Table Of Contents

Introduction.................................................................................................................2
Heterogeneous Application Design considerations.......................................................3
WSCR (Wireless/Web Student Course Registration) Application : ..............................4
Technology used .........................................................................................................5
Modes Of Application : ...............................................................................................6
Generating XML (eXtensible Markup Language) content.......................................... 10
The Mapping File ......................................................................................................11
Style Sheet Transformations......................................................................................13
A . The other end.......................................................................................................19
WSCR: As J2ME-J2EE Application..........................................................................20
A . Benefits of J2EE-J2ME........................................................................................22
Deploying Web/Wireless Application........................................................................24
The Application Platform ..........................................................................................25
Known Issues ............................................................................................................26
Conclusion ................................................................................................................27
References and Resources .........................................................................................29
Introduction

Today access to internet is not restricted to desktop. Almost any handheld device, including Psion, Palm Pilots, and cellular phones, provide users with a medium for accessing the myriad of Web services available today. Information is critical in today’s Internet age, and managers, employees, customers, and suppliers need access to data 24 hours a day, 7 days a week, regardless of their location.

This paper will discuss and provide a demonstration of how an application “WSCR”, based on J2EE technologies and using a J2EE Application Server, is easily accessible from cell phones or Java technology-enabled devices like Psion or Palm Pilot as well as traditional web browsers.

The Java Platform Enterprise Edition (J2EE) architecture provides a standard framework for enterprises to develop and deploy end-to-end e-commerce applications more rapidly and efficiently.

This paper particularly presents the Web Tier and all other Client Tier implementations of the “WSCR” application based on Jsp, Servlet, MIDP (Mobile Information Device Profile), XML (eXtensible Markup Language) and XSLT (XML Stylesheet Language Transformations) technologies on the Web Tier. The Application Tier is handled by J2EE standard compliant server and serves as a common Business/database layer for all clients. The paper also demonstrates the technical aspect of development and deployment of such an application.

This sample Application is based on all necessary open standards available, particularly JAVA based technology.

The paper will also show some of the negative aspects of designing WAP based Applications with JSP and Servlets in particular.
Heterogeneous Application Design considerations

When it comes to serving heterogeneous clients where the developer is uncertain about what kind of client he is dealing with the following are some of the known issues worth noting when designing such applications.

*How many lines can I display on a screen?*

Actually, there is no specific limitation on the number of lines that can be displayed as long as the maximum deck size limit (which varies from device to device) is not exceeded. However, in order to avoid too much scrolling, five to seven lines per screen (i.e., card) is a good limit.

*What security issues should I consider?*

Some phones do not support the POST method of sending form data. Hence, a username and password must be sent by the GET method. At the WAP gateway, if the logging function is activated and if the request is logged, the administrator will be able to see the username and password. This would be particularly problematic if the gateway were hosted by an ISP or other third party. As noted, one way to get around this problem is to have the WAP gateway at your own end instead of at the ISP's. In that case, a trusted person could handle the gateway and the logging function could be turned off. You could also use a custom encryption algorithm to encrypt the username and the password at the client end using WMLScript. This is possible only if you use simple algorithms; WMLScript is not powerful enough to support algorithms of the DES category.

*How can I maintain a session?*

Some WAP clients do not support cookies. Thus, in order to maintain information about the client on the server side as he navigates through the various pages of your site, a session ID must be passed as a parameter along with every request sent to the server. The parameter name for this session ID is different for different servlet engines.
Sometimes, the default session ID length increases the length of each request considerably. As a result, the client or the WAP gateway may reject the request as a malformed URL. It is necessary to reduce the length of the session ID.

**WSCR (Wireless/Web Student Course Registration) Application:**

WSCR is a sample application that demonstrates the use of Enterprise JavaBeans (EJB) technology by modeling an online Student Course Registration service. It allows an end user to:

- Login in to the system.
- Search for Semesters.
- Search for courses offered in the selected semester.
- Register for the course.
- Confirm/Cancel the course.

