

Ajax

jQuery对XMLHttpRequest组件的调用接口实现了封装，更加方便调用。默认是**异步**请求。

GET请求

flask中蓝图代码改动如下

```
1 from flask import Blueprint, jsonify, render_template, request
2
3 books = Blueprint('book', __name__)
4
5 @books.route('/')
6 def list():
7     books = {
8         'count':3,
9         'results':[
10             (1, 'python'),
11             (2, 'java'),
12             (3, 'javascript')
13         ]
14     }
15     print(*dir(request), sep='\n') # request用小写的
16     print('~' * 30)
17     print(request.url)
18     print(request.full_path)
19     print(request.method)
20     print(request.cookies)
21     print(request.content_type)
22     print(request.args)
23     print(request.query_string)
24     print(request.form)
25     print(request.is_json, request.get_json, request.json)
26     print('~' * 30)
27     return jsonify(books)
```

也修改一下注册蓝图 `app.register_blueprint(books, url_prefix='/books')`

```
1 <!DOCTYPE html>
2 <html lang="zh-CN">
3
4 <head>
5     <meta charset="UTF-8">
6     <script src="{{ url_for('static', filename='js/jquery-3.6.0.min.js') }}"></script>
7     <title>马哥教育jQuery测试</title>
8 </head>
9
10 <body>
11     <div id="root">
12         <h2>测试事件</h2>
13         <div><button class="testclick">点击事件</button></div>
```

```

14     <button class="ajaxget">Ajax GET请求</button>
15     <div class="console" style="border: 1px solid #000;"></div>
16 </div>
17 </body>
18
19 </html>
20 <script>
21     $(function () {
22         $('button.testclick').click(function (event) {
23             console.log(event.target);
24             var c = $('<div>hello</div>'); // 包装成jquery对象
25             c.css({ 'background-color': '#F0F0F0', 'margin': '10px' });
26             $('.console').append(c); // 使用jquery对象
27         });
28
29         // AJAX GET请求
30         $('button.ajaxget').click(function (event) {
31             // http://127.0.0.1:5000/books/
32             $.get('/books/', { k1: 1, k2: 2, k2: 3 },
33                 function (data) {
34                     console.log(data);
35                 }
36             )
37         })
38     })
39
40 </script>

```

```

1 GET /books/?k1=1&k2=3 HTTP/1.1
2 Host: 127.0.0.1:5000
3 Connection: keep-alive
4 Accept: */*
5
6 X-Requested-With: XMLHttpRequest
7 User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like
  Gecko) Maxthon/5.0 Chrome/55.0.2883.75 Safari/537.36
8 Referer: http://127.0.0.1:5000/
9 Accept-Encoding: gzip, deflate
10 Accept-Language: zh-CN

```

访问蓝图中的list()视图函数，返回Json数据。

```

1 {
2     "count": 3,
3     "results": [
4         [
5             1,
6             "python"
7         ],
8         [
9             2,
10            "java"
11        ],
12        [
13            3,
14            "javascript"

```

```
15     ]
16   ]
17 }
18
```

GET请求跨域

如果你发起HTTP请求时，使用了不同的域名或端口，这就不是同域了。

为了演示效果，使用不同域来访问。

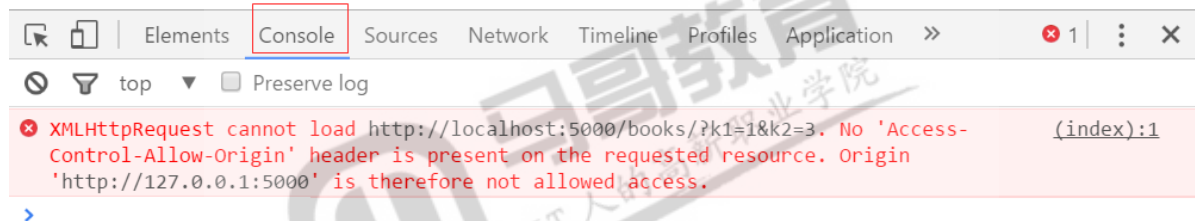
如果浏览器地址栏使用 `127.0.0.1`，那么Ajax请求就使用 `localhost`。总之两个不一样就行。

