常见的类型有两种：  
1、form类型，即：application/x-www-form-urlencoded  
2、json类型，即：application/json  
一张图片说明：



如果是通过页面表单方式提交，那就是"application/x-www-form-urlencoded"；如果是json（要反序列化成字符串），那就是"application/json"。

对于"application/x-www-form-urlencoded"，其参数组织形式，是键值对，类似于get方式的参数形式：name=张三&sex=男&tel=5354169，

而application/json大家都知道，是这样的：

{ name:“张三”, sex:“男”, tel:“5354169”,}

所以，如果你传递的参数是json，而你声明的header是"application/x-www-form-urlencoded"，或者没有声明这个”Content-Type”（似乎默认就是”application/x-www-form-urlencoded”），则服务器拿不到提交的数据。数据是传过去了，但读不出来。

反之应该也一样。

# PHP content-type为"application/json"的post过来的数据$\_POST接受不到的问题

ajax默认是以application/x-www-form-urlencoded方式提交。也就是常见的表单提交方式。在PHP中使用$\_POST方式可以轻松获取。

但如果将ajax的请求头强制指定为 application/json，那么你的 $\_POST 就接受不到了。必须使用$GLOBALS['HTTP\_RAW\_POST\_DATA']或者file\_get\_contents('php://input')取出来，然后再json\_decode就行了。

比如：

**json\_decode(file\_get\_contents('php://input'), true);**

如fetch、axios默认的请求头就是application/json，所以要注意一下。

还有一些的细节需要了解一下

1、后端必须允许前端定义Content-Type之类的头请求。

header('Access-Control-Allow-Headers:x-requested-with,content-type');

**定义和用法**

enctype 属性规定在**发送到服务器之前**应该如何**对表单数据**进行**编码**。  
默认地，表单数据会编码为 "application/x-www-form-urlencoded"。就是说，在发送到服务器之前，所有字符都会进行编码（空格转换为 "+" 加号，特殊符号转换为 ASCII HEX 值）。

**enctype属性值**

|  |  |
| --- | --- |
| **值** | **描述** |
| application/x-www-form-urlencoded | 在发送前编码所有字符（默认） |
| multipart/form-data | 不对字符编码。  在使用包含文件上传控件的表单时，必须使用该值。 |
| text/plain | 空格转换为 "+" 加号，但不对特殊字符编码。 |

**application/x-www-form-urlencoded**

这应该是最常见的 POST 提交数据的方式了。浏览器的原生 <form> 表单，如果**不设置 enctype 属性**，那么最终就会以 application/x-www-form-urlencoded 方式提交数据。

<form action="form\_action.asp" enctype="text/plain">

<p>First name: <input type="text" name="fname" /></p>

<p>Last name: <input type="text" name="lname" /></p>

<input type="submit" value="Submit" />

</form>

此时Form提交的请求数据，抓包时看到的请求会是这样的内容（无关的请求头在本文中都省略掉了）：

**BASH**POST http://www.example.com HTTP/1.1  
   Content-Type: application/x-www-form-urlencoded;charset=utf-8  
   title=test&sub%5B%5D=1&sub%5B%5D=2&sub%5B%5D=3

首先，Content-Type 被指定为 application/x-www-form-urlencoded；其次，提交的数据按照**key1=val1&key2=val2 的方式进行编码**，key 和 val 都进行了 URL 转码。  
大部分服务端语言都对这种方式有很好的支持。例如 PHP 中，$\_POST['title'] 可以获取到 title 的值，$\_POST['sub'] 可以得到 sub 数组。

很多时候，我们用 Ajax 提交数据时，也是使用这种方式。  
例如 [JQuery](http://jquery.com/) 和 [QWrap](http://www.qwrap.com/) 的 Ajax，Content-Type **默认值**都是「application/**x-www-form-urlencoded**;charset=utf-8」。

**multipart/form-data**

这又是一个常见的 POST 数据提交的方式。我们使用表单**上传文件**时，必须让 <form> 表单的enctype 等于 multipart/form-data。直接来看一个请求示例：

**BASH**POST http://www.example.com HTTP/1.1

Content-Type:multipart/form-data; boundary=----WebKitFormBoundaryrGKCBY7qhFd3TrwA

------WebKitFormBoundaryrGKCBY7qhFd3TrwA

Content-Disposition: form-data; name="text"

title

------WebKitFormBoundaryrGKCBY7qhFd3TrwA

Content-Disposition: form-data; name="file"; filename="chrome.png"

Content-Type: image/png

PNG ... content of chrome.png ...

