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

```python
from flask import Blueprint, jsonify, render_template, request

books = Blueprint('book', __name__)

@books.route('/')
def list():
    books = {
        'count':3,
        'results':[
            (1, 'python'),
            (2, 'java'),
            (3, 'javascript')
        ]
    }
    print(*dir(request), sep='\n') # request用小写的
    print('~' * 30)
    print(request.url)
    print(request.full_path)
    print(request.method)
    print(request.cookies)
    print(request.content_type)
    print(request.args)
    print(request.query_string)
    print(request.form)
    print(request.is_json, request.get_json, request.json)
    print('~' * 30)
    return jsonify(books)
```

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

```html
<!DOCTYPE html>
<html lang="zh-CN">

<head>
    <meta charset="UTF-8">
    <script src="{{ url_for('static', filename='js/jquery-3.6.0.min.js') }}"></script>
    <title>马哥教育jQuery测试</title>
</head>

<body>
    <div id="root">
        <h2>测试事件</h2>
        <div><button class="testclick">点击事件</button></div>
```

```
        <button class="ajaxget">Ajax GET请求</button>
        <div class="console" style="border: 1px solid #000;"></div>
    </div>
</body>

</html>
<script>
    $(function () {
        $('button.testclick').click(function (event) {
            console.log(event.target);
            var c = $('<div>hello</div>'); // 包装成jquery对象
            c.css({ 'background-color': '#F0F0F0', 'margin': '10px' })
            $('.console').append(c); // 使用jquery对象
        });

        // AJAX GET请求
        $('button.ajaxget').click(function (event) {
            // http://127.0.0.1:5000/books/
            $.get('/books/', { k1: 1, k2: 2, k2: 3 },
                function (data) {
                    console.log(data);
                }
            )
        })
    })

</script>
```

```
GET /books/?k1=1&k2=3 HTTP/1.1
Host: 127.0.0.1:5000
Connection: keep-alive
Accept: */*

X-Requested-With: XMLHttpRequest
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
Referer: http://127.0.0.1:5000/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN
```

访问蓝图中的list()视图函数，返回Json数据。

```
{
  "count": 3,
  "results": [
    [
      1,
      "python"
    ],
    [
      2,
      "java"
    ],
    [
      3,
      "javascript"
```

```
    ]
  ]
}

```

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

这是HTTP请求跨域访问产生的。

请求头如下

```
GET /books/?k1=1&k2=3 HTTP/1.1
Host: localhost:5000
Connection: keep-alive
Accept: */*
Origin: http://127.0.0.1:5000
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
DNT: 1
Referer: http://127.0.0.1:5000/
Accept-Encoding: gzip, deflate
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。

```python
from flask import Blueprint, jsonify, render_template, request

books = Blueprint('book', __name__)

@books.route('/')
def list():
    books = {
        'count':3,
        'results':[
            (1, 'python'),
            (2, 'java'),
            (3, 'javascript')
        ]
    }
    ret = jsonify(books)
    ret.headers['Access-Control-Allow-Origin'] = '*'
    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

```
<!DOCTYPE html>
<html lang="zh-CN">

<head>
    <meta charset="UTF-8">
    <script src="{{ url_for('static', filename='js/jquery-3.6.0.min.js') }}"></script>
    <title>马哥教育jQuery测试</title>
</head>

<body>
    <div id="root">
        <h2>测试事件</h2>
        <div><button class="testclick">点击事件</button></div>
        <button class="ajaxget">Ajax GET请求</button>
        <button class="ajaxpost">Ajax POST请求</button>
        <div class="console" style="border: 1px solid #000;"></div>
    </div>
</body>

</html>
<script>
    $(function () {
        $('button.testclick').click(function (event) {
            console.log(event.target);
            var c = $('<div>hello</div>'); // 包装成jquery对象
            c.css({ 'background-color': '#F0F0F0', 'margin': '10px' })
            $('.console').append(c); // 使用jquery对象
        });

        // AJAX GET请求
        $('button.ajaxget').click(function (event) {
            // http://127.0.0.1:5000/books/
            $.get('http://localhost:5000/books/', { k1: 1, k2: 2, k2: 3 },
                function (data) {
                    console.log(data);
                }
            )
        })

        // AJAX POST请求
        $('button.ajaxpost').click(function (event) {
            // http://127.0.0.1:5000/books/
            $.post('http://localhost:5000/books/?k1=v1&k2=v2&k2=v3', { p1: 100, p2: 200 },
                function (data, status) {
                    console.log(data, status);
                }
            )
        })
    })

</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'] = '*'`

