

## Introducing WALL: a Library to Multiserve Applications on the Wireless Web

by Luca Passani  
passani at eunet dot no  
luca dot passani at openwave dot com

More and more new WAP 2.0 devices hit the market each week. WAP 2.0 introduced XHTML MP 1.0 (Mobile Profile) as the mark-up for developing wireless applications.

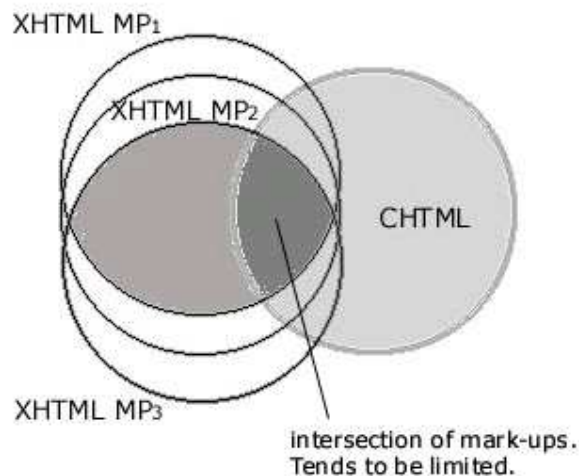
While XHTML MP gives up on many of the usability extensions introduced with WML 1.X, it promises to bring convergence with HTML, the mark-up of the big Web.

Unfortunately, XHTML-MP implementations differ from browser to browser. These difference are sometimes tiny and neglegible. Other times they are subtle and can make your applications fail on certain devices totally unexpectedly. Other times still, the differences are so big that you can hardly deploy an XHTML MP application which works usably enough on all popular WAP 2.0 devices.

To add to to that, another mark-up language is showing up in GSM regions too: Compact HTML (CHTML), the mark-up language of NTT Docomo's I-Mode. CHTML also derives from HTML, but it's different enough that an application can hardly work on any WAP 2.0 device and an i-mode device without problems.

### XHTML Mobile Profile dialects and Compact HTML

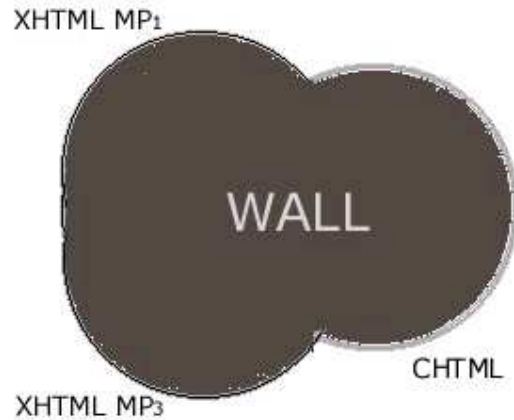
abstracting away HTML-derived mark-ups.



**Figure 1:** Intersection of different mark-up languages

**Figure 1** shows the intersection of the different mark-up visually. In theory, developers should code their XHTML apps using tags that are interpreted equally by all the devices. Not only does this approach detract substantially from the expressive power of HTML, but there is a risk that future device will make the intersection even smaller.

The WALL tag-library addresses this issue. By exploiting the power of the WURFL device capability database (<http://wurfl.sourceforge.net>), WALL can transparently detect what a device can do and make sure that the best possible mark-up is delivered to that device. **Figure 2** below illustrates this by 'covering' the mark-up range of the sum of all possible HTML-derived mark-ups.



**Figure 2:** WALL covers the sum of all mark-ups

The basic idea behind WALL is better explained with one example. Let's consider a tag as banal and obvious as the time-honored line break "br". In the old HTML days, the tag was as simple as `<br>`. A few years back, XML took every aspect of computer programming by storm and HTML was no exception. W3C turned HTML 4.0 into an XML-application called, XHTML 1.0 (from which XHTML Basic and XHTML MP were successively derived). As a result, a tag as simple as `<br>` had to be expressed as `<br/>` under new XML rules.

While XML may be a standard, having to add extra trailing 'slashes' to singleton tags went against established HTML rules.

When it comes to wireless devices, here is the situation:

- most CHTML devices honor the `<br>` tag and simply ignore `<br/>` tags
- most XHTML MP devices will honor both `<br/>` and `<br>` tags.
- some XHTML MP device will **throw an error** if a single unslashed `<br>` is found and no information will be displayed (except a pretty useless error message)

If you want to stick to a single XHTML page and expect it to work on all devices, you are left with the only option of using `<br/>` and making your users endure lousy page layout on CHTML devices.

The wall library provides an elegant solution by supporting the `<wall:br/>` tag.

This tag will:

- look at the user-agent for you,
- query the WURFL to figure out which mark-up the requesting device prefers, and
- issue `<br/>` or `<br>`, depending on the capability of that device.

The WURFL capability that is queried to decide which mark-up to produce is `preferred_markup`.

