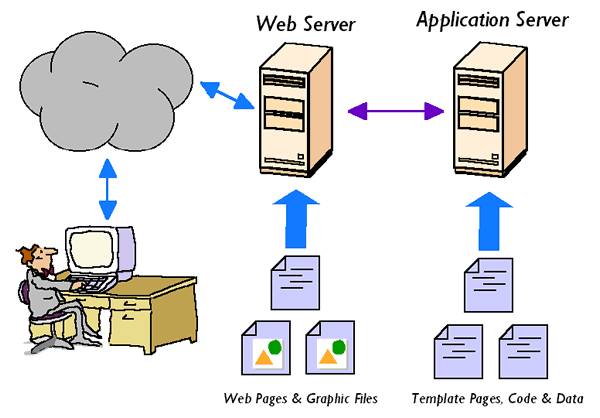
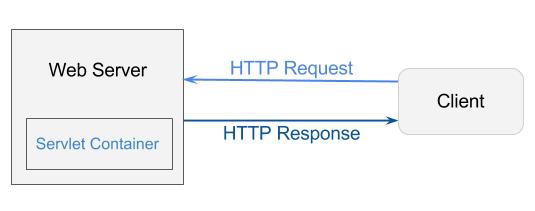
1. 简介
2. Web服务器和应用服务器的概念
3. Web容器的概念
4. Servlet的概念
5. Servlet工作原理
6. Tomcat容器和Servlet的工作流程
7. Servlet的生命周期
8. 注意
9. ServletContext的概念
10. ServletRequest和ServletResponse进入生命周期
11. Seesion和Cookie
12. doGet方式和doPost方式的区别
13. JSP工作原理
14. JSP的概念
15. Servlet和JSP的联系
16. JSP的工作流程

---------------------------------------------------------------------------------------------------------------------------------

1. 简介
2. Web服务器和应用服务器的概念



1. Web服务器的基本功能就是提供Web信息浏览服务，专门处理HTTP请求(request)，一般只需支持HTTP协议、HTML文档格式及URL。仅仅提供一个可以执行服务器端(server-side)程序和返回(程序所产生的)响应(response)的环境。
2. 应用程序服务器让多个用户可以同时使用应用程序，通过很多协议（比Web服务器多）来为应用程序提供商业逻辑。服务器端(server-side)程序通常具有事务处理(transaction processing)，数据库连接(database connectivity)和消息(messaging)等功能。
3. 以Java EE为例，Web服务器主要是处理静态页面处理和作为 Servlet容器，解释和执行servlet/JSP，而应用服务器是运行业务逻辑的，主要是EJB、 JNDI和JMX API等J2EE API方面的，还包含事务处理、数据库连接等功能，所以在企业级应用中，应用服务器提供的功能比WEB服务器强大的多。Tomcat作为Web服务器也兼有一部分应用服务器的功能，也被称为轻量级应用服务器。
4. Web服务器和应用服务器界限不是那么明确，只是侧重点不同而已。
5. Web容器的概念



