Dynamic WAP Content Generation with the Use of Java Server Pages

Karl Kurbel[1], Andrzej Dabkowski[1]

[1] European University Viadrina, Business Informatics
D-15230 Frankfurt (Oder), Germany
{kurbel, adabkowski}@euv-frankfurt-o.de

Abstract. As wireless mobile applications are becoming more and more popular, an increasing number of applications requires that the content to be displayed on the user's device is generated dynamically. Just like many electronic-commerce applications need to generate web content depending on user input and data retrieved from a database, WAP (Wireless Application Protocol) based applications as well have to access databases and produce output in WML (Wireless Markup Language) or its scripting language WMLScript. In this paper, the basic technologies for creating static and dynamic WAP content are discussed. JSP (Java Server Pages) is chosen as an appropriate technology that supports the generation of dynamic user-tailored content. A mobile-shopping example is used to illustrate the concepts. The underlying database is accessed through a JDBC-ODBC driver. Parameter passing across the heterogeneous architecture of the mobile solution is discussed.

1 Background and Problem Description

The Wireless Application Protocol (WAP) supports mobile services on small devices like cellular phones and PDAs (Personal Digital Assistants). While it is quite simple to provide static content for such devices, generating dynamic content can prove to be a challenging task. In most cases dynamic content is more valuable for the user because that content can be tailored to user-specific requirements or interests. To create the desired content, interaction with the user based on WAP has to be supported so that the user can indicate what exactly he or she wants.

Many real-world mobile applications require access to data which are stored in a database or created by an information system in the user's organization. In a business environment, such a source of information may be an ERP (Enterprise Resource Planning) system, for example. Consider a sales representative who works and deals with the company's customers in a given region. He is usually out of his company and obviously he has no wired Internet connection with the company's computer systems and with the corporate ERP system. Still he needs information from that system and it would be quite helpful for him if he could access it. For example, a sales representative might need up-to-date information like the current quantity on stock of

a product if the customer he is just visiting would like to order that product. Or if the customer asks what the status of an order he placed earlier is and when that order will be dispatched, it would be useful if the sales person could track that information on his mobile device down in the back-office ERP system.

In such cases the mobile service has to generate dynamic WAP content depending on some input by the user (e.g. product number, order ID). The problem discussed in this paper is how to create that content when the data needed come from a corporate database management system.

Several server-side programming technologies support dynamic content creation, e.g. Common Gateway Interface (CGI), Active Server Pages (ASP), PHP Hypertext Preprocessor (PHP) and Java Server Pages (JSP). All these technologies allow constructing dynamically generated pages on the fly. Server-side scripts may be mixed with statements for the presentation layer of WAP to provide dynamic content. In other words, both WML (Wireless Markup Language) code and code in WMLScript, the associated scripting language, can be dynamically generated with the use of the mentioned technologies.

The paper is organized as follows: In the next section, basic technologies to display static content on mobile devices are briefly outlined. Section 3 discusses the concepts of creating dynamic content from a database and displaying it on a mobile device. A comprehensive example is presented in section 4 for illustration. A summary of our experience is given in section 5. Some code used for reference is included in the appendix.

2 Basic Technologies for Static WAP Content

2.1 WAP Protocols

WAP defines a set of standards that enable users of mobile devices to access mobile network services. "WAP is a communications protocol and application environment for the development of information resources, advanced telephony services and Internet access from mobile devices" [1]. One feature of WAP is access to and navigation through so-called decks written in WML [2]. Decks are composed of cards, the units of information displayed at one time, similar to pages in HTML. WAP also includes a scripting language, WMLScript [2], that supports dynamic interaction with a mobile user. Content which is based on WAP standards (WML, WMLScript) is often called WAP content.

WML, WMLScript and WBMP are components of the presentation layer. WAP actually includes a number of sub-protocols for a hiearchy of layers. The binary representation formats for WML and WMLScript are Wireless Binary XML (WBXML) and WMLScriptc (or WMLScript "compiled"), respectively. WML content is normally parsed and encoded into WBXML binary format before it is transmitted from the content server to a WAP client. WMLScriptc is the binary encoded form of the WMLScript language.

