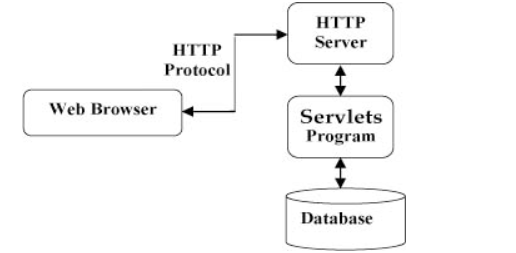
Servlet 是什么？

Java Servlet 是运行在 Web 服务器或应用服务器上的程序, 由Servlet容器所管理，用于生成动态的内容(是一个java类)



Web服务器(http服务器)主要是向浏览器发送HTML以供浏览，如Apache

应用程序服务器(各种协议)提供功能以供客户端应用程序使用

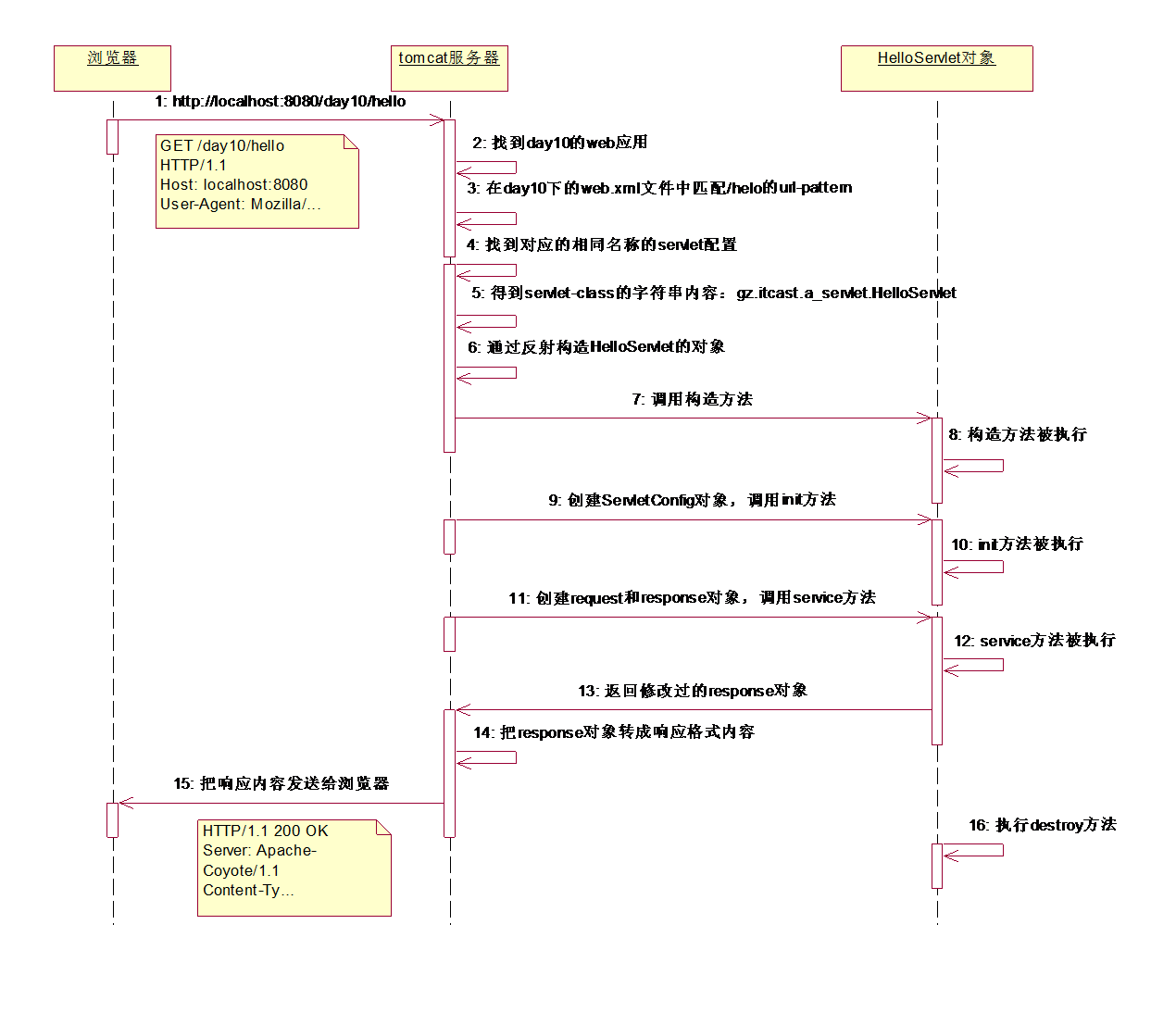
应用程序使用此逻辑就像你调用对象的一个方法(或过程语言中的一个函数)一样,Tomcat