*If you are curious about how this happens, here is the core part of the Java code which implements the support for the `<wall:br/>` tag.*

**Don't be scared. You are not required to deal with raw java code at all to use the tag-lib! This is just FYI.**

```

//get the user agent
UA = TagUtil.getUA(this.request);

device_id = uam.getDeviceIDFromUALoose(UA);

markup = cm.getCapabilityForDevice(device_id,
                                   "preferred_markup");

//XHTML <br/> with trailing slash
if ( markup.indexOf("xhtmlmp") != -1) {
    try {
        JspWriter out = pageContext.getOut();
        out.println("<br/>");
    } catch(IOException ioe) {
        System.out.println("Error in br tag: " + ioe);
    }
    return(SKIP_BODY);
}

//CHTML <br/> without trailing slash
if ( markup.indexOf("chtml") != -1) {
    try {
        JspWriter out = pageContext.getOut();
        out.println("<br>");
    } catch(IOException ioe) {
        System.out.println("Error in br tag: " + ioe);
    }
    return(SKIP_BODY);
}
}

```

Many WALL tags can be used in stand-alone mode, in the middle of a perfectly normal XHTML page. For this reason, the tag-library is simple to use, Developers can learn it on a need-to-know basis, by adopting the tags that helps with specific problems.

Not all tags can be used in stand-alone mode, though. Some tags require proper nesting to make sense. This will be clear in the examples I will show shortly.

### Bring in WML 1.X WML compatibility mode

Of course, wireless is not HTML only. Quite the contrary, in fact. Most wireless devices around are WAP 1.X devices which only understand some flavors of WML 1.X.

The good news is that you can use WALL to abstract away WML too, so you won't be forced to build a separate version of your application for WAP 1.X devices.

Of course, you need to use **\*more\*** WALL tags if you wish to abstract away WML tags too.

The following simple example will clarify this.

If all you need is to abstract away HTML-capable devices of some kind, you will get away with something as simple as the following mark-up, in which plain tags and WALL tags are freely intermixed:

```

<wall:form action="http://url" method="post">
    Your name:<wall:br/>
    <input type="text" name="somename" value=""/>
    <wall:br/>
    <input type="submit" value="Submit" />
</wall:form>

```

*(for the impatient, <wall:form> is necessary to enforce compliance to the XHTML-MP DTD on all devices or some XHTML MP devices will break)*

As one can expect, the code above will not work on WML devices, since the submit button is not understood and it's outside of the WALL taglib domain.

If you want to support WML too, you will need to use WALL in WML compatibility mode, which basically means that each and every tag will need to be expressed as a WALL tag.

Porting the example above to WML Compatibility Mode involves recoding it as follows:

```

<wall:form action="http://url" method="post" enable_wml="true">
  Your name:<wall:br/>
  <wall:input type="text" name="somename" value=""/>
  <wall:br/>
  <wall:input type="submit" value="Submit" />
</wall:form>

```

Of course, the WML produced by WALL will not be idiomatic. You can forget about explicitly programming softkeys, using WMLScript, timers and similar amenities. You will need to be happy with not letting anyone down, and that's not little in today's fragmented browser market.

**Note:** *if you are serious about multiserving good WML, you can refer to the Open Usability Interface library (OUI), available at <http://oui.sourceforge.net>.*

After all these words, it's time to consolidate the concepts with a few well-chosen example. **body.jsp** will show a bunch of simple WALL tags in action.

Following the JSP listing, you can see the three kinds of mark-up that are generated for WML, XHTML and CHTML respectively:

### body.jsp

*The wall:document tag needs to be on the very same top line right after the tag-lib declaration, if you want to support WML too.*

```

<%@ taglib uri="/WEB-INF/tld/wall.tld" prefix="wall" %><wall:document>
<wall:xmlpidtd />

<wall:head>
  <wall:title enforce_title="true">My Document</wall:title>
  <!--sent to all devices as it is -->
  <meta name="value" content="value" />
</wall:head>
<wall:body>
  <wall:block>
UA :
<wall:marquee>
  <%= request.getHeader("User-Agent") %>
</wall:marquee>
  <wall:br />
  Body part 2
  </wall:block>
</wall:body>
</wall:document>

```

## WML 1.X

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC
    "-//WAPFORUM//DTD WML 1.1//EN"
    "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<head>
  <meta name="taglib" content="WALL" />
  <!--sent to all devices as it is -->
  <meta name="value" content="value" />
</head>
<card title="My Document">
<p>My Document</p>
  <p>
    UA :SIE-SLCK/3.1 UP/4.1.19i
    <br/>
    Body part 2
  </p>
</card>
</wml>
```

## CHTML

```

<html>
<head>
<title>My Document</title>
  <!--sent to all devices as it is -->
  <meta name="value" content="value" />
</head>
<body>

UA :
<marquee>portalmmm/2.0 N341i(c10;TB)</marquee>
  <br>
  Body part 2

</body>
</html>

```

## XHTML MP 1.0

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC
  "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
  "http://www.wapforum.org/DTD/xhtml-mobile10.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
  xml:lang="en">
<head>
<title>My Document</title>
  <!--sent to all devices as it is -->
  <meta name="value" content="value" />
</head>
<body>
  <p>

UA :
<div mode="nowrap">
  SIE-M55/10 UP.Browser/6.1.0.5.c.6 (GUI) MMP/1.0
</div>
  <br/>
  Body part 2
  </p>
</body>
</html>

```

Enclosing the whole document inside a `<wall:document>` tag is advised for non-banal application (and absolutely necessary for WML Compatibility Model). This tag collects status info for the page while it renders. Status is needed by other tags to behave optimally.

