Installation and Integration

Openwave™ Usability Interface, Java Edition 1.0 Beta

Openwave Systems Inc.
1400 Seaport Boulevard
Redwood City, CA 94063
http://www.openwave.com

Part Number OUII-10-009
November 2001

LEGAL NOTICE

Contents

About This Book

This book describes how to install, configure, and try out a development and web hosting environment for the Openwave™ Usability Interface (OUI), Java Edition 1.0 Beta, which you can use to create a single Wireless Markup Language (WML) application that operates optimally on various types of Wireless Application Protocol (WAP) devices.

Audience

This book is intended for WAP developers who want to create and host WML applications that run well on more than one type of mobile browser.

You can write OUI applications using the Java language or the OUI XHTML or WML tag libraries. Depending on the method you choose, you will need a background in that technology.

Related Documentation

The Openwave Usability Interface comes with a large body of documentation, which is also available on the Openwave Developer web site:

```
http://developer.openwave.com
```

See the OUI *Getting Started* book for more information about the OUI documentation.

Technical Support

The best resource for up-to-date information on developing wireless web services for the market is the Openwave Developer site:

```
http://developer.openwave.com
```

You can download tools and find a variety of useful resources on this site, including Frequently Asked Questions, bug reporting, technical support, and an interactive developer forum.

Style and Typographical Conventions

This manual uses different fonts to represent the information that you enter:

- `Text that appears like this` identifies command names, path names, URLs, and specific text that you must enter.
- *`Text that appears like this`* identifies placeholders or variables that you should replace with values appropriate to your environment.

1 Preparing to Use OUI

You can use the Openwave Usability Interface (OUI) to build a single wireless application that delivers the best possible user experience to a wide variety of Wireless Application Protocol (WAP) devices. The current release of OUI is a developer library implemented in Java, plus tag libraries modeled on the Wireless Markup Language (WML) and XHTML Mobile Profile (XHTML-MP).

You host OUI applications on a web server that is configured to support Java Server Pages (JSP) and servlets. One example is the Tomcat Java server from the Apache Software Foundation, which is used in the steps and examples in this book.

To start using OUI, you must first follow the instructions included in this chapter to install and configure the Java and OUI software. You can skip steps that apply to components that are already installed on your system.

NOTE The installation and configuration instructions included in this chapter are provided for installing OUI, the Apache Tomcat Java server, and the Sun JDK on Windows 2000 and NT. If you are installing OUI or other Java software on another system (UNIX, Linux, or another version of Windows), you may find valuable information here that you can adapt for your installation.

Choosing a Java Server

Because OUI uses Java Server Page (JSP) and Java servlet files, you must install a Java application server to host your applications. Any Java application server should handle OUI files, but the following three have been tested successfully:

- Apache Jakarta Tomcat 3.2.1 or later: `http://jakarta.apache.org/`
- Allaire JRun 3.1: `http://www.jrun.com/`

  NOTE JRun adds white space before the XML processing instruction. There is an easy workaround. See the *Release Notes* for more information.

- Resin 2.0.0: `http://www.caucho.com`

NOTE For instructions on how to install your Java server, refer to the documentation for server you choose. If you choose Apache Tomcat, you can refer to the instructions in this book as well as the documentation included with the server.

Installing Tomcat

Tomcat is a free Java server available from the Apache Software Foundation.

1. **Get the Tomcat `jakarta-tomcat-3.2.3.zip` file from:**

   `http://jakarta.apache.org/`

   NOTE If you use a later version of Tomcat than 3.2.3, follow the installation instructions that come with that server and adjust the following steps accordingly.

2. **Unzip the Tomcat zip file to `c:\`.**

   This creates a `c:\jakarta-tomcat-3.2.3` directory and inserts all Tomcat files and directories into it.

Installing the Java 2 SDK

1. **Download the latest Java 2 SDK (1.2.2 or later) from:**

   `http://java.sun.com/j2se/`

2. **Execute the SDK installation file to install it.**

   If you have any questions, refer to the SDK documentation.

   At the time this document was published, the latest Java 2 SDK version was 1.3.1. The default installation directory for version 1.3.1 is `c:\jdk1.3.1_`*xx*, where *xx* may change. You'll need this information to set an environment variable for Tomcat.