```python
from flask import Blueprint, jsonify, render_template, request

books = Blueprint('book', __name__)

@books.route('/', methods=['get', 'post']) # 默认只支持GET、HEAD，不支持POST
def list():
    books = {
        'count':3,
        'results':[
            (1, 'python'),
            (2, 'java'),
            (3, 'javascript')
        ]
    }
    # print(*dir(request), sep='\n') # request用小写的
    # print('~' * 30)
    print(request.url)
    print(request.full_path)
    print(request.method)
    # print(request.cookies)
    print(request.content_type)
    print(request.args)
    print(request.query_string)
    print(request.form)
    # print(request.is_json, request.get_json, request.json)
    print('~' * 30)
    ret = jsonify(books)
    ret.headers['Access-Control-Allow-Origin'] = '*'
    return ret
```

```
http://localhost:5000/books/?k1=v1&k2=v2&k2=v3
/books/?k1=v1&k2=v2&k2=v3
POST
application/x-www-form-urlencoded; charset=UTF-8
ImmutableMultiDict([('k1', 'v1'), ('k2', 'v2'), ('k2', 'v3')])
b'k1=v1&k2=v2&k2=v3'
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。

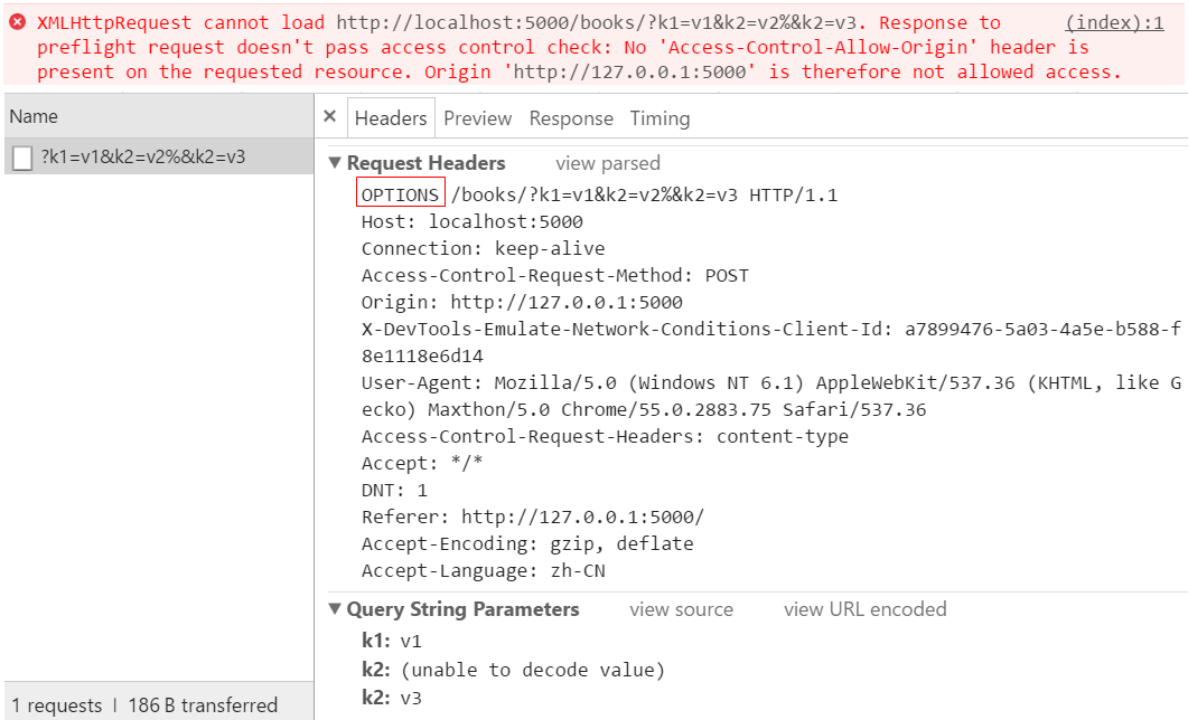
2、预检请求

POST请求提交JSON数据

```html
<!DOCTYPE html>
<html lang="en">

```

```
<head>
    <meta charset="UTF-8">
    <title>马哥教育</title>
    <script src={{ url_for('static', filename='js/jquery.js') }}></script>
</head>

<body>
    <h2 id="root">测试jQuery</h2>
    <button class="testclick">事件响应</button>
    <button class="ajaxget">Ajax GET请求</button>
    <button class="ajaxpost">Ajax POST请求</button>
    <button class="ajaxpostjson">Ajax POST请求提交Json数据</button>
    <div class="content" style="border:1px solid #000"></div>
    <script>
        $(function () {
            // 省略其它代码

            // AJAX POST 请求
            $('button.ajaxpost').click(function (event) {
                $.post('http://localhost:5000/books/?k1=v1&k2=v2%&k2=v3',
                    { p1: 100, p2: 200 },
                    /*http://127.0.0.1:5000/books/?k1=v1&k2=v2%&k2=v3如果同域可以不写*/
                    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);
                    })
            });

            // AJAX POST with JSON
            $('button.ajaxpostjson').click(function (event) {
                $.ajax({
                    type:'POST',
                    url:'http://localhost:5000/books/?k1=v1&k2=v2%&k2=v3',
                    contentType:"application/json",
                    data:{ p1: 100, p2: 200 },
                    /*http://127.0.0.1:5000/books/?k1=v1&k2=v2%&k2=v3如果同域可以不写*/
                    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);
                    }
                });
            });
        })
    </script>
</body>

</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



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

```
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
```

```
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 (KH
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

```python
