一、  
1、启动一个WEB项目，WEB容器会先去读取它的配置文件web.xml,读取<context-param>和<listener>两个节点。  
2、接着，容器创建一个ServletContext(servlet上下文)，这个web项目的所有部分都讲共享这个上下文  
3、容器将<context-param>转换为键值对，并交给servletContext。  
4、容器创建<listener>中的类实例，创建监听器。  
二、  
load-on-startup 元素在web应用启动的时候指定了servlet被加载的顺序，它的值必须是一个整数。如果它的值是一个负整数或是这个元素不存在，那么容器会在该servlet被调用的时候，加载这个servlet 。如果值是正整数或零，容器在配置的时候就加载并初始化这个servlet，容器必须保证值小的先被加载。如果值相等，容器可以自动选择先加载谁。  
在servlet的配置当中，<load-on-startup>5</load-on-startup>的含义是：  
标记容器是否在启动的时候就加载这个servlet。  
当值为0或者大于0时，表示容器在应用启动时就加载这个servlet；  
当是一个负数时或者没有指定时，则指示容器在该servlet被选择时才加载。  
正数的值越小，启动该servlet的优先级越高。  
三、  
在项目中总会遇到一些关于加载的优先级问题，近期也同样遇到过类似的，所以自己查找资料总结了下，下面有些是转载其他人的，毕竟人家写的不错，自己也就不重复造轮子了，只是略加点了自己的修饰。  
首先可以肯定的是，加载顺序与它们在 web.xml 文件中的先后顺序无关。即不会因为 filter 写在 listener 的前面而会先加载 filter。最终得出的  
结论是：listener –> filter –> servlet  
同时还存在着这样一种配置节：context-param，它用于向 ServletContext 提供键值对，即应用程序上下文信息。我们的 listener， filter 等在初始化时会用到这些上下文中的信息，那么 context-param 配置节是不是应该写在 listener 配置节前呢？实际上 context-param 配置节可写在任意位置，因此真正的加载顺序为：  
context-param –> listener –> filter –> servlet  
对于某类配置节而言，与它们出现的顺序是有关的。以 filter 为例，web.xml 中当然可以定义多个 filter，与 filter 相关的一个配置节是 filter-mapping，这里一定要注意，对于拥有相同 filter-name 的 filter 和 filter-mapping 配置节而言，filter-mapping 必须出现在 filter 之后，否则当解析到 filter-mapping 时，它所对应的 filter-name 还未定义。web 容器启动时初始化每个 filter 时，是按照 filter 配置节出现的顺序来初始化的，当请求资源匹配多个 filter-mapping 时，filter 拦截资源是按照 filter-mapping 配置节出现的顺序来依次调用 doFilter() 方法的。  
servlet 同 filter 类似 ，此处不再赘述。  
由此，可以看出，web.xml 的加载顺序是：context-param -> listener -> filter -> servlet ，而同个类型之间的实际程序调用的时候的顺序是根据对应的 mapping 的顺序进行调用的。  
web.xml文件详解  
Web.xml常用元素  
复制代码代码如下:  
<web-app>    
     <display-name></display-name> 定义了WEB应用的名字    
     <description></description> 声明WEB应用的描述信息    
     <context-param></context-param> context-param元素声明应用范围内的初始化参数。    
     <filter></filter> 过滤器元素将一个名字与一个实现javax.servlet.Filter接口的类相关联。    
     <filter-mapping></filter-mapping> 一旦命名了一个过滤器，就要利用filter-mapping元素把它与一个或多个servlet或JSP页面相关联。    
     <listener></listener> servlet API的版本2.3增加了对事件监听程序的支持，事件监听程序在建立、修改和删除会话或servlet环境时得到通知。Listener元素指出事件监听程序类。    
     <servlet></servlet> 在向servlet或JSP页面制定初始化参数或定制URL时，必须首先命名servlet或JSP页面。Servlet元素就是用来完成此项任务的。    
     <servlet-mapping></servlet-mapping> 服务器一般为servlet提供一个缺省的URL：http://host/webAppPrefix/servlet/ServletName。但是，常常会更改这个URL，以便servlet可以访问初始化参数或更容易地处理相对URL。在更改缺省URL时，使用servlet-mapping元素。    
     <session-config></session-config> 如果某个会话在一定时间内未被访问，服务器可以抛弃它以节省内存。可通过使用HttpSession的setMaxInactiveInterval方法明确设置单个会话对象的超时值，或者可利用session-config元素制定缺省超时值。    
     <mime-mapping></mime-mapping> 如果Web应用具有想到特殊的文件，希望能保证给他们分配特定的MIME类型，则mime-mapping元素提供这种保证。    
     <welcome-file-list></welcome-file-list> 指示服务器在收到引用一个目录名而不是文件名的URL时，使用哪个文件。    
     <error-page></error-page> 在返回特定HTTP状态代码时，或者特定类型的异常被抛出时，能够制定将要显示的页面。    
     <taglib></taglib> 对标记库描述符文件（Tag Libraryu Descriptor file）指定别名。此功能使你能够更改TLD文件的位置，而不用编辑使用这些文件的JSP页面。    
     <resource-env-ref></resource-env-ref> 声明与资源相关的一个管理对象。    
     <resource-ref></resource-ref> 声明一个资源工厂使用的外部资源。    