Other WAP sub-protocols, on lower levels, are the session layer protocols Wireless Session Protocol/B (WSP/B) and Wireless Session Protocol (WSP), and the transaction layer protocol Wireless Transaction Protocol (WTP). For security and client authentication WAP provides a security layer called Wireless Transport Layer Security (WTLS). The Wireless Datagram Protocol (WDP) is a network layer protocol on the data transport layer of the WAP protocol architecture. WDP is a datagram-oriented protocol that delivers data divided into datagrams.

All these sub-protocols cooperate with each other in a way that the information carried in datagrams reaches the mobile client and is presented on its device with the help of the presentation layer. For a more detailed investigation of WAP protocols we refer the reader to the WAP specification [2] or to an overview in [3].

2.2 Connection to Internet via Mobile Phones

Static or dynamic WAP content is usually stored and generated by ordinary web servers which are additionally configured to provide content for mobile devices. The normal access to such a server, i.e. via Internet, is that the web client sends an HTTP (Hypertext Transfer Protocol) request to the server and the server sends the requested information back as an HTTP response. Obviously this way of access is not immediately possible for a sales representative with a cellular phone because that phone is not connected to the Internet.

One way to provide the access needed is to link the mobile device to the Internet and then convert a mobile request into an HTTP request. Likewise, the HTTP response has to be translated into a mobile response for the cellular phone. In this way, communication between a mobile device and a web server can be established.

The WAP specification introduces the concept of a WAP gateway for this purpose. A WAP gateway acts like a proxy between the mobile device and the Internet. It is capable of both conveying the information to the mobile device over the existing wireless transport infrastructure, using the above WAP protocols, and communicating with a web server via HTTP or other Internet protocols. With the use of a WAP gateway each mobile device can place its requests for data. The gateway retrieves the content from a web server and sends it back to the mobile device in WAP format as response to the request (see figure 1).

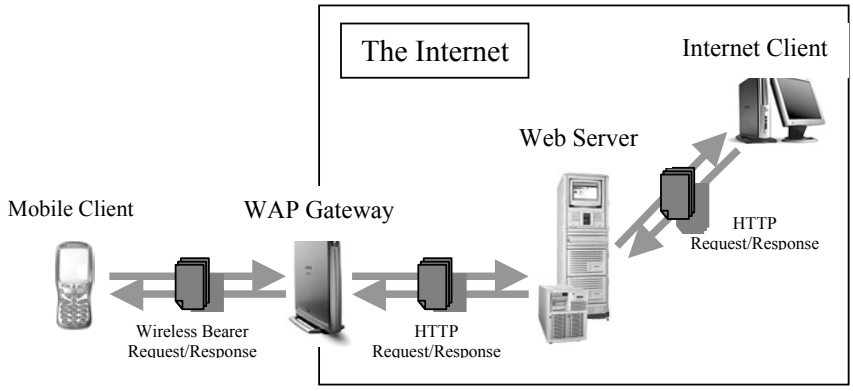


Fig. 1. Request and response from/to a mobile device

2.3 WML and WMLScript

WML is a markup language. Its syntax consists of a set of tags. The number of different tags is much smaller than in HTML. WML is used to describe the format of content to be displayed and thus its appearance on the user's display. It also specifies how documents are linked and how data are gathered in forms. In addition to textual content and layout, WML pages can include simple graphics in a WAP Bitmap format (WBMP).

WMLScript, the scripting language used in WML pages, has a syntax similar to JavaScript. It is sometimes called a light version of JavaScript. WMLScripts are not embedded in the WML pages but referenced through URL links. Before the script is sent to the WAP browser, WMLScript is compiled into byte code on the server. The main goals of WMLScript are to validate user input, and to manage and display message and dialog boxes.

Lexically, this scripting language consists of variables, basic flow control instructions, and internal and external functions. Pre-defined WMLScript libraries are responsible for alerts, displaying dialogs, implementations of mathematical functions, type manipulations, string and URL handlings, and others. Programming in WMLScript is mainly using those libraries. Both WML and WMLScript content can be generated dynamically with the use of JSP technology. Descriptions of the WML and WMLScript languages are found in [1], [4], [5].

2.4 Constraints of Mobile Devices

Developer of WAP applications face a number of difficulties which come primarily from the nature and size of mobile devices. WAP-enabled devices usually have a rather small display area. On cellular phones, a typical size is four lines of twelve characters, but the precise measurements depend on the specific device type. A single deck of cards is limited in size – for example up to 1400 bytes. Therefore information has to be spread across several decks.

