Java Servlet 实验

段 炼

2008-9-3

1.0 简介

内容：

1. 开发环境设定，包括安装 JDK 及 Tomcat。
2. 撰写一支简单的 servlet 程序。
3. 将 servlet 部署到 Tomcat 环境。

2.0 准备你的开发环境

2.1 安装 Java SDK

下载 J2SDK 和 JRE. 可以在 http://java.sun.com 下载. 注意 Tomcat6 需要的是 JRE 6

安装并设置环境变量. 设置 JAVA_HOME 变量为 java 的主目录. 把 java 的 bin 目录路径添加到 PATH 环境变量中.

测试 Java 环境. 进入 命令提示符(开始 - 运行 - cmd),输入 java -version,看版本对不对. 输入 javac -help 看是不是正确的提示,如果提示"不是内部活外部命令,也不是可运行的程序或批处理文件",则没有把 Path 路径设置好。

2.2 安装 Tomcat 6

1. 至 http://jakarta.apache.org/ 下载 Tomcat 安装程序（下载附文件名是 .exe 的档案）。

2. 将 Tomcat 安装至目录 "c:\Tomcat"。如果你希望每次开机就自动启动 Tomcat 服务，在安装过程中要将 "Service" 项目打勾，参考图 2。

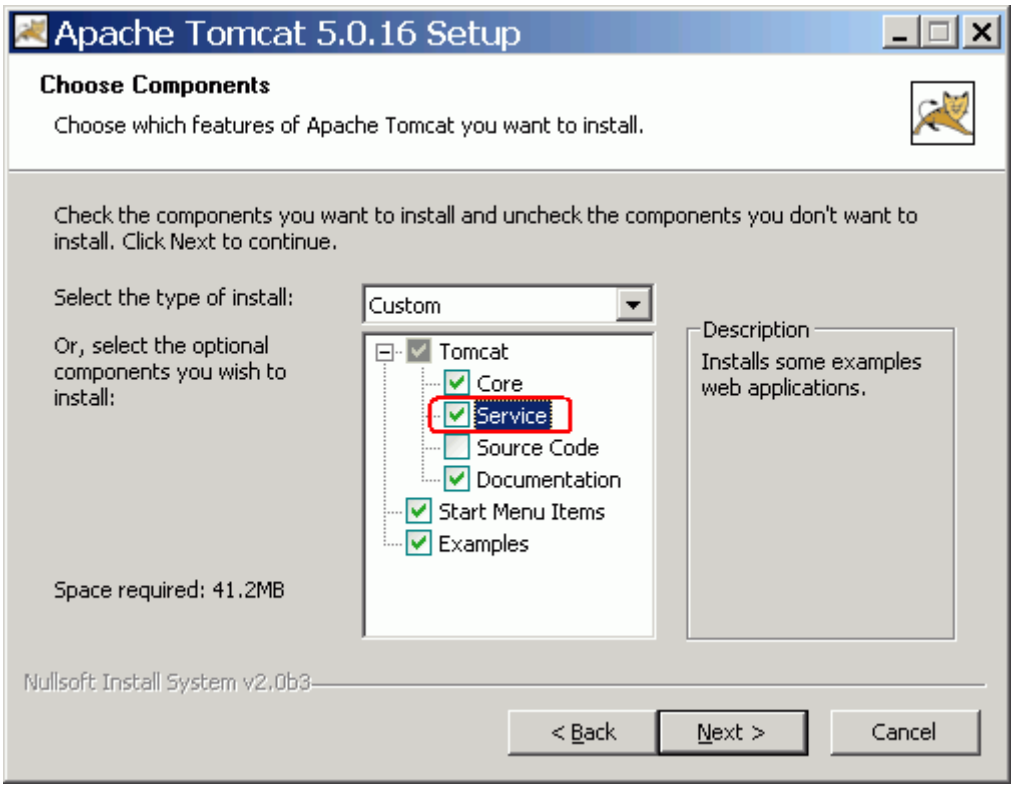

图 2

3. 安装完成后，增加一个系统环境变量 "CATALINA_HOME"，值为 "c:\tomcat"。
4. 现在 Tomcat 服务应该已经自动启动了，开启浏览器，并且输入 URL "http://127.0.0.1:8080"（http 一定要输入），如果安装无误，应会看到如图 3 的画面。