Setting System Variables

To make it possible for the Java compiler and utilities to run from any directory, you must add Java 2 SDK to the Windows system `Path` variable.

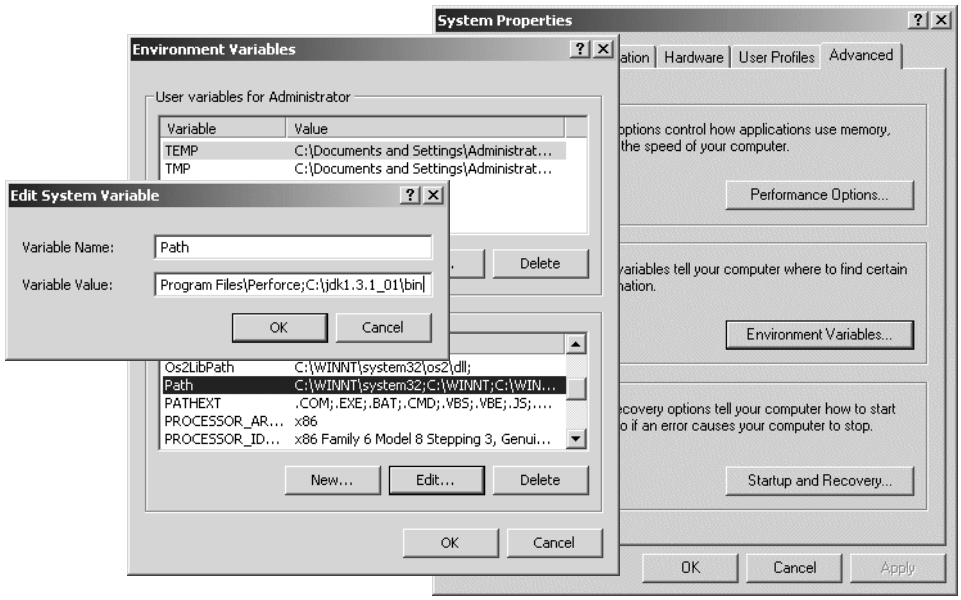
1. **Choose Start > Settings > Control Panel.**

2. **In the Control Panel window, start the System control panel.**

3. **Locate the system variables.**

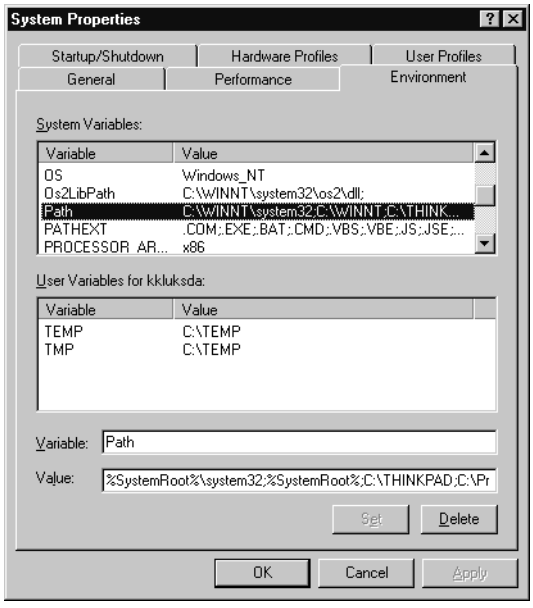
   - If you are running Windows 2000 (shown in Figure 1-1), click the Advanced tab and then click the Environment Variables button. In the list of system variables, select the `Path` system variable and then click Edit to open the Edit System Variable dialog box.

Figure 1-1. Setting Windows 2000 system environment variables



- If you are running Windows NT (shown in Figure 1-2), click the Environment tab. Then select the `Path` variable in the list of system variables.

Figure 1-2. Setting Windows NT system environment variables



4 Add the path for the Java 2 SDK bin directory to the `Path` system variable (use a semicolon to separate values):

```
;C:\jdk1.3.1_01\bin
```

5 **Add system environment variables for the Java server home directories:**

If you are using Windows 2000, you enter a new system variable by clicking the New button below the list of system variables and entering the variable name and value in the New System Variable dialog box.