The methods of user input are limited and uncomfortable. Scrolling the small display area, for example, to fill a number of data fields is awkward. Furthermore, most WAP devices have little RAM memory and CPU processing power.

A mobile developer has to take these limitations into account when designing a WAP application, regardless if the content to be produced is static or dynamic. He/she has to provide for a larger number of screens than on a WWW client and adjust them properly according to the size restriction and limited memory of WAP devices. In other words, WAP applications should be designed as lightweight and uncomplicated.

2.5 Mobile Simulators and Toolkits

Toolkits for developing and testing WML code and simulating the application on a computer are available. The most popular toolkits are provided for download by the market leaders in the mobile solutions field like Nokia, Ericsson and Phone.com. Simulators emulate real mobile devices. They have display areas and buttons like the

real mobile devices. Mobile toolkits offer a variety of capabilities. They facilitate the development, testing and debugging of mobile applications with integrated editors, mobile simulators, and debuggers.

Toolkits usually include several mobile simulators. Those simulators differ from each other just like real mobile phones are different. It is good practice to test a mobile application on a variety of mobile simulators to make sure that it will run correctly not only on one device type [6]. (The reader is encouraged to try out the examples given below on his/her own device to see what they will look like.)

2.6 An Example of Static WML/WMLScript Content

A small example of a WAP application created with static WML and WMLScript code is presented for illustration. This example, preparing an order from a simple mobile store, will be later extended to dynamic content generation. A collection of WML cards is organized in a deck. The application covers all basic WML techniques. It uses WML templates, actions, links, and allows the user to enter values into input boxes. WML tables are employed for arranging the layout etc. and a WMLScript function is included to perform calculations.

Figure 2 shows some screenshots to demonstrate the look and feel of WML display features. In the second and third screen, the user enters an article number. Data belonging to that article number are returned in the fourth screen. Finally the user enters the quantity to be ordered and initiates calculation of the total price.

Due to the limited space in this paper the sample code is not included. It is provided for download on the authors' website instead for readers who would like to study and test it in detail [http://www.wiwi.euv-frankfurt-o.de/wi-www/de/research/project/fp_mcommerce.html].



Fig. 2. A mobile store example with static WML/WMLScript content

3 Creating Dynamic WAP Content from a Database

3.1 Java Server Pages as Basic Technology

All technologies for server-side programming mentioned before - Java Server Pages (JSP), Common Gateway Interface (CGI), Active Server Pages (ASP), and PHP Hypertext Preprocessor (PHP) - provide similar functionalities, in particular for developing and maintaining dynamic web content.

JSP was chosen for our purpose because it has some advantages over the other technologies. The major advantage is that JSP is based on the Java language, allowing the developer to make use of many powerful features of Java, for example multi-threading, network operations, and security. Another advantage of JSP is that a JSP file is compiled once and then kept in the server's memory. The compiled file is used each time the page is requested, making Java Server Pages a very efficient technology with better performance than other technologies.

JSP technology is an extension of the Java servlet technology. Java servlets are server-side modules employed to extend the capabilities and functionality of a web server. Servlets do not require the developer to reflect about platform-specific characteristics, providing platform independence, enhanced performance, and ease of administration and use [7]. The stages of processing a Java Server Page from request to response are illustrated in figure 3.

In terms of syntax a JSP document can be composed of HTML [8], XML [9], WML, or other elements which may contain JSP specific elements. A WML page with no JSP elements is also a valid JSP document if the file extension is changed from "wml" to "jsp". Such a document will be displayed after processing by the JSP engine in the same way as the wml file before. The JSP engine leaves WML (and likewise HTML/XML) statements untouched, executing only JSP elements. The JSP parts consist of Java code that is delimited by special JSP tags.