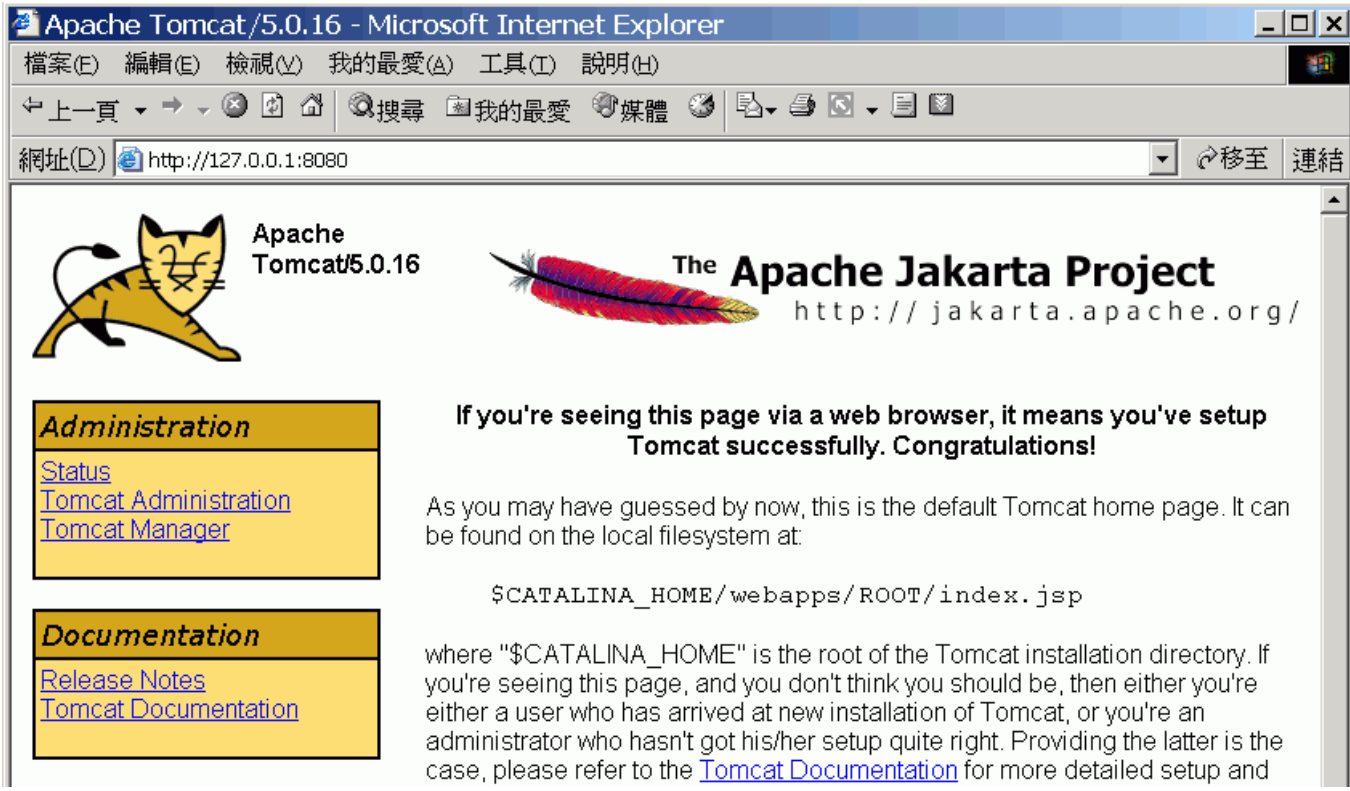

图 3

2.3 设定 CLASSPATH 环境变量

编译 serlvet 程序时必须让 javac 编译器找得到 HttpServlet 和相关的类别，有两种方法：

- 把 %CATALINA_HOME%\common\lib 目录下的 servlet-api.jar 和 jsp-api.jar 复制到 %JAVA_HOME%\jre\lib\ext 目录下。
- 把 .;%CATALINA_HOME%\common\lib\servlet-api.jar;%CATALINA_HOME%\common\lib\jsp-api.jar" 加入系统环境变量 CLASSPATH 中。注意前面的 ".;" 一定要加，它代表的是每次编译时的现行目录。

两种方法都可以。

此外，为了在不修改 web.xml 文件的情况下也能运新 servlet，具体做法如下，在 tomcat 的根目录下，如安转目录是 D:\Apache Software Foundation\Tomcat 6 找到

conf 文件夹下的 web.xml 文件.把其中如下的 servlet 和 servlet-mapping 元素注释去掉:

```
<servlet>
<servlet-name>invoker</servlet-name>
<servlet-class>
org.apache.catalina.servlets.InvokerServlet
</servlet-class>
...
</servlet>
...
<servlet-mapping>
<servlet-name>invoker</servlet-name>
<url-pattern>/servlet/*</url-pattern>
</servlet-mapping>
```

但这一步不提倡。

3.0 撰写 Servlet

开启文字编辑器（建议使用 UltraEdit，也可以使用 Netbeans），创建一个输出“Hello! 这是我的第一次 Java servlet 程序。”的 Servlet 类，类名为 HelloServlet.java。输入下列程序代码，并且存盘成 HelloServlet.java。

```
// Filename: HelloServlet.java
// Compile: javac HelloServlet.java
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloServlet extends HttpServlet {
```

```
    public void doGet(HttpServletRequest request, HttpServletResponse response)
                                throws ServletException, IOException {