**Architecture of the Application:**

[Diagram of the architecture of the application showing the interaction between the browser, WML/WAP, XML/DOM, and J2EE Application Server.]
Technology used

- J2EE Reference Implementation Server 1.3.1
- Tomcat v4.0.2 and Cloudscape v4.0.6
- J2EE Standards – EJB 2.0, Servlets 2.3, JSPs 1.3
- JDK 1.3.1
- JAXP 1.1 APIs for XML Parsing
- XALAN /Xerces
- XSLT, WML, HTML, Javascript, WMLScript

WSCR application is based on J2EE technologies. The WSCR application can be deployed on any J2EE compliant server; however, the paper describes it specifically running on a J2EE Reference Implementation server 1.3.1.

The application consists of a combination of Session and Entity beans that provide the business logic to handle transactions. WSCR accesses database that contains the student Information, Semester Listing and Course Listing. The student first logs in to the registration system using his userId and password. If authenticated, from their search results, clients are able to select and register for the courses. The results from the EJB are passed on to the Servlets, Jsp or Midlets (MIDP), depending upon which client is accessing the application tier, to process the content of the results in a format supported by the client.

The Application was designed with few goals in the mind such as:

- The user should have flexibility in terms of location and the kind of device he is using in accessing the registration system.
- Number of keypresses that a user must use should be limited when accessing through mobile devices.
- Due to the limited nature of a handheld device's user interface, the amount of data that a user
is asked to enter should be kept to a minimum.

- Should be able to handle WAP clients with no cookie support.
- There should be minimum amount of transactions between Client and the Server to overcome the bandwidth limitations.

**Modes Of Application :**

To serve heterogeneous Clients we are using three sets of implementation :

a) Jsp based Implementation.
b) Servlet based Implementation.
c) MIDP J2ME Implementation.

**Jsp based Implementation :**

- Jsp for HTML Browser :
  
  StudentLogin.jsp, SemesterList.jsp, Register.jsp, Cancel.jsp, ThankYou.jsp

  Each one of it is specially designed to generate HTML page content. It uses session bean at the server end to generate the necessary data manipulation.

- Jsp for WAP/WML Browser :
  
  WSCRWML.jsp, StudentLoginWml.jsp, StudentLoggedWml.jsp, SemesterListWml.jsp, RegisterWml.jsp, wmls.jsp, ThankYouWml.jsp, reLoginwml.jsp.

  Each one of these are responsible to generate WML content for WAP browsers. It uses session beans at the server end for necessary data manipulation.

  Only WAP browsers that support cookies at client end can successfully browse the site with these JSPs.

  The alternate approach for this is to pass a parameter called SessionID with the request. This must be passed with every request sent to the server.

  The Java method `HttpServletResponse.encodeURL(String URL)` automatically appends the
session ID.

The second approach is using an object of the session bean at the server end. This approach is used in Servlet based Implementation to handle WAP browsers in our WSCR application.

**Servlet based Implementation**

The most crucial and complex part of our application was to develop one single Web Tier to serve diverse audience of clients.

Following is the set of servlets used for this purpose:

AbstractServlet.java, Common.java, IndexHtml.java, IndexWML.java, StudentLogin.java, SemesterList.java, Register.java, Cancel.java, ThankYou.java. The following figure shows some interaction between servlets and EJBs.

![Interaction between WSCR web and application server tiers](image-url)
Servlet Web Server Tier

The content dynamically generated by the servlets must be in a format supported by the requesting client. The identification of the capabilities of a requesting client means, in our case, specifically identifying the proper format in which to respond. The selection of the proper content type is made according to a configuration table that maps a given category of clients, or a client’s advertised category of accepted MIME types to a particular MIME type. This MIME type is used as the response content type. The two steps are:

1. Identify the client and its capabilities.
2. Map the client to a MIME type.

- The identification of the capabilities of a requesting client is based on the values of both fields of the HTTP request header: User-Agent and Accept. The values of the User-Agent and Accept header fields can be retrieved by a Abstract servlet in WSCR.

- The mapping from a category of clients to the corresponding desired MIME type must be configurable. This is needed in order to add the mapping of new categories of clients that are unknown at development time, and in order to be able to add more supported MIME types when the Web application evolves, without requiring any recoding. A default property called "MIMETYPEMap.properties is provided with the Application. At deployment time the deployment tool can assign values to Resource References used inside a Web application. These Resource References are added in the Naming service and made accessible to the servlet through the java:comp/env context. A servlet (i.e AbstractServlet in WSCR) can use the Context lookup() method to get the value of such a reference from the Naming service.

