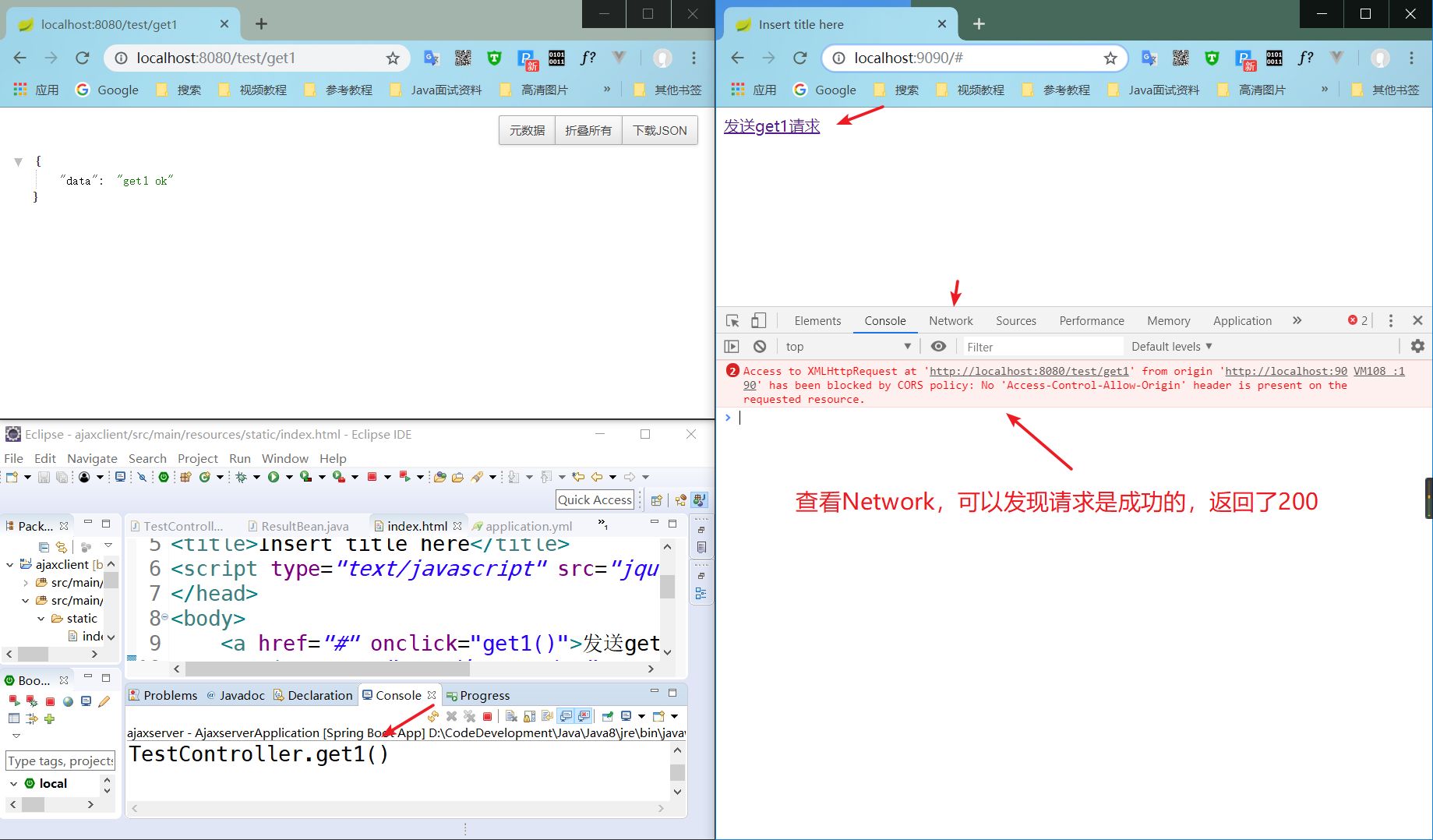
**0、什么是跨域问题**

前端调用的后端接口不属于同一个域（域名或端口不同），就会产生跨域问题，也就是说你的应用访问了该应用域名或端口之外的域名或端口。



**1、为什么会发生跨域问题**

要同时满足三个条件才会产生跨域问题，这也就是为什么会产生跨域的原因。

* 1、浏览器限制，而不是服务端限制，可以查看Network，请求能够正确响应，response返回的值也是正确的
* 2、请求地址的域名或端口和当前访问的域名或端口不一样
* 3、发送的是XHR（XMLHttpRequest）请求，可以使用 a 标签（模拟xhr请求）和 img 标签（模拟json请求）做对比（控制台只报了一个跨域异常）