If you are using Windows NT, you enter the new system variable name and value in the fields below the list of system variables and then click Set.

For Tomcat, add:

Name	Value
TOMCAT_HOME	c:\jakarta-tomcat-3.2.3

For the Java 2 SDK, add:

Name	Value
JAVA_HOME	c:\jdk1.3.1_01

6 **Add the paths to the Java server** `servlet.jar` **file and to the** `oui.jar` **file to the** `CLASSPATH` **system variable.**

This makes it possible for the Java compiler to find the servlet and OUI classes.

Although you have not yet installed the `oui.jar` file, you can set the value for `CLASSPATH` because you will install it in the same Java server directory as the `servlet.jar` file. This is usually in the server's `lib` directory.

For Tomcat, add (or modify) the `CLASSPATH` system variable name and value:

Name	Value
CLASSPATH	%TOMCAT_HOME%\lib\servlet.jar;%TOMCAT_HOME%\lib\oui.jar

You may need to set environment variables for other Java servers. See the server documentation for more information.

Starting and Stopping the Java Application Server

The first time you start Tomcat, it creates files and directories you need to install OUI files. If you don't find the directories needed to install OUI files with another Java server, you may need to start the server to create these directories. See the server documentation for more information.

You start and stop Tomcat by entering commands in a DOS command prompt window.

To open a DOS command prompt window

1 **Choose Start > Run.**

2 **In the Run dialog box, type** `cmd` **and click OK.**

To Start Tomcat:

1 **Open a DOS command prompt window.**

2 **Change to the Tomcat** `bin` **directory:**

```
cd %TOMCAT_HOME%\bin
```

3 **Enter:**

```
startup
```

The server starts and opens a second DOS command prompt window, which displays information about the Tomcat server.

The first time you start Tomcat, it takes a few seconds for it to create the files and directories it needs. You know that Tomcat is running and ready to serve Java files when you see the messages in the second DOS command prompt window indicating that ports 8080 and 8007 are being handled.

NOTE If you close the second window, Tomcat shuts down abruptly.

You can create a batch file to start Tomcat. For example, you can create a text file named `starttomcat.bat` that contains:

```
c:
cd \
cd jakarta-tomcat-3.2.3\bin
call startup
```

To test that Tomcat is configured correctly:

1 **Start Tomcat.**

2 **Open a web browser and go to the following Tomcat URL:**

```
http://localhost:8080/
```

3 **From the Tomcat server home page, use the Tomcat JSP and servlet examples to make sure that Tomcat is correctly configured to serve them.**

To Stop Tomcat:

1 **Open a DOS command prompt window.**

2 **Change to the Tomcat** `bin` **directory:**

```
cd %TOMCAT_HOME%\bin
```

3 **Enter:**

```
shutdown
```

This closes the second Tomcat DOS command prompt window and stops the server.

You can create a batch file to stop Tomcat. For example, you can create a text file named `stoptomcat.bat` that contains:

```
c:
cd \
cd jakarta-tomcat-3.2.3\bin
call shutdown
```

Installing OUI

The OUI package includes library files, Release Notes, a Readme file, documentation, and example JSP files. The OUI package is available at the Openwave Developer web site:

`http://developer.openwave.com`

To prepare for installation:

1. **Download the appropriate OUI package for your operating system.**

   The Windows package is `oui.zip`

   The UNIX package is `oui.tar.gz`

2. **Unzip the OUI archive file to a new directory (such as `c\:OUIfiles`).**

To install the OUI library files:

You must install the following three files in the correct directories to make it possible for the Java server to support OUI applications:

- `oui.jar`
- `oui.tld`
- `xhtmloui.tld`

To install these files:

1. **Copy the `oui.jar` file to the location where your application server stores JAR files.**

   Usually, this is in the `lib` directory in the server's home directory.

   For Tomcat, install `oui.jar` in `%TOMCAT_HOME%\lib`.

2. **Create a new directory called `tld` in the Java server's web applications `WEB-INF` directory and copy `oui.tld` and `xhtmloui.tld` to the new directory.**

   For Tomcat, create the new `tld` directory in:

   `%TOMCAT_HOME%\webapps\ROOT\WEB-INF`

3. **Restart the server**

   See "Starting and Stopping the Java Application Server" on page 4.