The `wall:document` tag needs to be on the very same top line immediately after the tag-lib declaration, if you intend to support WML too. The reason for this is that you don't want to give your application server a chance to produce the default MIME-Type (usually "text/html"). This would cause an error on WML devices.

You may find that the `xmlpi` tag needs to be on the first line as well. This ensures that the XML processing instruction starts on the very first line as mandated by the XML spec (while this is insignificant in most cases, some old WAP gateways will break on this).

`<wall:xmlpi>` (Mnemonic: XML Processing Instruction and DTD) makes sure that the right XML headers are produced for the mark-up that is being generated. Just remember to put it there and be happy.

`<wall:head>` this is a good place to add your meta tags and, of course, the title of your document.

`<wall:body>` and `<wall:block>` need to enclose each page (block can also be replaced by a menu or a form. More later). You can have multiple blocks but only one body (note: wrt non-idiomatic WML, this implies that WALL will only produce WML decks with one single card).

The block tag is necessary to ensure well-formedness and validity for both WML and XHTML-MP. As a matter of fact, some WAP 2.0 browsers fail to display non-valid XHTML MP content, even when the lack of validity is due to missing `<p>` tags, which are mandated by the XHTML basic DTD (in spite of the fact, that those paragraph tags feel unnatural to programmers with an HTML background).

<wall:marquee> is a tribute to the CHTML tag to scroll text horizontally (this is actually a Microsoft invention for MSIE 2 back in 96). Using this tag will also try to render marquee on XHTML devices which support that through CSS syntax.

<wall:br> is straightforward. <br> to CHTML devices and <br /> to WML and XHTML devices.

The next example is about the most common navigation pattern: a menu. While all devices support some basic menus, it would be a pity to have to feed spartan lists of hyperlinks also to the newest XHTML color devices too. The WALL library has some good abstractions.

### Menu.jsp

The wall:menu\_css tag enables advanced menus for those XHTML devices which support it.

```
<%@ taglib uri="/WEB-INF/tld/wall.tld" prefix="wall" %><wall:document>
<wall:xmlpidtd />
<wall:head>
  <wall:title>Entertainment</wall:title>
  <wall:menu_css />
</wall:head>

<wall:body>
  <wall:menu colorize="true" autonumber="true">
    <wall:a href="http://url1" title="Games">Games</wall:a>
    <wall:a href="http://url2" title="Horos">Horoscopes</wall:a>
    <wall:a href="http://url1" title="Kids">Kids</wall:a>
    <wall:a href="http://url2" title="Movies"><wall:b>Movies</wall:b></wall:a>
    <wall:a href="http://url1" title="Music">Music</wall:a>
    <wall:a href="http://url2" title="Radio">Radio</wall:a>
    <wall:a href="http://url2" title="TV">TV</wall:a>
  </wall:menu>
</wall:body>
</wall:document>
```

### WML 1.X (Openwave UP.Browser 4)

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC
  "-//WAPFORUM//DTD WML 1.1//EN"
  "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<head>
  <meta name="taglib" content="WALL" />
</head>

<card id="w" title="Entertainment">
  <p align="left" mode="nowrap">
<select>
  <option onpick="http://url1" title="Games">Games</option>
  <option onpick="http://url2" title="Horos">Horoscopes</option>
  <option onpick="http://url1" title="Kids">Kids</option>
  <option onpick="http://url2" title="Movies">Movies</option>
  <option onpick="http://url1" title="Music">Music</option>
  <option onpick="http://url2" title="Radio">Radio</option>
  <option onpick="http://url2" title="TV">TV</option>
</select>
</p>
</card>
</wml>
```

### WML 1.X (non Openwave)

```

<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC
    "-//WAPFORUM//DTD WML 1.1//EN"
    "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<head>
    <meta name="taglib" content="WALL" />
</head>

<template>
<do type="prev" label="Back">
    <prev/>
</do>
</template>
<card id="w" title="Entertainment">
<p>
    <a href="http://url1" title="Games">Games</a><br/>
    <a href="http://url2" title="Horos">Horoscopes</a><br/>
    <a href="http://url1" title="Kids">Kids</a><br/>
    <a href="http://url2" title="Movies">Movies</a><br/>
    <a href="http://url1" title="Music">Music</a><br/>
    <a href="http://url2" title="Radio">Radio</a><br/>
    <a href="http://url2" title="TV">TV</a><br/>
</p>
</card>
</wml>

```

### XHTML MP 1.0 (plain)

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC
    "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
    "http://www.wapforum.org/DTD/xhtml-mobile10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xml:lang="en">
<head>
<title>Entertainment</title>
</head>
<body>

<p>
<ol>
    <li><a acceskey="1" href="http://url1" title="Games">Games</a></li>
    <li><a acceskey="2" href="http://url2" title="Horos">Horoscopes</a></li>
    <li><a acceskey="3" href="http://url1" title="Kids">Kids</a></li>
    <li><a acceskey="4" href="http://url2" title="Movies"><b>Movies</b></a></li>
    <li><a acceskey="5" href="http://url1" title="Music">Music</a></li>
    <li><a acceskey="6" href="http://url2" title="Radio">Radio</a></li>
    <li><a acceskey="7" href="http://url2" title="TV">TV</a></li>
</ol>
</p>
</body>
</html>

