[AJAX POST请求中参数以form data和request payload形式在servlet中的获取方式](http://blog.csdn.net/mhmyqn/article/details/25561535)

分类： [java](http://blog.csdn.net/mhmyqn/article/category/935732)2014-05-11 21:20 5957人阅读 [评论](http://blog.csdn.net/mhmyqn/article/details/25561535#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/mhmyqn/article/details/25561535#report)

[HTTP AJAX POST请求](http://www.csdn.net/tag/HTTP%20AJAX%20POST%e8%af%b7%e6%b1%82)[request payload](http://www.csdn.net/tag/request%20payload)[form data](http://www.csdn.net/tag/form%20data)[Content Type](http://www.csdn.net/tag/Content%20Type)

HTTP请求中，

如果是get请求，那么表单参数以name=value&name1=value1的形式附到url的后面，

如果是post请求，那么表单参数是在请求体中，也是以name=value&name1=value1的形式在请求体中。

通过chrome的开发者工具可以看到如下（这里是可读的形式，不是真正的HTTP请求协议的请求格式）：

get请求：

**[plain]** [view plaincopy](http://blog.csdn.net/mhmyqn/article/details/25561535)[在CODE上查看代码片](https://code.csdn.net/snippets/339675)

1. RequestURL:http://127.0.0.1:8080/test/test.do?name=mikan&address=street
2. Request Method:GET
3. Status Code:200 OK
5. Request Headers
6. Accept:text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8
7. Accept-Encoding:gzip,deflate,sdch
8. Accept-Language:zh-CN,zh;q=0.8,en;q=0.6
9. AlexaToolbar-ALX\_NS\_PH:AlexaToolbar/alxg-3.2
10. Connection:keep-alive
11. Cookie:JSESSIONID=74AC93F9F572980B6FC10474CD8EDD8D
12. Host:127.0.0.1:8080
13. Referer:http://127.0.0.1:8080/test/index.jsp
14. User-Agent:Mozilla/5.0 (Windows NT 6.1)AppleWebKit/537.36 (KHTML, like Gecko) Chrome/33.0.1750.149 Safari/537.36
16. Query String Parameters
17. name:mikan
18. address:street
20. Response Headers
21. Content-Length:2
22. Date:Sun, 11 May 2014 10:42:38 GMT
23. Server:Apache-Coyote/1.1

Post请求：

1. RequestURL:http://127.0.0.1:8080/test/test.do
2. Request Method:POST
3. Status Code:200 OK
5. Request Headers
6. Accept:text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8
7. Accept-Encoding:gzip,deflate,sdch
8. Accept-Language:zh-CN,zh;q=0.8,en;q=0.6
9. AlexaToolbar-ALX\_NS\_PH:AlexaToolbar/alxg-3.2
10. Cache-Control:max-age=0
11. Connection:keep-alive
12. Content-Length:25
13. Content-Type:application/x-www-form-urlencoded
14. Cookie:JSESSIONID=74AC93F9F572980B6FC10474CD8EDD8D
15. Host:127.0.0.1:8080
16. Origin:http://127.0.0.1:8080
17. Referer:http://127.0.0.1:8080/test/index.jsp
18. User-Agent:Mozilla/5.0 (Windows NT 6.1)AppleWebKit/537.36 (KHTML, like Gecko) Chrome/33.0.1750.149 Safari/537.36
20. Form Data
21. name:mikan
22. address:street
24. Response Headers
25. Content-Length:2
26. Date:Sun, 11 May 2014 11:05:33 GMT
27. Server:Apache-Coyote/1.1

这里要注意post请求的Content-Type为application/x-www-form-urlencoded，参数是在请求体中，即上面请求中的Form Data。