You are now ready to serve OUI JSP and servlet files.

To Install the OUI example files:

The OUI examples are included with the OUI package in the `SampleCode` directory, which contains the following directories:

- `JavaAPI_Samples`: OUI Java servlet examples
- `WML_TagLibrary`: OUI WML JSP examples
- `XHTML_TagLibrary`: OUI XHTML JSP examples

1. **Create a new directory in the server's root directory from which JSP applications are served.**

   For Tomcat, create the new directory in the `ROOT` directory, for example:

   ```
   %TOMCAT_HOME%\webapps\ROOT\oui_examples
   ```

2. **Copy the `SampleCode` directory to the new directory.**

To try out the OUI JSP examples:

- **Open a web browser of the Openwave IDE and enter the URL for the example you want to open.**

  If you are using the Tomcat server, enter:

  ```
  http://localhost:8080/oui_examples/SampleCode/
  WML_TagLibrary/index.jsp
  ```

  Or

  ```
  http://localhost:8080/oui_examples/SampleCode/
  XHTML_TagLibrary/index.jsp
  ```

To try out one of the OUI servlet examples:

Before starting one of the OUI servlet files, you need to first compile them and then copy the resulting `.class` files to the Java server class directory.

1. **Choose Start > Run.**

2. **In the Run dialog box, enter `cmd` and click OK.**

3. **In the DOS command prompt window, change to the `JavaAPI_Samples` directory.**

   For Tomcat, directory path is:

   ```
   %TOMCAT_HOME%\webapps\ROOT\oui_examples\Sample_Code\
   JavaAPI_Samples
   ```

4. **Enter:**

   `javac` *`filename`*`.java`

   Replace *`filename`* with the name of the OUI Java example you want to compile.

5. **Repeat this for all of the java files.**

6 Copy the class files to the server class directory.

For Tomcat, the directory is:

```
%TOMCAT_HOME%\webapps\ROOT\WEB-INF\classes
```

The server class directory now contains the *filename*.class files, which you can run using a web browser or a mobile browser simulator.

7 Enter the URL for one of the Java example class files in a web browser or in the Go To Address dialog box in the Openwave IDE.

The WMLAll example provides links to the other servlet examples. When accessing the class files, don't include the .class extension in the URL.

For example, for Tomcat use the following URL:

```
http://localhost:8080/servlet/WMLAll
```

Using the Openwave SDK

While you can view OUI applications using your web browser, it's far more useful to install the Openwave SDK and use the Openwave IDE's mobile browser simulator to view the examples as they appear on mobile browsers. This allows you to fine tune your applications specifically for mobile browsers.

You can use the Openwave IDE to create new JSP files with the proper headings already in place and uses syntax coloring in the editor.

You can download the Openwave SDK free of charge from the Openwave Developer web site:

```
http://developer.openwave.com
```

NOTE Make sure that the Openwave IDE mobile browser simulator is set to HTTP Direct mode when you run OUI applications.

Using OUI 2

You can experiment with the following examples to determine which OUI technology fits your programming needs. For a more complete introduction to creating OUI services, see the OUI *Getting Started* book.

Java Server Pages

The following examples show how to create and use OUI JSP files.

NOTE Make sure that the Openwave IDE mobile browser simulator is set to HTTP Direct mode when you run OUI applications.

WML

1. **In the Java server's** `webapps` **directory, create a new directory called** `TestWml`**.**

   For Tomcat, the directory is:

   `%TOMCAT_HOME%\webapps\ROOT\test_wml`

2 Create the following OUI WML tag file and save it in the new directory as `mjindex.jsp`**:**

```
<%@ taglib uri="/WEB-INF/tld/oui.tld" prefix="oui" %>
<oui:wml>
    <oui:card title="Michael Jordan" id="main">
      <oui:p mode="nowrap">
        <oui:menu>
          <oui:menu_item href="nba_stats.jsp" title="NBA"
              text="NBA Statistics" />
          <oui:menu_item href="col_stats.jsp" title="College"
              text="College Statistics" />
          <oui:menu_item href="titles.jsp" title="NBA Titles"
              text="Championships" />
          <oui:menu_item href="return.jsp" title="Comeback"
              text="Out of Retirement" />
        </oui:menu>
      </oui:p>
    </oui:card>
</oui:wml>
```

