Content Adaptation Tag Library – An Approach for User Interface Adaptation for Different Devices

Anna Maria Jankowska, Andrzej Dabkowski

European University Viadrina
Chair of Business Informatics
D-15230 Frankfurt (Oder), Germany

{jankowska,adabkowski}@euv-frankfurt-o.de

Abstract. The proliferation of devices with the ability to deliver information anywhere at any time has increased the users' flexibility and the quality of services. It has also caused the need for development and deployment of new infrastructures supporting multiple platforms. As a result, new techniques for delivering content according to device features and even specific languages such as User Interface Markup Language (UIML) emerged. This paper discusses an innovative approach for device-dependent content delivery based on JSP tag libraries. A special tag library that generates appropriate markup elements depending on the markup language supported by a specific device was developed and is described in detail. A comprehensive example illustrates the proposed technique.

1 Introduction and Problem Description

In 1991, in his paper Weiser announced the era of ubiquitous computing and described a vision of proliferation of computational resources that provide access to information when and where desired [1]. This proliferation has indeed occurred, with commonly used devices such as mobile phones, personal digital assistants (PDAs), Palmtops or laptops. According to MobileInfo the number of handsets sold in 2002 amounted to 440 million units worldwide.[1] Wireless appliances increase user's convenience, accessibility and flexibility but bring new challenges regarding applications development.

The major technical requirement concerning the access to information systems from various devices is the presentation of information in multiple formats and content tailoring to the capabilities of particular device types. Mobile and wired devices are equipped with browsers that support various media formats. It is therefore necessary to deliver the data in different markup languages such as WML [2], XHTML [3] or HTML [4]. Information presentation on mobile devices needs to address the shortcomings of wireless appliances concerning small display sizes, different features for

[1] Compare http://www.mobileinfo.com/Market/market_outlook_2003.htm.

data input, limited graphical capabilities, etc. In order to display the same number of information different number of pages may be needed depending on the device type. For example, one HTML page in an Internet browser may be the equivalent of 3 PocketPC pages and 10 WML cards for a mobile phone. The navigation methods also differ across devices. Web pages use lists of hyperlinks to navigate from one page to another; WML pages often use softkeys. Since WAP browsers have to deal with smaller screens than Internet browsers, it is not possible to display the same number of details on different devices. Breaking up the content into displayable portions is therefore essential.

An intuitive solution to the problem of device-dependent content delivery would consist in the development of many different views on the same data and applying them according to the supported formats and presentation features of devices. This approach has, however, many shortcomings. It results in rewriting applications for various browsers, markup languages and device types, maintaining large code bases and gathering design expertise at least for the most popular appliances available on the market. In order to avoid creating of separate user interfaces for each type of device, alternative techniques have to be considered.

This paper introduces a new method for building applications that, depending on the user's device and its settings, present the same data in HTML, XHTML or WML format. The approach is based on a developed in our project specific JSP tag library that detects device features and generates appropriate markup elements for mobile devices and desktop computers. It offers a fast, easy-to-use, and well-designed method to create dynamic user interfaces for various devices and separates business logic from presentation logic.

The paper is structured as follows: Section two provides an overview of existing methods for device-dependent content delivery. The subsequent section introduces the JSP and JSP tag library technology used in our approach and provides a detailed description of the Content Adaptation Tag Library (CATL). Section four illustrates the proposed technique by means of a comprehensive example. In last section issues for further research are discussed and some concluding remarks are provided.

2 State of the Art

In last years several methods for tailoring data to various devices were proposed. One approach consists of retrieving data from an information system in XML [5] format and converting them to the appropriate markup language with eXtensible Stylesheet Language Transformations (XSLT) [6]. A transformation expressed in XSLT describes rules for converting the input (source) document's tree into a structure called a result tree consisting of result objects. The conversion is achieved by associating patterns with templates. Each template matches some set of elements in the source tree and then describes the contribution that the matched element makes to the result tree. In constructing the result tree, elements from the source tree can be filtered as well as reordered, and new elements can be added. To perform the transformation, a developer needs a stylesheet and an XSLT processor invoked from JSP or a custom tag.