        // 下面两行让中文字能正确显示
        response.setContentType("text/html;charset=UTF-8");
        request.setCharacterEncoding("UTF-8");

        PrintWriter out = response.getWriter();
        out.println("<HTML>");
        out.println("<BODY>");
        out.println("<p>Hello! 这是我的第一次 Java servlet 程序。</p>");
        out.println("</BODY>");
        out.println("</HTML>");
  }
}
```

存档完后，编译 HelloServlet.java，在 DOS 窗口下输入命令：

javac HelloServlet.java

若编译无误，会产生一个 HelloServlet.class。请参考相关书籍以了解程序代码结构以及相关细节。

若编译时出现错误，错误讯息中包含字符串 "package javax.servlet.http does not exist"，表示 Java 编译器找不到 javax.servlet.http 这个 package，请回到步骤 2.3 检查看看哪里出问题了。

4.0 布署

Tomcat 安装完成后，已经有一些编译好的 servlet 范例可以执行，这些范例都在 "c:\Tomcat\webapps\examples\" 目录以下，但我们希望自己的程序布署在独立的目录下，假设我们希望自己写的 servlet 都放在 myapp 目录下，请按下列步骤进行布署：

1. 建立以下目录结构："%Tomcat 安装目录%\webapps\myapp\WEB-INF\classes\"。请注意大小写是有区别的。
2. 把之前编译后产生的 HelloServlet.class 复制到 "%Tomcat 安装目录%\webapps\myapp\WEB-INF\classes\" 目录下。
3. 开启文字编辑器，将下列文字输入（剪贴亦可）并存盘，档名为 %Tomcat 安装目录%\webapps\myapp\WEB-INF\web.xml"。

表 1

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
    <servlet>
      <servlet-name>
          HelloServlet
      </servlet-name>
      <servlet-class>
          HelloServlet
      </servlet-class>
    </servlet>