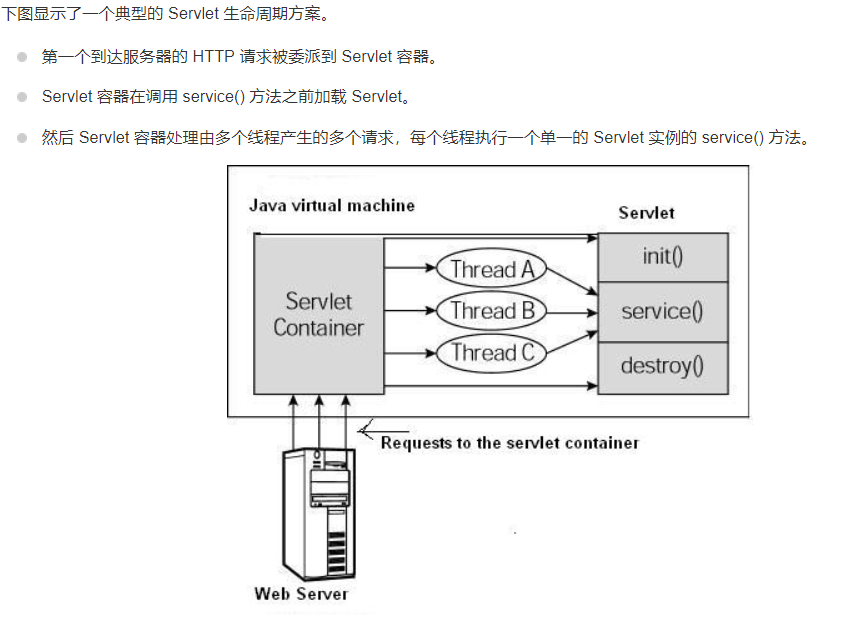
特点：

* Servlet的生命周期有四个阶段：加载并实例化、初始化、请求处理、销毁
* servlet由servlet容器调用，不能独立运行。客户端多次请求，服务器只创建一个servlet实例，之后驻留内存中继续服务直至web容器退出才销毁它。
* 请求:服务器针对客户端的每一次请求都会创建新的request和response对象(它们的生命周期很短)，传给service方法。

servlet的调用过程和生命周期

当用户在浏览器输入一个url,如 http://localhost:8080/Servlet\_Requset时，[服务器](https://www.baidu.com/s?wd=%E6%9C%8D%E5%8A%A1%E5%99%A8&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)将其提交到Tomcat，Tomcat根据用户提交的url在web.xml中进行查找，首先是在<servlet-mapping>中查找，当找到<url-pattern>中有跟用户输入的url匹配时，获取mapping中的<servlet-name>中的值，这里是Servlet-Request，并且根据获取得到的值在<Servlet>中查找一个<servlet-name>与其匹配的值，然后获取<servlet-class>中的确切的类，然后通过反射加载，生成Servlet对象，然后进行访问