The AbstractServlet then lookup appropriate mime Type and returns the contentType to respective servlets which actually extends AbstractServlet.

This is how the client type and mime support is determined.
Servlets workflow and session management

The Web application is structured around the four steps of our sample process:

1. **Login** to the system.

2. **Search** for semesters that match student division.

3. **Search** for courses that match selected semester.

4. **Register** for courses.

5. **Cancel/Confirm** courses.

A new HttpSession is typically created and bound to the originating client at step 1.

StudentLogin also associates useful information (a reference to a Controller session bean) to the client session. This information is used for the subsequent requests by SemesterList servlet, Register and cancel Servlet. When the session expires due to a timeout, it is invalidated and the information associated with it is lost. Any subsequent request automatically creates a new session.

The above scenario works fine for browsers supporting cookies. As mentioned earlier, some WAP browsers do not support cookies. To handle such clients we have used objects of server side session beans which can in turn hold object references for us. The object can then be invalidated whenever a new student login is encountered for that session.

Note: However this approach is costly in terms of object creation. The second most widely used approach is to pass sessionID as parameter in the request, as discussed in Jsp based Implementation.
Generating XML (eXtensible Markup Language) content

In our application, XML is used only to generate the response content. For request content, we rely on the HTTP Query String. Note that the request content could have been posted in XML as well.

However, the classical Query String parsing proved sufficient and ensured a single common way of processing the request parameters, no matter what client was used (i.e., generic HTML browser or WML/WAP browsers).

The XML content for each response is generated:

1. By constructing the corresponding DOM (Document Object Model) tree.
2. By serializing it to generate the XML stream.

The XML document is created from database using a reference mapping file. The map file determines what data is retrieved and how it is ultimately represented in the XML file.

The map file determines which data is extracted, the name and structure of the new file, and what data is stored where.

The mapping file contains several pieces of information:

* The original data, in the form of a data element. For maximum flexibility, this is in the form of an SQL statement. In this way, you can use the mapping file to specify that data should be drawn from more than one table.

* The overall structure of the new document. This is in the form of the root element, which, through attributes, specifies the name of the destination root element and the name of the elements that are to represent database rows.

* The names and contents of data elements. These are contained in a series of element elements. The elements include the name of the new element and any attribute or content elements.
The Mapping File

The final mapping file is as follows

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

-<mapping>

<data sql="select u.student_name, u.password, u.student_id, u.course_level from studentprefs u where student_id = ? and course_level = ? and password = ?" />

-<root name="studentInfo" rowName="student">

  <element name="STUDENT_ID">
    <content>STUDENT_ID</content>
  </element>

  <element name="STUDENT_NAME">
    <content>STUDENT_NAME</content>
  </element>

  <element name="COURSE_LEVEL">
    <content>COURSE_LEVEL</content>
  </element>

- </root>

</mapping>
```

WSCR semesterref.xml Mapping file

Creating an XML document using SQL results:

The algorithm

The process for creating the new XML document is as follows:

1. Parse the map file as above to make the necessary information, including the data to be retrieved, available.

2. Retrieve the source query. This allows for the dynamic retrieval of data based on the map file.
3. Store the data in a Document object. This temporary document will then be available to pull the data from to create the destination document according to the mapping.

4. Retrieve the data mapping to make it available to the application.

5. Loop through the original data. Each row of the data is analyzed and re-mapped to the new structure.

6. Retrieve element mappings. The mapping file determines what data is pulled from the temporary document, and in what order.

7. Add elements to the new document. Once the data is retrieved, add it to the new document under new names.

8. Add attributes to the new document. Finally, add any attributes to the appropriate elements.

At the end of this process, newDoc holds the old information in the new format. From here, it can be used in another application or transformed further using XSLT or other means.

The final document looks like as follows:

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

-<studentInfo>
  -<student>
    <STUDENT_ID>D10-1234-1234</STUDENT_ID>
    <STUDENT_NAME>Allen.K.Williams</STUDENT_NAME>
    <COURSE_LEVEL>UnderGraduate</COURSE_LEVEL>
  </student>
-<studentInfo>
```

WSCR Student.xml
Style Sheet Transformations

XSL transformations allow defining templates (rules) that will be applied on elements of an XML Source document to transform it into another document. The resulting document could be another well-formed XML document (XML, WML...), an HTML document, or any other format provided that the proper output method is available.