The Jakarta Taglibs Project offers two libraries for transforming XML into various formats: JSTL and XTags [7]. The entire approach is described in detail in [8]. Using XML and XSLT for the generation of appropriate markup elements separates content from presentation and allows for the same data to be presented in different ways. It enables reusing of fragments of data as well as generating multiple output formats and styles tailored to the device types. The most important drawback of this method is the need to maintain numerous stylesheets and the necessity to perform updates to each stylesheet separately if the view changes.

Another approach for delivering information to different devices in a device-dependent way is the User Interface Markup Language (UIML) [9]. UIML is an XML language that permits a declarative description of a user interface and separates the interface from the application logic and device type. Its usage is simple, even for non-programmers and it reduces the time spent on rapid prototyping of user interfaces for multiple devices. One of the objectives of UIML is to permit a UIML document to be mapped to any type of user interface (e.g. Java AWT, WML, HTML, VoiceXML). UIML proposes five main elements of an interface: the interface structure, presentation style, content (words, images, etc.), actions taken in response to user interaction, and interconnection of the interface to application logic. Special meta-renderers for each user interface type are responsible for generating appropriate application code from UIML (for example UIML2WML converter produces WML markup elements, UIML2HTML creates HTML, etc.). UIML is not an open-source project and the technology for generating specific output for particular devices from UIML was not revealed. It is therefore difficult to evaluate the effectiveness or efficiency of this method as well as the possible extensibility of the language.

Third approach introduces a JSP tag libraries technology [10] instead of entirely new language with numerous elements. In this method JSP tags are interpreted on the server side. When the JSP engine encounters a custom tag, it executes Java code that has been specified to go with that tag and a specific output is produced. Some tag libraries were already developed for particular markup languages such as WML or XHTML.[2] For example, the enterprise Coldbeans Software proposes custom JSP tags library for WML developers.[3] The library consists of three tags: WmlOn, WmlOff and WAP DTD. With WmlOn a developer can mark parts of JSP code executed when the JSP page is requested from a WAP browser and with WmlOff these code fragments that should be seen only in an Internet browser. WAP DTD tag writes a standard header for WML files. Although the library is available for free, it offers a very limited functionality. A comprehensive tag library for XHTML was developed by Openwave and it was named Openwave Usability Interface (OUI) [11]. It encapsulates all tags included in XHTML language specification. OUI is a server-side programming library that allows developers to employ a uniform application user interface model that abstracts implementation details from the underlying platform and is provided not only as the JSP tag library but also as PHP and COM for ASP.[4]

2 See http://jsptags.com/tags/index.jsp for an index of existing tag libraries.

3 Compare http://coldjava.hypermart.net/servlets/wmltags.htm for available tags.

4 See http://oui.sourceforge.net/download.php

Device-dependent content delivery can be based on the Model-View-Controller (MVC) model that was designed specifically to separate business logic from presentation logic through the use of a controller. [12] The most widely used MVC implementation for Java Web application developers is the Jakarta Struts framework.[5] Struts makes use of standard technologies, including Java servlets, JSP, JavaBeans and XML, to allow an application developer to define mappings between business logic (the "model") and presentation logic (the "view"). These mappings are defined within an XML configuration file that is loaded by a "controller" servlet at run-time. The Struts concept enables the developer to flexibly define multiple presentation outputs for single requests. It requires, however, a solid knowledge of the framework and some experience in the area of the used programming technologies.

The problem of proper rendering of content depending on a device's browser is also not solved by JavaServer Faces (JSF) [13] technology that makes it simpler to build user interfaces (UI) for JavaServer applications. There are no comprehensive UI components in JSF that would automatically detect the type of browser and supported markup languages in order to generate appropriate content.

All presented approaches do not provide a suitable solution for delivering device-dependent content for various devices. Usage of the XML and XSLT technologies results in code maintenance problems. The Struts framework is hard to use for non-programmers. UIML is a proprietary solution which cannot be extended by developers and the offered JSP tags were designed only for particular markup languages. JSP tag libraries are, however, a very promising technology for device–specific content tailoring. They can be also re-implemented to fit the design and implementation principles of both JSF and Struts frameworks and the functionality of the tag library can be incorporated into these both technologies.