//先初始化，再多线程多线程调用同一个servlet实例的service（）

JavaBean

遵循特定写法的java类，用于封装数据

1. 所有属性为private

2.具有无参数的构造器

（若无构造方法，则默认添加一个无参构造方法，若写了一个构造方法，无参不会自动添加，要手动添加）

3. 这个类的属性使用getter和setter来访问，其方法遵从标准命名规范

4.可序列化，实现Serializable接口

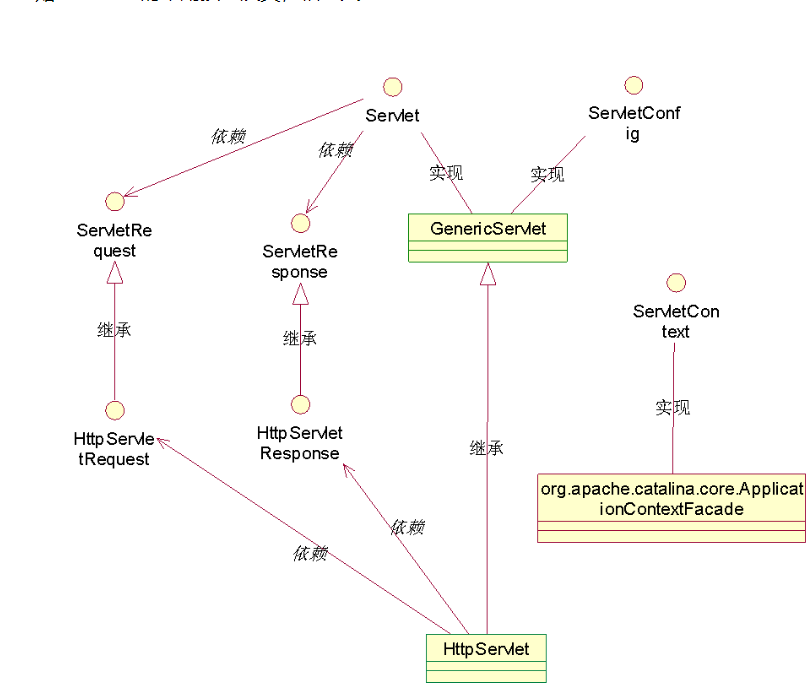
只要满足这些其他随意

POJO****（简单java对象）****

有一些属性及其getter setter方法的类,

有简单的运算方法也是可以的,但不允许有业务方法

关系图



JavaWeb 4大作用域

PageContex ,request、session、applicationion（ServletContext）

PageContex作用范围：整个JSP页面，是四大作用域中最小的一个

超过这个页面就不能够使用了

(所以使用pageContext对象向其它页面传递参数是不可能的.)

request是表示一个请求，只要发出一个请求就会创建一个request

****它的作用域：仅在当前请求中有效。  
用处：常用于服务器间同一请求在不同页面之间的参数传递，常应用于表单的值传递****

session服务器会为每个会话创建一个session对象

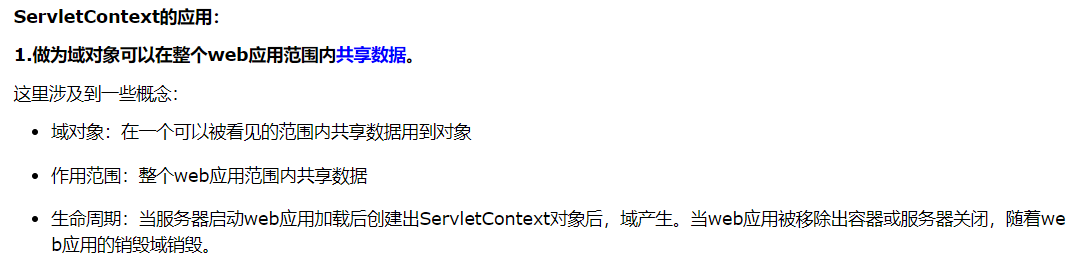
****session中的数据可供当前会话中所有servlet共享  
会话：用户打开浏览器会话开始，直到关闭浏览器会话才会结束。一次会话期间只会创建一个session对象****

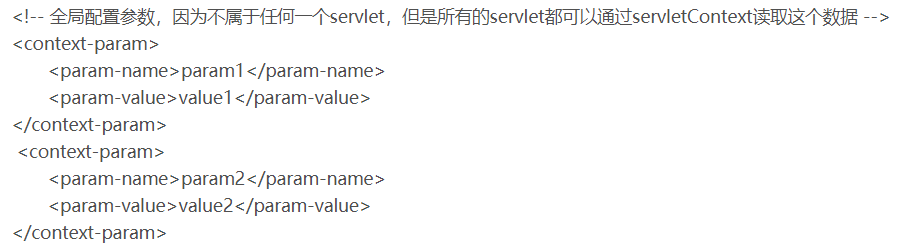
ServletContext 代表当前web应用（多种获取方法）

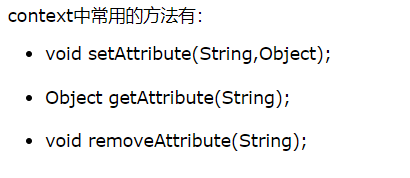
* ****WEB容器在启动时，它会为每个WEB应用程序都创建一个对应的ServletContext对象****
* ****一个WEB应用中的所有Servlet共享同一个ServletContext对象，因此Servlet对象之间可以通过ServletContext对象来实现通讯****