An XSLT processor reads both a source XML document and an XSL style sheet. The XSL style sheet is itself a well-formed XML document. In our application, the XSL transformations occur whenever such a style sheet, both for the targeted MIME type and the corresponding XML file is available. The appropriate style sheet is loaded and then applied to the XML file.

Style Sheet Engine Adapter

In WSCR, AbstractServlet act as Style Sheet Adapter for all the servlets. All necessary method to do the transformation are defined in this class.

There are four main tasks which actually deal with the transformation of XML are:

- Loading a style sheet.
- Clearing the style sheet parameters.
- Setting the style sheet parameters.
- Applying a style sheet to a document and writing the result on the output stream.

Each servlet such as StudentLogin etc.specifies respective style sheet for transformation. This style sheet is in turn passed onto the setXML_XSL(...) method in AbstractServlet, which is responsible for the final transformation of XML file.

The output is then directed to the client in either HTML or WML format depending upon content type and XSL being processed.

WSCR has two sets of XSL defined for HTML and WML respectively. We will see WML for instance.
StyleSheet for WML (WSCR WmlSemesterInfo.xsl)

WML (Wireless Markup Language) is based on XML and is intended to specify the way information is presented on wireless client devices. Similar to HTML, WML specifies user interface components (formatted text, links, text fields, list of options) to support interaction with the user. Those components are laid out within cards. A card corresponds to one screen of information to be displayed. To cope with the limited capabilities of the wireless client devices, the content to be displayed may be divided among different cards; and several related cards may be gathered within a deck to be sent to the client device as a single HTTP response. A deck contains at least one card. The user can navigate between WML cards much the same way as between HTML pages. Upon loading a deck of cards, the client device display the first defined card. Wireless client devices may have different capabilities and the same cards may be displayed differently depending on the device. Therefore, the WML document to be sent may first have to be specifically tailored for some devices to render the content appropriately.
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

<xsl:output method="xml" doctype-public="-//WAPFORUM//DTD WML 1.1//EN"
doctype-system="http://www.wapforum.org/DTD/wml_1.1.xml" encoding="ISO-8859-1"/>

<xsl:template match="/">

<xsl:variable name="student_Id" select="studentInfo/student/STUDENT_ID"/>
<xsl:variable name="course_Level" select="studentInfo/student/COURSE_LEVEL"/>

<wml>

<card id="S3" title="WSCR:Login Page">

<p align="center"><br/>

*<b>Login Successful</b>*
***
<br/>

Student Name :<b><xsl:value-of select="studentInfo/student/STUDENT_NAME"/></b><br/>
Student Id :<b><xsl:value-of select="studentInfo/student/STUDENT_ID"/></b><br/>
Division :<b><xsl:value-of select="studentInfo/student/COURSE_LEVEL"/></b><br/>

***<br/>proceed to register<br/>

***************

</p>

</card>

</wml>

</xsl:template>

</xsl:stylesheet>

WSCR WmlStudentInfo.xsl style sheet
Example of an XML source document to be processed by WmlStudentInfo.xsl style sheet

When the template (<xsl:template match="/"/> ) is applied to the root node, it instructs the style sheet processor in AbstractServlet class (through the xsl:apply-templates statement ) to process the templates in the style sheet that match its child nodes. For the above example of XML source document, only the template matching the student element will be applied. The <xsl:value-of > statements in the style sheet will be replaced in the output document with the values of the Student_ID and Student_Name d10-1234-1234 and Allen.K.Williams respectively.
Splash screen for WSCR

List of semesters

List of courses

The deck of cards generated by AbstartServlet using the WML xsl style sheet and displayed in the KBrowser on a Palm device.
The deck of cards generated by Servlets and displayed in the KBrowser on a Palm device.

Note:

We are not generating WML content through XSL for all the decks. There will be overhead at the server end if we want to generate each and every page using XSL. Transformation of XML to any other form utilizes lot of server resources and memory since we are using DOM tree to generate XML.
A. The other end

a) J2EE primarily targets clients with browsers. It requires special techniques of screen
scraping or transcoder to handle other non-browser clients.
b) Requires lot of custom coding to handle diverse client environment.
c) Cannot handle session management, data recovery due to network glitch efficiently.
d) Does not give flexibility to swap between devices while in session serving different
markup language.
e) The server scalability, performance and latency can be at high stake if the application
is not handled properly due to sloppy coding and bad architectural design.