```

### XHTML MP 1.0 (cool)

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC
    "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
    "http://www.wapforum.org/DTD/xhtml-mobile10.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
    xml:lang="en">
<head>
<title>Entertainment</title>

<style>
    .bgcolor1 { background-color:#99CCFF;}
    .bgcolor2 { background-color:#FFFFFF;}
</style>

</head>

<body>
<p>
<table>

<tr>
<td class="bgcolor1"> 1 <a acceskey="1" href="http://url1" title="Games">
    Games</a>
</td>
</tr>
<tr>
<td class="bgcolor2"> 2 <a acceskey="2" href="http://url2" title="Horos">
    Horoscopes</a>
</td>
</tr>
<tr>
<td class="bgcolor1"> 3 <a acceskey="3" href="http://url1" title="Kids">
    Kids</a>
</td>
</tr>
<tr>
<td class="bgcolor2"> 4 <a acceskey="4" href="http://url2" title="Movies">
    <b>Movies</b></a>
</td>
</tr>
<tr>
<td class="bgcolor1"> 5 <a acceskey="5" href="http://url1" title="Music">
    Music</a>
</td>
</tr>
<tr>
<td class="bgcolor2"> 6 <a acceskey="6" href="http://url2" title="Radio">
    Radio</a>
</td>
</tr>
<tr>
<td class="bgcolor1"> 7 <a acceskey="7" href="http://url2" title="TV">
    TV</a>
</td>
</tr>
</table>

</p>
</body>
</html>

```

## CHTML

```

<html>
<head>
    <title>Entertainment</title>
</head>

<body>

    <br clear="all">
    &#59106;&nbsp;<a acceskey="1" href="http://url1">Games</a><br>
    &#59107;&nbsp;<a acceskey="2" href="http://url2">Horoscopes</a><br>
    &#59108;&nbsp;<a acceskey="3" href="http://url1">Kids</a><br>
    &#59109;&nbsp;<a acceskey="4" href="http://url2"><b>Movies</b></a><br>
    &#59110;&nbsp;<a acceskey="5" href="http://url1">Music</a><br>
    &#59111;&nbsp;<a acceskey="6" href="http://url2">Radio</a><br>
    &#59112;&nbsp;<a acceskey="7" href="http://url2">TV</a><br>
</body>
</html>

```

I will describe the new tags used in the menu example shortly, but first a couple of pictures to illustrate visually how the menu is shown on the different devices.



**Figure 3:** Menu on simple WML devices and primitive XHTML MP devices.



**Figure 4:** Menu on XHTML devices with decent table and CSS support.



**Figure 5:** Menu on Imode CHTML devices.



**Figure 6:** Menu on devices featuring UP.Browser 4 by Openwave (WML 1.X).

Let's look at the tags used in **menu.jsp**.

In order to have the nice, cool and colorized menus, you'll need to inject a few lines of CSS into your page (refer to the cool XHTML mark-up listed above). `<wall:menu_css>` does exactly this. The CSS lines will only be produced for XHTML MP devices that have decent table and CSS support. The library relies on two WURFL capabilities for this:

```
xhtml_supports_css_cell_table_coloring and
xhtml_supports_table_for_layout.
```

If you wonder where those two color values in the example come from, the answer is still the WURFL:

```
xhtml_readable_background_color1 and
xhtml_readable_background_color2.
```

These properties were introduced because it is very difficult to find "readable" colors which works ok across different devices, so this looked like a typical resourcification job for the WURFL.

Of course, you may want to vary the color in your JSPs. For these reason a couple of attributes will turn up to be handy:

```
<wall:menu_css bgcolor1="blue" bgcolor2="pink">
```

One extra note is about the fact that using this tag is not enough to get colorized menus. You also need to enable the `colorized="true"` part in the `<wall:menu>` tag.

`<wall:menu colorize="true" autonumber="true">` tells your JSP that you are about to define a menu. This has an implication on how the hyperlinks inside the menu actually behave wrt graphics and handling control of accesskeys definition over to the menu itself. The `autonumber` attribute instructs the menu to generate accesskey automatically. If false, or not set, the programmer can still set accesskeys manually on each single hyperlink. The `colorize` attribute is the one that authorizes the menu to produce cool graphics for devices that support it.

You should note that there is no `wall:block` tag as in the **body.jsp** example.

`wall:block` and `wall:menu` (and `wall:form` to be introduced later) are, so to speak, on the same level, and you should use one or the other, but avoid nesting them. If you are used to HTML, this may seem a bit unnatural. The fact is that some XHTML MP devices only parse perfectly valid XHTML-MP mark-up. Using blocks, menus and forms as presented in the examples is necessary to keep the page valid (which doesn't mean you won't have several chances to break validation other pages in your document. The whole validation thing makes your pages pretty fragile when compared to HTML).

The `<wall:a>` tag is pretty straightforward. It's a hyperlink as we know it from virtually any mark-up we have seen around. There is nothing special to say about this tag, if not that it has a double life depending on whether it's located inside a menu tag or not. Also, no matter if you define accesskeys manually or automatically, accesskeys will not be produced for WML 1.X devices which do not support them.

WURFL capabilities:

```
menu_with_select_element_recommended
```

The reason for `<wall:b>` tag (as opposed to using a plain `<b>` tag) is that you can't use the `b` tag inside an anchor, or the WML will break. Using `<wall:b>` fixes this nicely by making sure that it's not WML and it's not inside an anchor, before rendering your `<b>`. `<wall:i>` also exists.

Let's now move to forms. As you can expect, forms in WML differ substantially from what we know from HTML. For this reason, forms come in two flavors. **form.jsp** illustrate the simple case (XHTML/CHTML only):

### form.jsp

*This example is simple. No WML Compatibility Mode. Will only work for XHTML MP and CHTML devices.*

```
<%@ taglib uri="/WEB-INF/tld/wall.tld" prefix="wall" %><wall:document>
<wall:xmlpiDTD />

<wall:head>
  <wall:title>Form 1</wall:title>
</wall:head>

<wall:body>
  <wall:form action="url" method="post">
    <!-- Greatly simplified under the assumption that - ->
    <!-- WML is not supported -->
    Economy or Business:<br/>
    <input type="radio" name="course" value="economy" checked="checked"/>E
    <input type="radio" name="course" value="business"/>B<wall:br />

    Add comment: <wall:br />
    <textarea name="comment" rows="2" cols="10"></textarea>
    <wall:br />
    <input type="hidden" name="user_id" value="bhgfd65488769" />
    <input type="submit" value="Next"/>
  </wall:form>
</wall:body>
</wall:document>
```

### XHTML MP 1.0

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC
  "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
  "http://www.wapforum.org/DTD/xhtml-mobile10.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
  xml:lang="en">
<head>
  <title>Form 1</title>
</head>

<body>
<form action="url" method="post">
  <p>

    <!-- Greatly simplified under
      the assumption that - ->
    <!-- WML is not supported -->
    Economy or Business:<br/>
    <input type="radio" name="course"
      value="economy" checked="checked"/>E
    <input type="radio" name="course"
      value="business"/>B<br/>

    Add comment: <br/>
    <textarea name="comment" rows="2" cols="10">
    </textarea>
    <br/>

    <input type="hidden" name="user_id"
      value="bhgfd65488769" />
    <input type="submit" value="Next"/>
  </p>
</form>

</body>
</html>

```

## CHTML

```

<html>
<head>
  <title>Form 1</title>
</head>

<body>
<form action="url" method="post">

  <!-- Greatly simplified under
    the assumption that -->
  <!-- WML is not supported -->
  Economy or Business:<br/>
  <input type="radio" name="course"
    value="economy" checked="checked"/>E
  <input type="radio" name="course"
    value="business"/>B<br>

  Add comment: <br>
  <textarea name="comment" rows="2" cols="10">
  </textarea>
  <br>

  <input type="hidden" name="user_id"
    value="bhgfd65488769" />
  <input type="submit" value="Next"/>
</form>
</body>
</html>

```

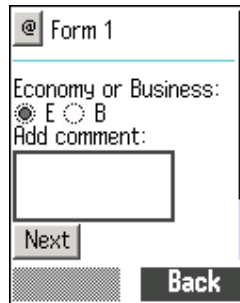
## WML 1.X

```

<?xml version="1.0"?>
<!DOCTYPE wml
PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<head>
<meta name="taglib" content="WALL" />
</head>

<card id="w" title="Form 1">
<p>This page is not available to
WAP 1.X devices.
If you think this is an error,
please contact your service provider.</p>
</card>
</wml>

```



**Figure 7:**Form on XHTML MP device

Are forms available in WML Compatibility Mode? they are, but there are some limitations. You only have text fields, hidden fields, passwords and select element. No radio and no check-buttons. This is due to the differences in how WML browsers post data as compared to XHTML and CHTML. The current solution ensures that data are posted to the receiving service in the same format regardless of the device. In addition, the syntax used is similar to XHTML, as illustrated by the **form2.jsp** example.