点击“Ajax GET请求”按钮，发现数据返回了，但是控制台中的数据不能打印了，并报了下面的错误

测试jQuery

事件响应

Ajax GET请求



这是HTTP请求跨域访问产生的。

请求头如下

```
1 GET /books/?k1=1&k2=3 HTTP/1.1
2 Host: localhost:5000
3 Connection: keep-alive
4 Accept: */*
5 Origin: http://127.0.0.1:5000
6 User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like
7   Gecko) Maxthon/5.0 Chrome/55.0.2883.75 Safari/537.36
8 DNT: 1
9 Referer: http://127.0.0.1:5000/
10 Accept-Encoding: gzip, deflate
11 Accept-Language: zh-CN
```

上面是GET请求，跨域请求后，其实数据都已经返回到了浏览器端，但是被浏览器拒绝了。

跨域CORS

Cross Origin Resource Sharing跨域资源共享，它是使用额外的HTTP头来告诉浏览器准许访问一个与当前所在origin (Domain) **不同源**的服务器上特定的资源。

不同源：指的是不同的协议，或不同域，或不同端口。

当浏览器从当前所在资源所在源，对不同源的资源发起HTTP请求的时候，这个HTTP请求就是跨域HTTP请求。

简单请求

简单请求：

1. 如果请求方式是，GET、POST、HEAD三种方法之一
2. Content-Type是，text/plain、multipart/form-data、application/x-www-form-urlencoded三种值之一

简单请求解决方案比较简单，只需要在服务器端**响应报文**中增加一个首部。

简单请求跨域解决

浏览器端请求报文中的首部发送字段Origin首部，服务器端响应首部增加Access-Control-Allow-Origin 首部允许即可。

```
1 from flask import Blueprint, jsonify, render_template, request
2
3 books = Blueprint('book', __name__)
4
5 @books.route('/')
6 def list():
7     books = {
8         'count':3,
9         'results':[
10             (1, 'python'),
11             (2, 'java'),
12             (3, 'javascript')
13         ]
14     }
15     ret = jsonify(books)
16     ret.headers['Access-Control-Allow-Origin'] = '*'
17     return ret
```

Access-Control-Allow-Origin: * 意思是什么域都被允许；

Access-Control-Allow-Origin: http://magedu.com 只允许指定域。

res.headers['Access-Control-Allow-Origin'] = 'http://127.0.0.1:5000'

预检请求

如果发送**不是**上面的简单请求，就是发送预检请求。

预检请求需要先使用OPTIONS方法先发送一个预检请求到服务器端，看看服务器端是否允许该请求。

POST请求跨域

1、简单请求

```
1  <!DOCTYPE html>
2  <html lang="zh-CN">
3
4  <head>
5      <meta charset="UTF-8">
6      <script src="{ url_for('static', filename='js/jquery-3.6.0.min.js')
7      }}"></script>
8      <title>马哥教育jQuery测试</title>
9  </head>
10
11 <body>
12     <div id="root">
13         <h2>测试事件</h2>
14         <div><button class="testclick">点击事件</button></div>
15         <button class="ajaxget">Ajax GET请求</button>
16         <button class="ajaxpost">Ajax POST请求</button>
17         <div class="console" style="border: 1px solid #000;"></div>
18     </div>
19 </body>
20 </html>
21 <script>
22     $(function () {
23         $('button.testclick').click(function (event) {
24             console.log(event.target);
25             var c = $('<div>hello</div>'); // 包装成jquery对象
26             c.css({ 'background-color': '#F0F0F0', 'margin': '10px' });
27             $('.console').append(c); // 使用jquery对象
28         });
29
30         // AJAX GET请求
31         $('button.ajaxget').click(function (event) {
32             // http://127.0.0.1:5000/books/
33             $.get('http://localhost:5000/books/', { k1: 1, k2: 2, k2: 3 },
34                 function (data) {
35                     console.log(data);
36                 }
37             )
38         })
39
40         // AJAX POST请求
41         $('button.ajaxpost').click(function (event) {
42             // http://127.0.0.1:5000/books/
43             $.post('http://localhost:5000/books/?k1=v1&k2=v2&k2=v3', { p1:
44             100, p2: 200 },
45                 function (data, status) {
46                     console.log(data, status);
47                 }
48             )
49         })
50     })
51 </script>
```

