<https://blog.csdn.net/qq_24122593/article/details/52858177>

**一 原理区别**

    一般在浏览器中输入网址访问资源都是通过GET方式；在FORM提交中，可以通过Method指定提交方式为GET或者POST，默认为GET提交

Http定义了与服务器交互的不同方法，最基本的方法有4种，分别是**GET，POST，PUT，DELETE**

URL全称是资源描述符，我们可以这样认为：一个URL地址，它用于描述一个网络上的资源，而HTTP中的GET，POST，PUT，DELETE就对应着对这个资源的查 ，改 ，增 ，删 4个操作。到这里，大家应该有个大概的了解了，GET一般用于获取/查询 资源信息，而POST一般用于更新 资源信息(**个人认为这是GET和POST的本质区别，也是协议设计者的本意，其它区别都是具体表现形式的差异** )。

　　根据HTTP规范，GET用于信息获取，而且应该是安全的和幂等的 。

　　1.所谓安全的意味着该操作用于获取信息而非修改信息。换句话说，GET请求一般不应产生副作用。就是说，它仅仅是获取资源信息，就像数据库查询一样，不会修改，增加数据，不会影响资源的状态。

　　\* 注意：这里安全的含义仅仅是指是非修改信息。

　　2.幂等的意味着对同一URL的多个请求应该返回同样的结果。这里我再解释一下**幂等**这个概念：

**幂等**（idempotent、idempotence）是一个数学或计算机学概念，常见于抽象代数中。  
幂等有以下几种定义：  
对于单目运算，如果一个运算对于在范围内的所有的一个数多次进行该运算所得的结果和进行一次该运算所得的结果是一样的，那么我们就称该运算是幂等的。 比如绝对值运算就是一个例子，在实数集中，有abs(a) = abs(abs(a)) 。  
对于双目运算，则要求当参与运算的两个值是等值的情况下，如果满足运算结果与参与运算的两个值相等，则称该运算幂等，如求两个数的最大值的函数，有在实数集中幂等，即max(x,x) = x 。

看完上述解释后，应该可以理解GET幂等的含义了。

　　但在实际应用中，以上2条规定并没有这么严格。引用别人文章的例子：比如，新闻站点的头版不断更新。虽然第二次请求会返回不同的一批新闻，该操作仍然被认为是安全的和幂等的，因为它总是返回当前的新闻。从根本上说，如果目标是当用户打开一个链接时，他可以确信从自身的角度来看没有改变资源即可。

　　根据HTTP规范，POST表示可能修改变服务器上的资源的请求 。继续引用上面的例子：还是新闻以网站为例，读者对新闻发表自己的评论应该通过POST实现，因为在评论提交后站点的资源已经不同了，或者说资源被修改了。

　　上面大概说了一下HTTP规范中，GET和POST的一些原理性的问题。但在实际的做的时候，很多人却没有按照HTTP规范去做，导致这个问题的原因有很多，比如说：

　　1.很多人贪方便，更新资源时用了GET，因为用POST必须要到FORM（表单），这样会麻烦一点。

　　2.对资源的增，删，改，查操作，其实都可以通过GET/POST完成，不需要用到PUT和DELETE。

　　3.另外一个是，早期的但是Web MVC框架设计者们并没有有意识地将URL当作抽象的资源来看待和设计 。还有一个较为严重的问题是传统的Web MVC框架基本上都只支持GET和POST两种HTTP方法，而不支持PUT和DELETE方法。

　　\* 简单解释一下MVC：MVC本来是存在于Desktop程序中的，M是指数据模型，V是指用户界面，C则是控制器。使用MVC的目的是将M和V的实现代码分离，从而使同一个程序可以使用不同的表现形式。

　　以上3点典型地描述了老一套的风格（没有严格遵守HTTP规范），随着架构的发展，现在出现REST(Representational State Transfer)，一套支持HTTP规范的新风格，这里不多说了，可以参考《RESTful Web Services》。

**二 表现形式区别**

   搞清了两者的原理区别，我们再来看一下他们实际应用中的区别：

    为了理解两者在传输过程中的不同，我们先看一下HTTP协议的格式：

    HTTP请求：

<request line>          //http请求行

<headers>               //http请求消息报头

<blank line>             // 回 车 换行

[<request-body>]     //http请求正文

在HTTP请求中，第一行必须是一个请求行（request line），用来说明请求类型、要访问的资源以及使用的HTTP版本。紧接着是一个首部（header）小节，用来说明服务器要使用的附加信息。在首部之后是一个空行，再此之后可以添加任意的其他数据[称之为主体（body）]。

GET与POST方法实例：  
GET /books/?sex=man&name=Professional HTTP/1.1  
Host: www.wrox.com  
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.6)  
Gecko/20050225 Firefox/1.0.1  
Connection: Keep-Alive  
  