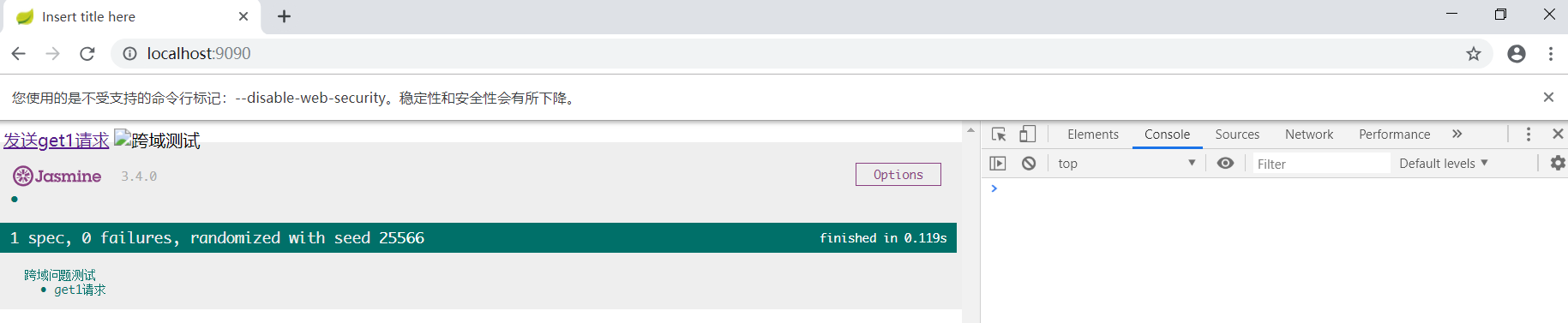
关于 XMLHTTPRequest 可以参看这篇文章 [《你真的会使用XMLHttpRequest吗？》](https://link.zhihu.com/?target=https%3A//segmentfault.com/a/1190000004322487)

**2、解决跨域问题的三种思路**

* **1、客户端浏览器解除跨域限制**（理论上可以但是不现实）
* **2、发送JSONP请求替代XHR请求**（并不能适用所有的请求方式，不推荐）
* **3、修改服务器端**（包括HTTP服务器和应用服务器）（**推荐**）

**2.1 客户端浏览器解除跨域限制**

浏览器默认都是开启跨域安全检查的，我们可以使用命令行启动浏览器，加上禁止安全检查的参数，以谷歌浏览器为例，chrome.exe --disable-web-security --user-data-dir=E:/temp --user-data-dir 为浏览器缓存临时目录，浏览器这时会提示安全问题。



**2.1.1 浏览器如何判断一个请求是不是跨域请求？**

**浏览器会根据同源策略来判断一个请求是不是跨域请求。**

**非跨域请求**，在请求头中会只包含请求的主机名。

**跨域请求**，在请求头中会既包含要请求的主机名还包括当前的源主机名，如果这两者不一致，那就是跨域请求了。

**2.1.2 浏览器对请求的分类**

在HTTP1.1 协议中的，请求方法分为GET、POST、PUT、DELETE、HEAD、TRACE、OPTIONS、CONNECT 八种。浏览器根据这些请求方法和请求类型将请求划分为简单请求和非简单请求。

**简单请求**：浏览器先发送（执行）请求然后再判断是否跨域。

请求方法为 GET、POST、HEAD，请求头header中无自定义的请求头信息，请求类型Content-Type 为 text/plain、multipart/form-data、application/x-www-form-urlencoded 的请求都是简单请求。

