Archived:Access the Mobile Web server from a Yahoo widget

Archived: This article is archived because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

The 'Mobile web server' was discontinued in February 2010. See here for more information.

Desktop widgets (also called gadgets) are small applications on the PC run by a hosting software the so called **widget engine**. The widgets usually show handy information (clock, todo, system indicators) or provide interface to popular and frequently used services (e.g.: weather, image/video services, news aggregators etc.) for the user. Most of them are free and downloadable from the Internet.

Here is list of several widget engines:

- Samurize
- Mozilla Prism
- Dashboard widgets of Apple Macintosh
- MS Windows Vista Sidebar and Gadgets
- Yahoo! Widgets
- Adobe Air

As this is a quite "hot" area so the landscape is changing quite dynamically. Here is a somewhat old comparison of the engines. Not fully independent but maybe this blog entry from Yahoo! Widgets blog might help you to get some overview.

Nokia Text Messenger

Nokia Text Messenger is a gadget for Windows Vista. The application fetches you the latest text messages from your connected Nokia device. Naturally, the latest messages are displayed first.

My Mobile Site widgets

This new application allows you to access your S60 phone with desktop widgets. It is a set of Yahoo! widgets accessing the phone via Mobile Web Server.

Known Issues in 1.0

- If you initiate a call with the My Mobile Site Call widget you can't hang up the call via the widget. The main problem here is that if the dataconnection is 2G then it gets suspended during the call and the widget can't send the command to the phone. The feature would be possible with 3G connection but currently the application does not check the type of the connection.
- Only the phone number is shown in the Messaging Sent items list for the SMSs sent by the SMS widget

How to develop your own widgets

This is a small example explaining how to implement a desktop widget that uses Mobile Web Server on the phone as data source. The widget is implemented using Yahoo! Widgets desktop widget engine and the implementation on the phone is in Python. The example is a small battery widget called “batterymini” which shows the battery level of the phone. The widget periodically queries the phone for the actual value, but there is also a right click menu to manually refresh the widget. The implementation shows also how to implement a dynamic dock item for the widget. The widget accesses the phone via an http connection which is handled by the Mobile Web Server and a small Python script creates the response.

You can download the full source code from here

In the description I assume that you have already installed Mobile Web Server and Yahoo! widgets.

Device implementation

Create a folder called “batterymini” in the web servers root folder. It is either on the Phone memory or the Memory card wherever the user installed it. The path to the folder is **X:\DATA\Web server\htdocs**. Place two files to this folder: **ht.acl** and **batterymini.py**.

ht.acl:

In this file you can set the access rights and define the Python handler for this folder. See details in the following documentations: How to Develop Content (Chapter 3.3 Authentication and Chapter 4. Dynamic Pages with Python). In the example we skip the authentication, so the phone provides the information for everyone. We only define the Python handler:

(C) Copyright Nokia 2013. All rights reserved.

```
AddHandler mod_python .py
PythonHandler batterymini
```

These two lines define that all .py calls in this folder will be handled by the batterymini file.

batterymini.py

The file contains the concrete request handler which calls the appropriate function to get the battery level.

```
def handler(req):
   try:
      response = ('%s' % getBattery())
      …
```

getBattery() function does the actual work. It asks the phone about the level of the battery:

```
from sysinfo import battery
b = battery()
```

Converts the result depending on the S60 version:

```
from e32 import s60_version_info
if s60_version_info < (3, 0):
   b = int(b / 7.0 * 100.0)
```

Earlier S60 versions gave the battery level in a simple 0-7 scale value where the newer versions give percentage. The code simply converts the earlier format to percentage.

Widget part

To start learn about Yahoo! widget implementation please go to the Reference manual. The files should be always in a folder called "Contents" which is the subfolder of the folder "<widget name>.widget". In the example there are three files and a folder for the images:

- **widget.xml**: provides basic information about the widget.
- **dockitem.xml**: defines the small image of the widget in the Yahoo! widgets dock. The widget alters the docitem image dynamically by changing this file.
- **Resources** folder contains the images.
- **Batterymini.kon** file contains the main implementation. See detailed explanation below.

Batterymini.kon

In this file we describe the widget in **XML**. There is also some **Javascript** code which implements the dynamic behavior of the UI and the communication with the phone. (For more complicated widgets it is better to put the Javascript code into separate .js files and leave only function calls in the .kon file.)

