了一条数据：




The screenshot shows a database client interface. At the top, a SQL query is entered: `SELECT * FROM pa.t_user where user_name='Zee_2';`. Below the query, the results are displayed in a table with 7 columns: id, user_number, user_name, org_id, email, mobile, and create_time. The first row contains the data for the user 'Zee_2', and the second row shows NULL values for all columns.

id	user_number	user_name	org_id	email	mobile	create_time
9c256477-11e9-11ea-af5f-00163e124cff	00009496	Zee_2	NULL	Zee_2@7dtest.com	18600000000	2019-11-28 22:16:03
NULL	NULL	NULL	NULL	NULL	NULL	NULL

然后，build 返回信息：

```
1     public ResultVO<T> build() {  
2         return new ResultVO<>(this.code, this.msg, this.data);  
3     }
```

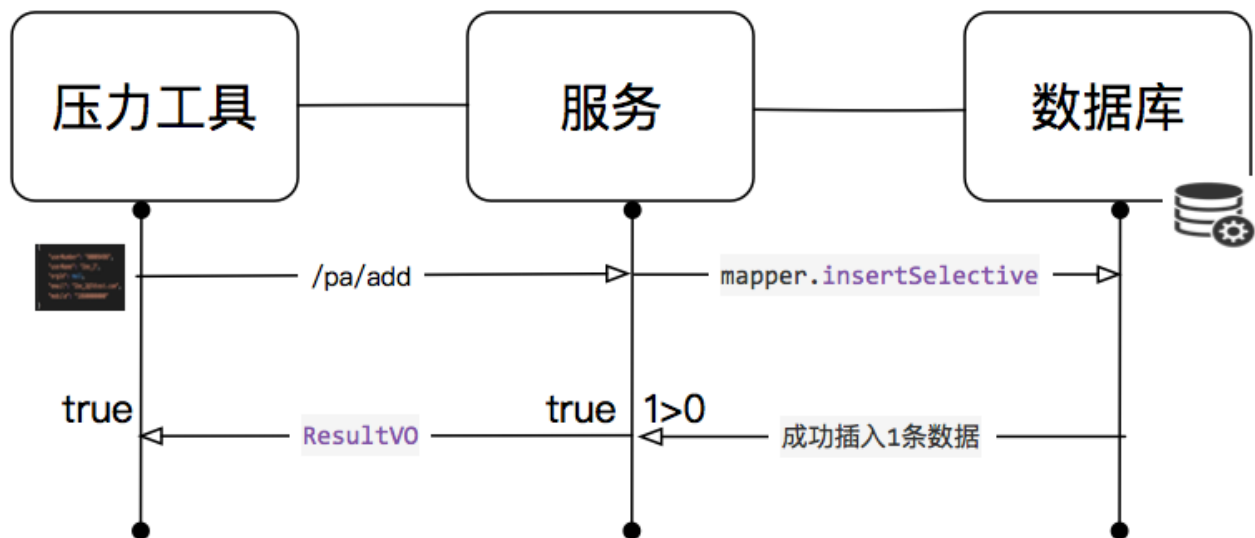
 复制代码

这个时候，我们才看到下面的提示信息：

```
1 {"data":true,"code":200,"msg":" 成功 "}
```

也就是说，在数据库中成功插入 1 条数据之后，把 $1 > 0$ 的判断结果，也就是 `true` 返回给 `result` 这个变量，然后通过 `public ResultVO<Boolean> add(@RequestBody User user)` 中的 `ResultVO` 返回给压力工具。

用图片来说的话，逻辑就是下面这样的：



通过这一系列的解释，我就是想说明一个问题：断言中的 `true` 是从哪来的。

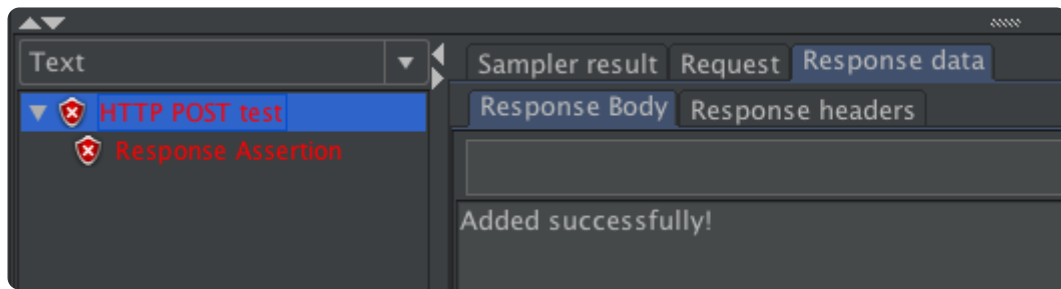
知道了这个问题的答案之后，我们就能理解，为什么这个 `true` 很重要了。因为有了它，就说明我们在数据库成功插入了数据。

断言是根据需要来设计的，而设计断言的前提就是完全理解这个逻辑。

当然我们也可以直接这样来写 Controller：

```
1 public String add(@RequestBody User user) {
2     Boolean result = paService.add(user);
3     return "Added successfully!";
4 }
```

