**一．过滤器**

**概述：Servlet**规范当中定义的一种特殊组件，用来拦截容器的调用过程。

是一个服务器端的组件，可以截取用户端的请求和与响应信息，并对这些信息过滤。

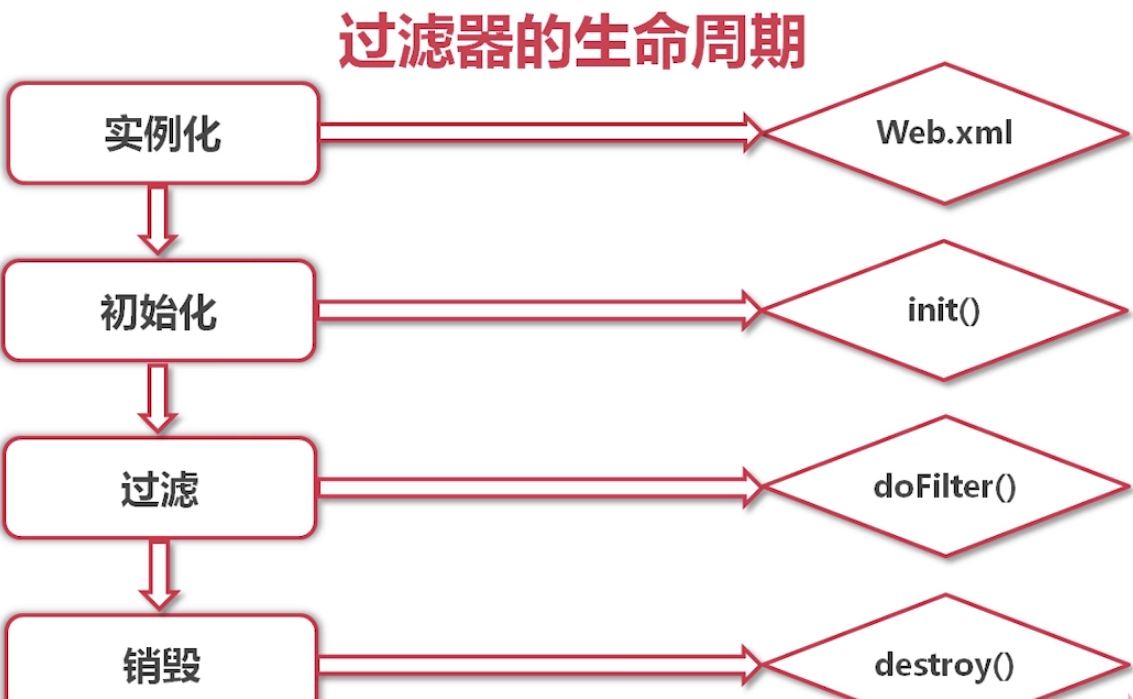
功能：过滤器可以改变用户请求的路径，但不能直接返回数据，不是一个标准的servlet