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

```
</web-app>
```

4. 其中 <servlet-mapping> 标签可以将网址和实际执行的 servlet 路径做转换对应，请特别注意 <url-pattern> 这个标签的内容是 "/HelloServlet"，前面的斜线是必要的，这个标签值代表使用者在浏览器上输入的网址的一部分，当使用者输入的网址符合这个 URL 样式时，就会转而呼叫你指定的 servelt，也就是 HelloServlet。因为此 servlet 范例程序的根目录是 "myapp"，所以加上 "/HelloServlet" 之后，浏览的网址就是 "http://127.0.0.1:8080/myapp/HelloServlet"。
5. 开启浏览器，在网址列输入 URL "http://127.0.0.1:8080/myapp/HelloServlet"，如果可以看到如图 4 的画面，就表示成功了。如果发生错误，请重头检查每一个步骤，看看哪里出问题了，例如：Tomcat 服务是否已经启动了，环境变量是否设定正确，大小写是否都输入正确....等。

图 4

以上的步骤主要是先建立子目录，复制文件，然后再编辑 web.xml 让 tomcat 可以根据网址找到对应的 servlet 档案。许多书籍和文件都是让人修改 "c:\tomcat\conf\server.xml"，在其中加入一个 <context> 卷标并设定相关路径，如果同学们有兴趣的话也可以尝试看看，但并不建议去改 server.xml，因为这是 Tomcat 专属的设定方式，若程序有一天要布署到别的 Web 服务器，还要费工夫重新设定，而 web.xml 则是通用的方式。

第一个范例成功后，如果你还不太了解程序代码的每个细节，请到网站和相关书籍中寻找线索，了解每个环节，并且试着动手修改一些程序代码，重新布署，看看修改后的结果。

每当你修改 servlet 程序，重新编译及布署之后，Tomcat 不会自动侦测到并重新加载新的 servlet class，解决方法请参阅后面。

5.0 练习撰写其它范例

如果这个范例已经没有问题，你可以试着练习写一个网页，可以让使用者输入名称，并且有一个〔送出查询〕按钮，如图 5：

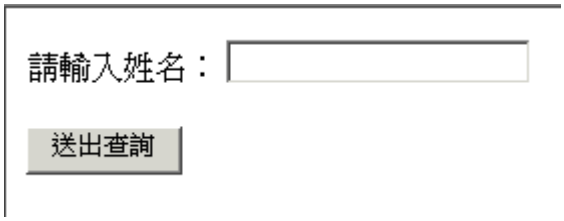

图 5

网页的原始码如表 2：

表 2. Hello.htm

```
<html>
<body>
    <form method=get action="/myapp/Hello">
    请输入姓名：
    <input type=text name="UserID"><p>
    <input type=submit>
    </form>
</body>
</html>
```

如果使用者在姓名栏输入 "残剑飞雪"，然后按下〔送出查询〕钮，该查询请求会传送到你的 servlet 程序，你的程序则会传回如图 6 的画面（注意网址列的内容）：

图 6

Servlet 的原始码列在表 3。

表 3. Hello.class

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Hello extends HttpServlet {
      public void doGet(HttpServletRequest request, HttpServletResponse response)
                  throws ServletException, IOException {

            // 下面两行让中文字能正确显示
            response.setContentType("text/html; charset=big5");
            request.setCharacterEncoding("big5");

            PrintWriter out = response.getWriter();

            String UserID = request.getParameter("UserID");
            out.println("<html><body>");
            out.println("您好, " + UserID);
```

```
            out.println("</body></html>");
        }

        public String getServletInfo() {
            return "A servlet that knows the user ID";
        }
}
```

编译之后的档案可以跟第一个范例布署到同一个目录下，但 web.xml 要加入下列内容：