状态码405，说明Flask中视图函数的方法不支持POST，增加方法就可以了。@books.route('/', methods=['GET', 'POST'])

注意观察请求头，跨域访问依然是**简单请求**，所以请求头依旧是POST。依然需要

```
ret.headers['Access-Control-Allow-Origin'] = '*'
```

```
1 from flask import Blueprint, jsonify, render_template, request
2
3 books = Blueprint('book', __name__)
4
5 @books.route('/', methods=['get', 'post']) # 默认只支持GET、HEAD，不支持POST
6 def list():
7     books = {
8         'count':3,
9         'results':[
10             (1, 'python'),
11             (2, 'java'),
12             (3, 'javascript')
13         ]
14     }
15     # print(*dir(request), sep='\n') # request用小写的
16     # print('~' * 30)
17     print(request.url)
18     print(request.full_path)
19     print(request.method)
20     # print(request.cookies)
21     print(request.content_type)
22     print(request.args)
23     print(request.query_string)
24     print(request.form)
25     # print(request.is_json, request.get_json, request.json)
26     print('~' * 30)
27     ret = jsonify(books)
28     ret.headers['Access-Control-Allow-Origin'] = '*'
29     return ret
```

```
1 http://localhost:5000/books/?k1=v1&k2=v2&k2=v3
2 /books/?k1=v1&k2=v2&k2=v3
3 POST
4 application/x-www-form-urlencoded; charset=UTF-8
5 ImmutableMultiDict([('k1', 'v1'), ('k2', 'v2'), ('k2', 'v3')])
6 b'k1=v1&k2=v2&k2=v3'
7 ImmutableMultiDict([('p1', '100'), ('p2', '200')])
```

method是POST，content_type是application/x-www-form-urlencoded，这是简单请求。

POST的简单请求的数据使用request.form即可提取。

2、预检请求

POST请求提交JSON数据

```
1 <!DOCTYPE html>
2 <html lang="en">
3
```

```

4 <head>
5   <meta charset="UTF-8">
6   <title>马哥教育</title>
7   <script src={{ url_for('static', filename='js/jquery.js') }}></script>
8 </head>
9
10 <body>
11   <h2 id="root">测试jQuery</h2>
12   <button class="testclick">事件响应</button>
13   <button class="ajaxget">Ajax GET请求</button>
14   <button class="ajaxpost">Ajax POST请求</button>
15   <button class="ajaxpostjson">Ajax POST请求提交Json数据</button>
16   <div class="content" style="border:1px solid #000"></div>
17   <script>
18     $(function () {
19       // 省略其它代码
20
21       // AJAX POST 请求
22       $('button.ajaxpost').click(function (event) {
23         $.post('http://localhost:5000/books/?k1=v1&k2=v2&k2=v3',
24           { p1: 100, p2: 200 },
25           /*http://127.0.0.1:5000/books/?k1=v1&k2=v2&k2=v3如果同域
可以写*/
           function (data, status) {// 成功回调函数，data就是返回的正文
内容
               console.log(data);
               console.log(status);
           })
       });
26
27       // AJAX POST with JSON
28       $('button.ajaxpostjson').click(function (event) {
29         $.ajax({
30           type: 'POST',
31           url: 'http://localhost:5000/books/?k1=v1&k2=v2&k2=v3',
32           contentType: "application/json",
33           data: { p1: 100, p2: 200 },
34           /*http://127.0.0.1:5000/books/?k1=v1&k2=v2&k2=v3如果同域
可以写*/
35           success: function (data, status) {// 成功回调函数，data就是
返回的正文内容
               console.log(data);
               console.log(status);
           }
36         });
37       });
38     });
39   </script>
40 </body>
41
42 </html>

```

直接点击 Ajax POST请求提交Json数据 按钮，在浏览器控制台出现下图错误

✖ XMLHttpRequest cannot load http://localhost:5000/books/?k1=v1&k2=v2%&k2=v3. Response to (index):1 preflight request doesn't pass access control check: No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://127.0.0.1:5000' is therefore not allowed access.

