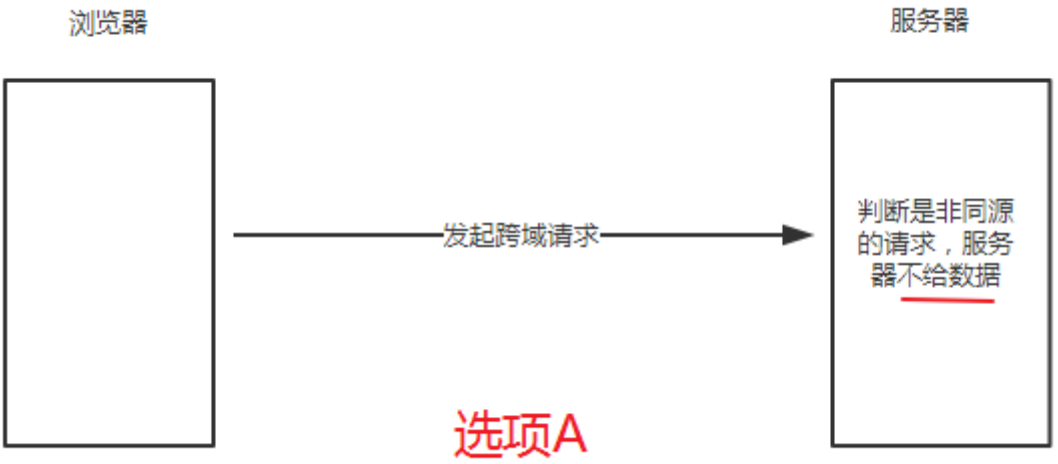
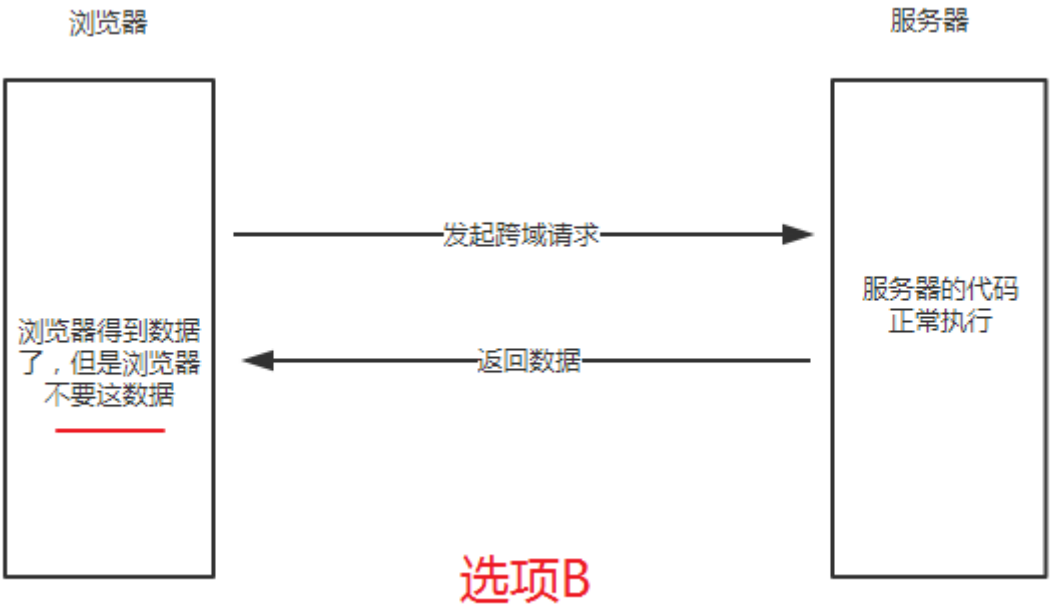
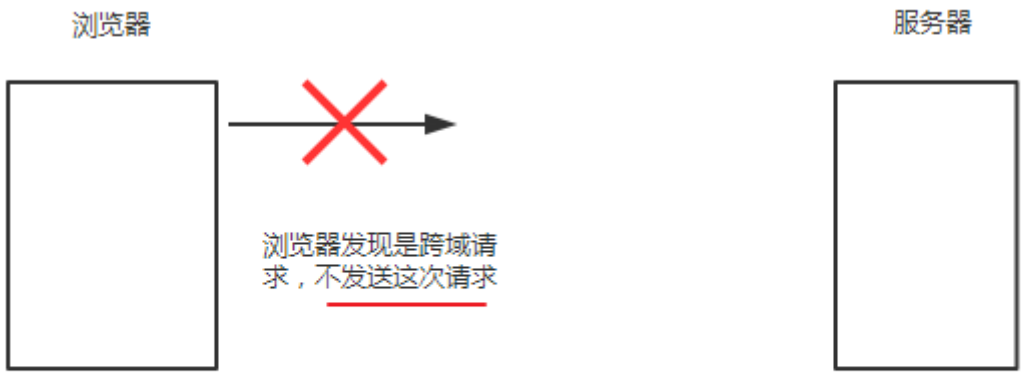
我们都知道，跨域请求会被同源策略所阻止，如界面
http://www.meiduo.site:8080/index.html这个界面中，使用js代码发起对
http://api.meiduo.site:9999/index_data这个地址的请求时，我们是得不到数据的，结论是很简单，也很好记，不过里面还是有一些细节很多人是比较模糊的。
到底是因为谁得不到数据呢？在下列的图片中选出你觉得正确的一项