**非简单请求**：浏览器先发送预检命令（OPTIONS方法），检查通过后才发送真正的数据请求。

{% asset\_img 非简单请求的预检命令.png 非简单请求的预检命令%}

预检命令会发送自定义头为Access-Control-Request-Headers: content-type的请求到服务器，根据响应头的中的 “Access-Control-Allow-Headers”: “Content-Type” 判断服务器是否允许跨域访问。预检命令是可以缓存，服务器端设置 “Access-Control-Max-Age”: “3600”，这样后面发送同样的跨域请求就不需要先发送预检命令了。

请求方法为 PUT、DELETE 的 AJAX 请求、发送 JSON 格式的 AJAX 请求、带自定义头的 AJAX 请求都是非简单请求。

**2.2 发送JSONP请求替代XHR请求**

**2.2.1 JSONP 是什么**

JSONP（JSON with Padding）是JSON的一种补充使用方式，不是官方协议，而是利用 Script 标签请求资源可以跨域的特点，来解决跨域问题的，是一种变通的解决方案。

2.2.2 使用 JSONP，服务器后台需要改动吗？

答案是**需要**，这里以Spring Boot为例，在 Spring Boot 1.5 大版本中，添加一个切面来支持JSONP请求，注意在 Spring Boot 2.x 大版本中已经废弃了 AbstractJsonpResponseBodyAdvice 类，不推荐这种方式解决跨域问题。

AJAX代码如下：

$.ajax({

url: baseUrl + "/get1",

dataType: "jsonp", // 关键字段

jsonp: "callback", // 前后端默认的约定

cache: true, // 表示请求结果可以被缓存，url中不会有下划线参数了

success: function(json) {

result = json;

}

});

服务端代码：

@ControllerAdvice

public class JsonpAdvice extends AbstractJsonpResponseBodyAdvice {

public JsonpAdvice() {

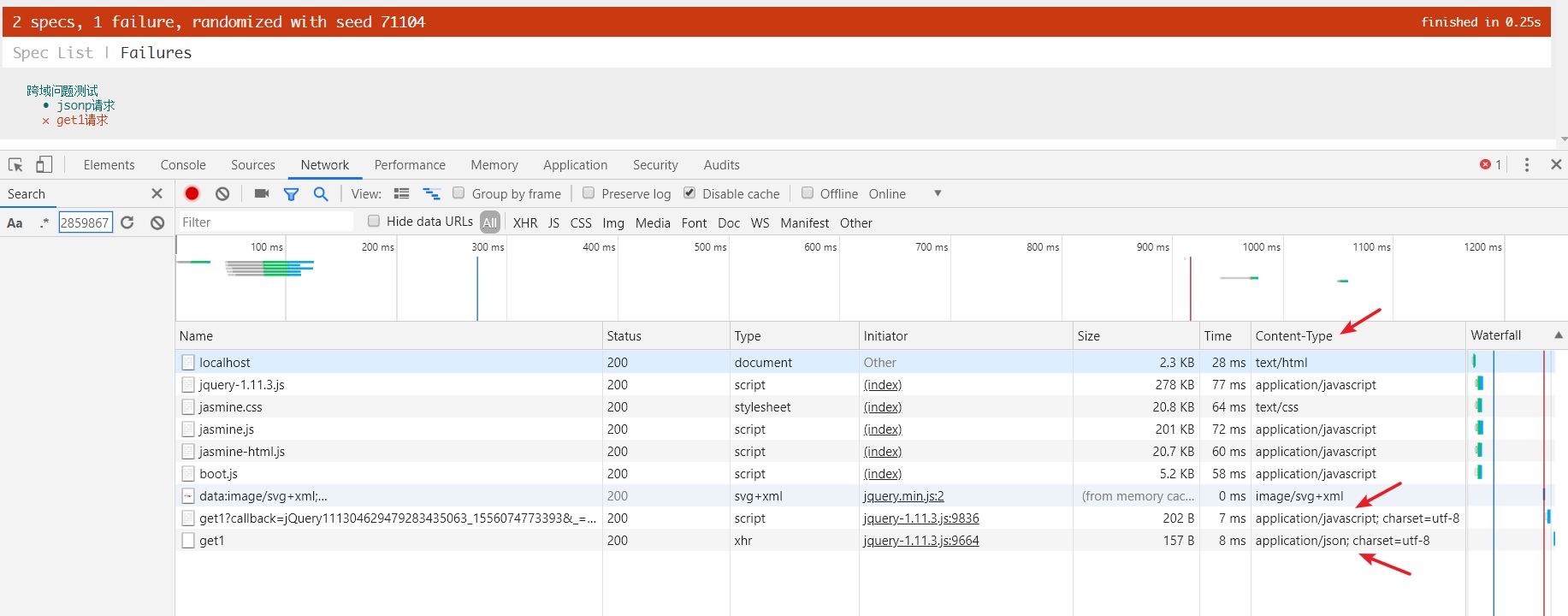
super("callback");