Name	×	Headers	Preview	Response	Timing
<input type="checkbox"/> ?k1=v1&k2=v2%&k2=v3		<div>▼ Request Headersview parsed</div> <div>OPTIONS /books/?k1=v1&k2=v2%&k2=v3 HTTP/1.1 Host: localhost:5000 Connection: keep-alive Access-Control-Request-Method: POST Origin: http://127.0.0.1:5000 X-DevTools-Emulate-Network-Conditions-Client-Id: a7899476-5a03-4a5e-b588-f8e1118e6d14 User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Maxthon/5.0 Chrome/55.0.2883.75 Safari/537.36 Access-Control-Request-Headers: content-type Accept: */* DNT: 1 Referer: http://127.0.0.1:5000/ Accept-Encoding: gzip, deflate Accept-Language: zh-CN</div> <div>▼ Query String Parametersview sourceview URL encoded</div> <div>k1: v1 k2: (unable to decode value) k2: v3</div>			

1 requests | 186 B transferred

在jQuery官方有这么一句

contentType (default: 'application/x-www-form-urlencoded; charset=UTF-8')

Type: [Boolean](#) or [String](#)

When sending data to the server, use this content type. Default is "application/x-www-form-urlencoded; charset=UTF-8", which is fine for most cases. If you explicitly pass in a content-type to `$.ajax()`, then it is always sent to the server (even if no data is sent). As of jQuery 1.6 you can pass `false` to tell jQuery to not set any content type header. **Note:** The W3C XMLHttpRequest specification dictates that the charset is always UTF-8; specifying another charset will not force the browser to change the encoding. **Note:** For cross-domain requests, setting the content type to anything other than `application/x-www-form-urlencoded`, `multipart/form-data`, or `text/plain` will trigger the browser to send a preflight OPTIONS request to the server.

也就是说跨域访问如果设置不是这3种Content-Type，也就是不是简单请求，那么会触发preflight OPTIONS请求（预检请求）。

请求头如下

1 OPTIONS /books/?k1=v1&k2=v2%&k2=v3 HTTP/1.1

2 Host: localhost:5000

3 Connection: keep-alive

4 Access-Control-Request-Method: POST

5 origin: http://127.0.0.1:5000

6 X-DevTools-Emulate-Network-Conditions-Client-Id: a7899476-5a03-4a5e-b588-f8e1118e6d14

7 User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Maxthon/5.0 Chrome/55.0.2883.75 Safari/537.36

8 Access-Control-Request-Headers: content-type

9 Accept: */*

10 DNT: 1

11 Referer: http://127.0.0.1:5000/

12 Accept-Encoding: gzip, deflate

13 Accept-Language: zh-CN


```
OPTIONS /books/?k1=v1&k2=v2%&k2=v3 HTTP/1.1
Host: localhost:5000
Connection: keep-alive
Access-Control-Request-Method: POST
Origin: http://127.0.0.1:5000
X-DevTools-Emulate-Network-Conditions-Client-Id: a7899476-5a03-4a5e-b588-f8e
User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gec
5.0 Chrome/55.0.2883.75 Safari/537.36
Access-Control-Request-Headers: content-type
Accept: */*
DNT: 1
Referer: http://127.0.0.1:5000/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN
```

正常请求头如下

Request Headers [view parsed](#)

```
POST /books/?k1=v1&k2=v2%&k2=v3 HTTP/1.1
Host: localhost:5000
Connection: keep-alive
Content-Length: 19
Accept: */*
Origin: http://127.0.0.1:5000
X-DevTools-Emulate-Network-Conditions-Client-Id: a7899476-5a03-
User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (K
5.0 Chrome/55.0.2883.75 Safari/537.36
Content-Type: application/json
DNT: 1
Referer: http://127.0.0.1:5000/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN
```