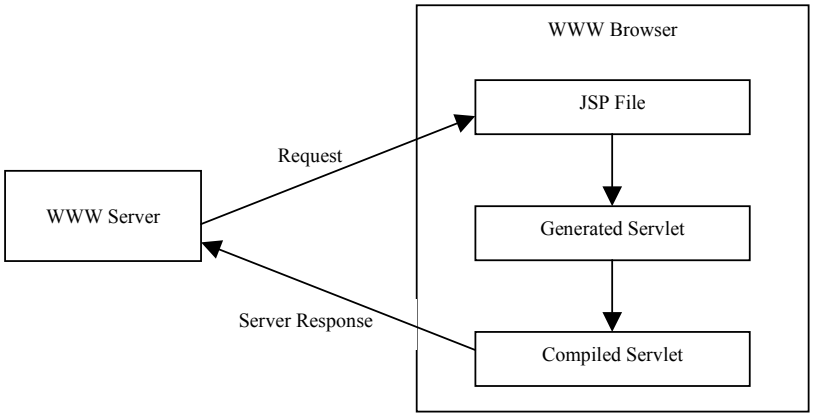


Fig. 3. JSP document processing phases

3.2 A Methodology for Creating Dynamic WAP Content

Starting with static content as in the section 2.6 example and then adding the dynamic parts proves to be a reasonable approach. The developer will first prepare static material focusing on the look, layout, and general functionality of the WML pages and on the processing mechanisms in WMLScript files. After these concepts are implemented, the static content is made dynamic. Note that it is easier to find and correct shortcomings and mistakes in the design if developing and testing the static content is done before. Subsequently the static pages serve as templates for layout, functionality, and navigation. In such a way, dynamic mobile applications have two major development stages: a static and a dynamic stage.

Creating the dynamic content typically goes through several phases: Setting the correct MIME type, obtaining user-defined parameters, establishing the connection to a database, querying the database for records matching the user's criteria, and writing out the results to a WML page. In more detail, the phases are characterized as follows.

Step 1: Setting the MIME type
For a WAP application to be served by a web server, the required MIME (Multipurpose Internet Mail Extensions) [10] type has to be set. Appropriate MIME types for WAP are listed in [11], [12], and in server documentations. The correct MIME type has to be set for the page in which dynamic WAP content will be generated. The MIME type determines the format of information that is exchanged between a WAP browser and the WAP server it connects to.

Step 2: Obtaining the user's parameters
Dynamic parts of a WAP page usually depend on some input given by the user. This input has to be obtained before the dynamic content can be generated. For example, the user will enter some value into a text box or select from a list in a WML page. These values may be used within the same page to adapt that page, or they may be passed as parameters to another page. Parameter passing will be discussed in section 4 below.

Step 3: Connecting to a database
In a business application as mentioned in the introduction, JSP pages often need to communicate with a database. In the example of figure 2, the article data returned to the user will probably be stored in the Mobile Store database. Upon retrieving the data from the database, the WAP page is generated according to those data.

The mechanisms that act in the middle and obtain the data use JDBC (Java Database Connectivity) features. The JDBC concept is based on the idea of database drivers that are tailored to specific database management systems. There are several kinds of JBDC drivers [13]. They differ from one another in their internal architectures and the methods of connecting an application with the database.

The Java Development Kit (JDK) includes one of these JDBC drivers called JDBC-ODBC bridge driver. This driver passes all SQL statements to an appropriate ODBC (Open Database Connectivity) driver. In this way Java based applications can interact with any database for which an ODBC driver is installed. A JDBC-ODBC

based approach is often deployed in an early project stage because it provides easy access to a database without much configuration effort. In a production environment this kind of bridge solution is usually replaced by a JDBC driver that can talk directly to the database and thus provide better performance.

Step 4: Sending queries to the database
The next phase is to query the database for records matching the criteria the user gave as input. JDBC statements that pass SQL statements to the database management system have to be created. The Java *Statement* interface is responsible for sending SQL statements to a database. An object with this interface (or its sub-interface *PreparedStatement*) provides the functionality to pass and execute SQL statements. In many cases, *PreparedStatement* is preferred for efficiency.

Step 5: Processing results
The last phase consists of processing the server's response that contains the results of the query found in the database. The response has to be written in WML format to a WAP page that is presented to the user. This is analogous to writing dynamic content to a WWW page in HTML format.

4 An Example of Creating Dynamic WAP Content

The steps of creating dynamic content are illustrated by the following example. We will extend the small mobile-store application of section 2.6, which had only static content, by JSP parts that create and shape the content in a dynamic way. The JSP scripts contain both WML tags and JSP specific tags. JSP tags and Java fragments are enclosed in `<% ... %>` marks. The final result produced by the scripts is WML content. The complete listing of the page discussed subsequently, ResultStore.jsp, is attached in the appendix.