1. 在Java方面，web容器一般是指Servlet容器。Servlet容器是与Java Servlet交互的web容器的组件。
2. web容器负责管理Servlet的生命周期、把URL映射到特定的Servlet、确保URL请求拥有正确的访问权限、当request到来时实例化servlet并且新建一个线程来处理请求。
3. 综合来看，Servlet容器就是用来运行你的Servlet和维护它的生命周期的运行环境。
4. Tomcat作为一个轻量级的服务器，就包含了servlet/jsp容器。一个web服务器可以包含多个容器，一般web服务器中会包含web容器。
5. Servlet的概念
6. 在Java里，Servlet是能够根据请求动态生成内容的服务端组件。事实上，Servlet是一个在javax.servlet包里定义的接口。自定义Servlet一般需要继承HttpServlet，而HttpServlet是继承GenericServlet，而GenericServlet是继承Servlet。
7. HttpServlet是特定于HTTP协议的类，Servlet接口和GenericServlet是不特定于任何协议的。Servlet接口中定义了五个方法，其中比较重要的三个方法涉及到Servlet的生命周期，分别是init(),service(),destroy()方法。GenericServlet是一个通用的，不特定于任何协议的Servlet,它实现了Servlet接口。而HttpServlet继承于GenericServlet，因此HttpServlet也实现了Servlet接口。所以定义Servlet的时候只需要继承HttpServlet即可。在HttpServlet中实现了service()方法，并将请求ServletRequest,ServletResponse强转为HttpServletRequest和HttpServletResponse。HttpServletRequest、 HttpServletResponse分别是ServletRequest和ServletResponse的子类
8. 类加载器通过懒加载（lazy-loading）或者预加载（eager loading）自动地把Servlet类加载到容器里。通常每个Servlet类在容器中只存在一个实例，每当请求到来时，则分配一个线程来处理该请求。每个请求都拥有自己的线程，而一个Servlet对象可以同时为多个线程服务。当Servlet对象不再被使用时，它就会被JVM当做垃圾回收掉。
9. Servlet工作原理
10. Tomcat容器和Servlet的工作流程



1. （容器启动初始化）Servlet容器启动时读取web.xml配置文件中的信息，构造指定的Servlet对象，创建ServletConfig对象，同时将ServletConfig对象作为参数来调用Servlet对象的init方法。
2. （分析请求）客户端浏览器向Servlet容器发出一个Http请求，Servlet容器分析请求信息。创建一个请求对象HttpServletRequst，用来封装用户请求的信息。此外再创建一个响应对象HttpServletResponse，用于响应客户端请求，向客户端返回数据。
3. （创建Servlet）当客户端请求一个 Servlet 时，Servlet容器收到该请求后，首先到容器中检索与请求匹配的 Servlet 实例是否已经存在。

--若不存在，则 Servlet 容器负责加载并实例化出该类 Servlet的一个实例对象，接着容器框架负责调用该实例的 init() 方法来对实例做一些初始化工作，然后Servlet 容器运行该实例的 service() 方法。

--若 Servlet 实例已经存在，则容器框架直接调用该实例的 service() 方法。

1. Servlet容器把HttpServletRequst和HttpServletResponse对象作为参数传给用户所请求的HttpServlet。
2. service() 方法在运行时，自动匹配运行与用户请求相对应的 doXX() 方法来响应用户发起的请求。
3. Servlet利用HttpServletResponse包含的数据和自身的业务逻辑处理请求，并把处理好的结果写在HttpServletResponse中，最后Servlet容器把响应结果传给用户。
4. Servlet的生命周期

* 生命周期总共分为四个过程，其中(1)(2)(4) 在 Servlet 的生命周期中只会被执行一次。

1) 加载和实例化

如果Servlet容器还没实例化一个Servlet对象，此时容器装载和实例化一个 Servlet。创建出该 Servlet 类的一个实例。如果已经存在一个Servlet对象，此时不再创建新实例。

2) 初始化 -- init()

在产生 Servlet 实例后，容器负责调用该 Servlet 实例的 init() 方法，在处理用户请求之前，来做一些额外的初始化工作。

3) 处理请求

当 Servlet 容器接收到一个 Servlet 请求时，便运行与之对应的 Servlet 实例的 service() 方法，service() 方法根据用户的请求调用相对应的doGet或doPost 方法来处理用户请求。然后再进入对应的方法中调用逻辑层的方法，实现对客户的响应。

4) 销毁 -- destroy()

当 Servlet 容器决定将一个 Servlet 从服务器中移除时 ( 如 Servlet 文件被更新 )，便调用该 Servlet 实例的 destroy() 方法，在销毁该 Servlet 实例之前，来做一些其他的工作。

