# [cookie 和session 的区别详解](https://www.cnblogs.com/shiyangxt/articles/1305506.html)

## 定义

Cookies是服务器在本地机器上存储的小段文本并随每一个请求发送至同一个服务器，是一种在客户端保持状态的方案。Cookie的主要内容包括：名字，值，过期时间，路径和域

Session是存在服务器的一种用来存放用户数据的类似于HashTable结构。浏览器第一次发送请求时，服务器自动生成了一个session对象和一Session ID来唯一标识这个这个对象，并将其通过响应发送到浏览器。浏览器第二次发送请求会将前一次服务器响应中的Session ID放在请求中一并发送到服务器上，服务器从请求中提取出Session ID，并和保存的所有Session ID进行对比，找到这个用户对应的session对象。

一般情况下，服务器会在一定时间内（默认30分钟）保存这个 Session，过了时间限制，就会销毁这个Session。在销毁之前，程序员可以将用户的一些数据以Key和Value的形式暂时存放在这个 Session中。

## 区别

1、存在的位置：

cookie 存在于客户端，临时文件夹中；  session存在于服务器的内存中，一个session域对象为一个用户浏览器服务

2、安全性

cookie是以明文的方式存放在客户端的，安全性低，可以通过一个加密算法进行加密后存放；  session存放于服务器的内存中，所以安全性好

3、网络传输量

cookie会传递消息给服务器；  session本身存放于服务器，不会有传送流量

4、生命周期(以20分钟为例)

cookie的生命周期是累计的，从创建时，就开始计时，20分钟后，cookie生命周期结束；session的生命周期是间隔的，从创建时，开始计时如在20分钟，没有访问session，那么session生命周期被销毁。但是，如果在20分钟内（如在第19分钟时）访问过session，那么，将重新计算session的生命周期。关机会造成session生命周期的结束，但是对cookie没有影响

5、访问范围

cookie为多个用户浏览器共享；  session为一个用户浏览器独

# get 和 post请求的区别

## HTTP协议

说道GET和POST，就不得不提HTTP协议，因为浏览器和服务器的交互是通过HTTP协议执行的，而GET和POST也是HTTP协议中的两种方法。

HTTP全称为Hyper Text Transfer Protocol，中文翻译为超文本传输协议，目的是保证浏览器与服务器之间的通信。HTTP的工作方式是客户端与服务器之间的请求-应答协议。

HTTP协议中定义了浏览器和服务器进行交互的不同方法，基本方法有4种，分别是GET，POST，PUT，DELETE。这四种方法可以理解为，对服务器资源的查，改，增，删。

GET：从服务器上获取数据，也就是所谓的查，仅仅是获取服务器资源，不进行修改。

POST：向服务器提交数据，这就涉及到了数据的更新，也就是更改服务器的数据。

PUT：PUT的英文含义是放置，也就是向服务器新添加数据，就是所谓的增。

DELETE：从字面意思也能看出，这种方式就是删除服务器数据的过程。

## Get

1、GET交互方式是从服务器上获取数据，而并非修改数据，所以GET交互方式是安全的。就像数据库查询一样，从数据库查询数据，并不会影响数据库的数据信息，对数据库来说，也就是安全的。

2、GET交互方式是幂等的，幂等是一个数学概念，幂等函数就是可以使用相同参数重复执行，并且能获得相同结果的函数。在GET交互这里就是，对同一个URL的多个请求，得到的结果是相同的。就像数据库查询，不同的数据库连接对同一个数据库表用相同条件查询时，得到的结果也是一样的。

3、请的求的数据会附加在URL之后，以？分隔URL和传输数据，多个参数用&连接。URL编码格式采用的是ASCII编码，而不是Unicode，即所有的非ASCII字符都要编码之后再传输。

4、因为URL的长度限制，GET方式传输的数据大小有所限制，传送的数据量不超过2KB

5、GET方式服务器端用Request.QueryString获取变量的值

6、GET方式传输的参数安全性低，因为传输的数据会显示在请求的URL中

7、GET请求能够被缓存，且会保存在历史记录中

## Post

1、POST交互是可以修改服务器数据的一种方式，涉及到信息的修改，就会有安全问题。就像数据库的更新，Update一个数据库表时，如果条件没有写对，就可能把不需要修改的数据给修改了，得到的数据就是错误的了。