解决方案

- 1、不跨域，使用同域来访问网页即可
- 2、修改视图函数，增加OPTIONS方法支持，并返回相应的访问控制头

```
1 from flask import Blueprint, jsonify, render_template, request
2
3 books = Blueprint('book', __name__)
4
5 @books.route('/', methods=['get', 'post', 'options']) # 默认只支持GET、HEAD，不
   支持POST
6 def list():
7     books = {
8         'count':3,
9         'results':[
10             (1, 'python'),
11             (2, 'java'),
12             (3, 'javascript')
13         ]
14     }
```

```

15     # print(*dir(request), sep='\n') # request用小写的
16     # print('~' * 30)
17     print(request.url)
18     print(request.full_path)
19     print(request.method)
20     # print(request.cookies)
21     print(request.content_type)
22     print(request.args)
23     print(request.query_string)
24     print(request.form)
25     if request.is_json:
26         print(request.data)
27         print(request.json) # 转换json失败
28
29     print('~' * 30)
30     ret = jsonify(books)
31     ret.headers['Access-Control-Allow-Origin'] = '*'
32     ret.headers['Access-Control-Allow-Headers'] = 'content-type'
33     return ret

```

跨域访问成功了，注意是2次HTTP请求，第一次就是提起预检请求OPTIONS方法，如果对方允许，才发起第二次POST请求。

Json数据处理

服务器端并没有成功的解析出Json数据，观察发现发过来的数据不是Json。

```

1  <!DOCTYPE html>
2  <html lang="zh-CN">
3
4  <head>
5      <meta charset="UTF-8">
6      <script src="{ url_for('static', filename='js/jquery-3.6.0.min.js')
7  }"></script>
8      <title>马哥教育jQuery测试</title>
9  </head>
10
11 <body>
12     <div id="root">
13         <h2>测试事件</h2>
14         <div><button class="testclick">点击事件</button></div>
15         <button class="ajaxget">Ajax GET请求</button>
16         <button class="ajaxpost">Ajax POST请求</button>
17         <button class="ajaxpostjson">Ajax POST请求提交Json数据</button>
18         <div class="console" style="border: 1px solid #000;"></div>
19     </div>
20 </body>
21 </html>
22 <script>
23     $(function () {
24         $('button.testclick').click(function (event) {
25             console.log(event.target);
26             var c = $('<div>hello</div>'); // 包装成jquery对象
27             c.css({ 'background-color': '#F0F0F0', 'margin': '10px' })

```

```

28         $('#console').append(c); // 使用jquery对象
29     });
30
31     // AJAX GET请求
32     $('#button.ajaxget').click(function (event) {
33         // http://127.0.0.1:5000/books/
34         $.get('http://localhost:5000/books/', { k1: 1, k2: 2, k2: 3 },
35             function (data) {
36                 console.log(data);
37             }
38         )
39     })
40
41     // AJAX POST请求
42     $('#button.ajaxpost').click(function (event) {
43         // http://127.0.0.1:5000/books/
44         $.post('http://localhost:5000/books/?k1=v1&k2=v2&k2=v3', { p1:
100, p2: 200 },
45             function (data, status) {
46                 console.log(data, status);
47             }
48         )
49     })
50
51     // AJAX POST with JSON
52     $('#button.ajaxpostjson').click(function (event) {
53         $.ajax({
54             type: 'POST',
55             url: 'http://localhost:5000/books/?k1=v1&k2=v2&k2=v3',
56             contentType: "application/json",
57             data: JSON.stringify({ p1: 100, p2: 200 }), // 转换为Json
58             success: function (data, status) { // 成功回调函数, data就是返回
的正文内容
59                 console.log(data);
60                 console.log(status);
61             }
62         });
63     });
64 })
65
66 </script>

```

Django静态配置

Django的静态文件配置，如下

- 1、settings中INSTALLED_APPS确保有django.contrib.staticfiles
- 2、settings中定义静态路径，STATIC_URL = '/static/'
- 3、settings中定义
 STATICFILES_DIRS = [
 os.path.join(BASE_DIR, "static"),
]

习题

Ajax方式提交GET方法，从服务器返回数据，使用表格显示，要求使用JS对表格进行动态追加。