创建：java类——interface实现javax.servlet——重写doFilter——配置xml【**置顶**】

优点： a.在不修改原有代码的基础上，增加新的功能。

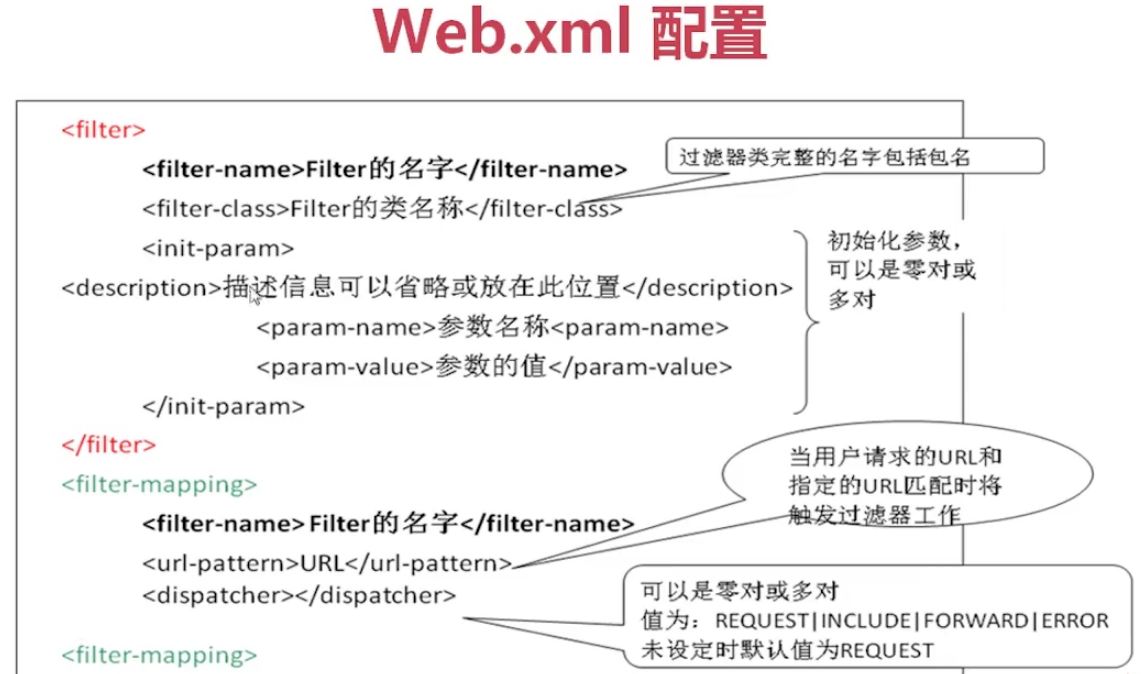
b.可以将多个组件相同的处理逻辑集中写在过滤器中，方便代码的维护。





**实例化：在web容器启动的时候会自动加载过滤器，只会实例化一次**





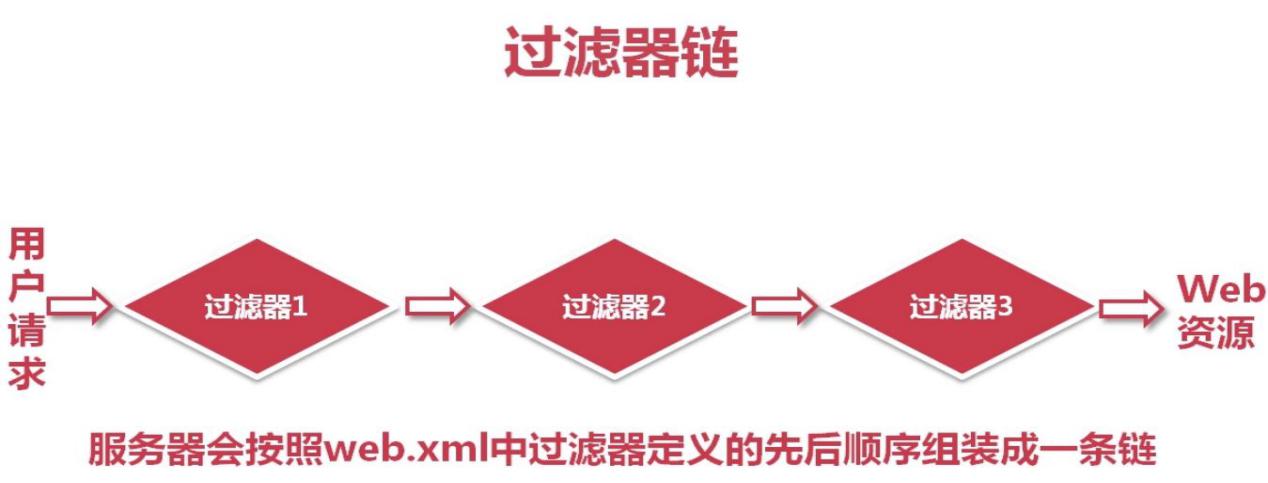