选项A




选项B




选项C

获取不到数据到底是服务器不给数据，还是浏览器不要这数据？或者是连这次请求都不向服务器发送？
正确答案是B，是服务器不要这个数据，服务器的代码是正常执行的。

所以同源策略是浏览器的策略，和服务器没关系。不过我们可以通过对服务器的响应头配置，让浏览器接受这次数据。

写个小例子：
服务器的代码如下：

```
@app.route("/index_data", methods=["GET"])
def index_data():
    print("服务器接收到这次请求了...")
    response = make_response("HelloWorld")
    print("服务器即将把数据返回给浏览器")
    return response


if __name__ == '__main__':
    app.run(debug=True, port=9999, host='0.0.0.0')
```

index_data这个函数要通过http://api.meiduo.site:9999/index_data才能访问到，从而得到HelloWorld这个字符串数据

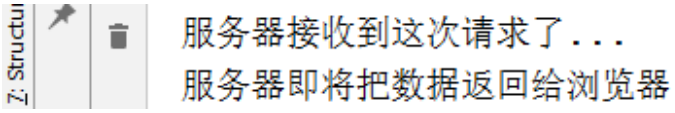
前端的代码和效果如下：


这是此界面的代码 →

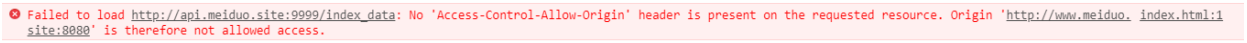
```
    <script>
    $(function(){
        $("#btn").click(function(){
            $.ajax({
                url:"http://api.meiduo.site:9999/index_data",
                type:"get",
                success:function(dat){
                    console.log(dat)
                }
            })
        })
    })
    </script>
</head>
<body>
    <input type="button" value="点我试试" id="btn">
</body>
```

当我们点击按钮来看看前后端的反应：

服务器：




浏览器：




服务器代码是执行完了，但是浏览器没有将数据给Ajax中的success，并且还报了一个错误：

No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://www.meiduo.site:8080' is therefore not allowed access.