2、一般的POST交互是必须要用到表单的，但是表单提交的默认方法是GET，如果改为POST方式，就需要修改表单提交时的Method。

3、POST方式将表单内各个字段和内容放置在HTML HEADER中一起传送到Action属性所指定的URL地址，用户是看不到这个过程的

4、POST方式传送的数据量比较大，一般被默认为没有限制，但是根据IIS的配置，传输量也是不同的

5、POST方式在服务器端用Request.Form获取提交的数据

6、POST方式传输的数据安全性较高，因为数据传输不是明显显示的

## 编码方式

1、GET请求不存在请求实体部分，键值对参数放置在 URL 尾部，因此请求头不需要设置 Content-Type 字段

2、POST请求有以下编码方式

### content-type 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。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

### ****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 的 enctyped 等于这个值。

### ****application/json****

用来告诉服务端消息主体是序列化后的 JSON 字符串，可以支持参数结果非常复杂的情况

后台获取数据方式：

1、@RequestBody

使用该注解自动获取参数

2、通过读取流转换成字符串最后解析成json对象来获取

BufferedReader br = request.getReader(); String str, wholeStr = ""; while((str = br.readLine()) != null){ wholeStr += str; } System.out.println(wholeStr); user user= (user)JSONObject.toBean(JSONObject.fromObject(wholeStr),user.class);

### ****text/xml****

# MVC的构成和实现

 model：应用的业务逻辑（如：数据库的操作），通过JavaBean实现   
（hibernate、mybatis、ibatis）

 view：视图层，用于与用户的交互，主要由jsp页面产生。   
（jsp、FreeMarker、tails、taglib、EL、Velocity ）

 controller：处理过程控制，一般是一个servlet。

它可以分派用户的请求并选择恰当的视图以用于显示

同时它也可以解释用户的输入并将它们映射为模型层可执行的操作。   
（severlet、struts、spring、action）

# Java RMI与RPC的区别

## RPC 远程过程调用

RPC（Remote Procedure Call Protocol）远程过程调用协议，通过网络从远程计算机上请求调用某种服务。

一次RPC调用的过程大概有10步：

1.执行客户端调用语句，传送参数

2.调用本地系统发送网络消息

3.消息传送到远程主机

4.服务器得到消息并取得参数

5.根据调用请求以及参数执行远程过程（服务）

6.执行过程完毕，将结果返回服务器句柄

7.服务器句柄返回结果，调用远程主机的系统网络服务发送结果

8.消息传回本地主机

9.客户端句柄由本地主机的网络服务接收消息

10.客户端接收到调用语句返回的结果数据

## RMI 远程方法调用

RMI:远程方法调用(Remote Method Invocation)。能够让在客户端**Java**虚拟机上的对象像调用本地对象一样调用服务端java 虚拟机中的对象上的方法。

RMI远程调用步骤：

1，客户调用客户端辅助对象stub上的方法

2，客户端辅助对象stub打包调用信息（变量，方法名），通过网络发送给服务端辅助对象skeleton

3，服务端辅助对象skeleton将客户端辅助对象发送来的信息解包，找出真正被调用的方法以及该方法所在对象

4，调用真正服务对象上的真正方法，并将结果返回给服务端辅助对象skeleton

5，服务端辅助对象将结果打包，发送给客户端辅助对象stub

6，客户端辅助对象将返回值解包，返回给调用者

7，客户获得返回值

## RPC与RMI的区别

1：方法调用方式不同：

 RMI中是通过在客户端的Stub对象作为远程接口进行远程方法的调用。每个远程方法都具有方法签名。如果一个方法在服务器上执行，但是没有相匹配的签名被添加到这个远程接口(stub)上，那么这个新方法就不能被RMI客户方所调用。

  RPC中是通过网络服务协议向远程主机发送请求，请求包含了一个参数集和一个文本值，通常形成“classname.methodname(参数集)”的形式。RPC远程主机就去搜索与之相匹配的类和方法，找到后就执行方法并把结果编码，通过网络协议发回。

2：适用语言范围不同：

RMI只用于Java；

RPC是网络服务协议，与操作系统和语言无关。

3：调用结果的返回形式不同：

Java是面向对象的，所以RMI的调用结果可以是对象类型或者基本数据类型；

RMI的结果统一由外部数据表示 (External Data Representation, XDR) 语言表示，这种语言抽象了字节序类和数据类型结构之间的差异。