**Dispather是定义过滤器的类型，过滤什么样的请求**

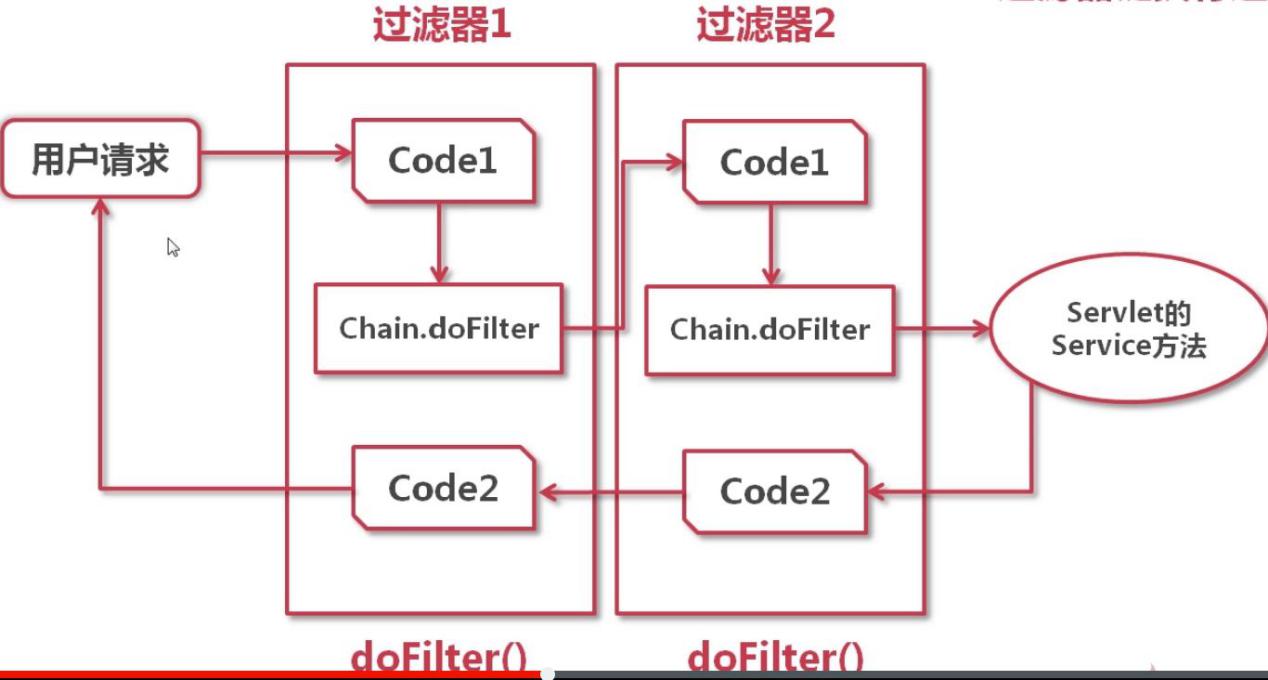
forward方法是把请求的内容转发到另外的一个servlet.而include是把另一个servlet处理过后的内容拿过来

forward与include基本上都是转发到容器内部的资源，而redirect可以重定向到外部的资源

调用forward() 的话,有关response对象的一切方法或者属性都会失去作用..只有request能被转向到下一个页面.

