# [web.xml配置详解](http://blog.csdn.net/guihaijinfen/article/details/8363839)

# 引文：对于一个J2EE领域的程序员而言，基本上每天都会和web应用打交道。什么是web应用？最简单的web应用什么样？给你一个web应用你该从何入手？

1、什么是web应用？

web应用是一种可以通过Web访问的应用程序。在J2EE领域下，web应用就是遵守基于**[Java](http://lib.csdn.net/base/17" \o "Java EE知识库" \t "http://blog.csdn.net/guihaijinfen/article/details/_blank)**技术的一系列标准的应用程序。

2、最简单的web应用什么样？

**2个文件夹、1个xml文件就能成为一个web应用**

第一个文件夹：应用名，如test

第二个文件夹：在test文件夹中创建名为WEB-INF的文件夹

XML文件：在WEB-INF文件夹下创建web.xml文件，文件内容只需<web-app></web-app>

3、给你一个web应用你该从何入手？

这属于仁者见仁智者见智的问题，我通常看一个web应用是从web.xml入手的IMG_258

正文：

1、web.xml学名叫部署描述符文件，是在Servlet规范中定义的，是web应用的配置文件。

2、部署描述符文件就像所有XML文件一样，必须以一个XML头开始。这个头声明可以使用的XML版本并给出文件的字符编码。DOCYTPE声明必须立即出现在此头之后。这个声明告诉服务器适用的servlet规范的版本（如2.2或2.3）并指定管理此文件其余部分内容的语法的DTD(Document Type Definition，文档类型定义)。所有部署描述符文件的顶层（根）元素为web-app。请注意，XML元素不像HTML，他们是大小写敏感的。因此，web-App和WEB-APP都是不合法的，web-app必须用小写。

      web.xml样例：

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. <!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN" "http://java.sun.com/dtd/web-app\_2\_3.dtd"**>**
3. **<web-app>**
4. **</web-app>**

 3、web.xml中XML 元素不仅是大小写敏感的，而且它们还对出现在其他元素中的次序敏感。例如，XML头必须是文件中的第一项，DOCTYPE声明必须是第二项，而web- app元素必须是第三项。在web-app元素内，元素的次序也很重要。服务器不一定强制要求这种次序，但它们允许（实际上有些服务器就是这样做的）完全拒绝执行含有次序不正确的元素的Web应用。这表示使用非标准元素次序的web.xml文件是不可移植的。

       元素顺序列表：



      元素标签详解：

**元素1：<icon>**

含义

icon元素包含small-icon和large-icon两个子元素.用来指定web站台中小图标和大图标的路径.  
<small-icon>/路径/smallicon.gif</small-icon>  
small-icon元素应指向web站台中某个小图标的路径,大小为16 X 16 pixel,但是图象文件必须为GIF或JPEG格式,扩展名必须为:.gif或.jpg.

<large-icon>/路径/largeicon-jpg</large-icon>  
large-icon元素应指向web站台中某个大图表路径,大小为32 X 32 pixel,但是图象文件必须为GIF或JPEG的格式,扩展名必须为; gif或jpg.

范例

<icon>  
   <small-icon>/images/small.gif</small-icon>  
   <large-icon>/images/large.gif</large-icon>  
</icon>

**元素2、3：<display-name>，<description>**

含义

<display-name>应用名称</display-name>  
定义应用的名称。

<description>应用描述</discription>  
对应用做出描述。

范例

<display-name>test</display-name>

<description>测试应用V1.0</discription>

**元素4：<context-param>**

含义

context-param 元素用来设定web应用的环境参数(context),它包含两个子元素:  
param-name和param-value.  
<param-name>参数名称</param-name>  
设定Context名称  
<param-value>值</param-value>  
设定Context名称的值

范例  
<context-param>  
   <param-name>param\_name</param-name>  
   <param-value>param\_value</param-value>  
</context-param>  
此所设定的参数,在JSP网页中可以使用下列方法来取得:  
${initParam.param\_name}  
若在Servlet可以使用下列方法来获得:  
String param\_name=getServletContext().getInitParamter("param\_name");