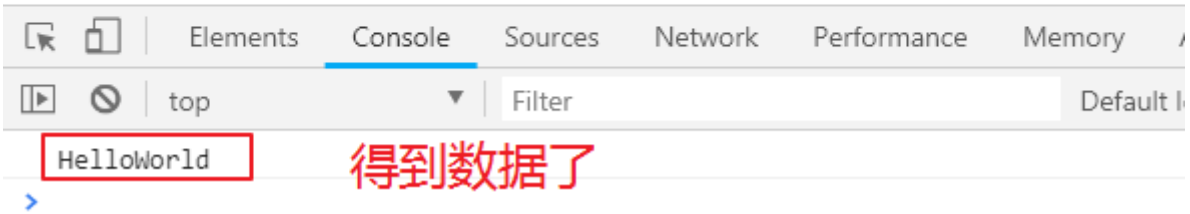
这个错误翻译过来就是，在你所请求的资源http://api.meiduo.site:9999/index_data的响应头中，并没有Access-Control-Allow-Origin的这项，所以http://api.meiduo.site:9999/index_data这个地址你是没有权限得到数据的

那现在我们就在响应头中加上Access-Control-Allow-Origin这一项试试

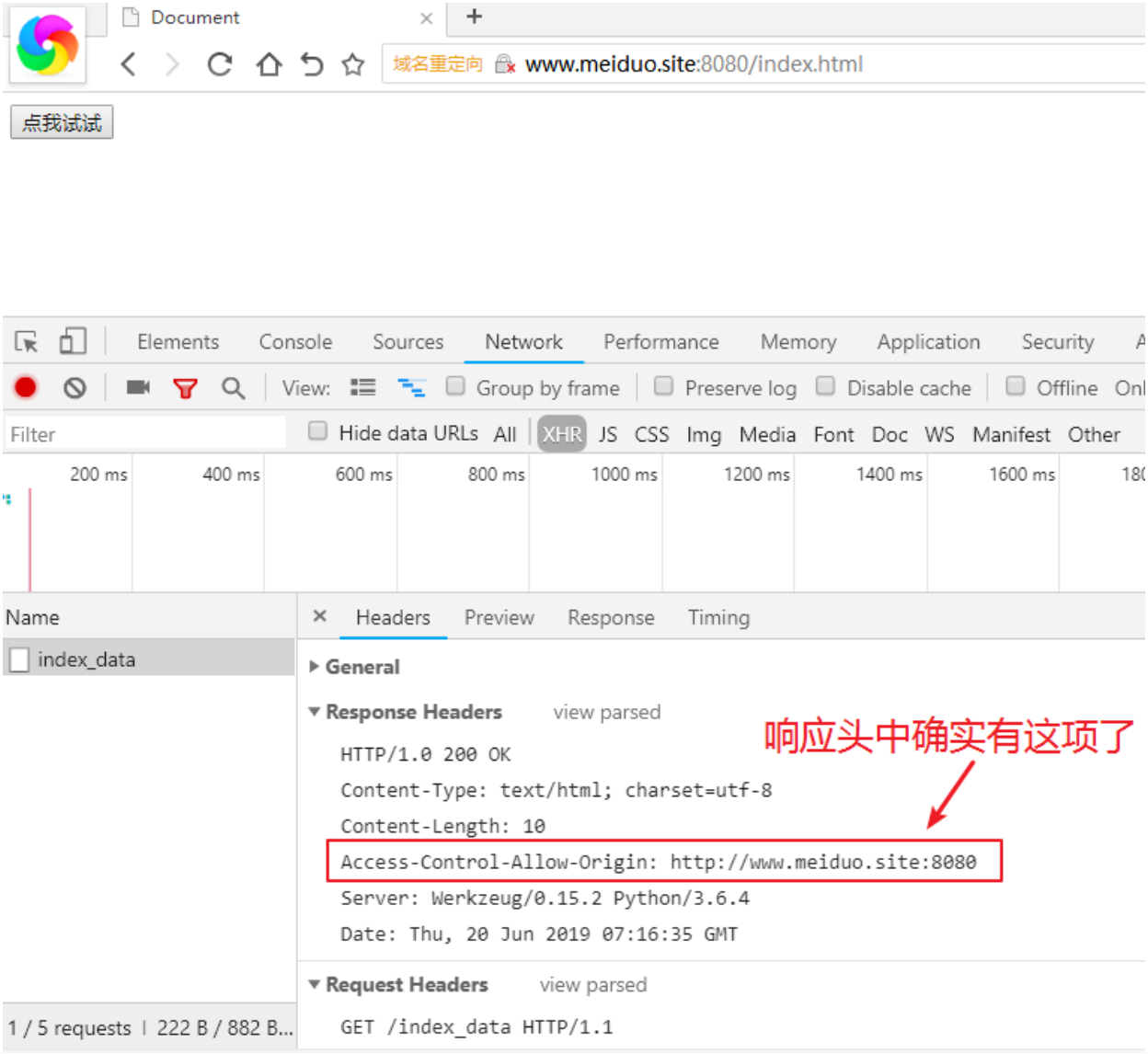
```python
@app.route("/index_data", methods=["GET"])
def index_data():
    print("服务器接收到这次请求了...")
    response = make_response("HelloWorld")
    response.headers["Access-Control-Allow-Origin"] = "http://www.meiduo.site:8080"
    print("服务器即将把数据返回给浏览器")
    return response


if __name__ == '__main__':
    app.run(debug=True, port=9999, host='0.0.0.0')
```