### **form2.jsp**

*Forms are available in WML Compatibility Mode, but they are more limited. Say goodbye to radio and check-boxes.*

```

<%@ taglib uri="/WEB-INF/tld/wall.tld" prefix="wall" %><wall:document>
<wall:xmlpidtd />

<wall:head>
  <wall:title>Form 2</wall:title>
</wall:head>

<wall:body>
  <wall:form action="url" method="post" enable_wml="true">
Pin Code:
  <wall:input type="text" name="pincode" value="" format="NNNN" maxlength="4" />
<wall:br />
Choose:
  <wall:select title="Day" name="day">
    <wall:option value="11/28/04">Yesterday
  </wall:option>
    <wall:option value="11/29/04" selected="selected">Today
  </wall:option>
    <wall:option value="11/30/04">Tomorrow
  </wall:option>
  </wall:select>
<wall:br />
  <wall:input type="hidden" name="session" value="gfsa87837" />

  <wall:input type="submit" value="Go" />
  </wall:form>
</wall:body>
</wall:document>

```

### **XHTML MP 1.0**

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
    "http://www.wapforum.org/DTD/xhtml-mobile10.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
    xml:lang="en">
<head>
  <title>Form 2</title>
</head>

<body>
<form action="url" method="post">
  <p>

Pin Code:
  <input type="text" name="pincode" value="" format="NNNN" maxlength="4" />
<br/>

Choose:
  <select name="day" title="Day">
    <option value="11/28/04">Yesterday</option>
    <option value="11/29/04" selected="selected">Today</option>
    <option value="11/30/04">Tomorrow</option>
  </select>
<br/>

  <input type="hidden" name="session" value="gfsa87837"/>
  <input type="submit" value="Go" />
  </p>
</form>
</body>
</html>

```

## CHTML

```

<html>
<head>
  <title>Form 2</title>
</head>

<body>
<form action="url" method="post">

Pin Code:
  <input type="text" name="pincode" value="" maxlength="4" istyle="4">
<br>

Choose:
  <select name="day" title="Day">
    <option value="11/28/04">Yesterday</option>
    <option value="11/29/04" selected>Today</option>
    <option value="11/30/04">Tomorrow</option>
  </select>
<br>

  <input type="hidden" name="session" value="gfsa87837">

  <input type="submit" value="Go" />
  </form>
</body>
</html>

```

## WML 1.X (Openwave Mobile Browser)

```

<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
    "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<head>
  <meta name="taglib" content="WALL" />
</head>

<card id="w" title="Form 2">
  <p>
Pin Code:
  <input type="text" name="pincode" value="" format="NNNN" maxlength="4" />
<br/>

Choose:
  <select name="day" title="Day">
    <option value="11/28/04">Yesterday</option>
    <option value="11/29/04">Today</option>
    <option value="11/30/04">Tomorrow</option>
  </select>
<br/>

<do type="accept" label="Go">
  <go href="url" method="post">
    <postfield name="session" value="gfsa87837" />
    <postfield name="pincode" value="$pincode" />
    <postfield name="day" value="$day" />
  </go>
</do>
</p>
</card>
</wml>

```

## WML 1.X (Non-Openwave Browser)

```

<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
    "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<head>
  <meta name="taglib" content="WALL" />
</head>

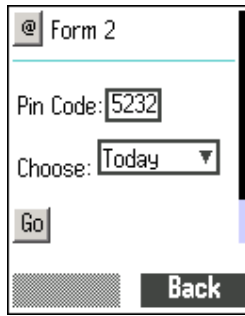
<template>
  <do type="prev" label="Back">
    <prev/>
  </do>
</template>
<card id="w" title="Form 2">
  <p>

Pin Code:
  <input type="text" name="pincode" value="" format="NNNN" maxlength="4" />
<br/>

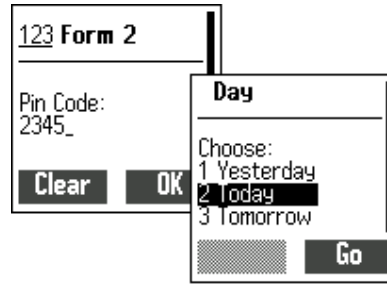
Choose:
  <select name="day" title="Day">
    <option value="11/28/04">Yesterday</option>
    <option value="11/29/04">Today</option>
    <option value="11/30/04">Tomorrow</option>
  </select>
<br/>

<anchor>Go
  <go href="url" method="post">
    <postfield name="session" value="gfsa87837" />
    <postfield name="pincode" value="$pincode" />
    <postfield name="day" value="$day" />
  </go>
</anchor>
</p>
</card>
</wml>

