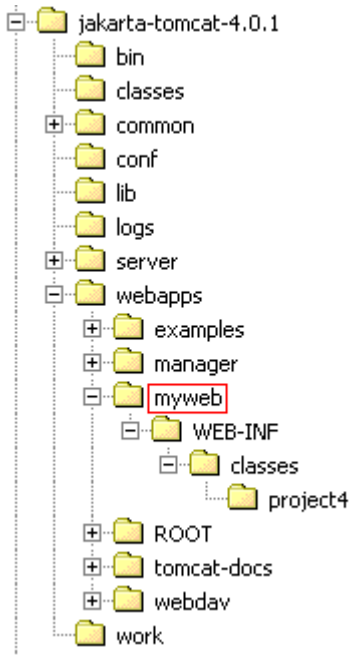
Tomcat 的配置

(王帅 swang@censoft.com.cn)

增加一个虚拟目录

在 server.xml 文件中增加
<Context path="/oicq" docBase="myweb" debug="0" reloadable="true">
</Context>
myweb 说明其相对 webapps 的位置，是物理存在的目录；
/oicq 说明其相对 web URL 的路径，是一个虚拟的路径，如：http://localhost/oicq

配置 JSP 及 Servlet



JSP 文件直接放在 myweb 下；
编译好的 JavaBean、Servlet 放在 WEB-INF 下的 classes 目录，而且包的路径要与目录路径一致。

配置服务器的端口

在 server.xml 文件的第 56 行，修改 port = “8080” 为你所希望使用的端口号，如：80

web.xml 文件的设置

默认(欢迎）文件的设置

在 h:\tomcat4\conf\web.xml 中，<welcome-file-list>与 IIS 中的默认文件意思相同。

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

报错文件的设置

```
<error-page>
    <error-code>404</error-code>
    <location>/notFileFound.jsp</location>
</error-page>
<error-page>
    <exception-type>java.lang.NullPointerException</exception-type>
    <location>/null.jsp</location>
</error-page>
```

如果某文件资源没有找到，服务器要报 404 错误，按上述配置则会调用 H:\tomcat4\webapps\ROOT\notFileFound.jsp。

如果执行的某个 JSP 文件产生 NullPointException ，则会调用 H:\tomcat4\webapps\ROOT\null.jsp

典型的 JSP 错误页面应该这样写：

```
<%@ page isErrorPage=" true" %>
出错了： (</p>
错误信息: <%= exception.getMessage() %><br>

Stack Trace is : <pre><font color="red"><%

 java.io.CharArrayWriter cw = new java.io.CharArrayWriter();

 java.io.PrintWriter pw = new java.io.PrintWriter(cw,true);
```

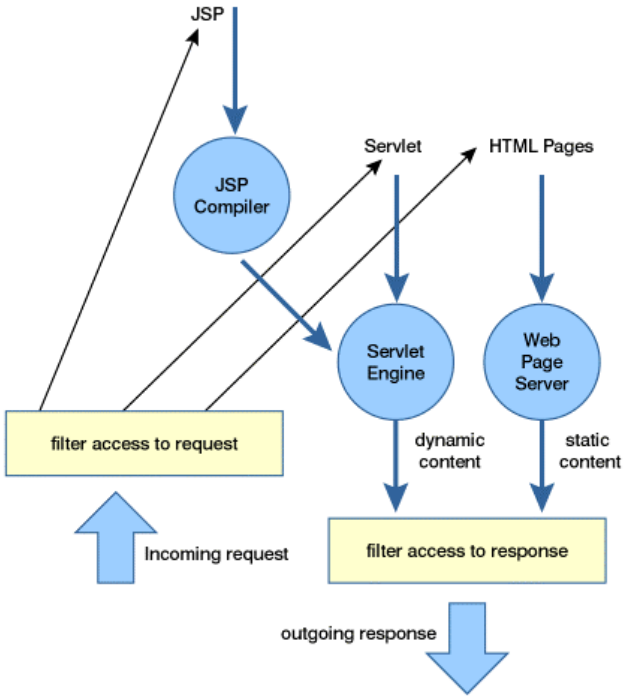
```
exception.printStackTrace(pw);

out.println(cw.toString());

%></font></pre>
```

会话超时的设置

设置 session 的过期时间，单位是分钟；

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

过滤器的设置




```
<filter>
    <filter-name>FilterSource</filter-name>
    <filter-class>project4. FilterSource </filter-class>
  </filter>
  <filter-mapping>
    <filter-name>FilterSource</filter-name>
    <url-pattern>/WwwServlet</url-pattern>
    (<url-pattern>/haha/*</url-pattern>)
```

```
    </filter-mapping>
 </filter>
```

过滤：
1) 身份验证的过滤 Authentication Filters
2) 日志和审核的过滤 Logging and Auditing Filters
3) 图片转化的过滤 Image conversion Filters
4) 数据压缩的过滤 Data compression Filters
5) 加密过滤 Encryption Filters
6) Tokenizing Filters
7) 资源访问事件触发的过滤 Filters that trigger resource access events
8) XSL/T 过滤　　XSL/T filters
9) 内容类型的过滤　　Mime-type chain Filter
注意监听器的顺序，如：先安全过滤，然后资源，然后内容类型等，这个顺序可以自己定，但最好要合理。

监听器的设置

```
<listener>
<listener-class>project4.SALListenerServlet</listener-class>
</listener>
```

监听器分四种，分别是：
ServletContextListener　:对上下文(全局)对象的创建和销毁进行监听
ServletContextAttributeListener：对上下文对象某一属性的增加、替换、删除进行监听
HttpSessionListener：对 Session 的创建和销毁进行监听
HttpSessionAttributeListener：对 Session 某一属性的增加、替换、删除进行监听