POST / HTTP/1.1  
Host: www.wrox.com  
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.6)  
Gecko/20050225 Firefox/1.0.1  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 40  
Connection: Keep-Alive  
（----此处空一行----）  
name=Professional%20Ajax&publisher=Wiley

有了以上对HTTP请求的了解和示例，我们再来看两种提交方式的区别：

   （1）GET提交，请求的数据会附在URL之后（就是把数据放置在HTTP协议头中），以?分割URL和传输数据，多个参数用&连接；例 如：login.action?name=hyddd&password=idontknow&verify=%E4%BD%A0 %E5%A5%BD。如果数据是英文字母/数字，原样发送，如果是空格，转换为+，如果是中文/其他字符，则直接把字符串用BASE64加密，得出如： %E4%BD%A0%E5%A5%BD，其中％XX中的XX为该符号以16进制表示的ASCII。

　　POST提交：把提交的数据放置在是HTTP包的包体中。上文示例中红色字体标明的就是实际的传输数据

     因此，GET提交的数据会在地址栏中显示出来，而POST提交，地址栏不会改变

(2)传输数据的大小：首先声明：HTTP协议没有对传输的数据大小进行限制，HTTP协议规范也没有对URL长度进行限制。

而在实际开发中存在的限制主要有：

GET:特定浏览器和服务器对URL长度有限制，例如 IE对URL长度的限制是2083字节(2K+35)。对于其他浏览器，如Netscape、FireFox等，理论上没有长度限制，其限制取决于操作系 统的支持。

因此对于GET提交时，传输数据就会受到URL长度的 限制。

POST:由于不是通过URL传值，理论上数据不受 限。但实际各个WEB服务器会规定对post提交数据大小进行限制，Apache、IIS6都有各自的配置。

(3)安全性：

POST的安全性要比GET的安全性高。注意：这里所说的安全性和上面GET提到的“安全”不是同个概念。上面 “安全”的含义仅仅是不作数据修 改，而这 里安全的含义是真正的Security的含义，比如：通过GET提交数据，用户名和密码将明文出现在URL上，因为(1)登录页面有可能被浏览器缓存， (2)其他人查看浏览器的历史纪录，那么别人就可以拿到你的账号和密码了，除此之外，使用GET提交数据还可能会造成Cross-site request forgery攻击

（4）Http get,post,soap协议都是在http上运行的  
1）get：请求参数是作为一个key/value对的序列（查询字符串）附加到URL上的  
查询字符串的长度受到web浏览器和web服务器的限制（如IE最多支持2048个字符），不适合传输大型数据集同时，它很不安全  
2）post：请求参数是在http标题的一个不同部分（名为entity body）传输的，这一部分用来传输表单信息，因此必须将Content-type设置为:application/x-www-form- urlencoded。post设计用来支持web窗体上的用户字段，其参数也是作为key/value对传输。  
但是：它不支持复杂数据类型，因为post没有定义传输数据结构的语义和规则。  
3）soap：是http post的一个专用版本，遵循一种特殊的xml消息格式  
Content-type设置为: text/xml   任何数据都可以xml化

**三 HTTP响应**  
1．HTTP响应格式：

<status line>          //http响应状态行  
<headers>             //http响应消息报头  
<blank line>            //回车换行  
[<response-body>]  //http响应正文  
  
在响应中唯一真正的区别在于第一行中用状态信息代替了请求信息。状态行（status line）通过提供一个状态码来说明所请求的资源情况。   
  
HTTP响应实例：  
  
HTTP/1.1 200 OK  
Date: Sat, 31 Dec 2005 23:59:59 GMT  
Content-Type: text/html;charset=ISO-8859-1 Content-Length: 122

＜html＞  
＜head＞  
＜title＞Wrox Homepage＜/title＞  
＜/head＞  
＜body＞  
＜!-- body goes here --＞  
＜/body＞  
＜/html＞  
2．最常用的状态码有：  
  
◆200 (OK): 找到了该资源，并且一切正常。  
◆304 (NOT MODIFIED): 该资源在上次请求之后没有任何修改。这通常用于浏览器的缓存机制。  
◆401 (UNAUTHORIZED): 客户端无权访问该资源。这通常会使得浏览器要求用户输入用户名和密码，以登录到服务器。  
◆403 (FORBIDDEN): 客户端未能获得授权。这通常是在401之后输入了不正确的用户名或密码。  
◆404 (NOT FOUND): 在指定的位置不存在所申请的资源。

**四 完整示例：**

例子：

**HTTP GET**   
  
发送  
  
GET /DEMOWebServices2.8/Service.asmx/CancelOrder?UserID=string&PWD=string&OrderConfirmation=string HTTP/1.1  
Host: api.efxnow.com  
  