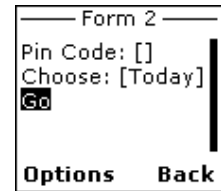
```



**Figure 8:** Form 2 on XHTML MP Device.



**Figure 9:** Form 2 on WML Browsers from Openwave (UP.Browser 4 family).



**Figure 10:** Form 2 on non-Openwave browsers.

The most notable thing in **form2.jsp** is the `enable_wml="true"` attribute and value. This is necessary if you want your form not to throw an error like the one shown in the WML output for **form.jsp** above. Keep in mind that WML content must be well-formed and valid, or nothing will work. Only go for WML Compatibility Model for forms, if you know exactly what you are doing.

When using WML Compatibility Model, your forms are parsed and rebuilt from scratch for WML phones, so that WML variables can be used behind the scenes to post user data transparently for your server.

This is not the whole story though. WALL uses the power of WURFL to perform some extra optimization for WML directed at terminals with different capabilities.

Some devices are better served by a `<template>` tag which explicitly programmes their back key. For other devices this may be useless or even harmful. WALL will detect this and only program the back button for devices which benefit from this.

Another important optimization is the submit button. While XHTML is generally better served with a submit push button, WML devices vary a lot in this regard. Openwave UP.Browser 4.X has great support for softkeys (which make entering data quick and intuitive), while trying to program softkeys will make users' lives harder on other WML browsers. WALL fixes this transparently by programming softkeys for UP.browser 4 and providing a submit anchor to other devices.

This is achieved by looking at the WURFL `softkey_support` capability for WML devices.

All of this is visualized in the two WML versions produced by **form2.jsp** (listed above) and by comparing **Figure 9** and **10**.

While the WALL tags are powerful, you are more than likely to come across specific needs to multiserve content based on WURFL properties. WALL offers integration with the JSTL (Java Standard Tag-Library). The following example illustrates how powerful WALL can be, by implementing a cross-device wireless portal for a fictional operator.

**Note:** *The JavaServer Pages Standard Tag Library (JSTL) encapsulates as simple tags the core functionality common to many Web applications. JSTL has support for common, structural tasks such as iteration and conditionals, tags for manipulating XML documents, internationalization tags, and SQL tags. It also provides a framework for integrating existing custom tags with JSTL tags. For more info, check out the following documents:*

<http://java.sun.com/products/jsp/jstl/> and

<http://jakarta.apache.org/taglibs/doc/standard-doc/intro.html> .

## portal.jsp

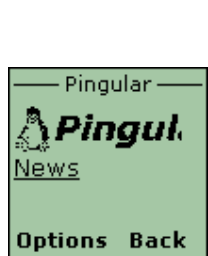
The `wall:load_capabilities` tag lets you use the JSTL to build expressive conditional JSP templates. JSTL and WALL tags can be intermixed

```
<%@ taglib uri="/WEB-INF/tld/wall.tld" prefix="wall" %><wall:document>
<wall:xmlpidtd />
<%@ taglib uri="/WEB-INF/tld/c.tld" prefix="c" %>
<wall:load_capabilities />
<wall:head>
  <wall:title>Entertainment</wall:title>
  <wall:menu_css />
</wall:head>

<wall:body>
  <wall:block>
  <c:choose>
    <c:when test="{capabilities.gif}">
      
    </c:when>
    <c:otherwise>
      
    </c:otherwise>
  </c:choose>
  </wall:block>
  <wall:menu colorize="true" autonumber="true">
  <c:if test="{capabilities.midp_10}">
    <wall:a href="http://url1" title="Games">Java Games</wall:a>
  </c:if>
  <c:if test="{capabilities.midp_10} && {capabilities.j2me_colors > 8}">
    <wall:a href="http://url1" title="Games">Cool Java Games</wall:a>
  </c:if>
  <c:if test="{capabilities.wap_push_support}">
    <wall:a href="http://url2" title="Sport">Real-Time Sport Updates</wall:a>
  </c:if>
  <c:if test="{capabilities.receiver}">
    <wall:a href="http://url2" title="MMS">Cool Pics</wall:a>
  </c:if>
  <c:if test="{capabilities.sender}">
    <wall:a href="http://url2" title="MMS">Send MMS to your friends</wall:a>
  </c:if>
    <wall:a href="http://url2" title="News">News</wall:a>
  </wall:menu>
</wall:body>
</wall:document>
```

In the example above, the portal will be rendered with a suitable mark-up for the requesting device. In addition to that, only the links to services that make sense for that device are rendered. In other words, owners of non MMS-capable devices are not presented with links to MMS services.

For this example, we will only show how the page is rendered on different actual devices by different vendors.



**Figure 11:** Nokia 7110 (WML 1.1, GSM EU)



**Figure 12:** Nokia 7210 (WML 1.2, GSM EU/US)



**Figure 13:** Siemens SL45 (WML 1.1 GSM EU)



**Figure 14:** Siemens SL45i (WML 1.1 GSM EU, software upgrade of SL45 which enables J2ME)



**Figure 15:** Sharp GX10 (XHTML MP, Vodafone)



**Figure 16:** Nec 341i (CHTML, IMode GSM EU)



**Figure 17:** SonyEricsson T616/T610 (XHTML MP, GSM EU/US)



**Figure 18:** LG LX-5350 (XHTML MP, CDMA US)

Some important observations:

- we multiserive two pictures: a WBMP and a GIF version of the pingular logo. There is no other optimization based on the graphical capability of the device, even though, the WURFL models those capabilities. In future versions, I would like to integrate ImageMagick and WURFL to multiserive images, but at this time, there is no plan for that. If you feel you can contribute, send me a note at luca dot passani at openwave dot com.
- The length of the sentences you can afford on each device also varies greatly from device to device. I am planning some support in the tag-library for that, but nothing I can announce yet.

While WALL will help you support a whole lot of devices, there may be devices which fail on your templates. One quick test is to point a web browser to your application. If there is an obvious error in your JSP or somewhere else on the server, you will see it in your web browser and you can debug there.

In other cases, the problem could be at the level of the WML, CHTML or XHTML MP produced by WALL. In those cases, you need a tool that lets you request a page using the User Agent string of the device which exhibits the problem.

## Debugging WALL applications

Such tools are easily available free of charge on the Internet. Some of these are:

- Curl: <http://curl.haxx.se/>
- Wget: <http://www.gnu.org/directory/wget.html> and (Windows) <http://www.interlog.com/~tcharron/wgetwin.html>

Personally, though, I prefer a Perl script which works nicely for me (Perl is available free of charge at <http://www.activestate.com> for Windows and other platforms). The script is just a few lines, even though I keep a few user-agent strings and URLs handy by commenting them out :