 在servlet中，可以通过request.getParameter(name)的形式来获取表单参数。

 而如果使用原生AJAX POST请求的话：

1. **function** getXMLHttpRequest() {
2. **var** xhr;
3. **if**(window.ActiveXObject) {
4. xhr= **new** ActiveXObject("Microsoft.XMLHTTP");
5. }**else** **if** (window.XMLHttpRequest) {
6. xhr= **new** XMLHttpRequest();
7. }**else** {
8. xhr= **null**;
9. }
10. **return** xhr;
11. }
13. **function** save() {
14. **var** xhr = getXMLHttpRequest();
15. xhr.open("post","http://127.0.0.1:8080/test/test.do");
16. **var** data = "name=mikan&address=street...";
17. xhr.send(data);
18. xhr.onreadystatechange= **function**() {
19. **if**(xhr.readyState == 4 && xhr.status == 200) {
20. alert("returned:"+ xhr.responseText);
21. }
22. };
23. }

通过chrome的开发者工具看到请求头如下：

1. RequestURL:http://127.0.0.1:8080/test/test.do
2. Request Method:POST
3. Status Code:200 OK
5. Request Headers
6. Accept:\*/\*
7. Accept-Encoding:gzip,deflate,sdch
8. Accept-Language:zh-CN,zh;q=0.8,en;q=0.6
9. AlexaToolbar-ALX\_NS\_PH:AlexaToolbar/alxg-3.2
10. Connection:keep-alive
11. Content-Length:28
12. Content-Type:text/plain;charset=UTF-8
13. Cookie:JSESSIONID=C40C7823648E952E7C6F7D2E687A0A89
14. Host:127.0.0.1:8080
15. Origin:http://127.0.0.1:8080
16. Referer:http://127.0.0.1:8080/test/index.jsp
17. User-Agent:Mozilla/5.0 (Windows NT 6.1)AppleWebKit/537.36 (KHTML, like Gecko) Chrome/33.0.1750.149 Safari/537.36
19. Request Payload
20. name=mikan&address=street
22. Response Headers
23. Content-Length:2
24. Date:Sun, 11 May 2014 11:49:23 GMT
25. Server:Apache-Coyote/1.1

注意请求的Content-Type为text/plain;charset=UTF-8，而请求表单参数在RequestPayload中。

 那么servlet中通过request.getParameter(name)却是空。为什么呢？而这样的参数又该怎么样获取呢？

为了搞明白这个问题，查了些资料，也看了Tomcat7.0.53关于请求参数处理的源码，终于搞明白了是怎么回事。

HTTP POST表单请求提交时，使用的Content-Type是application/x-www-form-urlencoded，而使用原生AJAX的POST请求如果不指定请求头RequestHeader，默认使用的Content-Type是text/plain;charset=UTF-8。

 由于Tomcat对于Content-Type multipart/form-data（文件上传）和application/x-www-form-urlencoded（POST请求）做了“特殊处理”。下面来看看相关的处理代码。

Tomcat的HttpServletRequest类的实现类为org.apache.catalina.connector.Request（实际上是org.apache.coyote.Request），而它对处理请求参数的方法为protected void parseParameters()，这个方法中对Content-Type multipart/form-data（文件上传）和application/x-www-form-urlencoded（POST请求）的处理代码如下：

**[java]** [view plaincopy](http://blog.csdn.net/mhmyqn/article/details/25561535)[在CODE上查看代码片](https://code.csdn.net/snippets/339675)

1. protectedvoid parseParameters() {
2. //省略部分代码......
3. parameters.handleQueryParameters();// 这里是处理url中的参数
4. //省略部分代码......
5. **if** ("multipart/form-data".equals(contentType)) { // 这里是处理文件上传请求
6. parseParts();
7. success = **true**;
8. **return**;
9. }
11. **if**(!("application/x-www-form-urlencoded".equals(contentType))) {// 这里如果是非POST请求直接返回，不再进行处理
12. success = **true**;
13. **return**;
14. }
15. //下面的代码才是处理POST请求参数
16. //省略部分代码......
17. **try** {
18. **if** (readPostBody(formData, len)!= len) { // 读取请求体数据
19. **return**;
20. }
21. } **catch** (IOException e) {
22. // Client disconnect
23. **if**(context.getLogger().isDebugEnabled()) {
24. context.getLogger().debug(
25. sm.getString("coyoteRequest.parseParameters"),e);
26. }
27. **return**;
28. }
29. parameters.processParameters(formData, 0, len); // 处理POST请求参数，把它放到requestparameter map中（即request.getParameterMap获取到的Map，request.getParameter(name)也是从这个Map中获取的）
30. // 省略部分代码......
31. }
33. **protected** **int** readPostBody(**byte** body[], **int** len)
34. **throws** IOException {
36. **int** offset = 0;
37. **do** {
38. **int** inputLen = getStream().read(body, offset, len - offset);
39. **if** (inputLen <= 0) {
40. **return** offset;
41. }
42. offset += inputLen;
43. } **while** ((len - offset) > 0);
44. **return** len;
45. }