```
    <servlet>
      <servlet-name>
          Hello
      </servlet-name>
      <servlet-class>
          Hello
      </servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Hello</servlet-name>
        <url-pattern>/Hello</url-pattern>
    </servlet-mapping>
```

修改 web.xml 之后请记得重新启动 Tomcat 服务，再以浏览器检视网址：http://127.0.0.1/myapp/hello.htm。

如果这个范例也顺利完成的话，你在布署 servlet 方面应该都没问题了，可以试着练习写复杂一点的范例程序。

6.0 其它事项

6.1 让 Tomcat 重新加载 servlet

在撰写 servlet 时，你可能会反复修改程序，部署档案，并且在浏览器中观看修改后的结果。但是在预设的情况下，基于效能的考虑，Tomcat 不会自动加载新的 servlet，这对我们在开发除错时期很不方便。要让新的 servlet 生效，有下列几种方法可以使用，依我个人的喜好顺序列出：

1. 修改 Tomcat 的 server.xml 档案内容，该档案存在 Tomcat 的 conf 目录下。做法是在档案中加入一个新的 context 卷标，透过该卷标来设定你的 web 应用程序的环境参数。最简单的方法就是先以字符串搜寻的方式找到 "Tomcat Root Context" 这个标签，然后在这段文字的上面加入一行 <DefaultContext reloadable="true"/>，参考下面的范例：

   ```
   <DefaultContext reloadable="true"/>