4.1 Phases of Creating Dynamic WAP Content

Step 1: According to the above methodology, the first step is to establish an appropriate MIME type for responses in WML. This is done at the beginning of each dynamically generated WAP page. The type chosen here is `"text/vnd.wap.wml"`:

```
response.setContentType("text/vnd.wap.wml");
```

Step 2: The values entered by the user are passed as parameters from the request object. In this example it is an article number (parameter name `"code")` that is assigned to Java variable (also named `"code")` embedded in JSP:

```
String code = request.getParameter("code");
```

Step 3: Connection to the database requires a driver first. The JDBC-ODBC driver is obtained by the statement:

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

The `forName()` method dynamically loads the `JdbcOdbcDriver` class into the Java Virtual Machine (JVM). When the driver is loaded the connection to the database can be obtained with the help of the `getConnection()` method of the `DriverManager` class. This method passes three parameters: The name of the database, the user name, and the password.

```
Connection con;
con = DriverManager.getConnection
          ("jdbc:odbc:MobileStore","user","password");
```

Step 4: In order to run an SQL statement that selects and retrieves information from the database the `executeQuery()` method of a `Statement` object is usually used. The data matching the query criteria are returned in a Java `ResultSet` object. Likewise, SQL insert, update, and delete are done with the `executeUpdate()` method. The JSP statements for retrieving the data belonging to the article number `code` and storing them in `ResultSet` object `rs` are the following ones:

```
Statement statement;
ResultSet rs;
statement = con.createStatement();
rs = statement.executeQuery("select description, price,
stock from goods ” + “where code = '" + code + "'" );
```

Step 5: Now the data found in the database have to be processed and the WML page to be displayed has to be created. Since the article number is unique in our case there is at most one row in the `rs` object. This row is processed by the following statements:

```
if (rs == null) { %>
   Database error while processing article <%= code %>
<%  }
else if (code == null || code.equals("") ||
code.length() == 0) { %>
   Provide a proper article number
<% }
else if (!rs.next()) { %>
   There is no article with number <%= code%>
<% }
else {
         desc =  rs.getString(1);
         price = rs.getFloat(2);
         stock = rs.getFloat(3);
}%>
```

If all checks are satisfied, `get.String()` and `get.Float()` methods are used to obtain the values in the first, second, and third column, respectively; i.e. the article's description, price, and quantity on stock. (If the article number were not unique, the `next()` method of the `ResultSet` object could be used to move the cursor to the next row.) The variables `desc, price`, and `stock` are used to display the respective values in the WML table in the second half of listing 1 (see appendix).

Not only WML files but also WMLScript files can be generated in a dynamic manner. It is important to note how these dynamic scripts works. First the script code is generated on the server, perhaps tailored to specific parameters provided by the user. Afterwards the produced source code of the script is compiled into virtual byte code on the WAP gateway (server), loaded, and run in the environment of the virtual machine on the WAP client device.

In general, developing a dynamic WMLScript is similar to developing a dynamic WML file, but there is no need to write out database results. An appropriate MIME type for WMLScript files has to be chosen, e.g. `"text/vnd.wap.wmlscript"`. Dynamic WMLScripts consist of Java and JSP tags embedded in WMLScript code. If it is needed to transfer dynamic data from a data source into a WMLScript the mechanisms for communicating with the database are the same as explained above.

To demonstrate a dynamic script the function to calculate the order total has been extended to obtain some data from the database (see listing 2 in the appendix). An article number is passed to the database and the article's price and description are returned. The script then calculates the total from the price and the quantity of that item.

4.2 Exchanging Information by Passing Parameters

In most applications data entered by the user are required. These data are usually needed in different pages. Instead of making the user enter the data repeatedly it is better to pass them as parameters between different pages.

Both languages used in our type of mobile application, WML and JSP/Java, provide variables. Those variables can be used as parameters to exchange information between documents containing mobile content. Obviously both kinds of variables can serve as parameters.