}

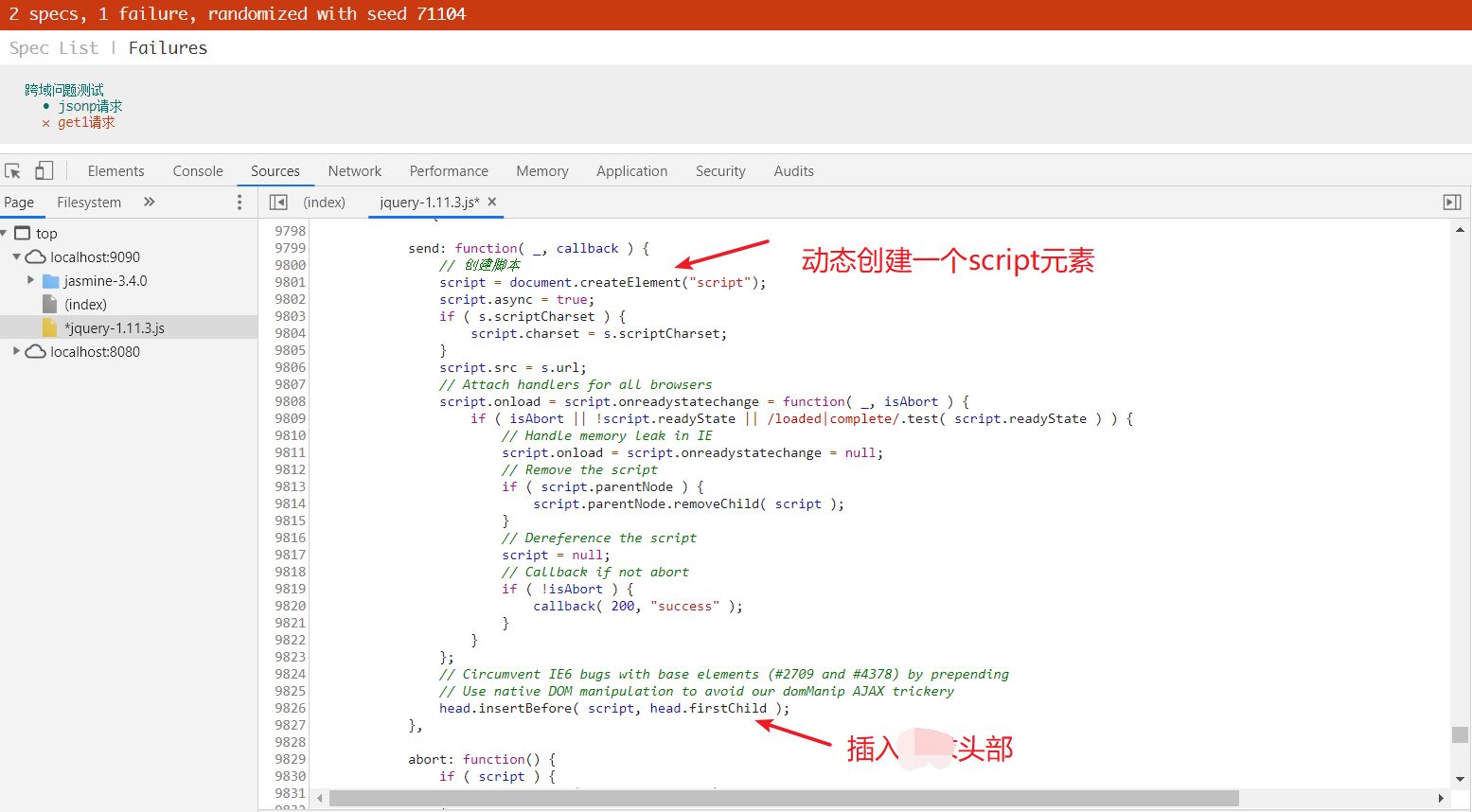
}

2.2.3 JSONP 实现原理

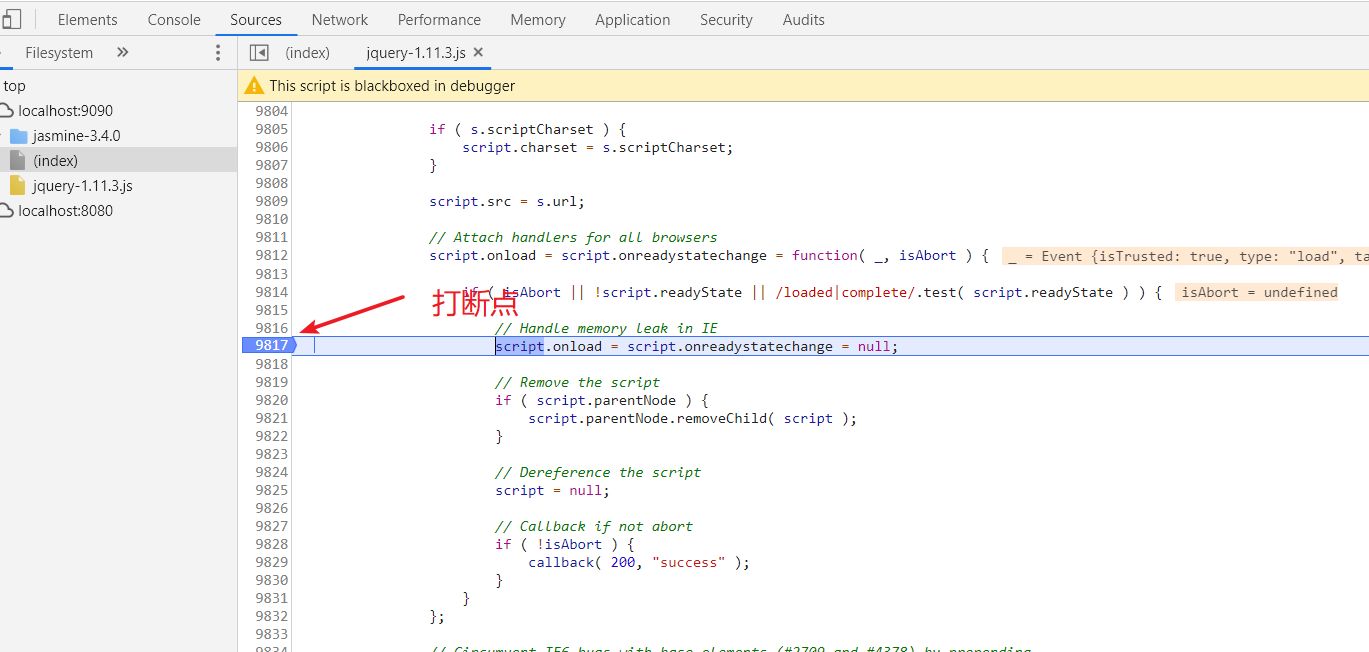
JSONP请求的类型是JavaScript脚本（callback 作为前后端的约定，callback的值做为方法名，json内容作为方法的参数），而XHR请求的类型是json类型。