增加了这一行代码

在浏览器中再次点击按钮：


此时我们在浏览器的network中看这次HTTP请求：




浏览器会拿到这个响应头中的Access-Control-Allow-Origin的值，和当前的界面的地址做比较，看看是否满足同源策略，即：同协议、同域名、同端口，如满足，则让Ajax中的success得到数据

貌似和OPTIONS请求没啥事？其实在进行跨域请求的时候，浏览器并不是都会发起OPTIONS的请求，跨域的请求分为两种，简单的跨域请求和复杂的跨域请求,
简单的跨域请求：

- 请求方法为 `HEAD`、`GET`、`POST` 中的 1 种
- 请求的 header 中没有自定义的请求头

- Content-Type 为以下几种：`application/x-www-form-urlencoded`、`multipart/form-data`、`text/plain`等

非简单的跨域请求：

- header 中包含自定义请求头的 AJAX 请求
- `PUT`、`DELETE` 形式的 AJAX 请求
- Content-Type 字段的类型是 `application/json` 等

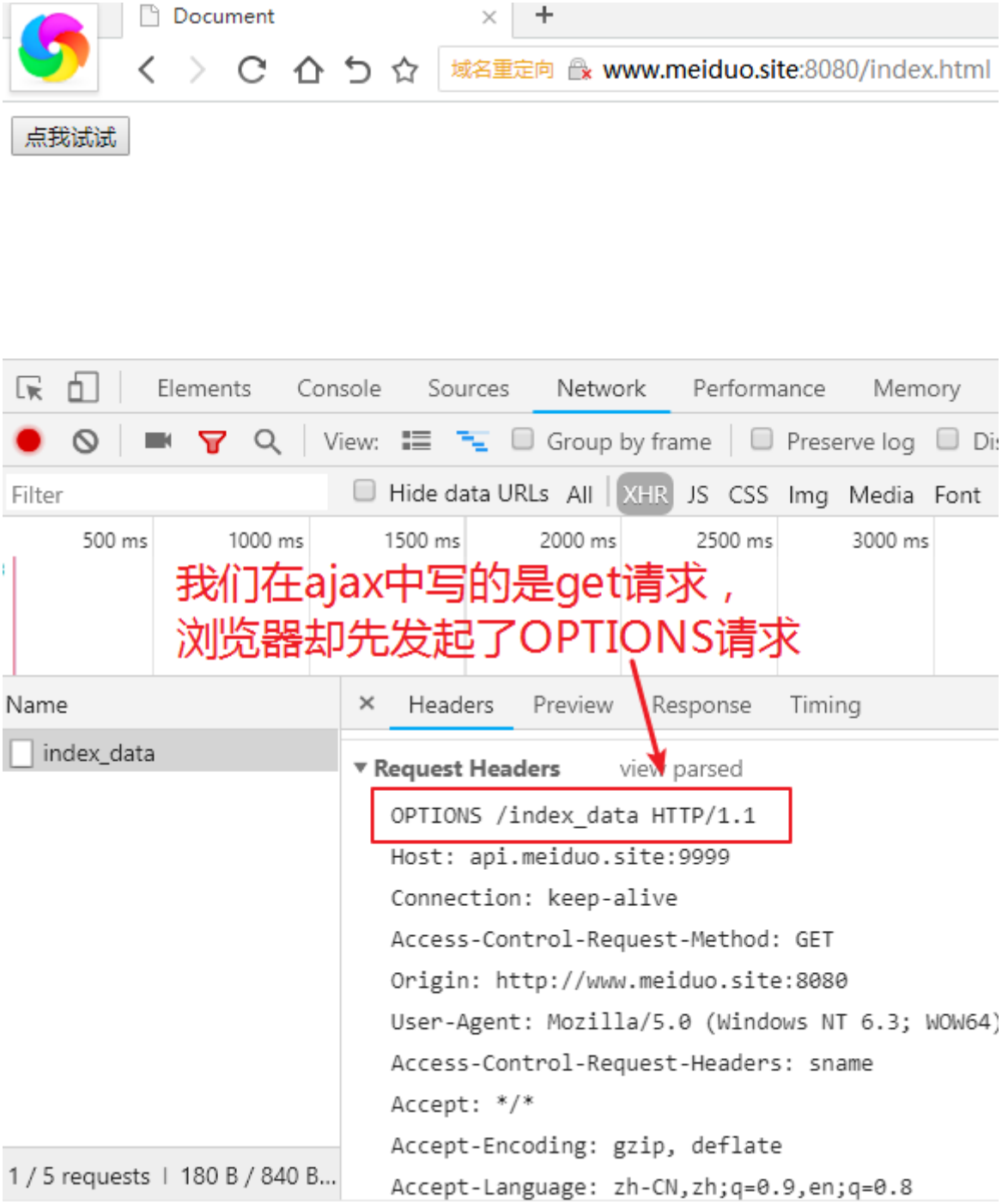
简单的跨域请求浏览器是不会发起OPTIONS请求的，复杂的跨域请求浏览器才会发起OPTIONS请求