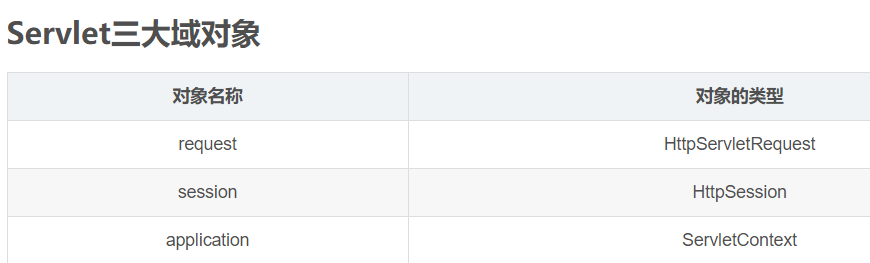
****ServletContext对象通常也被称之为context域对象****

JSP作用域与上同









保持状态(实现会话跟踪的技术)的4中方法：

网址重写（URL rewriting）、隐藏域(表单)、cookie、及HTTPSession对象

前两个不适合需要跨越多个页面的信息

由于HTTP协议本身是无状态的，服务器为了区分不同的用户，就需要对用户会话进行跟踪，一般为用户分配唯一的ID，下一次用户在请求中包含此ID，服务器据此判断到底是哪一个用户

Cookie

cookie有两种

* 一种是会话Cookie，浏览器窗口关闭后，cookie就没有了；
* 另一种是永久Cookie，保存在硬盘，并设置存在的时间

当用户通过浏览器和服务器建立一次会话后，会话ID等信息就会随响应信息返回并存储在客户端cookie中，只要浏览器没有关闭或者会话没有超时，下一次请求时这个会话ID又会随Cookie提交给服务器来识别用户身份

Cookie设置在http头部信息中



*不可跨域名性：*

*哪个服务器返回的cookie，那么这个cookie就只能用于哪个服务器*

*即www.google.com返回的cookie不能被www.baidu.com读取*

使用cookie时注意：

cookie存储的数据量有限（4k），不能将过多的内容存储cookie中；

浏览器通常只允许一个站点最多存放20个cookie

HttpSession

当一个用户第一次访问某个网站时会自动创建一个HttpSession

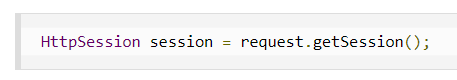
有以下三种方式来维持 Web 客户端和 Web 服务器之间的 session 会话

* ****一个 Web 服务器可以分配一个唯一的 session 会话 ID 作为每个 Web 客户端的 cookie，对于客户端的后续请求可以使用接收到的 cookie 来识别****
* 一个 Web 服务器可以发送一个隐藏的 HTML 表单字段，以及一个唯一的 session 会话 ID，如下所示



* ****可以在每个 URL 末尾追加一些数据来标识 session 会话，服务器会把该 session 会话标识符与已存储的有关 session 会话的数据相关联。****

例如，<http://w3cschool.cc/file.htm;sessionid=12345>



****与Cookie区别****

* ****会话中的内容放在服务器内存中,**** cookie保存客户端内存/硬盘中
* HttpSession中保存的值不发送到客户端，
* Servlet容器为它所创建的每一个HttpSession生成一个唯一标识符  
  并将这个标识符作为一个token（令牌）发送给浏览器，一般是作为一个名为JSESSIONID的cookie，或者作为一个jsessionid参数添加到URL后面。在后续的请求中，浏览器会将这个token发送回服务器，使服务器能够知道是哪个用户在发送请求****（session底层依赖于cookie）****

****特点：****

* 添加到HttpSession中不一定是String，可以是任意java对象，只要它的类实现了java.io.Serializable接口就好，以便保存的对象可以序列化成一个文件或者保存到数据库中  
      --例如容器的内存将要用完时，仍然可以将非序列化的对象保存在HttpSession中，但是如果Servlet容器试图将其序列化时，将会抛出异常
* *HttpSession放在服务器的内存中，因此不要将过大的对象放在里面*