In the following code sample, the dynamically created WMLScript function `CalcOrder()` is invoked with an actual parameter list that contains WMLScript variables as arguments (e.g. `total`). At the same time a JSP variable (`code`) is passed to the `StoreFunctions_wmls.jsp` page as a WML parameter in the `<postfield>` tag. Actually this is also an example of calling a dynamically created WMLScript function (`CalcOrder`, see appendix).

```
<do type="accept" name="Calc" label="Calculate">
   <go href="StoreFunctions_wmls.jsp#CalcOrder
           ('total','price','desc','$(quantity)')">
<!– WML variables -->
     <postfield name="code" value="<%= code %>"/>
<!– JSP variable  -->
         </go>
</do>
```

An alternative way to combine the two different types of parameters in the example would be to concatenate the parameter `code` and the invocation of the WMLScript function instead of using `postfield` technique:

```
<go href="StoreFunctions_wmls.jsp?code=<%= code %>
&amp;#CalcOrder('total','price','desc','$(quantity)')"/>
```

An analogous combination of WML and JSP variables could be passed as parameters to a dynamically generated WML file. The sample code illustrates how to create a WML link to a dynamic WML page and pass parameters to it:

```
<a href="Order.jsp?code=<%= code %>&amp;
        quantity=$(quantity)" title="Order">Order</a>
```

5 Summary and Outlook

The mobile WAP technology offers new opportunities for many useful application domains like mobile shopping (m-shopping), m-news, m-entertainment, m-banking, and more. In the business-to-business area, sales representatives and service personnel can be effectively supported by mobile services as mentioned in the introduction. Most of these applications require dynamic generation of content from databases. We discussed the process of generating dynamic WAP content starting from a request of the mobile device through all phases until the requested content is displayed on the device's screen. Java Server Pages is the technology used to create WML and WMLScript files on the fly.

The presentation layer concepts for mobile clients are still evolving. Eventually the presentation layer will be enhanced by a more sophisticated language than WML with more powerful display features. One promising candidate is the eXtensible Hypertext Markup Language (XHTML) [14]. However, the methodology of developing the presentation layer and the dynamic user-tailored content we presented will most likely remain the same. In other words, one can develop a mobile application in the same way, taking advantage of XHTML instead of WML to describe the presentation layer. Although our examples were quite small, the methodology we followed and the resulting architecture of the application are the same for larger, real-world applications.

References

1. Arehart, C. et al.: Professional WAP; WROX Press, Chicago 2000.
2. See WML overview: http://www.wapforum.org/what/WAPWhite_Paper1.pdf.
3. Ewing, J.: The Wireless Application Protocol; http://www.topxml.com/wap/articles/wapart2/.
4. See WAP Overview: http://download.forum.nokia.com/download/WAP2000Overview.pdf.
5. Wugofski, T., Lee, W.M., Foo, S.M., Watson, K.: Beginning WAP: Wireless Markup Language & Wireless Markup Language Script; WROX Press, Chicago 2000.
6. Kurbel, K., Dabkowski, A., Zajac, P.: Software Technology for WAP Based M-Commerce - A Comparative Study of Toolkits for the Development of Mobile Applications (to appear).
7. See JavaServer Pages: http://java.sun.com/products/jsp/
8. See HTML: http://www.w3.org/MarkUp/
9. See XML: http://www.w3.org/TR/xmlbase/
10. NACS Network & Academic Computing Services, University of California: MIME types references' link : http://www.nacs.uci.edu/indiv/ehood/MIME/MIME.html

11. Watson, K.: Wap for ASP Developers:
http://www.vbxml.com/conference/wrox/2000_vegas/text/karli_wap.pdf.
12. See a list of common MIME types at: http://hostutopia.com/support/s058.html.
13. Goodwill, J.: Pure Java Server Pages; Sams Publishing, Indianapolis 2000.
14. See XHTML: http://www.w3.org/TR/xhtml-basic/