3 Content Adaptation Tag Library (CATL) – a New Approach for Device-Dependent Content Delivery

JavaServer Pages is a technology for creating dynamic Web-based content using server-side processing [14]. It combines static markup elements with elements created by Java objects. JSP separates the application logic from the page design and encapsulates logic in portable, reusable Java components.

JSP technology supports a unique feature called JSP tag libraries. A developer can create a custom tag library with commonly used functions and distribute it to a wide range of authors. Tags from a library are resolved in the server and do not introduce client-side dependencies. Tag libraries are reusable modules that can build and access programming language objects and influence the output stream. They usually encapsulate frequent tasks and can be used across applications, increasing the speed and quality of development. The benefits of custom tags also include the creation of an abstraction layer between logic and presentation. This abstraction is a kind of an interface that allows Java developers to fix bugs, add new features, and change implemen-

[5] Jakarta Struts is available at http://jakarta.apache.org/struts.

tation without requiring any changes to the JSPs that include those tags. Tag libraries have access to all objects available to JavaServer Pages, they can communicate with each other and can be nested, allowing for complex interactions within a page.

Although some simple tag libraries for mobile applications are already provided by vendors, JSP tag libraries which enable the development of applications targeted at different devices are currently not available. A special tag library – the Content Adaptation Tag Library (CATL) - generating appropriate markup elements depending on device context was therefore developed. It separates the presentation format from the presentation logic and encapsulates most of the functionalities used in HTML, WML and XHTML pages. However, it is still possible to mix the tags with the elements of a specific markup language such as WML or HTML and to add Java source code (cf. Appendix). CATL is easy to use and enables rapid prototyping of applications designed for different devices since page developers have to include only one tag for displaying a particular element in three different formats. It also means less presentation code to manage. Applying this tag library divides the work of developing multiple views of the same data between JSP programmers responsible for the functionalities encapsulated in tag libraries (presentation logic) and developers of Web pages who are in charge of the layout and maintenance of Web sites.

The tags in the library are designed with two adaptation techniques – selective content inclusion/exclusion and page systems. In the first method inclusion means that some content is shown only for certain devices. Exclusion hides content for certain devices.

Content inclusion and exclusion addresses the problem of disparate content sets for diverse devices and varying media support. To create different content sets, all the information is first included for equivalent pages into one document. Specific data can then be hidden or shown depending on the device type. This adaptation technique can be applied with the help of WMLOn and WMLOff tags from the CATL library. WMLOnTag indicates that elements enclosed in the WMLOn tags should be included in the output for browsers supporting WML. WMLOffTag enables the elements enclosed by these tags to be excluded in the output for browsers supporting WML and presented only in browsers with XHTML or HTML support.

The second mechanism which was considered during the development of CATL is the so-called page systems. They provide a method for generating different numbers of output pages from the same document. This is particularly important for mobile devices which have varying display capabilities. For example, one information-rich PC page has to be divided and shown as many WML pages. We provide a way of displaying the same data on a different number of pages in the form of a CardTag. This tag produces a WML card or a deck of two cards. For each card a card title and card name is generated. It also offers the possibility of specifying the time amount after which the user will be redirected from the first card to another card with a given name.

Information about the devices' features is retrieved in a SiteTag which is always the first tag used in a page. The tags are able to communicate with other tags on the same page. Therefore, the SiteTag detects the browser type as well as its capabilities and all other tags use this information for content adaptation. This decreases the number of

data exchanged between the client and server and enables faster processing of requests.

The most popular way to obtain delivery context is to use the HTTP standard Accept headers [15]. These headers include the supported media types (MIME types [16]), character sets, content encoding, and languages. Additionally, the User-Agent header contains information about the device manufacturer, the version number, the device hardware and the browser used. Information delivered by HTTP headers is, however, subject to different limitations. Since there is no standard specifying the structure of the User-Agent header, delivery context is up to the device manufacturer.