     <security-constraint></security-constraint> 制定应该保护的URL。它与login-config元素联合使用    
     <login-config></login-config> 指定服务器应该怎样给试图访问受保护页面的用户授权。它与sercurity-constraint元素联合使用。    
     <security-role></security-role> 给出安全角色的一个列表，这些角色将出现在servlet元素内的security-role-ref元素的role-name子元素中。分别地声明角色可使高级IDE处理安全信息更为容易。    
     <env-entry></env-entry> 声明Web应用的环境项。    
     <ejb-ref></ejb-ref> 声明一个EJB的主目录的引用。    
     <ejb-local-ref></ejb-local-ref> 声明一个EJB的本地主目录的应用。    
 </web-app>   
相应元素配置  
1、Web应用图标：指出IDE和GUI工具用来表示Web应用的大图标和小图标  
  
  
复制代码代码如下:  
<icon>    
     <small-icon>/images/app\_small.gif</small-icon>    
     <large-icon>/images/app\_large.gif</large-icon>    
 </icon>   
2、Web 应用名称：提供GUI工具可能会用来标记这个特定的Web应用的一个名称  
复制代码代码如下:  
<display-name>Tomcat Example</display-name>       
3、Web 应用描述：给出于此相关的说明性文本  
复制代码代码如下:  
<disciption>Tomcat Example servlets and JSP pages.</disciption>  
4、上下文参数：声明应用范围内的初始化参数。  
复制代码代码如下:  
<context-param>  
     <param-name>ContextParameter</para-name>  
     <param-value>test</param-value>  
     <description>It is a test parameter.</description>  
 </context-param>  
在servlet里面可以通过 getServletContext().getInitParameter(“context/param”)得到  
5、过滤器配置：将一个名字与一个实现javaxs.servlet.Filter接口的类相关联。  
复制代码代码如下:  
<filter>  
     <filter-name>setCharacterEncoding</filter-name>  
     <filter-class>com.myTest.setCharacterEncodingFilter</filter-class>  
     <init-param>  
         <param-name>encoding</param-name>  
         <param-value>GB2312</param-value>  
     </init-param>  
 </filter>  
 <filter-mapping>  
     <filter-name>setCharacterEncoding</filter-name>  
     <url-pattern>/\*</url-pattern>  
 </filter-mapping>  
6、监听器配置  
复制代码代码如下:  
<listener>  
     <listerner-class>listener.SessionListener</listener-class>  
 </listener>  
7、Servlet配置  
基本配置  
复制代码代码如下:  
<servlet>  
     <servlet-name>snoop</servlet-name>  
     <servlet-class>SnoopServlet</servlet-class>  
 </servlet>  
 <servlet-mapping>  
     <servlet-name>snoop</servlet-name>  
     <url-pattern>/snoop</url-pattern>  
 </servlet-mapping>  
高级配置  
复制代码代码如下:  
<servlet>  
     <servlet-name>snoop</servlet-name>  
     <servlet-class>SnoopServlet</servlet-class>  
     <init-param>  
         <param-name>foo</param-name>  
         <param-value>bar</param-value>  
     </init-param>  
     <run-as>  
         <description>Security role for anonymous access</description>  
         <role-name>tomcat</role-name>  
     </run-as>  
 </servlet>  
 <servlet-mapping>  
     <servlet-name>snoop</servlet-name>  
     <url-pattern>/snoop</url-pattern>  
 </servlet-mapping>  
元素说明  
<servlet></servlet> 用来声明一个servlet的数据，主要有以下子元素：  
<servlet-name></servlet-name> 指定servlet的名称  
<servlet-class></servlet-class> 指定servlet的类名称  
<jsp-file></jsp-file> 指定web站台中的某个JSP网页的完整路径  
<init-param></init-param> 用来定义参数，可有多个init-param。在servlet类中通过getInitParamenter(String name)方法访问初始化参数  
<load-on-startup></load-on-startup>指定当Web应用启动时，装载Servlet的次序。  
当值为正数或零时：Servlet容器先加载数值小的servlet，再依次加载其他数值大的servlet.  