```

use LWP::UserAgent;
$ua = LWP::UserAgent->new;

$ua->agent("SIE-SLCK/3.1 UP/4.1.19i");
#wml11 agent
#$ua->agent("SIE-SL45/3.1 UP/4.1.19i UP.Browser/4.1.19i-XXXX");
#$ua->agent("Nokia7110/1.0 (04.78)");
#$ua->agent("Nokia7650/1.0 (03.09)");
#$ua->agent("Nokia7210/1.0(3.09)Profile/MIDP-1.0 Configuration/CLDC-1.0");

#wml13
#$ua->agent("CDM-8600/T10 UP.Browser/5.0.5 (GUI)");

#generic browser = chtml1 agent
#$ua->agent("portalmmm/2.0 N341i(c10;TB)");

#xhtmlmp_1 device (OPWV 6.1)
#$ua->agent("SIE-M55/10 UP.Browser/6.1.0.5.c.6 (GUI) MMP/1.0");
#$ua->agent("SonyEricssonT616/R-101 Profile/MIDP-1.0");
print $ua->agent."\n";

#urls
$req = HTTP::Request->new(GET => 'http://localhost:8080/wurfl/wall/body.jsp');
$req = HTTP::Request->new(GET => 'http://localhost:8080/wurfl/wall/menu.jsp');
$req = HTTP::Request->new(GET => 'http://localhost:8080/wurfl/wall/form.jsp');
$req = HTTP::Request->new(GET => 'http://localhost:8080/wurfl/wall/form2.jsp');
$req = HTTP::Request->new(GET => 'http://localhost:8080/wurfl/wall/portal.jsp');

$req->header('Accept' => 'text/html');
# send request
$res = $ua->request($req);
# check the outcome
if ($res->is_success) {
# print $res->content;
print $res->as_string;
} else {
print "Error: " . $res->status_line . "\n";
}

exit;

```

Here is the script in action (rp.pl, rp=retrieve page):

```

C:\WINNT\System32\cmd.exe
C:\projects>wurfl\XHTML-TagLib\test>perl rp.pl
SIE-SLCK/3.1 UP/4.1.19i
HTTP/1.1 200 OK
Connection: close
Date: Fri, 02 Apr 2004 12:53:21 GMT
Server: Apache Coyote/1.0
Content-Length: 492
Content-Type: text/vnd.wap.wml
Client-Date: Fri, 02 Apr 2004 12:53:21 GMT
Client-Peer: 127.0.0.1:8080
Client-Response-Num: 1
Set-Cookie: JSESSIONID=3DA1B5A6D67E559F9A8EAC1E09BBBBF99; Path=/wurfl

<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapf
/wml_1.1.xml">

<wml>
<head>
  <meta name="taglib" content="WALL" />
</head>
<card id="w" title="Pingular">
  <p>

  </p>
  <p align="left" mode="nowrap">
<select>

  <option onpick="http://url1" title="Games">Java Games</option>

  <option onpick="http://url2" title="News">News</option>
</select>
</p>
</card>
</wml>

```

**Figure 19:** Debugging a WALL-template.

**wget** (comes pre-installed on Linux RedHat) also performs similarly:

```

C:\>wget -O - -U "EricssonT610/R101" http://localhost:8080/wurfl/wall/portal.jsp

--12:36:08-- http://localhost:8080/wurfl/wall/portal.jsp => `-'
Resolving localhost... done.
Connecting to localhost[127.0.0.1]:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 774 [text/html]

 0% [                               ] 0          ---K/s
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN" "http://www.wapfor
um.org/DTD/xhtml-mobile1.0.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
  <title>Pingular</title>
</head>
<body>
  :

```

It goes without saying that if you come across a device which fails on the mark-up generated by WALL for it (assuming the device is correctly represented in the WURFL), I am very interested to hear about it. Please drop me a note at *luca dot passani at openwave dot com* with the following data:

- UA String of the device
- Info about the device (XHTML, WAP 1.X, etc...)
- mark-up generated by WALL
- what you expect or how the mark-up should be changed.

In addition, I am also interested about new abstractions. I can always implement them as long as they are general purpose enough and can be made degrade gracefully on legacy devices.

PS: WALL = WURFL-based Abstraction Library by Luca