In our approach content adaptation is based on information about device capabilities retrieved from HTTP headers. This information can be obtained with the help of the `getHeader` method of a `Request` object and is then available during the user's session. The `getHeader` method is a JavaServlet method. It returns the value of the specified header field as a `String` [14, p. 58]. All request header names can be obtained as an `Enumeration` object from the method `getHeaderNames()`. The SiteTag determines the form of presentation depending on the relevant features of a device.

Besides the WMLOnTag, WMLOffTag and the CardTag the Content Adaptation Tag Library contains the following tags:

- SiteTag – depending on the detected browser type generates for WML and XHTML the XML version number, the SGML public document identifier and the DTD location. It creates furthermore the root elements for HTML, WML or XHTML, the title for a page and specifies alternatively the link to an appropriate CSS stylesheet in XHTML and HTML. Since the applications are served by the server, the tag also sets a correct MIME type for each device with the `response.setContentType(String type)` method. This method determines the format of information which is exchanged between the server and the browser.
- TableTag – draws a table with such optional attributes as background color, border size as well as values for cellpading and cellspacing elements. For WML it also gives the number of columns.
- TableRowTag – draws a row in a table. The developer can specify a color for a row, its height and width.
- TableCellTag – draws cells in a row with specified height, width and background color.
- LinkTag – produces link elements in HTML, WML and XHTML.

The defined tags offer a web application developer the possibility to control how data is processed in back-end Java components without including any Java code in the JSP page. The tags allow furthermore for generation of completely different presentation layers depending on the device context: the same data can be split into many separate cards for browsers supporting WML and be offered on one page in Internet browsers. Content transformed by the Content Adaptation Tag Library can be in form of regular text or XML data. In the second case the XML source has to be transformed with the help of XSLT in order to extract specific data or modify them.

4 Example of CATL Usage

In order to demonstrate the possibilities of the implemented JSP tag library a sample application was developed. The source code for this application as well as the CATL library can be downloaded from the authors' website (http://www.bi.uni-ffo.de/de/research/project/fp_mcommerce-veroeff.html). The functionality of the program consists of retrieving information about orders and customers from a database as well as displaying some statistics about the products' turnover for items available in the enterprise database. All data are tailored to different device types and browsers (in particular to Internet browsers such as Netscape, IE or Opera and WAP browsers supporting XHTML or WML). Depending on the device capabilities the information is presented in a different way with regard to amounts of displayed data per screen, layouts, navigation elements and styles (graphical elements, colors). In Internet browsers the first page includes a greeting message and a list of links, for WAP browsers supporting WML this page was split into two cards due to the limited screen sizes of mobile phones. The user can navigate to the desired menu position by clicking on an appropriate link. If he/she chooses orders, a table with some detailed information about current orders will be displayed. The user will, however, obtain more details about orders in the Internet browser than in the WAP browser because all information included in the HTML table does not fit on the small screen of the mobile device. By choosing statistics or clients from the menu, one will reach a page on which data about customers and cumulated products' turnover are presented as a table. Some screenshots demonstrating the application in the Internet browser and in the WAP-enabled browser are shown in Figure 1 and 2.