Servlet 的设置

```
<servlet>
    <servlet-name>HelloServlet</servlet-name>
    <servlet-class>project4.HelloServlet</servlet-class>
    <init-param>
        <param-name>age</param-name>
        <param-value>26</param-value>
    </init-param>
    <init-param>
        <param-name>ip</param-name>
        <param-value>192.168.5.65</param-value>
    </init-param>
</servlet>
```

```
<servlet-mapping>
    <servlet-name>HelloServlet</servlet-name>
    <url-pattern>/HelloServlet</url-pattern>
</servlet-mapping>
```

解释：

<servlet>内的<servlet-name> 是一个逻辑名，可以是任何有效的标识名，可以将上述配置中的两个<servlet-name>HelloServlet</servlet-name>同时改成<servlet-name>qq</servlet-name>，得到的效果相同，注意要"同时改"，这样可以继续保持对应的关系。

<init-param>是 Servlet 初始参数，在 Servlet 的 init()方法中通过 getInitParameter("ip")取得，返回 String 型数据，

<servlet-mapping>内的<servlet-name>与<servlet>内的 <servlet-name>一一对应，把客户端对/HeeloServlet 的请求对应到<servlet-class>project4.HelloServlet</servlet-class>所指定的位置。

<url-pattern>/HelloServlet</url-pattern>指在 IE url 中的请求形式。这里的 / 是相对于当前的 web 目录的，如 H:\tomcat4\webapps\myweb

tomcat-users.xml 设置

```
<tomcat-users>
    <user name="tomcat" password="tomcat" roles="tomcat" />
    <user name="role1" password="tomcat" roles="role1" />
    <user name="both" password="tomcat" roles="tomcat,role1" />
    <user name="wang" password="tomcat" roles="admin" />
</tomcat-users>
```

定义用户名和用户所属的角色，在安全性访问中起作用，如 Basic，Form 等加密方式。

---

server.xml

```
<Context path="/icq" docBase="myweb" debug="0" reloadable="true">
```

```
    <Logger className="org.apache.catalina.logger.FileLogger"
     prefix="localhost_icq_log." suffix=".txt"
     timestamp="true"/>

</Context>
```

Logger 段，为 icq 这个应用目录建立一个日志文件；
Prefix 是日志文件的前缀；
suffix 是日志文件的后缀；
myweb 说明其相对 webapps 的位置，是物理存在的目录；
/icq 说明其相对 web URL 的路径，是一个虚拟的路径，如：http://localhost/icq
reloadable 在开发时比较有用，指自动载入新的 Servlet 类。

```
<Connector
className="org.apache.catalina.connector.http.HttpConnector"
port="8080" minProcessors="5" maxProcessors="75"
...
...
```

大概在 server.xml 中的第 55、56 行，配置服务器的端口。

配置日志

日志文件有四个，在 jakarta-tomcat-4.0.1\logs 目录下，

catalina_log 在 125 行
catalina_log.2002-06-20.txt
记录了 tomcat 服务器启动的相关信息

localhost_access_log 在 180 行：
localhost_access_log.2002-06-20.txt
用来记录客户端访问了哪些资源，格式如：
172.28.11.91 - - [20/Jun/2002:13:29:09 8000] "GET /web/WwwServlet HTTP/1.1" 500 149
200 149 是服务器反应的状态码

localhost_examples_log
localhost_examples_log.2002-06-20.txt 在 213 行
这个是 examples 的日志文件

localhost_log
localhost_log.2002-06-20.txt 在 190 行

这里记录了服务器所运行的程序的详细信息。包括错误信息，调试信息等。
用如下写法向这个日志文件写入调试信息
在 Servlet 中用　getServletContext().log("Servlet 中报的错！");
或在 JSP 中用　　application.log("哈哈，出错了：（");
(ServletContext 同 application 是一回事)

web.xml 文件中安全性的设置

```
<security-constraint>
    <display-name>test</display-name>
    <web-resource-collection>
    <web-resource-name>Success</web-resource-name>
    <url-pattern>/HelloServlet</url-pattern>
    <http-method>GET</http-method>
    <http-method>POST</http-method>
    </web-resource-collection>
    <auth-constraint>
    <role-name>admin</role-name>
    </auth-constraint>
    <user-data-constraint>
    <transport-guarantee>NONE</transport-guarantee>
    </user-data-constraint>
</security-constraint>
<login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>China of Beijing : ) </realm-name>
</login-config>
```

说明：

<display-name>和<web-resource-name>可以是任意，但最好起一个有意义的名。
<auth-constraint>中<role-name>约束了只有哪些角色可以访问由
<url-pattern>指定的资源。

<http-method>对指定方法的访问进行限定，未指出的不进行限定。

<transport-guarantee>指明对转输的数据的要求，有三个可选值：NONE，
INTEGRAL,CONFIDENTIAL

<login-config>:
内说明的是以何种方式进行身份验证，有三种可选值：
None,Digest,Client-Cert,Basic,Form。

<realm-name>：区域名，可以是你想给客户看的提示信息。

另一种验证方式：

```
<login-config>
<auth-method>FORM</auth-method>
<form-login-config>
<form-login-page>/login.htm</form-login-page>
<form-error-page>/loginError.htm</form-error-page>
</form-login-config>
</login-config>
```

login.htm 文件：

```
<form method="POST" action="j_security_check" >
用户名：<input type="text" name="j_username"> </p>
密  码：<input type="password" name="j_password"> </p>
<input type="submit" value="让我进去吧！">
</form>
```

其中红字部分不能更改，并且区分大小写