The J2ME MID Profile addresses many of the limitations of microbrowser-based
solutions that are faced in J2EE. MIDP addresses programming issues such as user
interface, networking, persistent storage, and application model.

**J2ME MIDP provides the following advantages:**

*In a microbrowser-based solution, the server is responsible for generating display
markup. This requires a round-trip every time the interface changes. In contrast, a MIDP
client's interface is contained within the device, so it can operate even when
disconnected. On the occasions the device does interact with a server, it incurs less
network traffic, because it downloads only application data, as opposed to application
data plus interface markup.*

*Markup languages such as WML and CHTML are, by design, restricted in the types of
interactions they can offer. With the MIDP GUI APIs, it is easy to implement customized
widgets and event-handling, opening up unlimited possibilities for mobile client
interfaces that are easier and more interesting to use.*
WSCR: As J2ME-J2EE Application

The basic Implementation is as follows:

*On the server side, you can develop and deploy the J2EE applications using any application server which fits one’s business strategic purpose and needs.*

*On the client side, you can develop your MIDP or MIDLET application using the J2ME wireless Toolkit and provision it onto any MIDP –compliant device(e.g mobile phone, palmtop etc). This gives the flexibility to serve large diversified wireless audience.*

WSCR is also developed to handle clients who support J2ME KVM. Usually most of the handhelds support the Java KVM.

The rest of this section describes the WSCR application in more detail, starting with the client-side classes and then looking at the server-side classes.

**Client-Side Classes**

The client part of the WSCR presents application views in a graphical user interface. It uses and subclasses widgets from the javax.microedition.lcdui package, which is part of the MID Profile. The views pull their data from the server by sending HTTP POST requests to the J2EE application server. The requests are sent using the MIDP networking APIs, which provides session management over HTTP using cookies. The J2EE part of the application interprets these requests and sends back an HTTP response containing the requested data.

The client-side controller, StudentRegistration1, listens for user interactions with the views in the GUI. When WSCR is notified of a user interaction:

- It decides what objects should handle the user input. (If the objects are remote, the Controller will send a message via an HTTP POST request.)
- It selects a view for the user (whether it is a view the user selected or the next view in the workflow).

The StudentRegistration1 class is by far the largest out of all the client classes as it contains all of
this high-level control logic. However, the large controller size is a reasonable price to pay for the benefit of having to look at only one place to understand and maintain client-side workflow and navigation.

The following table lists the main classes on the client side of the application.

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StudentAccountForm</td>
<td>A form for entering account information.</td>
</tr>
<tr>
<td>SemesterList,CourseList</td>
<td>Lists for selecting semesters and courses</td>
</tr>
<tr>
<td>ConfirmForm</td>
<td>A dialog asking students to confirm their course.</td>
</tr>
<tr>
<td>GaugeThread,ImageCanvas</td>
<td>A pop-up gauge displayed while a long-running thread executes</td>
</tr>
<tr>
<td>Session</td>
<td>Custom HTTP networking class that provides session management using cookies.</td>
</tr>
<tr>
<td>ApplicationException</td>
<td>An exception class that maps error codes to exception messages.</td>
</tr>
<tr>
<td>StudentRegistration1</td>
<td>The controller for the client side of the application.</td>
</tr>
</tbody>
</table>

Server-Side Classes

On the server side, a servlet (MIDPController) and its helper class (MIDPService) receive messages from MIDP clients and translate these into events on the enterprise data model. As noted in the previous section, the messages are packaged inside HTTP POST requests. If any information needs to be returned to the client, the servlet sends back an HTTP response containing the data.