from flask import Blueprint, jsonify, render_template, request

books = Blueprint('book', __name__)

@books.route('/', methods=['get', 'post', 'options']) # 默认只支持GET、HEAD，不支持POST
def list():
    books = {
        'count':3,
        'results':[
            (1, 'python'),
            (2, 'java'),
            (3, 'javascript')
        ]
    }
```

```python
    # print(*dir(request), sep='\n') # request用小写的
    # print('~' * 30)
    print(request.url)
    print(request.full_path)
    print(request.method)
    # print(request.cookies)
    print(request.content_type)
    print(request.args)
    print(request.query_string)
    print(request.form)
    if request.is_json:
        print(request.data)
        print(request.json) # 转换json失败

    print('~' * 30)
    ret = jsonify(books)
    ret.headers['Access-Control-Allow-Origin'] = '*'
    ret.headers['Access-Control-Allow-Headers'] = 'content-type'
    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。

```html
<!DOCTYPE html>
<html lang="zh-CN">

<head>
    <meta charset="UTF-8">
    <script src="{{ url_for('static', filename='js/jquery-3.6.0.min.js') }}"></script>
    <title>马哥教育jQuery测试</title>
</head>

<body>
    <div id="root">
        <h2>测试事件</h2>
        <div><button class="testclick">点击事件</button></div>
        <button class="ajaxget">Ajax GET请求</button>
        <button class="ajaxpost">Ajax POST请求</button>
        <button class="ajaxpostjson">Ajax POST请求提交Json数据</button>
        <div class="console" style="border: 1px solid #000;"></div>
    </div>
</body>

</html>
<script>
    $(function () {
        $('button.testclick').click(function (event) {
            console.log(event.target);
            var c = $('<div>hello</div>'); // 包装成jquery对象
            c.css({ 'background-color': '#F0F0F0', 'margin': '10px' })
```

```
            $('.console').append(c); // 使用jquery对象
        });

        // AJAX GET请求
        $('button.ajaxget').click(function (event) {
            // http://127.0.0.1:5000/books/
            $.get('http://localhost:5000/books/', { k1: 1, k2: 2, k2: 3 },
                function (data) {
                    console.log(data);
                }
            )
        })

        // AJAX POST请求
        $('button.ajaxpost').click(function (event) {
            // http://127.0.0.1:5000/books/
            $.post('http://localhost:5000/books/?k1=v1&k2=v2&k2=v3', { p1: 100, p2: 200 },
                function (data, status) {
                    console.log(data, status);
                }
            )
        })

        // AJAX POST with JSON
        $('button.ajaxpostjson').click(function (event) {
            $.ajax({
                type: 'POST',
                url: 'http://localhost:5000/books/?k1=v1&k2=v2&k2=v3',
                contentType: "application/json",
                data: JSON.stringify({ p1: 100, p2: 200 }), // 转换为Json
                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);
                }
            });
        });
    })

</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。