从上面代码可以看出，POST请求是不会读取请求体数据和进行相应的参数处理的，即把表单数据解析后，放到request parameter map中（即request.getParameterMap获取到的Map，request.getParameter(name)也是从这个Map中获取的）。所以通过request.getParameter(name)是获取不到的。

 那么这样提交的参数我们该怎么获取呢？

当然是使用最原始的方式，读取输入流来获取了，如下所示：

**[java]** [view plaincopy](http://blog.csdn.net/mhmyqn/article/details/25561535)[在CODE上查看代码片](https://code.csdn.net/snippets/339675)

1. privateString getRequestPayload(HttpServletRequest req) {
2. StringBuildersb = **new** StringBuilder();
3. **try**(BufferedReaderreader = req.getReader();) {
4. **char**[]buff = **new** **char**[1024];
5. intlen;
6. **while**((len = reader.read(buff)) != -1) {
7. sb.append(buff,0, len);
8. }
9. }**catch** (IOException e) {
10. e.printStackTrace();
11. }
12. returnsb.toString();
13. }

当然，设置了application/x-www-form-urlencoded的POST请求也可以通过这种方式来获取。

 所以，在使用原生AJAX POST请求时，需要明确设置Request Header，即：

1. xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");

另外，如果使用jquery，我使用1.11.0这个版本来测试，$.ajax post请求是不需要明确设置这个请求头的，其他版本的本人没有亲自测试过。相信在1.11.0之后的版本也是不需要设置的。不过之前有的就不一定了。这个没有测试过。

[java 获取 reuqest payload 里的数据](http://183615215-qq-com.iteye.com/blog/1846820)

Java代码1

1. **public** **static** String getBody(HttpServletRequest request) **throws** IOException {
2. String body = **null**;
3. StringBuilder stringBuilder = **new** StringBuilder();
4. BufferedReader bufferedReader = **null**;
5. **try** {
6. InputStream inputStream = request.getInputStream();
7. **if** (inputStream != **null**) {
8. bufferedReader = **new** BufferedReader(**new** InputStreamReader(inputStream));
9. **char**[] charBuffer = **new** **char**[128];
10. **int** bytesRead = -1;
11. **while** ((bytesRead = bufferedReader.read(charBuffer)) > 0) {
12. stringBuilder.append(charBuffer, 0, bytesRead);
13. }
14. } **else** {
15. stringBuilder.append("");
16. }
17. } **catch** (IOException ex) {
18. **throw** ex;
19. } **finally** {
20. **if** (bufferedReader != **null**) {
21. **try** {
22. bufferedReader.close();
23. } **catch** (IOException ex) {
24. **throw** ex;
25. }
26. }
27. }
29. body = stringBuilder.toString();
30. **return** body;
31. }

java代码2(测试没通过)

1. private String getRequestPayload(HttpServletRequest req) {
2. StringBuilder sb = **new** StringBuilder();
3. **try**(BufferedReader reader = req.getReader();) {
4. **char**[] buff = **new** **char**[1024];
5. int len;
6. **while**((len = reader.read(buff)) != -1) {
7. sb.append(buff,0, len);
8. }
9. }**catch** (IOException e) {
10. e.printStackTrace();
11. }
12. return sb.toString();
13. }