**元素5，6：<filter>，<filter-mapping>**

含义

filter元素用来设定web应用的过滤器，它的两个主要子元素filter-name和filter-class用来定义Filter所对应的class

<filter-name>Filter的名称</filter-name>  
定义Filter的名称  
<filter-class>Filter的类名称</filter-class>  
定义Filter的类名称

filter-mapping 元素的两个主要子元素filter-name和url-pattern.用来定义Filter所对应的URL.  
<filter-name>Filter的名称</filter-name>  
定义Filter的名称.  
<url-pattern>URL</url-pattern>  
Filter所对应的RUL.例如:<url-pattern>/Filter/\*</url-pattern>

范例

<filter>  
  <filter-name>Encoding</filter-name>  
  <filter-class>ghjf.test.filter.SetCharacterEncodingFilter</filter-class>  
  <init-param>  
     <param-name>encoding</param-name>  
     <param-value>GBK</param-value>  
  </init-param>  
</filter>

<filter-mapping>  
   <filter-name>Encoding</filter-name>  
   <url-pattern>/\*</url-pattern>  
</filter-mapping>

**元素7：<listener>**

含义

listener元素用来定义Listener接口,它的主要子元素为<listener-class>  
<listen-class>Listener的类名称</listener-class>  
定义Listener的类名称  
<listener>

范例

<listener>  
  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  
</listener>

**元素8、9：<servlet>、<servlet-mapping>**

含义

servlet元素的两个主要子元素servlet-name和servlet-class用来定义servlet所对应的class

<servlet-name>servlet的名称</servlet-name>  
定义servlet的名称  
<servlet-class>servlet的类名称</servlet-class>  
定义servlet的类名称

servlet-mapping元素包含两个子元素servlet-name和url-pattern.用来定义servlet所对应URL.  
<servlet-name>Servlet的名称</servlet-name>  
定义Servlet的名称.  
<url-pattern>Servlet URL</url-pattern>  
定义Servlet所对应的RUL.例如:<url-pattern>/Servlet/\*</url-pattern>  
</servlet-mapping>

范例

<servlet>  
 <servlet-name>dwr-invoker</servlet-name>  
 <display-name>DWR Servlet</display-name>  
 <description>Direct Web Remoter Servlet</description>  
 <servlet-class>uk.ltd.getahead.dwr.DWRServlet</servlet-class>  
 <init-param>  
     <param-name>debug</param-name>  
     <param-value>true</param-value>  
 </init-param>  
</servlet>

<servlet-mapping>  
 <servlet-name>dwr-invoker</servlet-name>  
 <url-pattern>/dwr/\*</url-pattern>  
</servlet-mapping>

**元素10：<session-cofing>**

含义

session-config包含一个子元素session-timeout.定义web应用中的session参数.  
<session-timeout>分钟</session-timeout>  
定义这个web站台所有session的有效期限.单位为分钟.

范例

<session-config>  
   <session-timeout>30</session-timeout>  
</session-config>

**元素11：<mime-mapping>**

含义

mime-mapping包含两个子元素extension和mime-type.定义某一个扩展名和某一MIME Type做对映.  
<extension>扩展名名称</extension>  
扩展名称  
<mime-type>MIME格式</mime-type>  
MIME格式.

范例

<mime-mapping>  
   <extension>doc</extension>  
   <mime-type>application/vnd.ms-word</mime-type>  
</mime-mapping>  
<mime-mapping>  
   <extension>xls</extension>  
   <mime-type>application/vnd.ms-excel</mime-type>  
</mime-mapping>  
<mime-mapping>  
   <extension>ppt</extesnion>  
   <mime-type>application/vnd.ms-powerpoint</mime-type>  
</mime-mapping>

**元素12：<welcome-file-list>**

含义

welcome-file-list包含一个子元素welcome-file.用来定义首页列单.  
<welcome-file>用来指定首页文件名称</welcome-flie>  
welcome-file用来指定首页文件名称.我们可以用<welcome-file>指定几个首页,而服务器会依照设定的顺序来找首页.