当值为负或未定义：Servlet容器将在Web客户首次访问这个servlet时加载它  
<servlet-mapping></servlet-mapping> 用来定义servlet所对应的URL，包含两个子元素  
<servlet-name></servlet-name> 指定servlet的名称  
<url-pattern></url-pattern> 指定servlet所对应的URL  
8、会话超时配置（单位为分钟）  
复制代码代码如下:  
 <session-config>  
     <session-timeout>120</session-timeout>  
 </session-config>  
9、MIME类型配置  
复制代码代码如下:  
 <mime-mapping>  
     <extension>htm</extension>  
     <mime-type>text/html</mime-type>  
 </mime-mapping>  
10、指定欢迎文件页配置  
复制代码代码如下:  
 <welcome-file-list>  
     <welcome-file>index.jsp</welcome-file>  
     <welcome-file>index.html</welcome-file>  
     <welcome-file>index.htm</welcome-file>  
 </welcome-file-list>  
11、配置错误页面  
一、 通过错误码来配置error-page  
复制代码代码如下:  
 <error-page>  
     <error-code>404</error-code>  
     <location>/NotFound.jsp</location>  
 </error-page>  
上面配置了当系统发生404错误时，跳转到错误处理页面NotFound.jsp。  
二、通过异常的类型配置error-page  
复制代码代码如下:  
 <error-page>  
     <exception-type>java.lang.NullException</exception-type>  
     <location>/error.jsp</location>  
 </error-page>  
上面配置了当系统发生java.lang.NullException（即空指针异常）时，跳转到错误处理页面error.jsp  
12、TLD配置  
复制代码代码如下:  
 <taglib>  
     <taglib-uri>http://jakarta.apache.org/tomcat/debug-taglib</taglib-uri>  
     <taglib-location>/WEB-INF/jsp/debug-taglib.tld</taglib-location>  
 </taglib>  
如果MyEclipse一直在报错,应该把<taglib> 放到 <jsp-config>中  
view source  
复制代码代码如下:  
 <jsp-config>  
     <taglib>  
         <taglib-uri>http://jakarta.apache.org/tomcat/debug-taglib</taglib-uri>  
         <taglib-location>/WEB-INF/pager-taglib.tld</taglib-location>  
     </taglib>  
 </jsp-config>  
13、资源管理对象配置  
复制代码代码如下:  
  
 <resource-env-ref>  
     <resource-env-ref-name>jms/StockQueue</resource-env-ref-name>  
 </resource-env-ref>  
14、资源工厂配置  
复制代码代码如下:  
 <resource-ref>  
     <res-ref-name>mail/Session</res-ref-name>  
     <res-type>javax.mail.Session</res-type>  
     <res-auth>Container</res-auth>  
 </resource-ref>  
配置数据库连接池就可在此配置：  
复制代码代码如下:  
 <resource-ref>  
     <description>JNDI JDBC DataSource of shop</description>  
     <res-ref-name>jdbc/sample\_db</res-ref-name>  
     <res-type>javax.sql.DataSource</res-type>  
     <res-auth>Container</res-auth>  
 </resource-ref>  
15、安全限制配置  
复制代码代码如下:  
 <security-constraint>  
     <display-name>Example Security Constraint</display-name>  
     <web-resource-collection>  
         <web-resource-name>Protected Area</web-resource-name>  
         <url-pattern>/jsp/security/protected/\*</url-pattern>  
         <http-method>DELETE</http-method>  
         <http-method>GET</http-method>  
         <http-method>POST</http-method>  
         <http-method>PUT</http-method>  
     </web-resource-collection>  
     <auth-constraint>  
         <role-name>tomcat</role-name>  
         <role-name>role1</role-name>  
     </auth-constraint>  
 </security-constraint>  
16、登陆验证配置  
复制代码代码如下:  
 <login-config>  
     <auth-method>FORM</auth-method>  
     <realm-name>Example-Based Authentiation Area</realm-name>  
     <form-login-config>  
         <form-login-page>/jsp/security/protected/login.jsp</form-login-page>  
         <form-error-page>/jsp/security/protected/error.jsp</form-error-page>  
     </form-login-config>  
 </login-config>  
17、安全角色：security-role元素给出安全角色的一个列表，这些角色将出现在servlet元素内的security-role-ref元素的role-name子元素中。  
分别地声明角色可使高级IDE处理安全信息更为容易。  
复制代码代码如下:  
 <security-role>  
     <role-name>tomcat</role-name>  
 </security-role>  
18、Web环境参数：env-entry元素声明Web应用的环境项  
复制代码代码如下:  