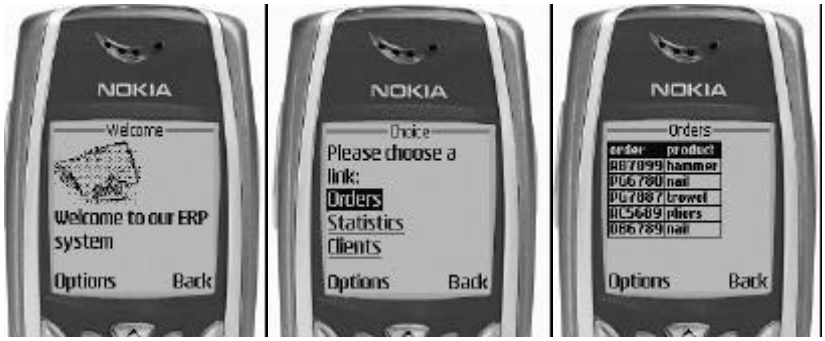

Fig. 1. An example of an application using CATL displayed in a WAP browser

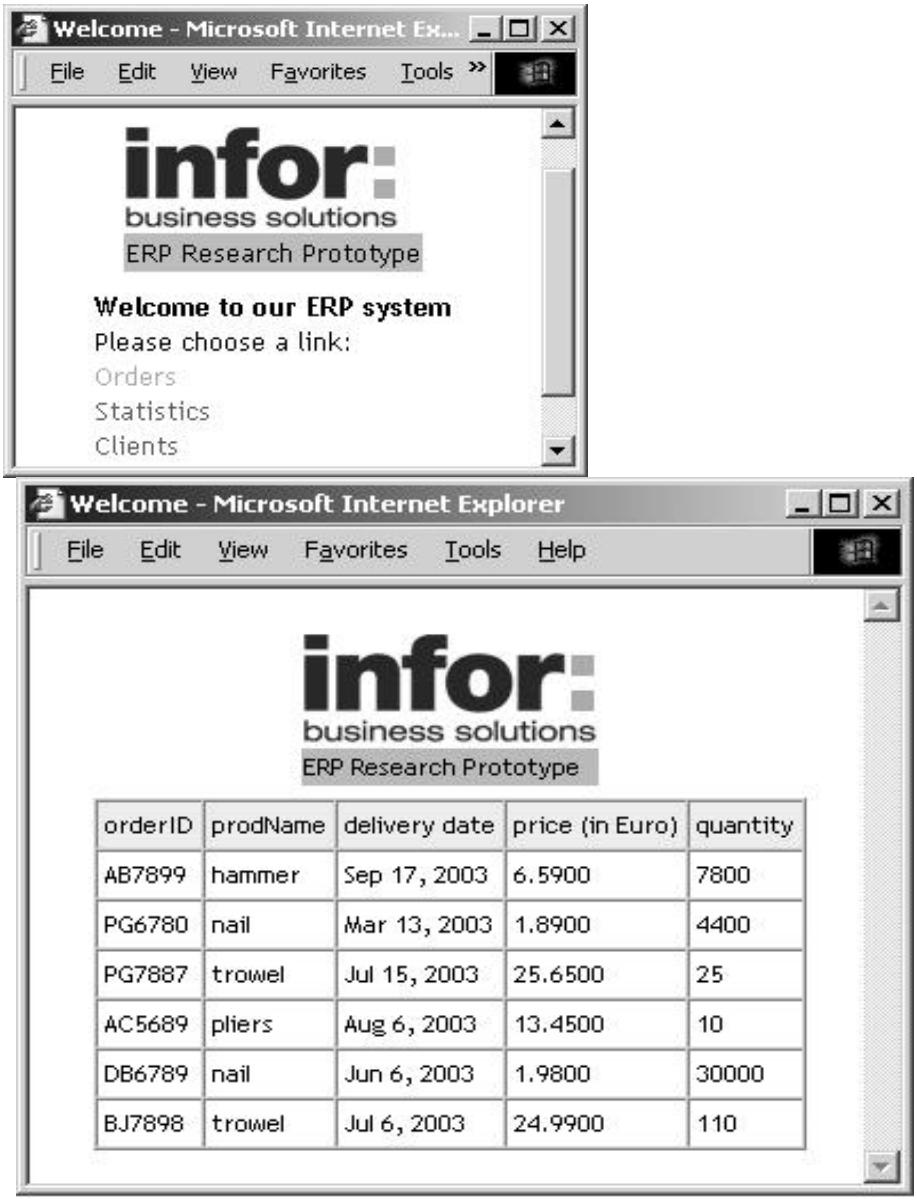

Fig. 2. The same application presented in an Internet browser

The developed application consists of JSP pages with JSP tag libraries. Besides the CATL library, the DBTags [17] and the I18N tag library (for data localization) [18] are used. As already mentioned the information about orders is dynamically retrieved from a database. The commonly used mechanism for communicating with a database is Java Database Connectivity (JDBC). It is based on the idea of database drivers which are tailored to specific database management system. The DBTags library

which is applied in the application to build a connection with the database and to retrieve appropriate data also relies on JDBC. Instead of writing Java code embedded in a JSP page, the developer can make use of specific tags for building a connection to database, sending queries or retrieving results. The following code builds a connection to a database and queries it for data about available orders:

```
<sql:connection id="conn1">
  <sql:url>jdbc:odbc:Infor</sql:url>
  <sql:driver>sun.jdbc.odbc.JdbcOdbcDriver</sql:driver>
</sql:connection>

<sql:preparedStatement id="stmt1" conn="conn1">
  <sql:query>
    SELECT orderID, <%=name%>, requestDate, price,
    quantity FROM [order] INNER JOIN product
    ON order.prodID = product.prodID
  </sql:query>
</sql:preparedStatement>
```