这时就没有 true 了。脚本运行结果如下：



这时断言看似是失败的，因为我们断言判断的是“true”，但服务端没有返回“true”这个字符。而实际上，当我们从数据库中查看时，插入是成功的。

但是这种写法是有问题的，不管数据有没有插入成功，只要在 add 方法执行了，就会提示 “Added successfully!” 。

在实际的工作中，也有开发这样写代码，这样的话，断言似乎都是对的，事务也是成功的，但实际上数据库中可能没有插进去数据。

总结

实际上，关联和断言的前半部分是一样的，都是从服务器返回信息中取出数据。但不同的是，关联取来的数据每次都会不同；而断言取出来的数据基本上都是一样的，除非出了错。

对服务端生成的，并且每次生成都不一样的动态变化的数据，那么将其**取回来之后，在后续的请求中使用**，这种逻辑就是关联。

对服务端返回的，可标识业务成功与否的数据，将其取回来之后，做判断。这种逻辑就是断言。

思考题

最后给你留道思考题吧，你能说一下关联和断言的逻辑是什么吗？它们取数据的特点又是什么呢？

欢迎你在评论区写下你的思考，我会和你一起交流，也欢迎把这篇文章分享给你的朋友或者同事，一起交流进步。

点击查看 

打卡学习，成为真正的性能测试高手



PC端用户扫码参与



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 08 | 案例：手把手教你编写最简单的性能脚本

下一篇 10 | 案例：在JMeter中如何设置参数化数据？

精选留言 (9)

 写留言



@zzw

2020-01-05

思考题：联和断言的逻辑是什么吗？它们取数据的特点又是什么呢？

关联：取出前序调用返回结果中的某些动态值，传递给后续的调用。最常见的是唯一标识客户端的「Session ID」。...

展开 

作者回复：合理。



3



rainbowzhouj

2020-01-06

关联：假设一个业务场景由多个请求构成，那么关联可以理解为前一个请求的输出作为后一个请求的输入。并且可以将关联的值参数化，例如Token, jobId等；

断言：一个请求从执行开始到结束之中，所经历每个步骤都可以“暂停”，那么暂停的这个动作可以理解为断言。通过断言你可以知道代码的运行逻辑，对应的输出是否合理，Debug的好帮手。

展开 ▾

作者回复: 理解的很对。



1



奔跑的栗子

2020-01-03

关联和断言，都是获取特定数据；关联将获取到的数据更新到下一次使用中；断言预知被解除数据的数值，判断执行结果是否正常；

展开 ▾

作者回复: 理解非常正确。



1



餘生

2020-01-13

关联，个人认为比较直观的解释:比如操作一个事件，需要前后分别请求A接口和B接口，B接口请求需要A接口返回参数的某些字段，这就是关联。

断言，没什么好说的，就是判断实际结果是否符合预期结果，并且测试中一定要加，因为有时候无论结果是否正确，response code都是200，不加断言无法直观判断

展开 ▾



小老鼠

2020-01-10

1, 获取csrftoken，然后把它写入hidden和cookies中，这样就可以用接口测试代码进行csrf攻击了。2, 听茹老师说过当断言比较多时可用diff这个工具去比较生成的文件(比如json文件)





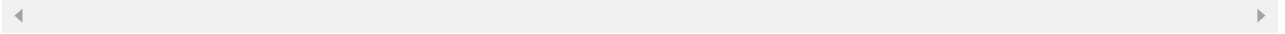
Geek 570c4c

2020-01-07

期待后面的内容

展开 ∨

作者回复: 我也期待。



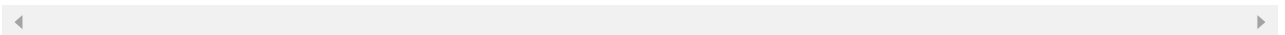
life_牛

2020-01-05

老师，性能压测和参数优化会讲解到吗？

展开 ∨

作者回复: 后面会有。但是是参数众多，只是举常见的场景，不会全覆盖。
主要是要说明分析的思路。



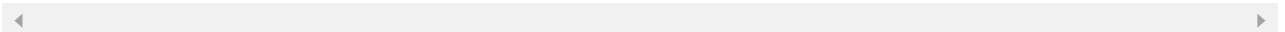
律飛

2020-01-03

关联，有关有联，该数据一定是根据前面的业务获取的，是一个变化动态的，从服务器获得的，否则就可以在脚本中直接写好，变成一个参数了；同时该数据也一定是后面业务得以进行的必须输入，否则就没有存在的意义了；因此，关联数据起了一个承上启下的作用。取数据特点，从服务器返回信息中取数据，这个数据是动态的，且是后续业务必须的输入数据，需要继续使用的。...

展开 ∨

作者回复: 写的非常好。



村夫

2020-01-03

老师，工具的使用还有几篇？

展开 ∨

作者回复: 还有参数化的逻辑和http协议的两篇算是和压力工具有关的。再往后就不讲压力工具了。

就开始讲场景，监控，分析什么的了。
是不是工具部分太简单了？ 😊😊