------WebKitFormBoundaryrGKCBY7qhFd3TrwA--

这个例子稍微复杂点。首先生成了一个 boundary 用于分割不同的字段，为了避免与正文内容重复，boundary 很长很复杂。然后 Content-Type 里指明了数据是以 multipart/form-data 来编码，本次请求的 boundary 是什么内容。消息主体里按照字段个数又分为多个结构类似的部分，每部分都是以 --boundary 开始，紧接着是内容描述信息，然后是回车，最后是字段具体内容（文本或二进制）。如果传输的是文件，还要包含文件名和文件类型信息。消息主体最后以 --boundary-- 标示结束。  
关于 multipart/form-data 的详细定义，请前往 [rfc1867](http://www.ietf.org/rfc/rfc1867.txt) 查看。

这种方式一般用来上传文件，各大服务端语言对它也有着良好的支持。

上面提到的这两种 POST 数据的方式，都是浏览器原生支持的，而且现阶段标准中原生 <form> 表单也[只支持这两种方式](http://www.w3.org/TR/html401/interact/forms.html" \l "h-17.13.4)（通过 <form> 元素的 enctype 属性指定，默认为 application/x-www-form-urlencoded。其实 enctype 还支持 text/plain，不过用得非常少）。

随着越来越多的 Web 站点，尤其是 WebApp，全部使用 Ajax 进行数据交互之后，我们完全可以定义新的数据提交方式，给开发带来更多便利。

**application/json**

application/json 这个 Content-Type 作为响应头大家肯定不陌生。实际上，现在越来越多的人把它作为请求头，用来告诉服务端消息主体是序列化后的 JSON 字符串。由于 JSON 规范的流行，除了低版本 IE 之外的各大浏览器都原生支持 JSON.stringify，服务端语言也都有处理 JSON 的函数，使用 JSON 不会遇上什么麻烦。

JSON 格式支持比键值对复杂得多的结构化数据，这一点也很有用。记得我几年前做一个项目时，需要提交的数据层次非常深，我就是把数据 JSON 序列化之后来提交的。不过当时我是把 JSON 字符串作为 val，仍然放在键值对里，以 x-www-form-urlencoded 方式提交。

Google 的 [AngularJS](http://angularjs.org/) 中的 Ajax 功能，默认就是提交 JSON 字符串。例如下面这段代码：

JSvar data = {'title':'test', 'sub' : [1,2,3]};

$http.post(url, data).success(function(result) {

...

});

最终发送的请求是：

BASHPOST http://www.example.com HTTP/1.1

Content-Type: application/json;charset=utf-8

{"title":"test","sub":[1,2,3]}

这种方案，可以方便的提交复杂的结构化数据，特别适合 RESTful 的接口。各大抓包工具如 Chrome 自带的开发者工具、Firebug、Fiddler，都会以树形结构展示 JSON 数据，非常友好。但也有些服务端语言还没有支持这种方式，例如 php 就无法通过 $\_POST 对象从上面的请求中获得内容。这时候，需要自己动手处理下：在请求头中 Content-Type 为 application/json 时，从 php://input 里获得原始输入流，再 json\_decode 成对象。一些 php 框架已经开始这么做了。

当然 AngularJS 也可以配置为使用 x-www-form-urlencoded 方式提交数据。如有需要，可以参考[这篇文章](http://victorblog.com/2012/12/20/make-angularjs-http-service-behave-like-jquery-ajax/)。