调用include()的话,response跟request都能被传递到转向的下一个页面

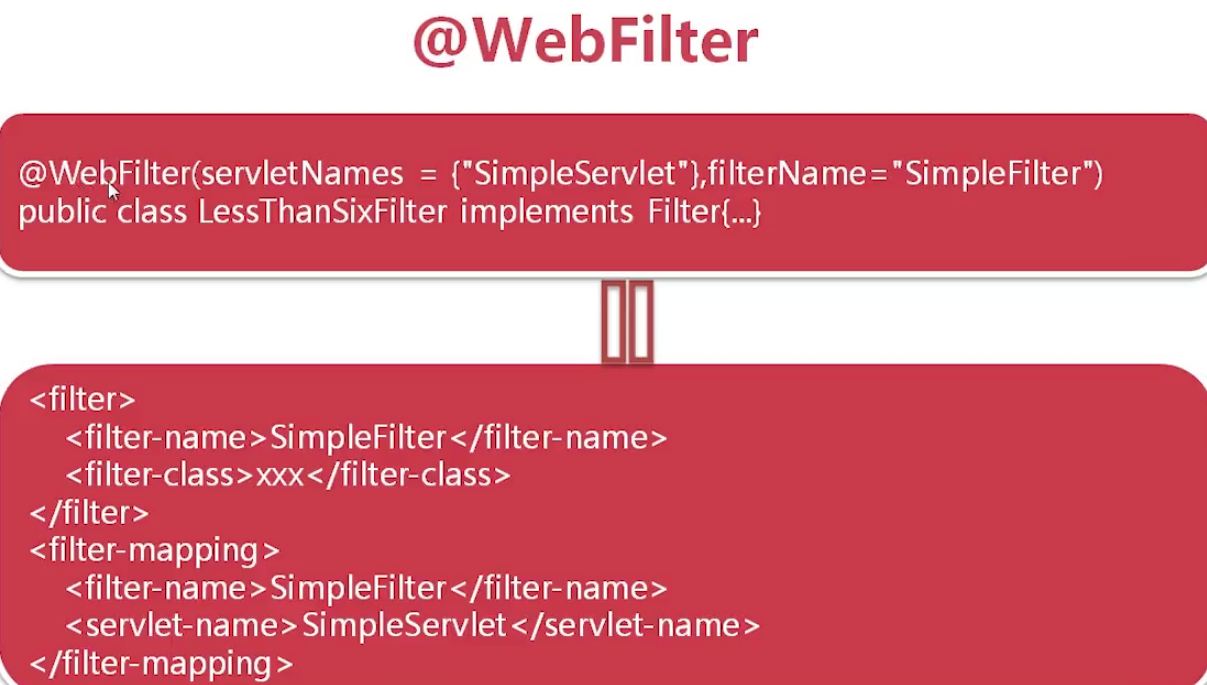




1).filter的init方法在容器初始化时加载。第一次加载容器执行顺序随机，以后再次加载顺序以第一次加载顺序为准。  
2).filter的doFilter方法在请求url时执行，如果有多个filter匹配，则按照<filter-mapping>顺序执行(前提是doFilter方法里面最后要调用FilterChain的doFilter方法，这个方法作用是继续执行下个filter,如果没有加，则不执行下面的filter)