范例

<welcome-file-list>  
  <welcome-file>index.jsp</welcome-file>  
  <welcome-file>index.htm</welcome-file>  
</welcome-file-list>

**元素13：<error-page>**

含义

error-page元素包含三个子元素error-code,exception-type和location.将错误代码(Error Code)或异常(Exception)的种类对应到web应用资源路径.  
<error-code>错误代码</error-code>  
HTTP Error code,例如: 404、403  
<exception-type>Exception</exception-type>  
一个完整名称的Java异常类型  
<location>/路径</location>  
在web应用内的相关资源路径

范例

<error-page>  
   <error-code>404</error-code>  
   <location>/error404.jsp</location>  
</error-page>  
<error-page>  
   <exception-type>java.lang.Exception</exception-type>  
   <location>/exception.jsp</location>  
</error-page>

**元素14：<jsp-config>**

含义

jsp-config元素主要用来设定JSP的相关配置,<jsp:config>包括<taglib>和<jsp-property-group>两个子元素.其中<taglib>元素  
在JSP 1.2时就已经存在了;而<jsp-property-group>是JSP 2.0新增的元素.

<taglib>  
taglib元素包含两个子元素taglib-uri和taglib-location.用来设定JSP网页用到的Tag Library路径.  
<taglib-uri>URI</taglib-uri>  
   taglib-uri定义TLD文件的URI,JSP网页的taglib指令可以经由这个URI存取到TLD文件.  
<taglib-location>/WEB-INF/lib/xxx.tld</taglib-laction>  
   TLD文件对应Web站台的存放位置.  
</taglib>

<jsp-property-group>  
jsp-property-group元素包含8个元素,分别为:  
<description>Description</descrition>  
此设定的说明

<display-name>Name</display-name>  
此设定的名称

<url-pattern>URL</url-pattern>  
设定值所影响的范围,如:/CH2 或者/\*.jsp

<el-ignored>true|false</el-ignored>  
若为true,表示不支持EL语法.

<scripting-invalid>true|false</scripting-invalid>  
若为true表示不支持<%scription%>语法.

<page-encoding>encoding</page-encoding>  
设定JSP网页的编码

<include-prelude>.jspf</include-prelude>  
设置JSP网页的抬头,扩展名为.jspf

<include-coda>.jspf</include-coda>  
设置JSP网页的结尾,扩展名为.jspf  
</jsp-property-group>  
</jsp-config>

范例

<jsp-config>  
<taglib>  
   <taglib-uri>Taglib</taglib-uri>  
   <taglib-location>/WEB-INF/tlds/MyTaglib.tld</taglib-location>  
</taglib>  
<jsp-property-group>  
   <description>  
      Special property group for JSP Configuration JSP example.  
   </description>  
   <display-name>JSPConfiguration</display-name>  
   <uri-pattern>/\*</uri-pattern>  
   <el-ignored>true</el-ignored>  
   <page-encoding>GB2312</page-encoding>  
   <scripting-inivalid>true</scripting-inivalid>  
</jsp-property-group>  
</jsp-config>

**元素15：<resource-env-ref>**

含义

resource-env-ref有两个子元素：  
<resource-env-ref-name>资源名</resource-env-ref-name>    
资源的名称 相对于java:comp/env  
<resource-env-ref-type>查找资源时返回的资源类名</resource-env-ref-type>  
当web应用查找该资源的时候，返回的Java类名的全称

范例

 <resource-env-ref>    
  <resource-env-ref-name>jdbc/mssql</resource-env-ref-name>    
  <resource-env-ref-type>javax.sql.DataSource</resource-env-ref-type>    
 </resource-env-ref>

**元素16：<resource-ref>**

含义

resource-ref元素包括五个子元素description,res-ref-name,res-type,res-auth,res-sharing-scope.利用JNDI取得应用可利用资源.  
<description>说明</description>  
资源说明  
<rec-ref-name>资源名称</rec-ref-name>  
资源名称  
<res-type>资源种类</res-type>  
资源种类  
<res-auth>Application|**[Container](http://lib.csdn.net/base/4" \o "Docker知识库" \t "http://blog.csdn.net/guihaijinfen/article/details/_blank)**</res-auth>  
资源由Application或Container来许可  
<res-sharing-scope>Shareable|Unshareable</res-sharing-scope>  
 资源是否可以共享.默认值为 Shareable

范例

<resource-ref>  
   <description>JNDI JDBC DataSource</description>  
   <res-ref-name>jdbc/data</res-ref-name>  
   <res-type>javax.sql.DataSoruce</res-type>  
   <res-auth>Container</res-auth>  
</resource-ref>