再做个例子:

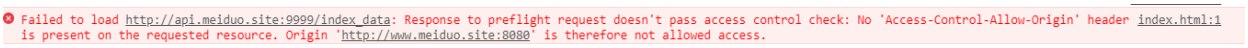
```
$(function(){
    $("#btn").click(function(){
        $.ajax({
            url:"http://api.meiduo.site:9999/index_data",
            type:"get",
            headers:{
                "sname":"itheima"
            },
            success:function(dat){
                console.log(dat)
            }
        })
    })
})
```

增加自定义请求头

点击按钮，看浏览器的network




浏览器此时依然报错：




所以我们需要先让OPTIONS请求得到正确的响应，浏览器才会发起正确的GET请求

改造服务器代码，如下：

```python
@app.route("/index_data", methods=["GET", "OPTIONS"])
def index_data():
    if request.method == "GET":
        response = make_response("HelloWorld")
        response.headers["Access-Control-Allow-Origin"] = "http://www.meiduo.site:8080"
        return response
    elif request.method == "OPTIONS":
        response = make_response("")
        response.headers["Access-Control-Allow-Origin"] = "http://www.meiduo.site:8080"
        return response
```

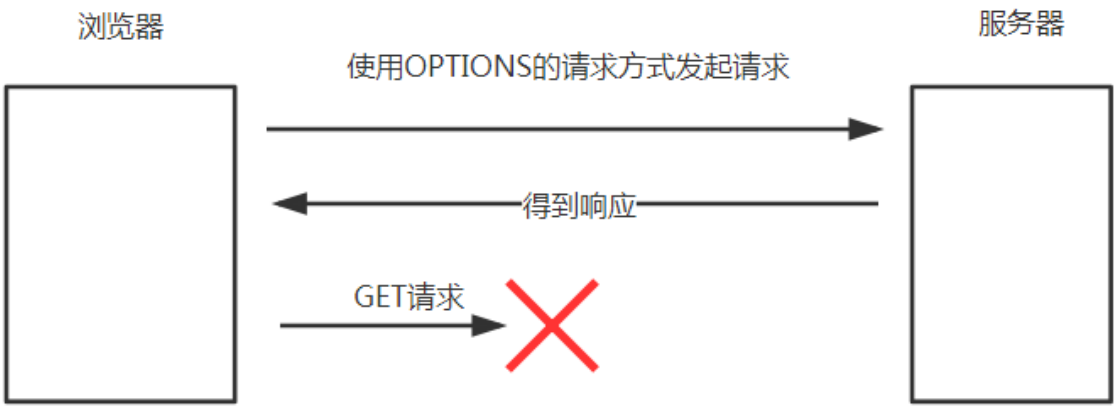
对于OPTIONS请求的响应，也增加了Access-Control-Allow-Origin的响应头信息