可以在浏览器中查看 Jquery 源码来验证 JSONP 是否将请求包装成了 script 脚本。



在 Jquery 源码中打断点。



刷新后查看 element 元素，可以看到 Jquery 在 html 源码中添加了 script 标签。



**2.2.4 JSONP 的缺点**

1、只支持 GET 方法请求，不管 AJAX 中实际的请求方法是不是 GET

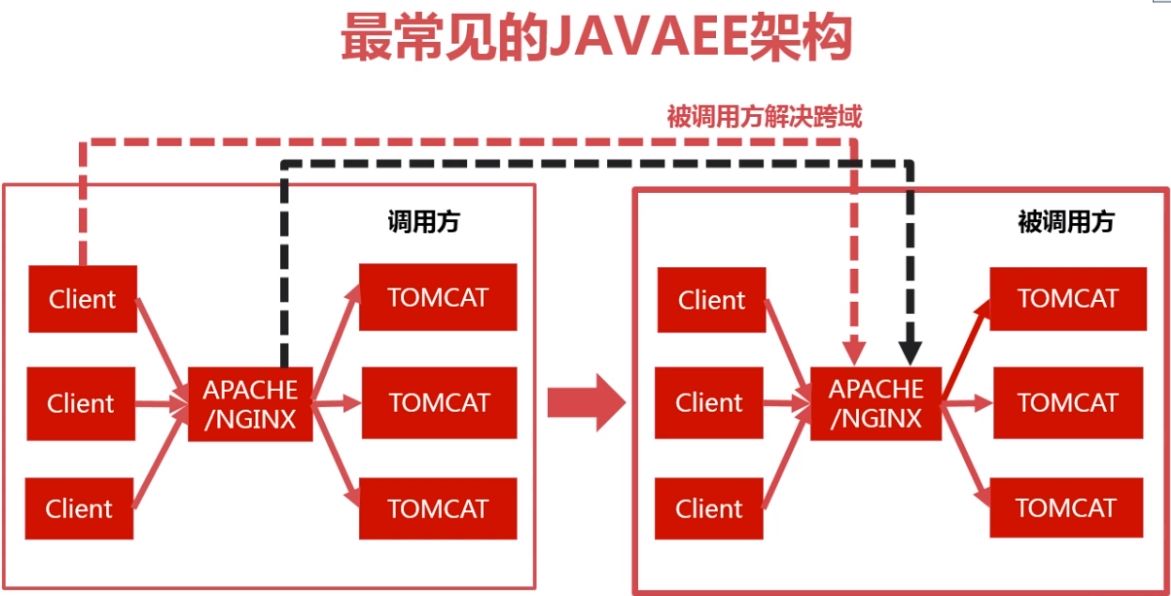
2、服务端还需要修改代码

3、发送的不是 XHR 请求，无法使用 XHR 对象（但这也是为什么可以解决跨域问题的根本）

总之，并不推荐使用 JSONP 方式来解决跨域问题，因为还有更好的解决方式。

**2.3 修改服务器端**

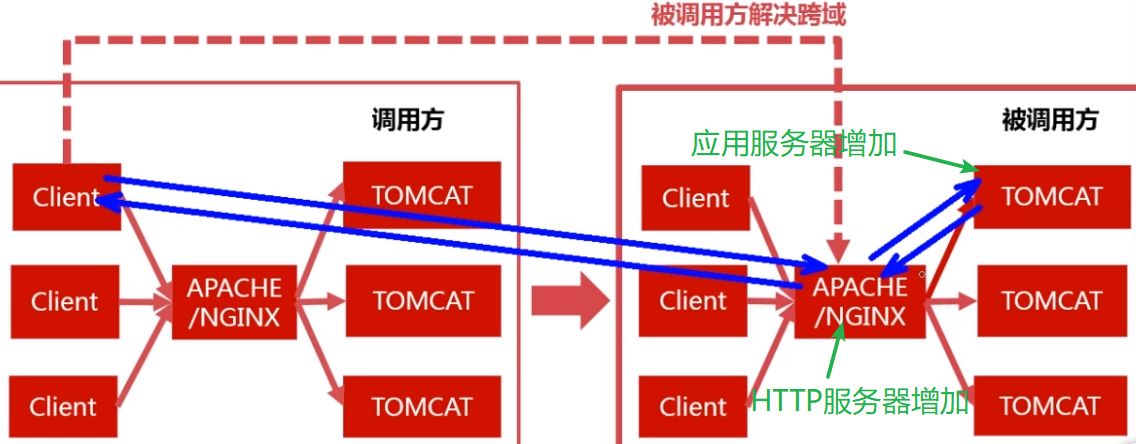
根据现如今网站架构设计，可以将前端应用看作调用方使用服务，将后端应用看作被调用方提供服务。