 <env-entry>  
     <env-entry-name>minExemptions</env-entry-name>  
     <env-entry-value>1</env-entry-value>  
     <env-entry-type>java.lang.Integer</env-entry-type>  
 </env-entry>  
19、EJB 声明  
复制代码代码如下:  
 <ejb-ref>  
     <description>Example EJB reference</decription>  
     <ejb-ref-name>ejb/Account</ejb-ref-name>  
     <ejb-ref-type>Entity</ejb-ref-type>  
     <home>com.mycompany.mypackage.AccountHome</home>  
     <remote>com.mycompany.mypackage.Account</remote>  
 </ejb-ref>  
20、本地EJB声明  
复制代码代码如下:  
 <ejb-local-ref>  
     <description>Example Loacal EJB reference</decription>  
     <ejb-ref-name>ejb/ProcessOrder</ejb-ref-name>  
     <ejb-ref-type>Session</ejb-ref-type>  
     <local-home>com.mycompany.mypackage.ProcessOrderHome</local-home>  
     <local>com.mycompany.mypackage.ProcessOrder</local>  
 </ejb-local-ref>  
21、配置DWR  
复制代码代码如下:  
 <servlet>  
     <servlet-name>dwr-invoker</servlet-name>  
     <servlet-class>uk.ltd.getahead.dwr.DWRServlet</servlet-class>  
 </servlet>  
 <servlet-mapping>  
     <servlet-name>dwr-invoker</servlet-name>  
     <url-pattern>/dwr/\*</url-pattern>  
 </servlet-mapping>  
22、配置Struts  
复制代码代码如下:  
 <display-name>Struts Blank Application</display-name>  
 <servlet>  
     <servlet-name>action</servlet-name>  
     <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>  
     <init-param>  
         <param-name>detail</param-name>  
         <param-value>2</param-value>  
     </init-param>  
     <init-param>  
         <param-name>debug</param-name>  
         <param-value>2</param-value>  
     </init-param>  
     <init-param>  
         <param-name>config</param-name>  
         <param-value>/WEB-INF/struts-config.xml</param-value>  
     </init-param>  
     <init-param>  
         <param-name>application</param-name>  
         <param-value>ApplicationResources</param-value>  
     </init-param>  
     <load-on-startup>2</load-on-startup>  
 </servlet>  
 <servlet-mapping>  
     <servlet-name>action</servlet-name>  
     <url-pattern>\*.do</url-pattern>  
 </servlet-mapping>  
 <welcome-file-list>  
     <welcome-file>index.jsp</welcome-file>  
 </welcome-file-list>  
 <!-- Struts Tag Library Descriptors -->  
 <taglib>  
     <taglib-uri>struts-bean</taglib-uri>  
     <taglib-location>/WEB-INF/tld/struts-bean.tld</taglib-location>  
 </taglib>  
 <taglib>  
     <taglib-uri>struts-html</taglib-uri>  
     <taglib-location>/WEB-INF/tld/struts-html.tld</taglib-location>  
 </taglib>  
 <taglib>  
     <taglib-uri>struts-nested</taglib-uri>  
     <taglib-location>/WEB-INF/tld/struts-nested.tld</taglib-location>  
 </taglib>  
 <taglib>  
     <taglib-uri>struts-logic</taglib-uri>  
     <taglib-location>/WEB-INF/tld/struts-logic.tld</taglib-location>  
 </taglib>  
 <taglib>  
     <taglib-uri>struts-tiles</taglib-uri>  
     <taglib-location>/WEB-INF/tld/struts-tiles.tld</taglib-location>  
 </taglib>  
23、配置Spring（基本上都是在Struts中配置的）

复制代码代码如下:

 <!-- 指定spring配置文件位置 -->  
 <context-param>  
     <param-name>contextConfigLocation</param-name>  
     <param-value>  
         <!--加载多个spring配置文件 -->  
         /WEB-INF/applicationContext.xml, /WEB-INF/action-servlet.xml  
     </param-value>  
 </context-param>  
 <!-- 定义SPRING监听器，加载spring --> <listener>  
     <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  
 </listener>  
 <listener>  
     <listener-class>org.springframework.web.context.request.RequestContextListener</listener-class>  
 </listener>