Appropriate markup language is generated with CATL tags. Consider the following code that produces the first page in Figure 1 and two WML pages in Figure 2:

```
<%@ page language="java"%>
<%@ taglib uri="http://www.uni-ffo.de/www-
wiwi/taglibs/markup-1.0" prefix="markup"%>

<markup:page title="Welcome" stylesheet="index.css">
   <markup:card title="Welcome" cardID="card1"
    timer="40" nextCard="#card2">
    <img src="infor.gif" alt="Infor" align="top"/>
     Welcome to our ERP system<br/>
    </markup:card>
    <markup:card title="Choice" cardID="card2">
        Please choose a link:<br/>
       <markup:link name="Orders" href="Orders.jsp"/>
       <markup:link name="Statistics"
                    href="Statistics.jsp"/>
      <markup:link name="Clients" href="Clients.jsp"/>
   </markup:card>
</markup:page>
```

In this case only three tags with appropriate attributes – the SiteTag (`<markup:page>`), the CardTag (`<markup:card>`) and the LinkTag (`<markup:link>`) are used. These tags generate different outputs for the WAP-enabled browsers and Internet browsers. The code snippet below presents the pages that are produced as WML and HTML. A comprehensive example that includes all used JSP tag libraries as well as almost all CATL tags is provided in the Appendix.

```
<!—-HTML output-->
<html>
<head><title>Welcome</title>
```

```
<link rel="stylesheet" href="index.css" type="text/css"/>
</head>

<body>
  <img src="infor.gif" alt="Infor" align="top"/>
   Welcome to our ERP system<br/>

   Please choose a link:<br/>
    <a href="Orders.jsp">Orders</a><br/>
    <a href="Statistics.jsp">Statistics</a><br/>
    <a href="Clients.jsp">Clients</a><br/>
</body>
</html>


<!—-WML output-->
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"
  "http://www.wapforum.org/DTD/wml13.dtd">

<wml>
  <template>
    <do type="options" label="Back">
      <prev/>
    </do>
  </template>
  <card id="card1" title="Welcome" newcontext="true">
    <onevent type="ontimer">
      <go href="#card2"/>
    </onevent>
    <timer value="40"/>
    <p>
      <img src="infor.gif" alt="Infor" align="top"/>
        Welcome to our ERP system
      <br/>
    </p>
  </card>

  <card id="card2" title="Choice" newcontext="true">
    <p>
       Please choose a link:
      <br/>
      <a href="Orders.jsp">Orders</a>
      <br/>
      <a href="Statistics.jsp">Statistics</a>
      <br/>
      <a href="Clients.jsp">Clients</a>
      <br/>
    </p>
  </card>
</wml>
```

5 Summary and Further Work

Developing cross-device user interfaces requires both programming and creative skills, as well as, knowledge about the particular device and application domain. Most people master a single platform or a single device and have difficulties to port their applications to other platforms or devices. A tag library for device-dependent content delivery can contribute to the faster dissemination of information on different devices. It might have a role similar to this one which HTML played in the popularization of electronic publishing. The Content Adaptation Tag Library described in this paper should enable people who are not familiar with the capabilities of various mobile devices to build user interfaces in HTML, WML and XHTML. The separation of presentation logic and presentation format allows different specialists to take control of their area of development and significantly facilitates the task of programming applications for variety of devices.

The Content Adaptation Tag Library has been successfully applied for tailoring the content to different devices in a prototypical implementation for an ERP system, in-for:COM. Some important points for improving the functionality of this library have become evident during this work.

The CATL takes into account only basic information transported in the HTTP headers for device-dependent content delivery. Integration of comprehensive context models would enable delivery of more personalized information to users' devices and better content tailoring to the properties of devices. This enhancement can be achieved by using new standards for contextual information like the CC/PP model developed by W3C [19] or UAProf invented by the WAP Forum [20].