再测试：

依然报错：

```
Failed to load http://api.meiduo.site:9999/index_data: Request header field sname is not allowed by Access-Control-Allow-Headers in preflight response.
```

意思是请求头中的sname在OPTIONS请求的响应头中没有被允许，所以拥有自定义请求头的跨域请求此时也无法得到发起，所以这个时刻，是浏览器阻止了GET请求的发送



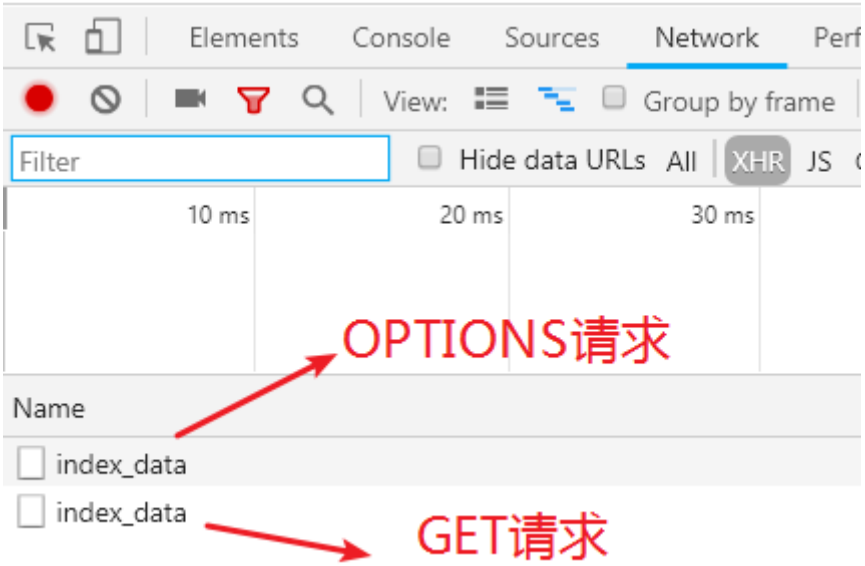

为了让GET请求能够发起，我们需要在OPTIONS的响应头中增加Access-Control-Allow-Headers即可

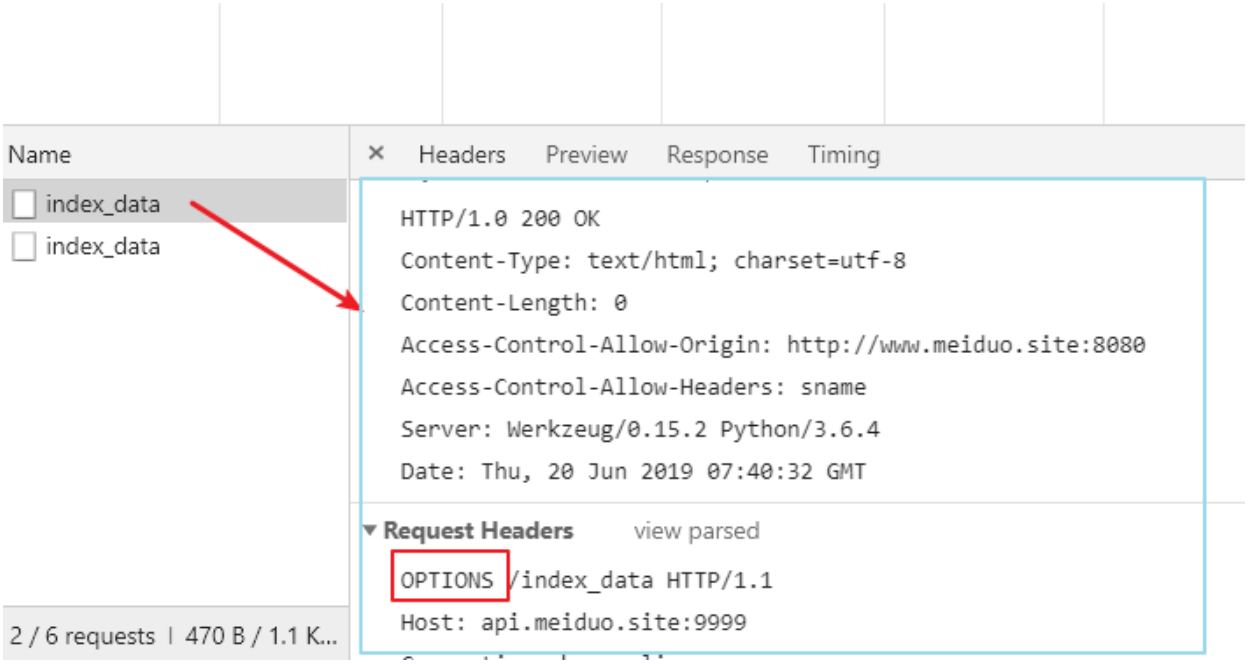
```python
@app.route("/index_data", methods=["GET", "OPTIONS"])
def index_data():
    print("接收到请求了...")
    if request.method == "GET":
        response = make_response("HelloWorld")
        response.headers["Access-Control-Allow-Origin"] = "http://www.meiduo.site:8080"
        return response
    elif request.method == "OPTIONS":
        response = make_response("")
        response.headers["Access-Control-Allow-Origin"] = "http://www.meiduo.site:8080"
        response.headers["Access-Control-Allow-Headers"] = "sname"
        return response
```