1. 注意

如果在web.xml中对servlet配置了load-on-startup，就是在容器加载Servlet初始化。



1. 一般的Servlet启动后只有一个实例（对同一Servlet只配置一个mapping），如果一个Servlet有多个mapping，那么会有多个Servlet实例。
2. 在web.xml文件中，某些Servlet只有<serlvet>元素，没有<servlet-mapping>元素，这样无法通过url的方式访问这些Servlet，这种Servlet通常会在<servlet>元素中配置一个<load-on-startup>子元素，让容器在启动的时候自动加载这些Servlet并调用init()方法，完成一些全局性的初始化工作。也就是说，如果要通过某个网址加载某些Servlet时，需要写<servlet-mapping>元素。如果Servlet是在容器加载时运行时，则不需要写<servlet-mapping>元素.
3. 当Servlet只有一个实例，当有多个客户端访问时，会多次调用已经实例化好的service方法处理请求。有时容器根据客户机请求生成Servlet对象实例或生成多个Servlet对象实例并将其加入SERVLET实例池中，来处理请求。
4. ServletContext的概念
5. 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。
6. 从Java文档可知，ServletContext定义了一组方法，Servlet使用这些方法来与它的Servlet容器进行通信。例如，用来获取文件的MIME类型、转发请求或者编写日志文件。
7. 在web应用的部署文件（deployment descriptor）标明“分布式”的情况下，web应用的每一个虚拟机都拥有一个上下文实例。在这种情况下，不能把Servlet上下文当做共享全局信息的变量（因为它的信息已经不具有全局性了）。可以使用外部资源来代替，比如数据库。
8. ServletRequest和ServletResponse进入生命周期
9. 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。
10. 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。
11. Seesion和Cookie

* jsp的Session和Servlet的Session本质上是一致的，区别是：jsp中session是作为隐式对象存在的，可以直接使用；Servlet中的session需要手动提取后才能使用。

1. **（session和cookie的生成）**当客户端第一次访问web应用或者第一次使用

request.getSession()获取HttpSession时，Servlet容器会创建Session，生成一个long类型的唯一ID（你可以使用session.getId()获取它）并把它保存在服务器的内存里。Servlet容器同样会在HTTP响应里设置一个Cookie，cookie的名是JSESSIONID并且cookie的值是session的唯一ID。

1. **（cookie的作用）**根据HTTP cookie规范（正规的web浏览器和web服务器必须遵守的约定），在cookie的有效期间，客户端（web浏览器）之后的请求都要把这个cookie返回给服务器。Servlet容器会利用带有名为JSESSIONID的cookie检测每一个到来的HTTP请求头，并使用cookie的值从服务器内容里获取相关的HttpSession。
2. **（生存时间）**HttpSession会一直存活着，除非超过一段时间没使用。你可以在web.xml里设定这个时间段，默认时间段是30分钟。因此，如果客户端已经超过30分钟没有访问web应用的话，Servlet容器就会销毁Session。之后的每一个请求，即使带有特定的cookie，都再也不会访问到同一个Session了。servletcontainer会创建一个新的Session。
3. 在客户端的session cookie拥有一个默认的存活时间，这个时间与浏览器的运行时间相同。因此，当用户关闭浏览器后（所有的标签或者窗口），客户端的Session就会被销毁。重新打开浏览器后，与之前的Session关联的cookie就再也不会被发送出去了。再次使用request.getSession()会返回一个全新的HttpSession并且使用一个全新的session ID来设置cookie。
4. doGet方式和doPost方式的区别

