如何在测试中模拟请求和响应?

你好，我是猿java。

在日常开发中，除了在服务器端进行单元测试之外，还经常需要做集成测试，为了能更好地做一些边界测试，我们常常需要`mock`一些`HTTP`请求或者响应，今天我们就来聊聊几种常见的方式。

服务器端设置

在开发中，我们会在服务器端编写一些便于测试的代码，然后通过增加一些开关来控制对应的代码分支，测试时，可以在配置中心设置对应的开关走对应的代码逻辑，这样可以更好的模拟真实场景。

如下伪代码，根据一个开关值来`mock`异常阻断：

```
import java.net.MulticastSocket;
boolean mock = globalConfigService.getBoolean(MOCK_KEY);//获取开关值
if(mock){
    throw new RuntimeException(”this is  mock error.”);
}
//  其他业务代码
```

尽管这种方式很灵活，但是也存在一些问题，总结如下：

优点

* 测试逻辑在服务器端设置，因此测试场景更加具体化
* 通过开关灵活的控制服务器的代码逻辑

缺点
--

* 代码里面会增加很测试的逻辑，代码侵入过多，影响代码的可读性和维护性。
* 编写的代码很可能造成事故

所以，对于在服务器端通过编写代码和开关这种方式，尽量避免在实际生产中使用，试想，万一把开关值设置错了，或者测试后忘了关闭，那可能造成线上事故。

使用代理工具
======

`Fiddler`或 `Charles Proxy`

可以使用`Fiddler`或 `Charles Proxy`等代理工具，拦截和修改客户端`HTTP`请求和响应，这些代理工具一般在做 App测试等时候使用的比较频繁。

Nginx/Apache

可以在`Nginx/Apache`的配置文件中添加规则，达到`mock`的目的，但是这种方式需要有权限去修改`Nginx/Apache`的配置文件

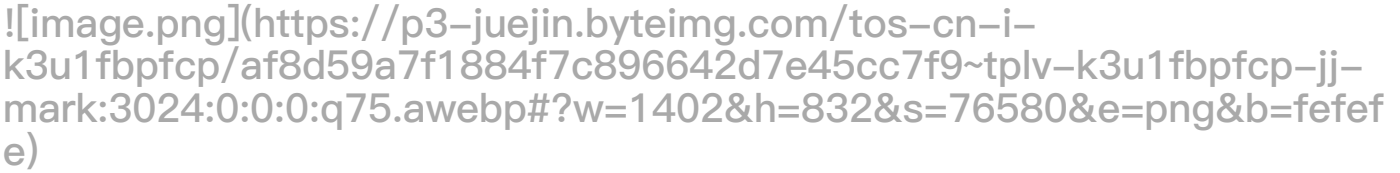
Postman

如果做 web测试，可以使用 Postman，在这个工具中设置规则，因此，Postman是一个很不错的测试工具。

使用浏览器插件
=======

如果是测试一些 sass平台或者网页版项目，使用浏览器插件也是个不错的选择，这里推荐一款使用比较多的`chrome插件

`: [modeHeader](http://cxyroad.com/ ”https://modheader.com/modheader”)，[modResponse](http://cxyroad.com/ ”https://modheader.com/modresponse”)，下图为`chrome`安装插件的一个截图:



下面分别讲解下日常开发中常用的功能。

ModHeader

`ModHeader`主要用于修改 HTTP请求和响应头，这里列举它的几个常用功能:

修改请求头和响应头

测试中，我们通常需要修改一些请求头和修改相应结果来来达到某些特定的测试目的，比如，修改`User-Agent`、`Authorization`等头部来模拟不同的客户端或授权信息。

模拟错误状态

web开发中，经常会通过不同的状态码进行不同的业务逻辑和业务交互，因此，我们常常会修改HTTP的响应状态码来进行缱绻交互的测试，

可以通过修改响应状态码（如404、500等）来模拟服务器错误，测试应用程序的错误处理逻辑。

模拟不同的用户代理

为了测试不同浏览器的兼容性，我们可以通过修改`User-Agent`头，这样就可以在测试网站的跨浏览器兼容性和响应式设计。

跨域资源共享 (CORS)

修改CORS头：可以添加或修改CORS相关的头部（如Access-Control-Allow-Origin），解决开发过程中遇到的跨域问题，进行跨域请求的调试。

请求重定向

有时候我们需要模拟请求重定向的场景，因此，可以通过修改头部，或者配合其他工具，可以实现请求重定向，测试重定向逻辑。

ModResponse

`ModResponse`专门用于修改 HTTP响应，大部分功能是针对`ModHeader`的增强，这里也列举它的几个常用功能：

修改 HTTP响应内容

这个功能和`ModHeader`一样，只是更具体化。

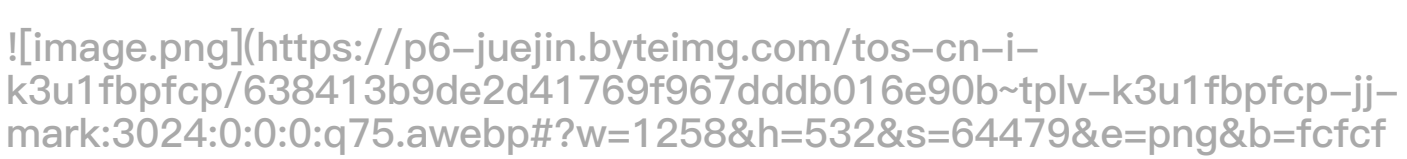
模拟错误状态

这个功能和`ModHeader`一样。

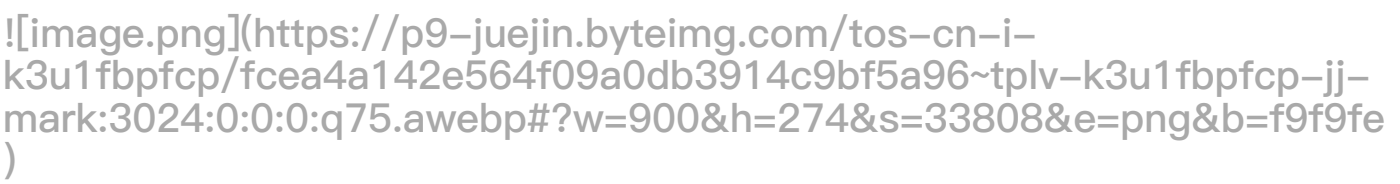
跨域资源共享 (CORS)

这个功能和`ModHeader`一样。

如下图，设置规则：对于`http://127.0.0.1:8080/test`请求，返回错误`{"error":"this is ModResonse error."}`



c)



因此，实际开发中，我们可以根据具体的业务需求设置相应的规则，规则可以设置多个。

总结
==

本文通过从客户端，代理，服务器 3个角度来分析如何`mock`HTTP请求和响应，一般代理的方式使用比较多，因为这种方式对于前后端的代码都没有侵入。对于某些特殊情况需要在服务端设置，也一定要谨慎再谨慎！

交流学习
====

最后，把我的座右铭送给你：`投资自己才是最大的财富`。 如果你觉得本文章对你有帮助，点赞，收藏不迷路，猿java，持续为你输出更多的硬核文章和面试经。

原文链接: https://juejin.cn/post/7369892677641388082