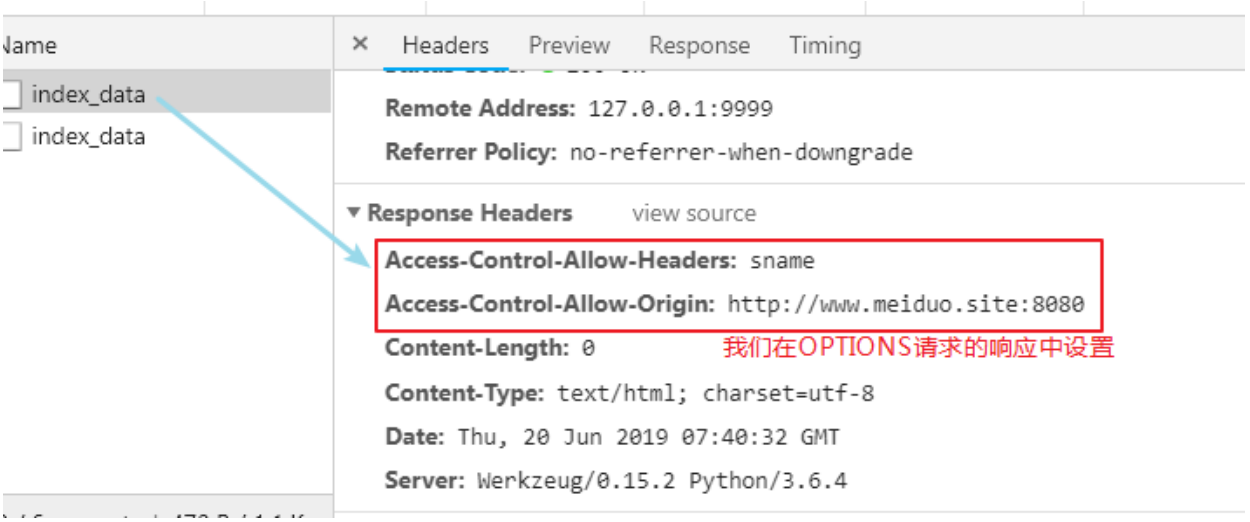
运行下一次请求的请求头中的字段

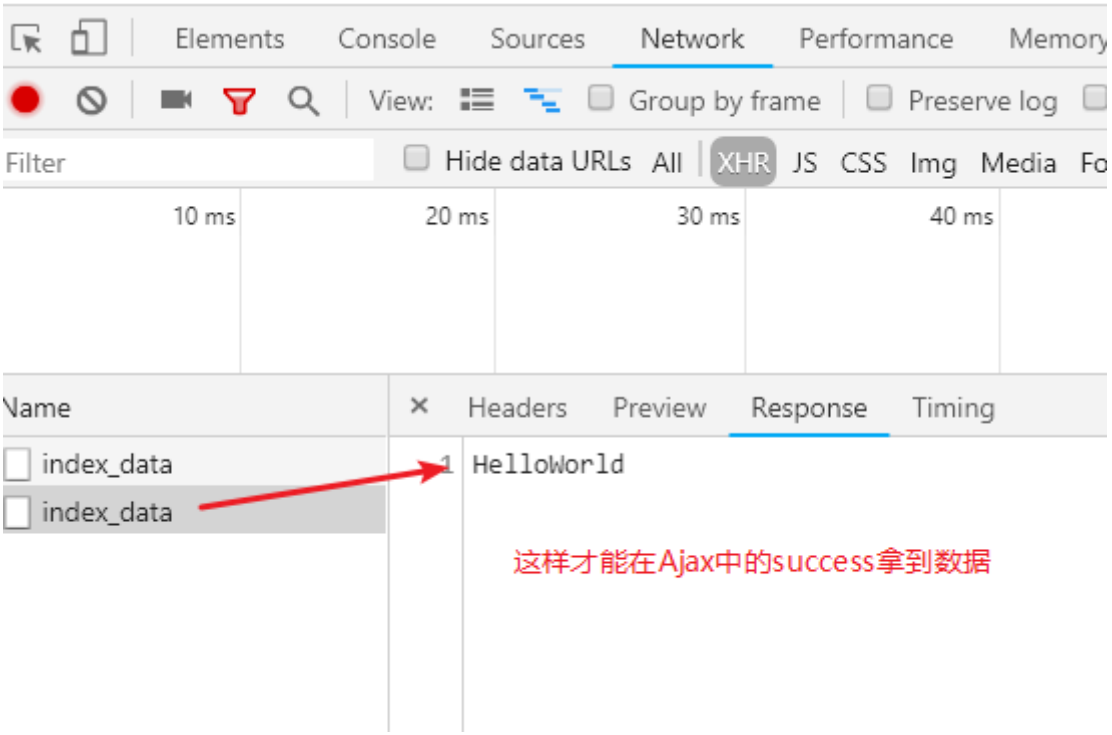
然后使用浏览器再点击按钮试一试：


域名重定向
www.meidu
点我试试


Elements
Console
Sources
Network
Perf
View:
Group by frame
Filter
Hide data URLs
All
XHR
JS
10 ms
20 ms
30 ms
OPTIONS请求
Name
index_data
index_data
GET请求


Name
index_data
index_data
Headers
Preview
Response
Timing
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 0
Access-Control-Allow-Origin: http://www.meiduo.site:8080
Access-Control-Allow-Headers: sname
Server: Werkzeug/0.15.2 Python/3.6.4
Date: Thu, 20 Jun 2019 07:40:32 GMT
Request Headers
view parsed
OPTIONS /index_data HTTP/1.1
Host: api.meiduo.site:9999
2 / 6 requests | 470 B / 1.1 K...




所以

1、服务器可以设置浏览器要不要跨域来的数据

2、服务器可以设置当浏览器在发起复杂请求头的请求的时候，哪些请求头能被允许

3、OPTIONS的请求相当于是一个询问请求