回复  
  
HTTP/1.1 200 OK  
Content-Type: text/xml; charset=utf-8  
Content-Length: length  
  
<?xml version="1.0" encoding="utf-8"?>  
<objPlaceOrderResponse xmlns="https://api.efxnow.com/webservices2.3">  
<Success>boolean</Success>  
<ErrorDescription>string</ErrorDescription>  
<ErrorNumber>int</ErrorNumber>  
<CustomerOrderReference>long</CustomerOrderReference>  
<OrderConfirmation>string</OrderConfirmation>  
<CustomerDealRef>string</CustomerDealRef>  
</objPlaceOrderResponse>  
**HTTP POST**   
  
发送  
  
POST /DEMOWebServices2.8/Service.asmx/CancelOrder HTTP/1.1  
Host: api.efxnow.com  
Content-Type: application/x-www-form-urlencoded  
Content-Length: length  
  
UserID=string&PWD=string&OrderConfirmation=string  
  
回复  
  
HTTP/1.1 200 OK  
Content-Type: text/xml; charset=utf-8  
Content-Length: length  
  
<?xml version="1.0" encoding="utf-8"?>  
<objPlaceOrderResponse xmlns="https://api.efxnow.com/webservices2.3">  
<Success>boolean</Success>  
<ErrorDescription>string</ErrorDescription>  
<ErrorNumber>int</ErrorNumber>  
<CustomerOrderReference>long</CustomerOrderReference>  
<OrderConfirmation>string</OrderConfirmation>  
<CustomerDealRef>string</CustomerDealRef>  
</objPlaceOrderResponse>  
  
  
  
**SOAP 1.2**   
  
发送  
  
POST /DEMOWebServices2.8/Service.asmx HTTP/1.1  
Host: api.efxnow.com  
Content-Type: application/soap+xml; charset=utf-8  
Content-Length: length  
  
<?xml version="1.0" encoding="utf-8"?>  
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">  
<soap12:Body>  
<CancelOrder xmlns="https://api.efxnow.com/webservices2.3">  
<UserID>string</UserID>  
<PWD>string</PWD>  
<OrderConfirmation>string</OrderConfirmation>  
</CancelOrder>  
</soap12:Body>  
</soap12:Envelope>  
  
回复  
  
HTTP/1.1 200 OK  
Content-Type: application/soap+xml; charset=utf-8  
Content-Length: length  
  
<?xml version="1.0" encoding="utf-8"?>  
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">  
<soap12:Body>  
<CancelOrderResponse xmlns="https://api.efxnow.com/webservices2.3">  
<CancelOrderResult>  
<Success>boolean</Success>  
<ErrorDescription>string</ErrorDescription>  
<ErrorNumber>int</ErrorNumber>  
<CustomerOrderReference>long</CustomerOrderReference>  
<OrderConfirmation>string</OrderConfirmation>  
<CustomerDealRef>string</CustomerDealRef>  
</CancelOrderResult>  
</CancelOrderResponse>  
</soap12:Body>  
</soap12:Envelope>

本文转自：http://hi.baidu.com/eveready/blog/item/585bbb30fbcee915eac4af27.html

  1. get是从服务器上获取数据，post是向服务器传送数据。   
get 和 post只是一种传递数据的方式，get也可以把数据传到服务器，他们的本质都是发送请求和接收结果。只是组织格式和数据量上面有差别，http协议里面有介绍  
  2. get是把参数数据队列加到提交表单的ACTION属性所指的URL中，值和表单内各个字段一一对应，在URL中可以看到。post是通过HTTP post机制，将表单内各个字段与其内容放置在HTML HEADER内一起传送到ACTION属性所指的URL地址。用户看不到这个过程。   
因为get设计成传输小数据，而且最好是不修改服务器的数据，所以浏览器一般都在地址栏里面可以看到，但post一般都用来传递大数据，或比较隐私的数据，所以在地址栏看不到，能不能看到不是协议规定，是浏览器规定的。  
3. 对于get方式，服务器端用Request.QueryString获取变量的值，对于post方式，服务器端用Request.Form获取提交的数据。   
没明白，怎么获得变量和你的服务器有关，和get或post无关，服务器都对这些请求做了封装  
  4. get传送的数据量较小，不能大于2KB。post传送的数据量较大，一般被默认为不受限制。但理论上，IIS4中最大量为80KB，IIS5中为100KB。   
post基本没有限制，我想大家都上传过文件，都是用post方式的。只不过要修改form里面的那个type参数  
  5. get安全性非常低，post安全性较高。   
如果没有加密，他们安全级别都是一样的，随便一个监听器都可以把所有的数据监听到，不信你自己下一个监听网络资源的软件