根据服务器的作用，可以将服务器分为 HTTP 服务器和应用服务器，所有修改服务器端既可以是修改应用服务器，也可以是修改 HTTP 服务器。

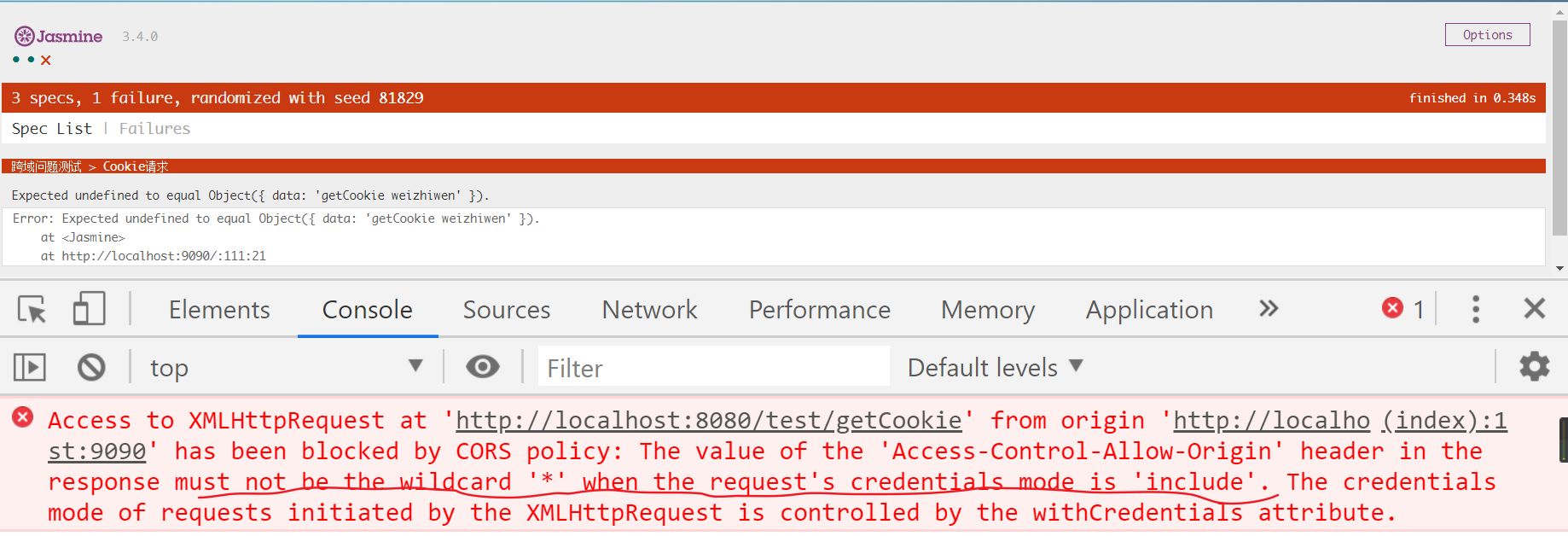
**2.3.1 被调用方修改**

**被调用方的解决思路是在**响应头中增加指定的字段允许调用方服务器跨域调用。



**在应用服务器增加指定字段**

对于不带 Cookie 的跨域请求，设置允许跨域的原始域名为任意域名，“Access-Control-Allow-Origin”: “\*”，设置允许跨域的方法为任意方法，“Access-Control-Allow-Methods”: “\*”，但是这样的星号设置不能满足带 Cookie 的跨域请求。



