Chapter 9
Web Applications and Application Servers
Learning Objectives

- To understand the services provided by an application server
- To understand the structure of both web and application archives for packaging Java web applications
- To be able to create a deployable web application using Ant
- To be able to deploy to the Tomcat application server within a JBoss environment
Application Servers

- Support dynamic content using HTTP over the Web
- Many technology options
  - Common Gateway Interface (CGI) scripts
  - LAMP (Linux, Apache, MySQL, PHP/Perl/Python)
  - Microsoft .NET platform
  - Java Enterprise Edition
- Java is efficient, scalable and cross platform
Apache Tomcat

- Open source Java application server
- Supports Java web applications
- Servlet engine (Catalina)
- JSP Compiler (Jasper)
- HTTP server (Coyote)
Tomcat Components

- HTTP client (browser)
- Coyote HTTP server
- Tomcat
  - Java EE components
    - Catalina servlet engine
    - Jasper JSP compiler
The JBoss Application Server

- JBoss is a Java Enterprise Edition (JEE) application server
  - Includes an http (web) server
  - Includes the Tomcat server for Java web application components
  - Includes an Enterprise JavaBean container and other enterprise services not included in Tomcat
Running JBoss

- JBoss has scripts to start and stop the server in its ‘bin’ folder
  - Start script is ‘run’
  - Stop script is ‘shutdown’
- The embedded Tomcat server runs on port 8080 by default (just like the standalone version)
- To deploy a web application, copy the war file to the ‘deploy’ folder:
  - JBoss\server\default\deploy
Starting JBoss

- Navigate to the bin folder of the JBoss installation directory, e.g.
  - C:\jboss-4.2.2.GA\bin
- Run the ‘run’ batch file
  - You should see a command window appear, with some log messages
‘localhost’ URL and Port Number

- When we run a server on the local machine, we use the ‘loopback’ address
  - IP address 127.0.0.1, also known as “localhost”
- The http server runs by default on port 8080
- Therefore the URL of the server is:
  - http://localhost:8080
JBoss Server Home Page
Web Application Structure

- A web application is a collection of related resources made available via an application server.
- An application is called a *context*.
- Each context may be started, stopped, and deployed, independently of any others.
- The resources of a context are deployed into a folder known as the *context root*.
- Everything in or under a context root is part of the same web application.
Static Content

- Static content is stored in the context root or subfolders
- It consists of files that are not dynamically created but are fixed content
- Static content can be HTML/XHTML files, XML linked with XSLT and CSS, images, sounds, videos etc.
Serving Static Content

HTTP request

http://localhost:8080/webhomecover/

http response

Welcome to WebHomeCover.com

Your home and possessions are important. We give you the very best insurance cover at the lowest prices.

© Cengage Learning 2008–2010
Java Web Application Deployment

To deploy a web application to a Java server we have to:

- Provide at least one XML deployment descriptor (‘web.xml’)
- Organise our files into a specified folder structure
- Package the files that make up the application into a Java archive (a ‘JAR’ file)
- Copy the archive file to a deployment directory on the server
Java EE Packaging

- Java EE application components are packaged in Java Archive (JAR) file formats.
  - These are similar to ZIP files, but the main motivation is grouping files into a single unit of deployment rather than using compression.

- Although they are all JAR files, different extensions are used to identify their purpose, e.g.
  - A jar file for a web application archive is given a .war extension.
  - A jar file for an enterprise application archive is given a .ear extension.
Web Application Folders and Resources

Context root (publicly servable files):
- HTML
- XML
- DTD
- JPG
- GIF
- WAV
- MPG
- etc.

/web-INF

Can be named anything
Document root for application
Can contain sub-directories other than WEB-INF

Deployment descriptor for application
Specifies configuration details

Must be called WEB-INF
Not part of document root
Server cannot directly serve contents

web.xml
Simple Example

Context called 'webhomecover' containing an html welcome file

webhomecover
  • welcome.htm

/WEB-INF

gweb.xml
Example Welcome File

• Static XHTML

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
  <title>Welcome to WebHomeCover</title>
</head>
<body>
  <h1><em>Welcome to</em> WebHomeCover.com</h1>
  <h2>Your home and possessions are important.<br/>
  We give you the very best insurance cover at the lowest prices.</h2>
</body>
</html>
```
Adding Style Sheets

- Style sheets can be added to the context root

h2 {font-style: italic; color: blue}
.companyname {color: red}

- If the XHTML file is also in the context root, only the CSS file name is required in the link element

```xml
<head>
<title>Welcome to WebHomeCover</title>
<link href="webhomecover.css" rel="stylesheet" type="text/css" />
</head>
```
XML Deployment Descriptors

- Deployment descriptors are written in XML
- Provide information about the application to the server
  - e.g. default web page to serve
- Must be well formed and valid against an XML Schema
  - web-app_2_5.xsd for ‘web.xml’
Example web.xml

- This example sets a default ‘welcome-file’ for the web application

```
<?xml version="1.0" ?>

<welcome-file-list>
  <welcome-file>/welcome.htm</welcome-file>
</welcome-file-list>
</web-app>
```
Welcome Files

- The ‘welcome-file’ element can contain multiple ‘welcome-file’ elements
- The server will use the first matching file that it finds