* Get方式的请求：直接在浏览器地址栏输入访问的地址所发送的请求或表单发送时没有指明post形式发送的(表单默认为get提交)都会调用doGet方法。

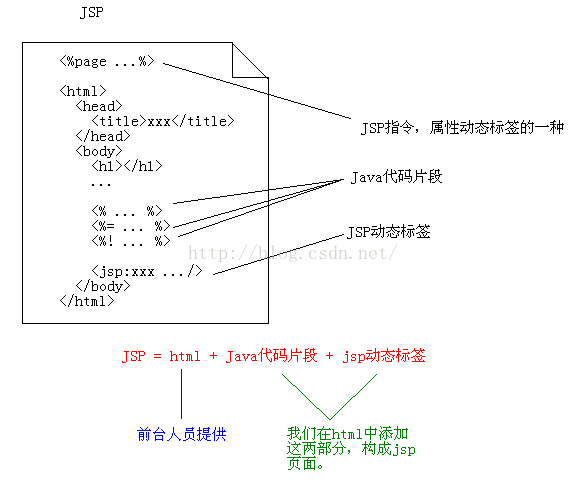
1. 由于get方式请求会将请求参数的名和值转换成字符串，并附在原URl之后，因此可以在地址栏上看见请求参数的名和值，安全性比较差。
2. get请求的数据量比较小，只能传递字符串，不能传递二进制数据
3. 服务器随机接受GET方法的数据，一旦断电等原因，服务器也不知道信息是否发送完毕。

* Post方式的请求：表单以post方式发送。

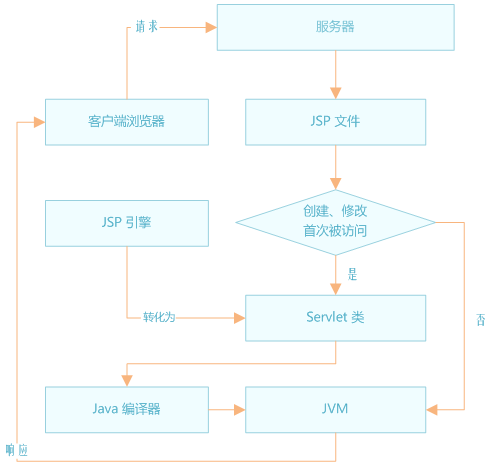
1. POST方式发送的请求参数以及对应的值放在html header中传输，安全性相对较高。
2. post传递的数据量比较大，通常认为请求参数大小不受限制，但往往取决于服务器端的限制。传递数据的类型没有限制，可以传递二进制数据
3. 而且此外，Post方法接受数据时，服务器先接受数据信息的长度，然后再接受数据，使用post方式，服务器可以知道数据是否完整。
4. JSP工作原理
5. JSP概念

JSP（Java Server Pages）是JavaWeb服务器端的动态资源。它与html页面的作用是相同的，显示数据和获取数据。

JSP = html +Java脚本（代码片段）+JSP动态标签



1. Servlet和JSP的联系
2. JSP 本质是一个 Servlet，它的运行也需要容器的支持。
3. 在 JSP 和 Servlet 文件中都可以编写 Java 和 HTML 代码，不同的是，Servlet 虽然也可以动态的生成页面内容，但更加偏向于逻辑的控制。
4. JSP 最终被转换成 Servlet 在 jvm 中解释执行，在 JSP 中虽然也可以编写 Java 代码，但它更加偏向于页面视图的展现。
5. 在 MVC 架构模式中，就 JSP 和 Servlet 而言，C 通常由 Servlet 充当，V 通常由 JSP 来充当。
6. JSP工作流程



1. 当客户端浏览器向服务器请求一个 JSP 页面时，服务器收到该请求后，首先检查所请求的这个JSP 文件内容 ( 代码 ) 是否已经被更新，或者是否是 JSP 文件创建后的第一次被访问。
2. --如果是，那么，这个 JSP 文件就会在服务器端的 JSP 引擎作用下转化为一个 Servlet 类的 Java 源代码文件。紧接着，这个 Servlet 类会在 Java 编译器的作用下被编译成一个字节码文件，并装载到 jvm 解释执行。剩下的就等同于 Servlet 的处理过程了。

--如果被请求的 JSP 文件内容 ( 代码 ) 没有被修改，那么它的处理过程也等同于一个 Servlet 的处理过程。即直接由服务器检索出与之对应的 Servlet 实例来处理。

1. 要注意的是，JSP 文件不是在服务器启动的时候转换成 Servlet 类的。而是在被客户端访问的时候才可能发生转换的 ( 如 JSP 文件内容没有被更新等，就不再发生 Servlet 转换 )。