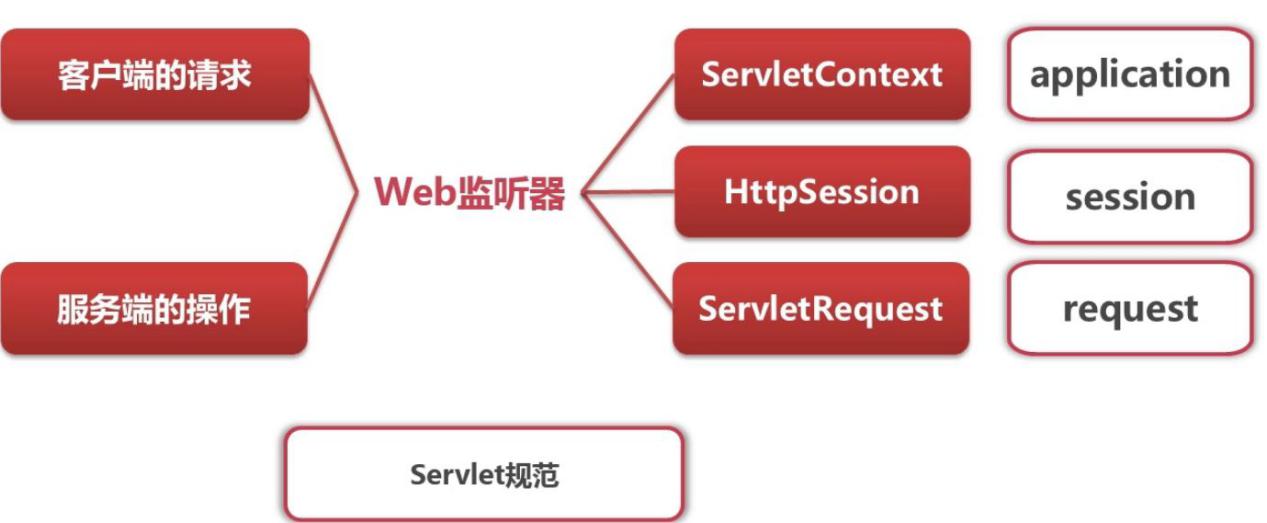




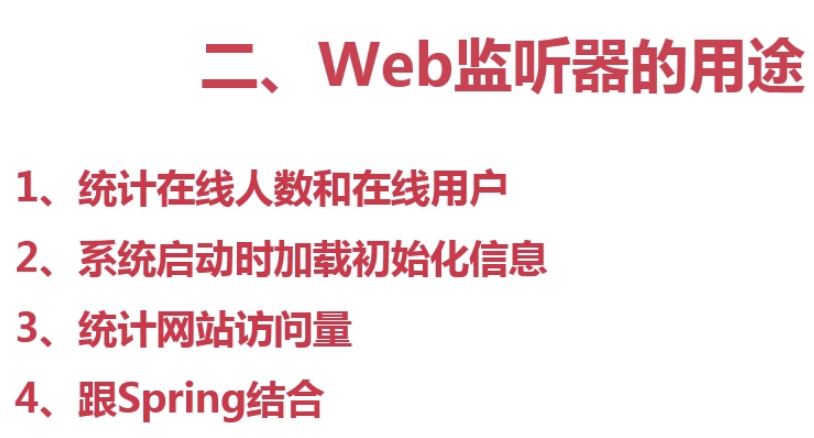




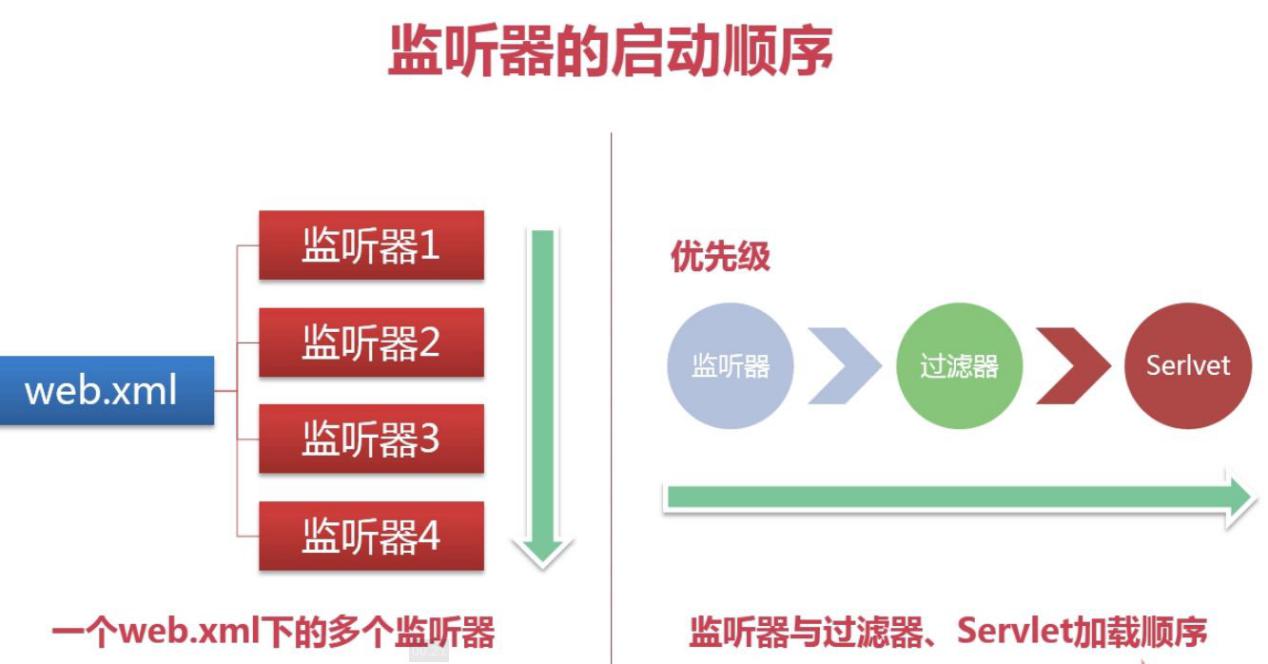
**二．监听器**



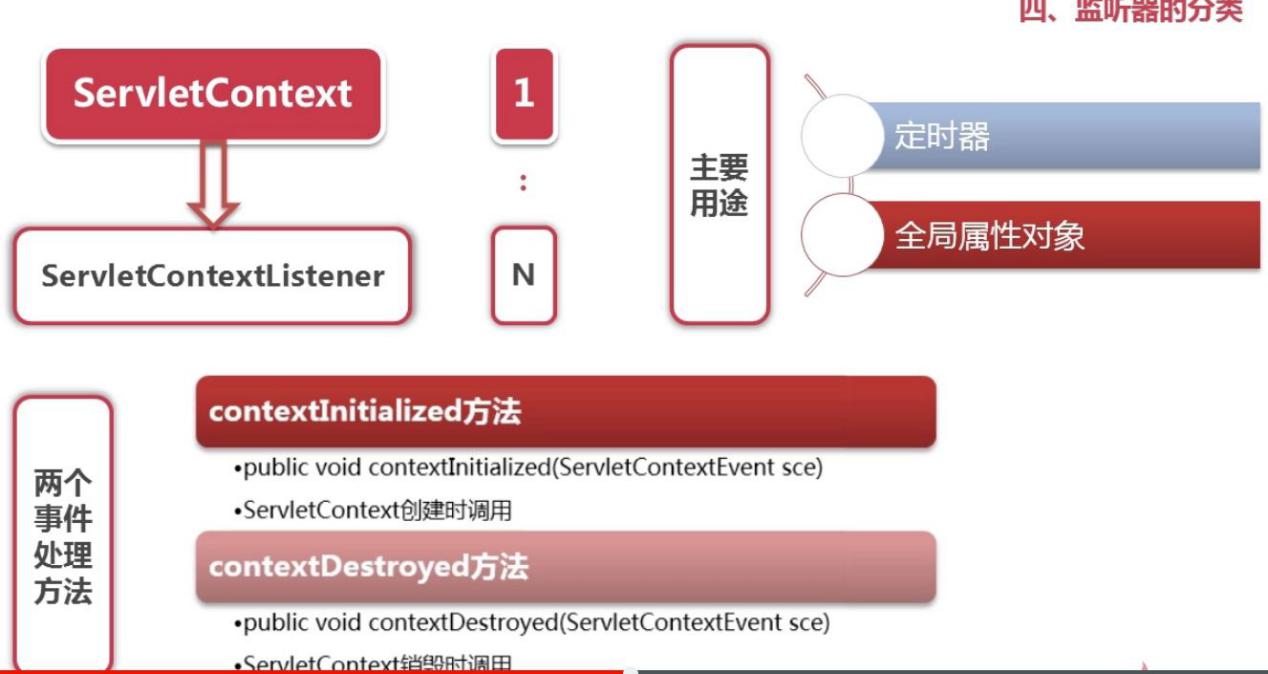


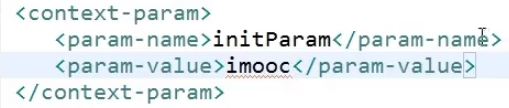








**--------------------------ServletContext【全局】--------------------** 





**servlet上下文**

1).什么是servlet上下文？

容器启动之后，会为每一个web应用创建唯一的

一个符合ServletContext接口要求的对象。

特点：

a.唯一性：一个web应用只会创建一个servlet上下文。

b.持久性：只要容器不关闭，应用没有被删除，servlet上下文就一直存在。

2). 如何获得servlet上下文？

GenericServlet,servletConfig,FilterConfig,HttpSession

都提供了getServletContext方法。

3). servlet上下文的作用？

作用1：绑定数据

setAttribute(),getAttribute(),removeAttribute()

