# Servlet的生命周期

1. Servlet一般在第一次访问的时候，被创建，但是通常，我们都会在web.xml中的servlet-mapping标签里写 <load-on-startup>以使得Servlet在服务器启动的时候，加载项目的时候创建Servlet。执行顺序是：
2. 服务器启动加载项目
3. 创建该项目的ServletContext对象
4. 创建该项目中存在的Servlet实例，并且相应的创建对应的ServletConfig，然后调用初始化方法 init();
5. 通常，web容器只会创建一个Servlet实例，存在内存中，然后面对不同的请求，该实例都会创建一个新的 request对象、response对象，并传给service(request,response),执行服务。
6. 服务器关闭，调用Servlet的destroy()方法，销毁servlet实例。
7. ServletContext的相关知识点：

ServletContext代表当前web应用，当服务器启动时，会依次加载web项目，且生成ServletContext，所以一个服务器中部署了多个项目的话，那么启动的时候，其他项目的ServletContext监听器也会有反应。

1. **过滤器Filter**

什么是过滤器：

过滤器 实际上是一个实现了Filter接口的类。

### Filter的生命周期

* **init(FilterConfig filterConfig)throws ServletException：**
  + 和我们编写的Servlet程序一样，Filter的创建和销毁由WEB**服务器负责**。 web 应用程序**启动时**，web 服务器将**创建Filter 的实例对象**，并调用其init方法进行初始化（注：**filter对象只会创建一次，init方法也只会执行一次。**示例 ）
  + 开发人员通过init方法的参数，可获得代表当前filter配置信息的FilterConfig对象。(filterConfig对象见下页PPT)
* **doFilter(ServletRequest,ServletResponse,FilterChain)**
  + 每次filter进行拦截都会执行
  + 在实际开发中方法中参数request和response通常转换为**HttpServletRequest**和**HttpServletResponse**类型进行操作
* **destroy()：**
  + 在Web容器卸载 Filter 对象之前被调用。

### Filter的Filter链FilterChain

* Filter接口中有一个doFilter方法，当开发人员编写好Filter，并配置对哪个web资源(拦截url)进行拦截后，WEB服务器每次在调用web资源之前，都会先调用一下filter的doFilter方法，因此，在该方法内编写代码可达到如下目的：
  + 调用目标资源之前，让一段代码执行
  + 是否调用目标资源（即是否让用户访问web资源）。
    - web服务器在调用doFilter方法时，会传递一个filterChain对象进来，filterChain对象是filter接口中最重要的一个对象，它也提供了一个doFilter方法，开发人员可以根据需求决定是否调用此方法，调用该方法，则web服务器就会调用web资源的service方法，即web资源就会被访问，否则web资源不会被访问。
  + 调用目标资源之后，让一段代码执行。

意思就是FilterChain.doFilter(request,response); 如果执行，那么就相当于放行了，通过当前过滤器，web资源走入FilterChain中下一个过滤器。

Filter链形成顺序是：web.xml中Filter的配置顺序。

|  |
| --- |
| <filter>  <filter-name>testFitler</filter-name>  <filter-class>org.test.TestFiter</filter-class>  <init-param>  <param-name>word\_file</param-name>  <param-value>/WEB-INF/word.txt</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>testFilter</filter-name>  <url-pattern>/index.jsp</url-pattern>  <dispatcher>REQUEST</dispatcher>  <dispatcher>FORWARD</dispatcher>  </filter-mapping>  上述是一个完整的filter在web.xml中的配置。黑色是必要，蓝色是可选。   * 用户在配置filter时，可以使用<init-param>为filter配置一些初始化参数，当web容器实例化Filter对象，调用其init方法时，会把封装了filter初始化参数的filterConfig对象传递进来。因此开发人员在编写filter时，通过filterConfig对象的方法，就可获得：   + String getFilterName()：得到filter的名称。   + String getInitParameter(String name)： 返回在部署描述中指定名称的初始化参数的值。如果不存在返回null.   + Enumeration getInitParameterNames()：返回过滤器的所有初始化参数的名字的枚举集合。   + public ServletContext getServletContext()：返回Servlet上下文对象的引用。 |

# listener监听器

1. 什么是监听器？

顾名思义，一件事情的发生，如果被监听器知道了，就会执行相应的操作。

如：警察看到小偷偷东西，那么就会去抓他。

事件源：小偷。

事件：偷东西

监听器：警察

Javaweb中，主要有三个事件源，servletContext、request、session对象创建和销毁。

一般用的比较少。

监听器的使用步骤：

创立一个自定义的类，实现相应的监听器接口，如下：

**public class MyListener implements** ServletContextListener {  
 @Override  
 **public void** contextInitialized(**ServletContextEvent** servletContextEvent) {  
 //在这个里面实现加载spring配置的代码。  
 ApplicationContext applicationContext=  
 **new** ClassPathXmlApplicationContext("demoContext.xml");  
 ServletContext servletContext=servletContextEvent.getServletContext();  
 servletContext.setAttribute("applicationContext",applicationContext);  
 }  
 @Override  
 **public void** contextDestroyed(**ServletContextEvent** servletContextEvent) {

}  
}