XML

The widget has one window describing the UI and the context menu:

```
<window title="Batterymini">
   <name>mainWindow</name>
   <contextMenuItems>
      <menuItem title="Refresh" onSelect="getBattery();"/>
   </contextMenuItems>
   ...
   <image src="Resources/bar_0.png">
      <name>bars</name>
   </image>
</window>
```

The last image with the name *"bars"* shows the actual battery level. The Javascript code changes this image dynamically.

The timer solves the periodic update:

```
<timer>
   <name>timer</name>
   <interval>10</interval>
   <ticking>true</ticking>
   <onTimerFired>
      print("timer fired");
      getBattery();
   </onTimerFired>
</timer>
```

As `<ticking>` is set to true it will start immediately calling the update. You can see there a `print()` command in the `<onTimerFired>` tag. This function puts its string argument to the trace window when the widgets runs in debug mode.

There are two event handlers (triggers) in the widget: {{{1}}} Creates the dock view of the widget dynamically when the dock item of the widget gets visible in the Yahoo! Widgets Dock.

```
<action trigger="onDockOpened">
   updateDock();
```


(C) Copyright Nokia 2013. All rights reserved.

```
    </action>
```

{{{1}}} This event is fired when the widget is loaded. The event handler defines the necessary Javascript variables and functions and initiates the widget. Here is the main code of the widget. See detailed description below.

Javascript

Variables

- `url`: An object of the `URL` class in the Yahoo Widgets Javascript library. Basically it implements similar functionality as the **XMLHttpRequest** class in AJAX.
- `barCount`: The actual battery level visible on the UI. There are eleven images in the Resource folder for the different battery levels. The value of this variable refers to the number in the image file name. e.g.: The value 2 refers to **bar_2.png**.
- `dockItemXML`: The DOM tree of the **dockitem.xml** file
- `dockItemBars`: The DOM object of the batterylevel image in the dock view of the widget.

Functions

`getBattery()`

The check at the beginning of the function assures that the widget does not call the server again when a request is ongoing.

```
    if (url != null)
       return;
```

(The variable `url` is set to null at the end of the callback function.)

Creates a new `URL` object, because one `URL` object can handle only one request.

```
    url = new URL();
```

Sends an HTTP request to the Mobile Web Server via the URL object.

```
  url.location = "http://<youraccount>.mymobilesite.net/Batterymini/.py";
  url.fetchAsync(getBatteryComplete);
```

It is an asynchron call so it sets also the callback function: `getBatteryComplete()`. The URL object will call this function when the response arrives.
IMPORTANT: Before running the widget replace `<youraccount>` in the code with the appropriate name on your phone.

`getBatteryComplete()`

Gets called when the response arrives. It checks the Http response. If not `200 OK` then it brings up an error message:

```
    if (url.response == 200)
    {
       ...
    }
    else
    {
       alert("Error during communication. Result code: " + url.result);
    }
    url = null;
```

Otherwise it converts the received string value to a number and calls `updateBattery()` function to update the UI:

```
    value = Number(url.result);
    print(value);
    updateBattery(value);
```

`updateBattery()`

Calculates which battery level to show and sets the right image in the widget window. Calls `updateDock()` which sets the dock item image.

```
    barCount = Math.floor((batteryvalue*11)/100);
    bars.src = "Resources/bar_"+barCount+".png";
    updateDock();
```

`updateDock()`

Changes dockitem to show the image of the actual battery level:

```
    if(!widget.dockOpen) return;
    dockItemBars.setAttribute("src","Resources/bar_"+barCount+".png");
    widget.setDockItem( dockItemXML );
```

Widget initialization:

Parses **dockitem.xml** and selects the tag which is the placeholder for the image of the battery level just like in the widget window.

(C) Copyright Nokia 2013. All rights reserved.

```
dockItemXML = XMLDOM.parse( filesystem.readFile( "dockitem.xml" ) );
dockItemBars = dockItemXML.getElementById( "bars" );
widget.setDockItem(dockItemXML);
```

Finally it calls the Mobile Web Server to get the battery level and starts the sequence described before which updates the UI:

```
getBattery();
```

To run the widget start Mobile Web Server on your phone and Yahoo! Widgets on the PC and simply double click **Batterymini.kon**. Don't forget to change the url in line 42! (See the description of the `getBattry()` function.) Or you can create a widget file by using the converter widget or call the **Converter.exe** from command line (see Yahoo! help for details):

```
Converter.exe -flat -o c:\ Batterymini.widget
```

Now you can double click the **Batterymini.widget** file to run the widget.

mszomol 12:18, 24 January 2008 (EET)

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.