3 **Open the file in a web browser or the Openwave IDE.**

For Tomcat, the URL is:

```
http://localhost:8080/test_wml/mjindex.jsp
```

XHTML

1 **In the Java server's** `webapps` **directory, create a new directory called** `TestXhtml`**.**

For Tomcat, the directory is:

```
%TOMCAT_HOME%\webapps\ROOT\test_xhtml
```

2 Create the following OUI XHTML tag file and save it in the new directory as `nbastats.jsp`**:**

```
<%@ taglib uri="/WEB-INF/tld/xhtmloui.tld" prefix="oui" %>
<oui:html xmlns="http://www.w3.org/1999/xhtml">
    <oui:head>
      <oui:title>MJ NBA Stats</oui:title>
    </oui:head>
    <oui:body>
      <oui:p align="center">
        Michael Jordan<oui:br/>
        NBA Stats<oui:br/>
        through 1998
      </oui:p>
      <oui:p align="left">
        <oui:table>
          <oui:tr>
            <oui:th>Category</oui:th>
            <oui:th>Total</oui:th>
          </oui:tr>
          <oui:tr>
            <oui:td>Points</oui:td>
            <oui:td>29,277</oui:td>
          </oui:tr>
          <oui:tr>
            <oui:td>Rebounds</oui:td>
            <oui:td>5,836</oui:td>
          </oui:tr>
          <oui:tr>
            <oui:td>Assists</oui:td>
            <oui:td>5,012</oui:td>
          </oui:tr>
          <oui:tr>
            <oui:td>Steals</oui:td>
            <oui:td>2,306</oui:td>
          </oui:tr>
          <oui:tr>
            <oui:td>Blocks</oui:td>
            <oui:td>828</oui:td>
          </oui:tr>
        </oui:table>
      </oui:p>
    </oui:body>
</oui:html>
```

3 Open the file in a web browser or the Openwave IDE.

For Tomcat, the URL is:

```
http://localhost:8080/test_xhtml/nbastats.jsp
```

Java Servlets

The following example shows you how to create and use OUI servlets.

1 **Enter the following code in a text editor.**

```
import java.util.*;
import com.openwave.oui.framework.*;
import com.openwave.oui.waomelements.*;
import javax.servlet.http.*;

public class MJReturns extends HttpServlet {

    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws javax.servlet.ServletException,
        java.io.IOException {

        performTask(request, response);
    }
    public void doPost(HttpServletRequest request,
        HttpServletResponse response)
        throws javax.servlet.ServletException,
        java.io.IOException {

        performTask(request, response);
    }
    public void performTask(HttpServletRequest request,
        HttpServletResponse response) {

        try {
            DeviceContext dc = new DeviceContext
                (request,response);
            Deck myDeck = new Deck();
            Card myCard = new Card("mjreturn","Retirement");
            myCard.setTitle("MJ Returns");
            myCard.beginParagraph();
            myCard.addText("In 2001, Michael Jordan came out
                out of retirement and joined the Washington
                Wizards, formerly known as the Washington
                Bullets.");
            myCard.endParagraph();
            myDeck.addCard(card);
            dc.render(myDeck);
        }
        catch (Throwable theException) {
            theException.printStackTrace();
        }
    }
}
```

2 **Save the code in a file named** `MJReturns.java` **in the server class directory.**

For Tomcat, the class directory is:

`%TOMCAT_HOME\webapps\ROOT\WEB-INF\classes`

3 **Choose Start > Run.**

4 **In the Run dialog box, enter** `cmd` **and click OK.**

5 In the DOS command prompt window, change to the server class directory.

For Tomcat, the class directory is:

```
%TOMCAT_HOME%\webapps\ROOT\WEB-INF\classes
```

6 Enter:

```
javac MJReturns.java
```

The class directory now contains the `MJReturns.class` file, which you can open with a web browser or Openwave IDE. (When opening this file, don't include the `.class` extension in the URL.)

7 Open the file in a web browser or the Openwave IDE.

For Tomcat, the URL is:

```
http://localhost:8080/servlet/MJReturns
```