对于带 Cookie 的跨域请求，要指名允许跨域请求的调用方主机名，Cookie 要加在调用方。

带自定义头的跨域请求，设置允许跨域的请求头自定义的请求头，“Access-Control-Allow-Headers”:“自定义的请求头”。

在 Java Web 中，可以添加一个过滤器来设置上面的参数。



而使用 Spring Boot 框架，只需要在 Controller 类上加上 @CrossOrigin 注解就可以轻松解决跨域问题了。

**在 HTTP 服务器增加指定字段**

以常用的 Nginx 服务器和 Apache 服务器为例。

Nginx 服务器允许跨域配置（注意不要手动直接点击Nginx.exe，否则停止和重新载入配置会失败的）：

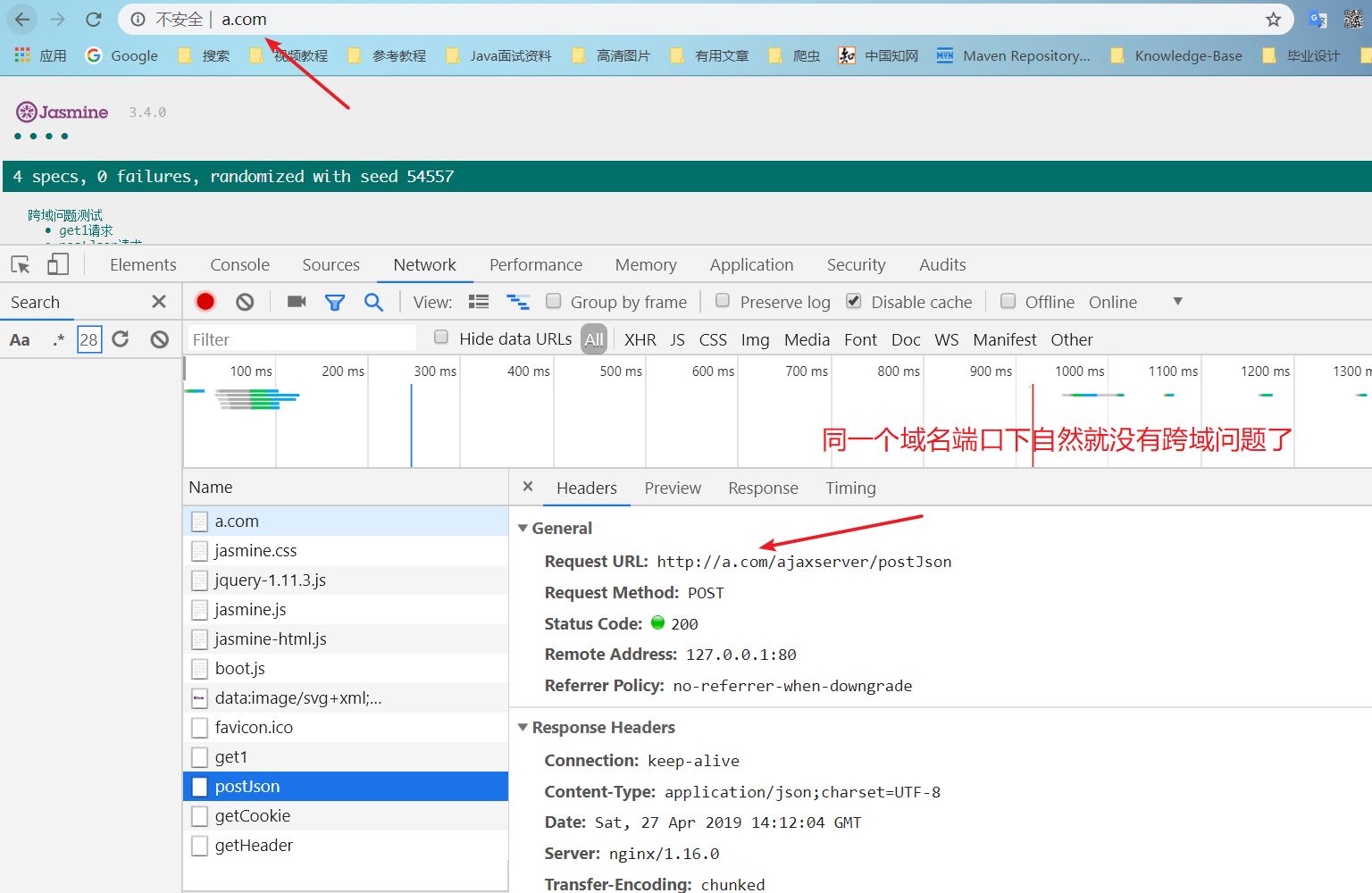


Apache 服务器允许跨域配置：



2.3.2 调用方修改

调用方的解决思路是反向代理，也即是将被调用方的域名代理到调用方域名下，这样就符合同源策略了，也就解决了跨域问题。



调用方修改一般都是直接修改 HTTP 服务器配置。

Nginx 服务器反向代理配置：



Apache 服务器反向代理配置：