When serving a client request, the servlet delegates tasks to Enterprise JavaBeans (EJB) components, or enterprise beans. These components provide business logic over the application's information (about students, semesters, and courses registered), which is kept in a database. The enterprise beans access this information using Bean-Managed Persistence (BMP) via the Java Database Connectivity (JDBC) API.
The following table lists the main server-side classes used in the demo

<table>
<thead>
<tr>
<th>Classes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIDPController and MIDPService</td>
<td>Servlet and helper translating messages from the MIDP client into application events.</td>
</tr>
<tr>
<td>StudentEJB</td>
<td>Enterprise bean representing a student and his or her account information (login, password, Student_Id, etc.). Provides methods for creating, changing, and accessing this information.</td>
</tr>
<tr>
<td>SemesterInfoEJB</td>
<td>Enterprise bean for retrieving information about semesters and courses.</td>
</tr>
<tr>
<td>CourseReservationEJB</td>
<td>Enterprise bean for reserving seats, cancelling registration, and confirming courses.</td>
</tr>
</tbody>
</table>

A . Benefits of J2EE-J2ME

a) Presentation layer is separated from the underlying logic, controller logic can be distributed amongst client and server.
b) The server side is not overloaded, this minimizes the latency resulting in good performance, improves scalability and throughput at the server end.
c) Since the application is built for a specific client whose profile is well known, there is no overhead defining multiple presentation layers.
d) In case of loss of connection data can be cached onto the client side which is retrieved and synchronized once the connection is re-established.

B . The other end

a) The user will have to download the application for the first time from the J2EE server exclusively if he/she does not want to pre-install JAR.
b) The user will have to install an XML parser for the data to be in the XML format. Which increases MIDLet’s size by 15 to 30 Kilobytes. These parsers also do not have utility to validate or report error since they offer only SAX for processing XML document.
c) Care has to be taken that the software must provision not just data but also code since it depends upon the memory of the device.
d) If the user wants to switch between wireless devices he will also have to switch to the appropriate device specific MIDLet along with some changes at the server end.
e) Also with every new device coming into market with a new JDK version makes hard to run old J2ME applications run on them. Whereas J2EE is a direct superset of J2SE, able to run every J2SE program, J2ME has diverging nooks and crannies. Every J2ME program will not run on J2SE, and within J2ME, programs written for certain classes will not run on others.

f) Configuration and Java virtual machines are closely related and are complex pieces of software. Even a small differences in a configuration’s specification can require large modifications to the internal design of a java virtual machine.
Deploying Web/Wireless Application

When deploying a wireless/web enterprise application, you follow the usual steps for deploying J2EE and MIDP applications.

On the J2EE side, JSP pages and Java servlets are packaged inside a Web archive (WAR) file, while EJB components are packaged inside a Java archive (JAR) file. The WARs and JARs go inside a single enterprise archive (EAR) file, together with descriptor files that describe the deployment requirements of the components. J2EE application servers typically include graphical tools for assembling and deploying EAR files.

Deploying a MIDP application is slightly simpler. A MIDP application is packaged inside a JAR file, which contains the application's class and resource files. You can deploy this JAR file using a couple of methods:

Pre-installing the JAR on the mobile device.

Downloading the JAR from the J2EE server once onto the mobile device.

When deploying the application, keep in mind that wireless devices generally have limited memory and work on networks with limited bandwidth. Smaller applications are desirable.

Accompanying the JAR file is a Java Application Descriptor (JAD) file, which describes the application and any configurable application properties. You might find it useful to make the URL of your wireless service a configurable property, for example.
The Application Platform

URLs for “Wireless/Web Student Course Registration”

a) Jsp for WML content:

http://csnet.pace.edu:8000/StudentRegistration1/jsp/WML/WSCRWML.jsp

b) Jsp for HTML content:

http://csnet.pace.edu:8000/StudentRegistration1/jsp/Index.html

c) Servlet for HTML and WML content:

http://csnet.pace.edu:8000/StudentRegistration1/entry

d) J2ME KVM

select StudentRegistration1.jad on any Palm device.

WSCR Application was tested and deployed on the following configuration

<table>
<thead>
<tr>
<th>Tiers</th>
<th>Hardware</th>
<th>Software</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Tier</td>
<td>Sun Sparc (Solaris 8)</td>
<td>Java Reference</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Windows 2000 (Professional</td>
<td>Implementation 1.3.1</td>
<td>JSP, Servlets</td>
</tr>
<tr>
<td></td>
<td>Version)</td>
<td></td>
<td>and EJB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Xalan-j 1.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Xerces 1.0.3</td>
<td>XML, XSLT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JAXP</td>
<td>XML, XSLT</td>
</tr>
<tr>
<td>Client Tier</td>
<td>Sun Sparc (Solaris 8)</td>
<td>Mozilla 6.0 beta</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Windows 2000 (Professional</td>
<td></td>
<td>HTML</td>
</tr>
<tr>
<td></td>
<td>Version)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Yospace SmartPhone</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Emulator</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>WinWAP3.1Pro</td>
<td>WML</td>
</tr>
<tr>
<td>Palm IIIxe</td>
<td></td>
<td>Klondike WAP Browser</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>J2ME (KVM)</td>
<td>WML</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Kbrowser1.0 (Java and</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>native)</td>
<td></td>
</tr>
</tbody>
</table>
Known Issues