   <!-- Tomcat Root Context -->
   <!--
   <Context path="" docBase="ROOT" debug="0"/>
   -->
   ```

2. 这个设定只需要做一次就行了，所以最方便，完成后记得要重新启动 Tomcat 服务器，新的设定才会生效。
3. 重新启动 Tomcat 服务器，你可以开启〔控制台〕的〔系统管理工具〕中的〔服务〕，找到 "Apache Tomcat" 这个服务项目，在此项目上点一下鼠标右键，选 "重新启动"。
4. 修改（touch）布署描述档 web.xml，只要 web.xml 档案有被修改过，Tomcat 会自动侦测并重新加载该档案所包含的 servlets。
5. 在浏览器的网址列输入 "http://localhost:8080/manager/reload?path=要重新启动的应用程序路径"，例如本文中的范例就是："http://localhost:8080/manager/reload?path=/myapp"。在执行新加载前会求

你输入管理者的账号和密码（只需输入一次），若执行成功，则会显示 "Ok - Reloaded application at context path ***"。

由于浏览器本身也有快取机制，如果重新整理仍无法看到修改后的结果，你可以试试看将浏览器的快取功能关闭，IE6 的设定在〔工具\因特网选项\一般〕的〔设定〕钮，预设是自动检查网页是否有新版本，你可以试着将它改为 "每次查阅画面时" 检查是否有新版的档案。

7.0 思考

做完练习后，问你自己以下问题，不知道的地方请查阅相关书籍。

1. 什么是 Web container？什么是 Java servlet？
2. Web.xml 档案的用途是？
3. Java 程序里面的的 import 是做什么用的？
4. 范例程序中的 Java 类别都继承自 HttpServlet，这个类别的主要用途是什么？它有哪些属性？经常需要改写的方法有哪些？
5. 试说明浏览器和 servlet 之间的互动过程。
6. 范例程序中的 Java 类别都有个 doGet 方法，它的作用是什么？
7. 说明 HTTP 的 GET 和 POST 方法的异同。
8. doGet 方法的签名（signature）后面都有宣告 throws ServletException, IOException，其作用为何？不宣告 exception 会怎么样？
9. 范例程序使用的是 HTTP 的 GET 方法，试改写成 POST 方法。
10. PrintWriter 类别的作用是？

附录：Tomcat目录结构说明

Standard Directory Layout

为了便于按要求格式产生网络应用程序档案,合适的办法就是按照 WAR 的格式(format)把你的应用程序里的“可执行”文件(就是 Tomcat 用来执行你的程序的文件)放在同样一个组织结构里。按照这种做法，下面的有关内容应该放置在你的应用程序的“文件根”目录里。

- ***.html, *.jsp, etc.** - HTML和JSP页面，还有其他相关文件(比如象JavaScript,stylesheet文件以及图象)应该被客户浏览器看到。对于较大的应用程序，你可以选择把这些文件分配到分支目录(subdirectory)中去，但对于小的应用程序，通常把这些文件放在一个目录里比较简单。

- **/WEB-INF/web.xml** - 网络应用程序调度描述符。这是一个可扩充标记语言(XML)文件，它描述了构成你的程序的servlets和其他部件，同时也描述了初始化参数(initialization parameters)，以及你想要服务器执行的有关容器管理安全性的限制。以后的章节里对这个文件会有详细的讨论。

- **/WEB-INF/classes/** - 这个目录包含应用程序里必需的Java类文件(以及相关的资源)，包括没有被组合到JAR文件里的servlet和non-servlet类。如果把你的类文件组织成Java包(package)，你必须把这点也反映在/WEB-INF/classes/ .下的目录里。例如，一个叫做com.mycompany.mypackage.MyServlet 的Java class,应该被放在文件/WEB-INF/classes/com/mycompany/mypackage/MyServlet.class里面。

- **/WEB-INF/lib/** - 这个目录包括由Java类文件(以及相关资源)组成的JAR文件，比如象其他第三类class libraries或JDBC驱动程式(drivers)。

当你把程序安装在Tomcat里(或其他 2.2/2.3 相兼容的服务器上)，那些在WEB-INF/classes/目录里的classes,以及其所有的WEB-INF/lib/目录里JAR文件里的classes,对于你的程序里的其他类来说都是可视的。这样，如果你把所必须的library classes放在上面所提到的地址之一（要检查一下你是否具有重新分配third

party libraries的权限），你就可以把你的程序安装简单化-- 不需要调整你的系统class path(或者在你的服务器上安装全局库文件(global library files))。

这里的许多信息摘录自第九章 Servlet 应用编程界面规范，2.3 版，详细内容请到那里参考。

Shared Library Files

和许多servlet容器一样，Tomcat支持这样一种机制，那就是安装library JAR文件(或未被包装的classes),使它们可被已安装的程序使用。关于Tomcat是怎样定位和共享这些类的详细内容在Class Loader HOW-TO 文件里有描述。我们这里仅讨论在Tomcat里安装共享编码时二个通用的地址。

- $CATALINA_HOME/common/lib - 放在这里的 JAR 文件既可被网络应用程序看见，又可被 Tomcat 内部编码看见(visible). 最好是把 JDBC 驱动程式放在这里，JDBC 驱动程式是你的程序和 Tomcat 内部使用所必需的东西(如 JDBCRealm)。