```xml
<welcome-file-list>
  <welcome-file>/welcome.htm</welcome-file>
  <welcome-file>/index.htm</welcome-file>
</welcome-file-list>
```
Deploying to JBoss

- To deploy a web application to Tomcat we:
  - Use the ‘jar.exe’ utility to create a web archive file (a ‘.war’ file)
  - copy this into the server’s ‘server\default\deploy’ folder

- JBoss will dynamically (re)deploy the application, even if it is running
Creating a Web Archive

- Use the ‘jar’ utility (Part of Java Standard Edition)
- Create the JAR from the root directory of the Web application
- Give it a ‘.war’ file extension
- Use the following options when invoking the ‘jar.exe’ program:
  - `c` → create a new archive file
  - `v` → display verbose output
  - `f` → specify the name of the file

```
path/webhomecover> jar cvf ..\webhomecover.war *
```
Deploying a Web Archive

- Copy the ‘war’ file to the server’s ‘deploy’ folder
- The application will be deployed using the name of the war file as the name of the application

http://localhost:8080/webhomecover
Hot Deployment

- To update the web application you need to rebuild the web archive and replace the existing file in the ‘deploy’ folder
- As long as JBoss is running, it will hot deploy the application and pick up any changes without needing to restart the server
Enterprise Application Deployment

- Enterprise servers enable the deployment of complex enterprise applications
  - These may include multiple Web applications and other Java EE components
- To package these components together, Java EE includes the concept of the enterprise application
- Packaged using an Enterprise Archive
  - This type of jar file has the extension ‘.ear’ to differentiate it from .war and .jar files.
Enterprise Application Folder Structure

- An enterprise application needs an XML deployment descriptor called ‘application.xml’
  - ‘application.xml’ file must appear in a folder called ‘META-INF’,
- This is included in the ear file along with the web archive
Enterprise Archive Structure

Document root
- *.htm, *.xml, *.css, *.xsl etc.

/web-INF

/web.xml

/META-INF

application.xml

Standard Java EE application deployment descriptors
application.xml

- Enterprise application deployment descriptor

```xml
<?xml version="1.0"?>
<application xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/application_5.xsd"
version="5">
<display-name>Insurance Application</display-name>
<module>
<web>
  <web-uri>webhomecover.war</web-uri>
  <context-root>/webhomecover</context-root>
</web>
</module>
</application>
```

- The name of the WAR file
- The context root of the web application - you can change this e.g. ‘/’ would make this the default application for the server.
Creating the Enterprise Archive

- Create the ear file using the jar utility
- Navigate to the ‘webhomecover\ear’ folder and create the enterprise archive file in the folder above:
  - `path\webhomecover\ear> jar cvf ..\webhomecover.ear *`
- The ‘ear’ file can be copied to the JBoss deploy folder, and JBoss will deploy the Web application.
  - Make sure you do not try to deploy the same Web application to JBoss as both a ‘war’ file and an ‘ear’ file or there will be deployment problems.
Running Web Applications on the Default HTTP Port

- Tomcat runs by default on HTTP port 8080, but web applications normally run on port 80
- Modify the ‘server.xml’ file
  - In JBoss, the ‘server.xml’ file is located at:
    - `Jboss\server\default\deploy\jboss-web.depl`er
- Locate the ‘Connector’ element

```
<Connector port="8080" address="${jboss.bind.address}" maxThreads="250" maxHttpHeaderSize="8192" …etc/>
```

- All you need to do is to change the port attribute value to ‘80’

```
<Connector port="80" …etc.
```
Automating Build and Deploy with Ant

- Apache Ant (Another Neat Tool) uses a combination of Java and XML to create platform independent build and deploy scripts
- Ant requires that you set up two environment variables
  - JAVA_HOME
    - root folder of your Java SDK installation
  - ANT_HOME
    - root folder of your Ant installation
- Add ‘ANT_HOME\bin’ to your system path
- It can be run from the command line simply by typing ‘ant’
  - By default it looks for an Ant build file called ‘build.xml’ in the current folder
The Ant Build File – ‘build.xml’

• ‘build.xml’ consists of a project, including
  – *tasks* that perform operations
  – *targets* with names, containing tasks
  – *properties* names used inside build.xml

• The root ‘project’ element specifies
  – the name of the project.
  – the default target
  – the base directory of the files used in this build.

```xml
<project name="webhomecover" default="copy-war" basedir="."/>
```
Ant Properties

- The project element can include any number of ‘property’ elements that give local names to various files and directories used in the Web application

```xml
<property name="root" value="." />
<property name="webapp" value="webhomecover" />
```

- Once a property has been declared, it can be referred to inside an Ant build file using this syntax: \$\{propertyname\}

- Here, for example, we use the ‘root’ and ‘webapp’ properties to create the ‘web-root’ property:

```xml
<property name="web-root" value="${root}\${webapp}" />
```
The Ant ‘jar’ task

- To build a jar file there is a ‘jar’ task
  - Takes the names of the output file (‘jarfile’) and the ‘base directory’ (‘basedir’) as parameters

```xml
<target name="package" depends="compile">
  <jar jarfile="${war-file}" basedir="${web-root}" />
</target>
```

- There is also a ‘war’ task in Ant useful if the source files for the Web application are not already organised in the correct structure for Web application deployment.
Deploying the War File with Ant

- Ant can copy the `.war` file to the server’s deployment folder.
- Here, the ‘copy-war’ target uses the ‘copy’ task to do this.
- Uses the ‘depends’ attribute of the ‘target’ element to ensure the war file is packaged before deployment.
  - If a target depends on another target, then Ant will execute the other target first.

```xml
<target name="jboss-copy-war" depends="package">
  <copy file="${war-file}" todir="${jboss-deploy}" />
</target>
```
Chapter Summary

- Tomcat web application server
- JBoss enterprise application server
- Web application file structures
- Creating an archive file with the ‘jar’ utility
- Deploying a web archive (war) file
- Enterprise archive (ear) files
- Using Ant for build and deploy