The developed library should be enriched with additional elements. New tags for specific markup structures, e.g. tags for generating forms, input fields, etc. and tags to support other formats (e.g. cHTML, SVG) will be programmed.

During the work of programming user interfaces for multiple devices we have noticed two essential aspects of device-dependent content delivery. Web application requests often return too much content for one page. Breaking up this content into digestible portions is essential. The CATL should be therefore enhanced by tags which will enable breaking of a large body of content into multiple pages with appropriate navigation elements. The content should be dynamically paginated at the runtime depending on the amount of data that should be displayed and screen sizes. Tailoring content to different device types usually requires a transformation of elements' structures and navigation methods. For example, in an Internet browser navigation elements can be displayed as breadcrumb trail at the top of web pages while in a WAP browser they may be replaced by softkeys. It should be possible to transform tables into lists or into other tables. Table-to-table transformation should allow a developer to reorder, include, or exclude content by rows or columns. The option to transpose rows into columns and columns into rows should be implemented as well. This would be useful for tables that have to display large quantities of tabular data for large devices, but only a subset of content for small ones. All the described functionalities will be implemented in the next version of CATL.

REFERENCES

1. Weiser, M.: The Computer for the 21st Century, Scientific American, Vol.265, No.3, (1991) 94–104
2. WAP Forum: Wireless Application Protocol WAP 2.0, Technical White Paper (2002). http://www.wapforum.org/what/WAPWhite_Paper1.pdf
3. W3C: XHTML™ 1.0 The Extensible HyperText Markup Language Specification (2002). http://www.w3.org/TR/xhtml1/
4. W3C: HTML 4.01 Specification (1999). http://www.w3.org/TR/html
5. W3C: Extensible Markup Language (XML) 1.0 (2000). http://www.w3.org/TR/REC-xml.
6. W3C: XSL Transformations (XSLT) Version 1.0 (1999). http://www.w3.org/TR/xslt.
7. Apache Jakarta: XTags Library: Tags for Working with XML, XPath and XSLT, Version 1.0 (2002). http://jakarta.apache.org/taglibs/doc/xtags-doc/index.html
8. Dabkowski, A., Jankowska, A.M., Kurbel, K.: Designing Global Applications for Wireless Devices with Java and XML. Proceedings of the 23rd Internationalization and Unicode Conference, Prague (2003)
9. Abrams, M., Phanouriou, C., Batongbacal, A., Williams, S., Shuster, J.: UIML: An Appliance-Independent XML User Interface Language, Proceedings of the 8th International World Wide Web Conference, Toronto (1999). http://www8.org/w8-papers/5b-hypertext-media/uiml/uiml.html
10. Shachor, G., Chace, A., Rydin, M.: JSP Tag Libraries, Manning Publications, Greenwich (2001)
11. Openwave Systems: XHTML Tag Library Reference (2001). http://developer.openwave.com/omdt/Opensource/oui.sourceforge.net/docs/xhtml_tag_lib_ref.pdf
12. Johnson, M. et al.: Designing Enterprise Applications with the J2EE™ Platform, Addison-Wesley Publishing Company, Boston, (2002)
13. Sun Microsystems: JavaServer Faces Technology, 2003. http://java.sun.com/j2ee/javaserverfaces/
14. Sun Microsystems: JavaServer Pages™ Specification, Version 1.2. (2001). http://java.sun.com/products/jsp/
15. W3C: Delivery Context Overview for Device Independence (2002). http://www.w3c.org/2001/di/public/dco
16. Network Working Group: Multipurpose Internet Mail Extensions (1996). http://www.nacs.uci.edu/indiv/ehood/MIME/2046/rfc2046.html
17. http://jakarta.apache.org/taglibs/doc/dbtags-doc/index.html
18. http://jakarta.apache.org/taglibs/doc/i18n-doc/index.html
19. W3C: Composite Capabilities/Preference Profiles: Terminology and Abbreviations, Working Draft (2000). http://www.w3.org/TR/2000/WD-CCPP-ta-20000721
20. WAP Forum: UAProf (2001). http://www1.wapforum.org/tech/ terms.asp?doc=WAP-248-UAProf-20010530-p.pdf