- **$CATALINA_BASE/shared/lib** - JAR files placed here are visible to all web applications, but not to internal Tomcat code. This is the right place for shared libraries that are specific to your application.

除此之外，标准的 Tomcat5 安装包括了一些先已安装好的共享文件库，它们包括：

- The *Servlet 2.4*和*JSP 2.0* APIs，它们是写servlets和JavaServer Pages的基础。
- 和JAXP(1.2 版)应用编程接口相兼容的可扩充标记语言语法分析程式(*XML Parser*)，这样你的程序可执行DOM-based 或者SAX-based XML文件。

Web Application Deployment Descriptor

在下面的描述中使用变量名$CATALINA_HOME 来代指Tomcat5 所安装的目录地址,这是一个基础目录(base directory),其他的相关路径由它而派生。不过，如果你已经把 Tomcat 设置成多个体实例(multiple instances),并且设立了一个

$CATALINA_BASE 目录，那么你就应使用$CATALINA_BASE 而不是 $CATALINA_HOME 作为参照。

如上面所提到的，/WEB-INF/web.xml 文件包括了你的程序所使用的网络应用调度描述符。从文件的附注档名(filename extension)就可以知道这是个XML文件，它描述了服务器所需要知道的关于你的程序的所有情况(*context path* 除外，它是由系统管理员在调度你的程序时指派的)。

完整的关于调度描述符的语法及语义在第 13 章 the Servlet 应用编程接口规范，2.3 版有定义。希望在将来能够有开发工具来帮你制作和编辑调度描述符。一个 basic web.xml 文件给你提供了一个起点。这个文件包含了用来解说每一个元素用途的注释。

注意— The Servlet规范为网络应用程序调度描述符提供了一个文档类型描述符(DTD)，同时Tomcat5 在运行你的应用程序/WEB-INF/web.xml 文件时，强制执行这里定义的有关规责。特别地，你**必须**按DTD规定的顺序输入你的描述符元素(如 <filter> , <servlet> , and <servlet-mapping>)(参看 13.3 章节)。

Tomcat Context Descriptor

在下面的描述中使用变量名$CATALINA_HOME 来代指 Tomcat5 所安装的目录地址,这是一个基础目录(base directory),其他的相关路径由它而派生。不过，如果你已经把 Tomcat 设置成多个体实例(multiple instances),并且设立了一个 $CATALINA_BASE 目录，那么你就应使用$CATALINA_BASE 而不是 $CATALINA_HOME 作为参照。

一个/META-INF/context.xml 文件可用来定义 Tomcat 特有的配置选项,如 loggers, data sources, session manager 的配置及其他。这个 XML 文件必须包括一个上下文元素(Context element)，这个元素是 Host 元素的子元素，Tomcat 配置文档中 Host 元素的注释里有 Context 元素的详细信息。

Deployment With Tomcat 5

为了能被执行，一个网络应用程序必须要被调度到 servlet 容器上。在开发程序

时也是这样。我们将要描述应用 Tomcat 来提供一个可执行环境。可以用下面的几种方法在 Tomcat 里调度你的程序。

- 把未包装的目录层次复制在*$CATALINA_HOME/webapps/*目录下面的分目录里。基于你选择的分目录名称， Tomcat会给你的程序指派一个上下文路径(context path)。因为这是开发程序时最快最简易的方法，我们把这个技术用在我们自己构造的build.xml 文件里。确保在安装和更新你的程序后重新启动Tomcat。

- 把网络应用程序档案文件复制在$CATALINA_HOME/webapps/目录里。当Tomcat启动时，它会自动把网络应用程序档案文件扩展成为未包装形式，同时执行它。这种处理办法通常用来把第三类卖主或者内部开发人员提供的额外的程序加入到已经存在的Tomcat安装里。**注意**—如果你用这种办法，又希望在以后更新你的程序，你必须更新网络应用程序档案文件,**和**同时删除Tomcat产生的扩展目录，然后重新启动Tomcat，这样才能反映出更改。

- *利用Tomcat5"管理员"网络应用程序来部署(deploy)和反部署(undeploy)网络应用程序*。Tomcat5 里面包含一个被部署在context path /manager 上的程序，允许你在运行中的Tomcat服务器上进行部署和反部署，而不需要重新启动它。关于使用管理员网络程序(the Manager web application)的更多信息，请参看 administrator 文档资料(TODO: hyperlink)。

- *使用"管理员"Ant Tasks来制作Script*。Tomcat5 包括一系列定义Ant build工具的用户化作业(task)，允许你对"管理员"网络程序自动执行命令。这些作业(tasks)被用在Tomcat调度员里。

- *使用Tomcat调度员*。Tomcat5 包含一个全包装的工具来绑定(bundling)Ant作业，也可以用于自动预编译JSPs——这些JSP在被调度到服务器之前是网络应用程序的一部分。

把程序调度到其他servlet容器上的方式对每个容器都是特定的，不过所有和

Servlet API Specification(2.2 以及其后版本)相适宜的容器，都被要求接受网络应用程序档案文件。注意，其他容器没有被要求(象Tomcat那样)接受未包装的目录结构，或者提供共享文件库的机制，不过它们通常都有这些特性。