- In order to write XML document on the server side, one will have to change the J2EE Server side policy (server.policy) file and give general File permission for read, write any file. In general the permission is given as following: However it is recommended to give permission to one specific folder or file in question.

```
permission java.io.FilePermission "<<ALL FILES>>", "read,write";
```

MalformedURLException exception is usually thrown if the file is not found or permission to write is denied.

- Jsp for WML content URL can only be viewed by devices supporting cookies.

- The application was designed such that all client details is contained in a single WML document generated by single JSP/Servlet. In this architecture, the semester list is contained in one card of a WML deck and the details for courses would be contained in separate cards in the same deck. But this single WML document contains too much data for a low-bandwidth WAP device to download at once. If the number of clients is high, the amount of data generated could easily exceed the maximum deck size permitted for a WML deck.

(Classic example of this is seen when we access WSCR on Palm pilots).

*The above issues were kept for the purpose of demonstrating some of the incapabilities displayed by handheld devices that could be encountered while designing WAP applications.*
Conclusion

The need for more sophisticated wireless devices will never end and so will the pursuit to develop wireless applications. This will give birth to even more sophisticated and intelligent Application Servers. Currently Wireless technology faces lot of challenges, not only in its overall infrastructure but also in the application development meant for it, which are currently being served by Application servers.

Web Application servers based on J2EE standards have become important part of the multitier application architecture for wireless Applications. Simplicity, portability, scalability and legacy integration all these features makes them an ideal candidate for all the applications. Many of these have been extended to support the new mobile, wireless clients.

For a traditional application server to act as a wireless application server, additional capabilities are required:

- Providing and rendering services to a much diverse wireless audience from a centralized system.
- Understanding the capabilities (memory, size, etc) and limitations (e.g. screen size, keyboard restrictions, etc) that is accessing the application content
- Understanding the available network bandwidth and accordingly process the contents to be served (e.g. avoiding huge data interaction, graphics etc)
- Reliable and secure transactions.
- Intelligent handeling of the data to and fro, from clients which are intermittently connected

This paper discussed the essential benefits and also the down-side of using various J2EE centric technologies by which one can develop Wireless Applications. Though they have many pit falls, when it comes to developing and deploying wireless Applications, much of the responsibility relies on the Vendor selection for Application servers, developers, architects and the business
requirements to make the Application successful.
References and Resources

Java Developers Connection: http://developer.java.sun.com

XSLT Programmer’s Reference by Micheal Kay

Professional Java Server Programming by Danny Ayers, Hans Bergsten, Micheal Bogovich, Jason Diamond, Mathew Ferris, Marc Fleury, Ari Halberstadt, Paul Houle, Piroz Mohseni, Andrew Patzer, Ron Phillips, Sing Li, Krishna Vedati, Mark Wilcox and Stefan Zeiger

4thpass Kbrowser: http://www.4thpass.com/kbrowser/download/


Yospace SmartPhone Emulator: http://www.yospace.com/

Addressing the Mobile Application Development and Deployment Challenge with J2EE.


M-1 Server Developers reference

Aligo Inc., M-1 Server reference document

Designing Wireless Enterprise Applications Using Java™ Technology


The Java Blueprints for wireless Program Charting the Wireless Way


Programming for Wireless Devices with the Java 2 Platform, Micro Edition

Roger Riggs, Antero Taivalsaari, Mark Vandenbrink / Paperback / Addison Wesley Longman,

Building Scalable and high – performance Java Web Application: Using J2ee

Greg Barish / Paperback / Addison Wesley Longman, Inc. / December 2001

J2EE in Practice: Building Business Applications with the Java 2 Platform Enterprise

Rick G. Cattell, Jim Inscore / Paperback / Addison Wesley Longman, Inc. / June 2001