Appendix: Source Code Demonstrating the Usage of CATL Library (Orders.jsp)

```
<%@ taglib uri="i18n.tld" prefix="i18n"%>
<%@ page language="java" import="java.sql.*,
                        java.text.*, java.util.Locale" %>
<%@ taglib uri="dbtags.tld" prefix="sql" %>
```

```
<%@ taglib uri="markup.tld" prefix="markup"%>

<%  //determing the preferred language
    String defaultLang="en";
    String name = "prodName_"+defaultLang;
    Locale locLg = new Locale(defaultLang,"");
%>

<i18n:bundle baseName="product"
             scope="page"
             locale="<%=locLg%>"
             changeResponseLocale="true"/>

<sql:connection id="conn1">
  <sql:url>jdbc:odbc:Infor</sql:url>
  <sql:driver>sun.jdbc.odbc.JdbcOdbcDriver</sql:driver>
</sql:connection>
<markup:page title="Orders" stylesheet="index.css">
<sql:preparedStatement id="stmt1" conn="conn1">
  <sql:query>
    SELECT orderID,<%=name%>,requestDate,price,quantity
    FROM [order]
    INNER JOIN product ON order.prodID = product.prodID
  </sql:query>
  <markup:wmlOff>
  <table>
     <tr>
        <td><img src="infor.gif" alt="Infor"
                                          align="left"/>
        </td>
     </tr>
     <tr>
        <td>Enterprise Web Research Prototype
        </td>
     </tr>
  </table>
  <br/>
  <sql:resultSet id="rset1">

  <markup:table border="1" cellpadding="4px">
     <markup:tr bgcolor="#8FBC8F">
        <markup:td>orderID</markup:td>
        <markup:td>prodName</markup:td>
        <markup:td>delivery date</markup:td>
        <markup:td>price (in Euro)</markup:td>
        <markup:td>quantity</markup:td>
     </markup:tr>
<%
   ResultSetMetaData rsmd = rset1.getMetaData();
   int numberOfColumns = rsmd.getColumnCount();
   while (rset1.next()){
```

```
%>
  <markup:tr bgcolor="#F0FFF0">
  <%
     for(int i=1 ; i<numberOfColumns+1 ; i++)
     {
        String cN = rsmd.getColumnName(i);
        String colType = rsmd.getColumnTypeName(i);
        if(cN.compareTo(name)==0) cN="prodName";
  %>
          <markup:td>
          <%
             if(colType.equals("DATETIME")){
          %>
              <i18n:locale language="<%=defaultLang%>">
              <i18n:formatDate value="<%= rset1.getDate(i)%>"
                               style="medium"/>
              </i18n:locale>
           <%}
             else
             {
               out.write(rset1.getString(i));
             }%>
          </markup:td>
     <%}%>
  </markup:tr>
  <%}%>
  </markup:table>
  </sql:resultSet>
  </markup:wmlOff>

  <markup:wmlOn>
    <markup:card title="Orders" cardID="card1">
      <sql:resultSet id="rset1">
        <markup:table columns="2" border="1"
                                    cellpadding="4px">
          <markup:tr>
            <markup:td>order</markup:td>
            <markup:td>product</markup:td>
          </markup:tr>
       <%
         ResultSetMetaData rsmd = rset1.getMetaData();
         int numberOfColumns = rsmd.getColumnCount();
         while (rset1.next()){
       %>
         <markup:tr>
          <%
            for(int i=1 ; i<3 ; i++)
            {
             String cN = rsmd.getColumnName(i);
             String colType=rsmd.getColumnTypeName(i);
             if (cN.compareTo(name)==0) cN="prodName";
```

```
          %>
              <markup:td>
                <%
                   out.write(rset1.getString(i));
                 %>
               </markup:td>
            <%}%>
          </markup:tr>
      <%}%>
      </markup:table>

  </sql:resultSet>
  </markup:card>
  </markup:wmlOn>
</sql:preparedStatement>
</markup:page>
```