# WebService

**WebService是一种跨编程语言和跨操作系统平台的远程调用技术。**

## ****优点****

1、跨平台调用

2、跨语言调用

3、远程调用

## 技术构成

### XML+XSD

WebService采用HTTP协议传输数据，采用XML格式封装数据（即XML中说明调用远程服务对象的哪个方法，传递的参数是什么，以及服务对象的 返回结果是什么）。XML是WebService平台中表示数据的格式。除了易于建立和易于分析外，XML主要的优点在于它既是平台无关的，又是厂商无关 的。无关性是比技术优越性更重要的：软件厂商是不会选择一个由竞争对手所发明的技术的。

XML解决了数据表示的问题，但它没有定义一套标准的数据类型，更没有说怎么去扩展这套数据类型。例如，整形数到底代表什么？16位，32位，64位？这 些细节对实现互操作性很重要。XML Schema(XSD)就是专门解决这个问题的一套标准。它定义了一套标准的数据类型，并给出了一种语言来扩展这套数据类型。WebService平台就 是用XSD来作为其数据类型系统的。当你用某种语言(如VB.NET或C#)来构造一个Web service时，为了符合WebService标准，所 有你使用的数据类型都必须被转换为XSD类型。你用的工具可能已经自动帮你完成了这个转换，但你很可能会根据你的需要修改一下转换过程。

### SOAP

WebService通过HTTP协议发送请求和接收结果时，发送的请求内容和结果内容都采用XML格式封装，并增加了一些特定的HTTP消息头，以说明 HTTP消息的内容格式，这些特定的HTTP消息头和XML内容格式就是SOAP协议。SOAP提供了标准的RPC方法来调用Web Service。

**SOAP协议 = HTTP协议 + XML数据格式**

 SOAP协议定义了SOAP消息的格式，SOAP协议是基于HTTP协议的，SOAP也是基于XML和XSD的，XML是SOAP的数据编码方式。打个比 喻：HTTP就是普通公路，XML就是中间的绿色隔离带和两边的防护栏，SOAP就是普通公路经过加隔离带和防护栏改造过的高速公路。

### WSDL

好比我们去商店买东西，首先要知道商店里有什么东西可买，然后再来购买，商家的做法就是张贴广告海报。 WebService也一样，WebService客户端要调用一个WebService服务，首先要有知道这个服务的地址在哪，以及这个服务里有什么方 法可以调用，所以，WebService务器端首先要通过一个WSDL文件来说明自己家里有啥服务可以对外调用，服务是什么（服务中有哪些方法，方法接受 的参数是什么，返回值是什么），服务的网络地址用哪个url地址表示，服务通过什么方式来调用。

　　WSDL(Web Services Description Language)就是这样一个基于XML的语言，用于描述Web Service及其函数、参数和返回值。它是WebService客户端和服务器端都 能理解的标准格式。因为是基于XML的，所以WSDL既是机器可阅读的，又是人可阅读的，这将是一个很大的好处。一些最新的开发工具既能根据你的 Web service生成WSDL文档，又能导入WSDL文档，生成调用相应WebService的代理类代码。

　　WSDL 文件保存在Web服务器上，通过一个url地址就可以访问到它。客户端要调用一个WebService服务之前，要知道该服务的WSDL文件的地址。 WebService服务提供商可以通过两种方式来暴露它的WSDL文件地址：1.注册到UDDI服务器，以便被人查找；2.直接告诉给客户端调用者。

## 适用场景

### 跨防火墙通信

如果应用程序有成千上万的用户，而且分布在世界各地，那么客户端和服务器之间的通信将是一个棘手的问题。因为客户端和服务器之间通常会有防火墙或者代理服 务器。在这种情况下，使用DCOM就不是那么简单，通常也不便于把客户端程序发布到数量如此庞大的每一个用户手中。传统的做法是，选择用浏览器作为客户 端，写下一大堆ASP页面，把应用程序的中间层暴露给最终用户。这样做的结果是开发难度大，程序很难维护。如果中间层组件换成WebService的话， 就可以从用户界面直接调用中间层组件。从大多数人的经验来看，在一个用户界面和中间层有较多交互的应用程序中，使用WebService这种结构，可以节 省花在用户界面编程上20%的开发时间。