注：

a.绑定到上下文的数据会一直存在

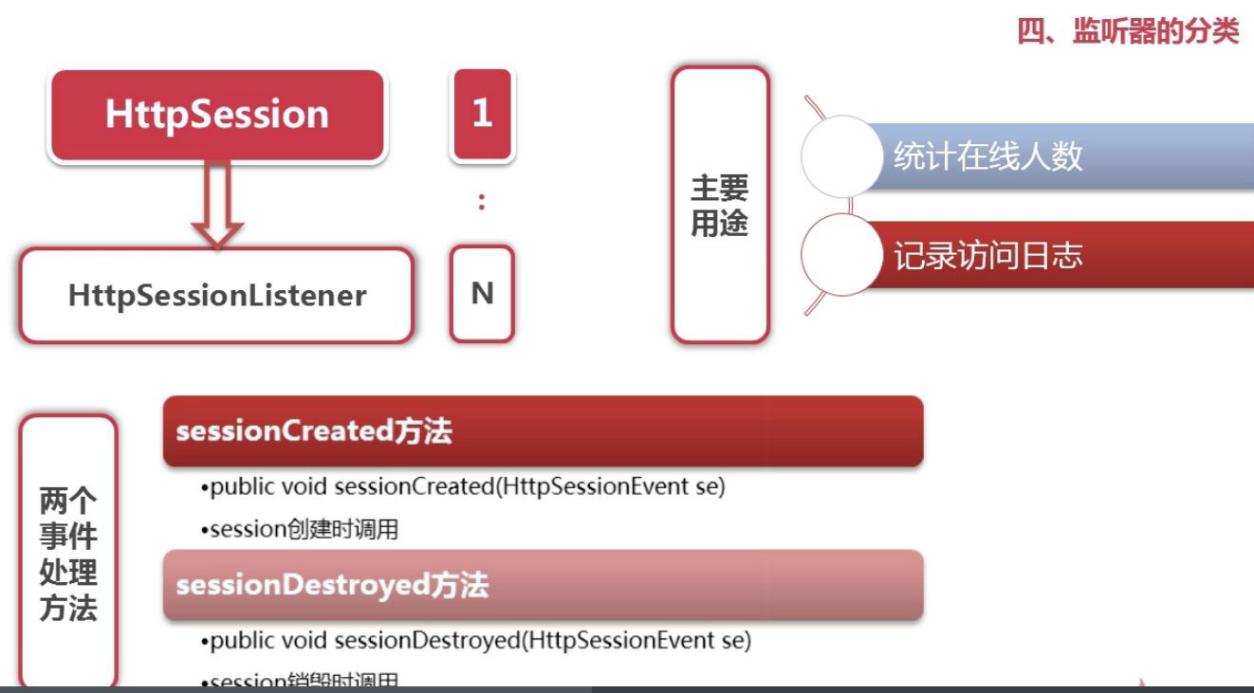
request<session<servlet上下文

b.绑定到上下文的数据可以被所有的用户共享

作用2：访问全局的初始化参数

ServletContext.getInitParameter(String paraName);

**---------------------------------HttpSession-------------------------------------**

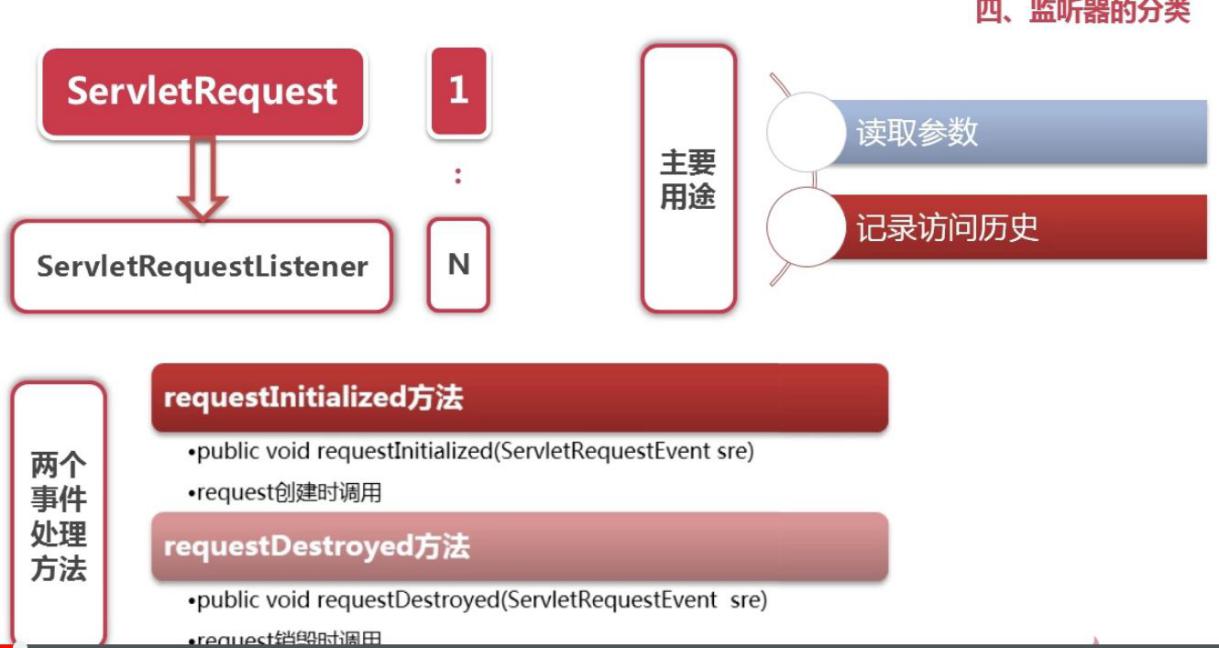




0：无超时限制

1：1分钟，实际时间比1分钟长，无关紧要

**---------------------------------ServletRequest-------------------------------------**



**------------------------按事件划分--------------------**

****