Appendix

Listing 1 ResultStore.jsp

```
<% // establish appropriate MIME type for WML content
    response.setContentType("text/vnd.wap.wml"); %>

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

  // JSP directive importing additional packages
<%@ page language="java" import="java.sql.*" %>

<% // beginning of Java block
  // variable code for article, no. passed as parameter
  String code = request.getParameter("code");
  String desc = null;  // article description
  float price, stock;  // price and current stock quant.

  Connection con = null; // capturing of database conn.
  ResultSet rs = null;  // for storing database results
// variable for sending SQL statement to database
  Statement statement = null;
%>  // end of Java block
<wml>            // WML content
   <card id="StoreResult" title="Article #<%= code%>">
      <do type="prev" name="Back" label="Back">
         <prev/>
      </do>
   <p mode="nowrap">
<%      // beginning of Java block
  try {
  // loading the JDBC-ODBC database driver
  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  // opening a connection to the database
  con = DriverManager.getConnection
                      ("jdbc:odbc:MobileStore","","");
  // creating statement object for querying the database
  statement = con.createStatement();
  // sending query to database
  rs = statement.executeQuery
      ("select description, price, stock from goods " +
       "where code = '" + code + "'" );
  if( rs == null) {
```

```
 %> // end of Java; writing Java variable in WML;
    Database error while processing article <%= code %>
   <%  // subsequently: more Java and WML intertwined
  }
  else if (code == null || code.equals("") ||
code.length() == 0)
  {
 %>   Provide a proper article number
 <% }  // there are no records matching our selection
  else if (!rs.next()) {  %>
       There is no article with number <%= code%>
 <% }
  else
  {   // processing the data retrieved from database
    desc  = rs.getString(1);
    price = rs.getFloat(2);
    stock = rs.getFloat(3);
 %>    // end of Java block
    // writing article data to a WML table
    <table title="Title" align="center" columns="2">
      <tr>
        <td><small><b>Code</b></small></td>
        <td><%= code %></td>
      </tr>
      <tr>
        <td><small><b>Description</b></small></td>
        <td><%= desc %></td>
      </tr>
      <tr>
        <td><small><b>Price</b></small></td>
        <td>$$<%= price %></td>
      </tr>
      <tr>
        <td><small><b>Stock</b></small></td>
        <td><%= stock %></td>
      </tr>
    </table>

 <%}   // end Java else and continue try-catch block
    }
    catch (Exception e) {
       out.println (e.getMessage());
    }
    finally
    { // freeing resources allocated for ResultSet
       rs.close();
       if (con != null)  // and Connection objects
         con.close();
    }
 %>
   </p>
   </card>
</wml>
```

Listing 2 StoreFunctions_wmls.jsp

```
<% // establishing app. MIME type for WMLScript content
  response.setContentType("text/vnd.wap.wmlscript"); %>

 // JSP directive importing additional packages of
//  java.lang
<%@ page language="java" import="java.sql.*" %>

<% // beginning of Java block
   // variable code for article; number passed as
   // parameter
   String code = request.getParameter("code");
   String description = null;  // article description
   Connection con = null;     // database connection
   ResultSet rs = null;    // storing database results
   // variable for sending SQL statement to database
   Statement statement = null;

try
{ // loading JDBC-ODBC database driver
  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  // opening a connection to the database
  con = DriverManager.getConnection
            ("jdbc:odbc:MobileStore","","");
  // creating a Statement obj for querying the database
  statement = con.createStatement();
  String sql = "select description, price from goods " +
               "where code = '" + code + "'";
  // concatenating SQL statement with a parameter
  // sending query to database
  rs = statement.executeQuery(sql);

%> // end of Java block and beginning of WMLScript
   // function

extern function CalcOrder(total, str_price, description,
quantity)
{ // WMLScript variables
  var amount = 0;
  var desc = 'Description : ';
  var str_p;

<% // start of Java block
  // there are no records matching our selection
  if (!rs.next())  {
%>  // end of Java block

  var p = -1;   // assignment to WML variable price
  desc = desc + 'not found';
<%  }
  else
  {
```

```
%>
  // price and description retrieved from database are
  // assigned to WMLScript variables
  var p = <%= rs.getFloat(2); %>;
  desc = desc + '<%= rs.getString(1) %>';
  <% }
%>   // end of Java; WMLScript operations begin

  // converting price (p) to String and concatenation
  str_p = 'Price : ' + String.toString(p);
  amount = quantity * p;

  // WMLScript variable values are passed to WML
  // browser by setting browser-defined variables
  WMLBrowser.setVar(description, desc);
  WMLBrowser.setVar(str_price, str_p);
  WMLBrowser.setVar(total, String.toString(amount));
  // WML browser content is explicitly refreshed to
  // make the current values visible
  WMLBrowser.refresh();

  } // end of WMLScript function;
<%  // continuation of Java try-catch block
}
catch (Exception e)
{
  out.println (e.getMessage());
}
finally
{
  // freeing resources allocated for ResultSet
  rs.close();
  if (con != null) // and Connection objects
    con.close();
}
%>
```