### 应用程序集成

企业级的应用程序开发者都知道，企业里经常都要把用不同语言写成的、在不同平台上运行的各种程序集成起来，而这种集成将花费很大的开发力量。应用程序经常 需要从运行在IBM主机上的程序中获取数据；或者把数据发送到主机或UNIX应用程序中去。即使在同一个平台上，不同软件厂商生产的各种软件也常常需要集 成起来。通过WebService，可以很容易的集成不同结构的应用程序。

### B2B集成

用WebService集成应用程序，可以使公司内部的商务处理更加自动化。但当交易跨越供应商和客户、突破公司的界限时会怎么样呢？跨公司的商务交易集成通常叫做B2B集成。WebService是B2B集成成功的关键。通过WebService，公司可以把关键的商务应用“暴露”给指定的供应商和客户。例如，把电子下单系统和电子发票系统“暴露”出来，客户就可以以电子的方式发送订单，供应商则可以以电子的方式发送原料采购发票。当然，这并不是一个 新的概念，EDI(电子文档交换)早就是这样了。但是，WebService的实现要比EDI简单得多，而且WebService运行在Internet 上，在世界任何地方都可轻易实现，其运行成本就相对较低。不过，WebService并不像EDI那样，是文档交换或B2B集成的完整解决方案。 WebService只是B2B集成的一个关键部分，还需要许多其它的部分才能实现集成。

用WebService来实现B2B集成的最大好处在于可以轻易实现互操作性。只要把商务逻辑“暴露”出来，成为WebService，就可以让任何指定 的合作伙伴调用这些商务逻辑，而不管他们的系统在什么平台上运行，使用什么开发语言。这样就大大减少了花在B2B集成上的时间和成本，让许多原本无法承受 EDI的中小企业也能实现B2B集成。

### 软件和数据重用

软件重用是一个很大的主题，重用的形式很多，重用的程度有大有小。最基本的形式是源代码模块或者类一级的重用，一种形式是二进制形式的组件重用。采用 WebService应用程序可以用标准的方法把功能和数据“暴露”出来，供其它应用程序使用，达到业务级重用。

# Web容器

Web容器一种服务程序，给处于其中的应用程序组件（JSP，SERVLET）提供一个环境，使JSP,SERVLET直接跟容器中的环境变量交互，不必关注其它系统问题。包括TOMCAT，WEBLOGIC，WEBSPHERE等。

## 作用

1.通信支持

通过web容器中的方法，只需简单的操作就能实现servlet与web服务器间的通信。而不需要自己创建socket，监听接口、新的流等一系列复杂的操作。Servlet容器包含在web服务器中，web服务器监听来自特定端口的HTTP请求，这个端口通常是80。当客户端（使用web浏览器的用户）发送一个HTTP请求时，Servlet容器会创建新的HttpServletRequest和HttpServletResponse对象，并且把它们传递给已经创建的Filter和URL模式与请求URL匹配的Servlet实例的方法，所有的这些都使用同一个线程。  
request对象提供了获取HTTP请求的所有信息的入口，比如请求头和请求实体。response对象提供了控制和发送HTTP响应的便利方法，比如设置响应头和响应实体（通常是JSP生成的HTML内容）。当HTTP响应被提交并结束后，request和response对象都会被销毁。

2.控制servlet生命周期

如何加载类，实例化和初始化servlet，调用servlet方法，并使servlet实例能够被垃圾回收。有了容器，我们就不用花精力去考虑这些资源管理垃圾回收之类的事情。当Servlet容器启动时，它会部署并加载所有的web应用。当web应用被加载时，Servlet容器会一次性为每个应用创建Servlet上下文（ServletContext）并把它保存在内存里。Servlet容器会处理web应用的web.xml文件，并且一次性创建在web.xml里定义的Servlet、Filter和Listener，同样也会把它们保存在内存里。当Servlet容器关闭时，它会卸载所有的web应用和ServletContext，所有的Servlet、Filter和Listner实例都会被销毁。

3.多线程支持

为每个servlet请求创建一个线程，servlet运行完成后容器就会自动结束这个线程。

## 各容器区别