Homescreen widget guidelines

This article provides guidelines for Symbian homescreen widgets. First, you will learn how to enable a widget for the homescreen. The rest of the article is about actual design and development guidelines for homescreen widgets.

20 Dec 2009

Before you start

To develop a home-screen widget for a mobile device, you need the following:

- Computer workstation — a PC with Microsoft Windows operating system, Linux, or an Apple Macintosh
- Text editor
- Zip-utility software capable of creating .zip files

For additional information, check Before you start (WRT widget development)

Enabling a WRT widget for the homescreen

This section describes how to enable the homescreen view for a WRT widget. It is a fairly simple operation, but our emphasis will be on slightly more advanced guidelines for a properly working homescreen widget.

Full-screen view and homescreen view of AccuWidget

To enable the mini mode for the WRT widget:

1. Read Create your first WRT widget and follow the guide to create a simple widget (or continue from your existing code if you have already completed this step).

2. In the info.plist file, make sure key "MiniViewEnabled" is set to true:

```
        <key>MiniViewEnabled</key>
        <true />  <!-- Enable home screen view -->
```

In case this key in not enabled (set to false), the widget cannot be added to homescreen UI.

3. Add a homescreen view area as a div element to your index.html (or whatever your main HTML file is called). Add CSS and JavaScript code to activate the view when needed (For more details, see following section "Activating the homescreen view and defining its layout".

4. Continue from the step "Create the widget package" of the article and make a package of your homescreen-enabled widget.

Running the widget on a mobile device

For instructions on installing the homescreen-enabled widget onto a mobile device, see section "Running the widget on a mobile device" in Create your first WRT widget. Notice that the home screen view is available only on certain Nokia devices, such as Nokia N97.

Using the homescreen

1. Add the widget to the homescreen by selecting Options > Add content > HelloWRT.

2. Answer "Yes" to the question: Allow blanket permission for all platform services? If this is the first network accessible WRT widget in homescreen, there

(C) Copyright Nokia 2013. All rights reserved.

is one more dialog, which appears asking "Allow Connection to network to enable updates to homescreen content. Updates can be disabled in settings". After answering to these questions, the widget appears on the homescreen.

Also note that for most homescreen-enabled widgets that are updating the content via network, the key "AllowNetworkAccess" must be set to true, otherwise the widget won't be able to connect and update the content.

3. Tap on the widget on the homescreen to activate the full screen view. Interacting with the widget is only possible in the full screen view.

> **Note:** Note that when the widget is added on the homescreen, it is running all the time. The widget is not closed until the user removes it from the homescreen.

Running the widget on the emulator

For instructions on running the widget on the emulator, see section "Running the widget on the emulator" in Create your first WRT widget. In order to develop a homescreen widget that displays gracefully on homescreen, an N97 SDK with emulator may be necessary to verify the widget behavior. Other option is to use the Nokia Web Tools, which offers preview support.

You may also want to consider Nokia Developer Remote Device Access services to see how your widget looks on the homescreen.

Homescreen widget guidelines

Activating the homescreen view and defining its layout

This section describes how to switch between the homescreen view and the full screen view of the widget.

If the widget is homescreen enabled, meaning that the MiniViewEnabled setting is true in the info.plist file, the widget needs to have two separate views defined in HTML page. In practice, this means a new div element for homescreen view,, which is hidden when the full screen mode is on. This is also true in reverse: when the homescreen view is visible on the home screen, the full screen view needs to be hidden.

```
<body>

<div id="full_view">
...
</div>

<div id="mini_view" class="view hidden">
...
</div>

</body>
```

Adding a new div element to the HTML is not enough, however. You also need to implement some JavaScript™ code to make the view changes when needed. The system fires an event when the user interacts with the widget by changing to full-screen or homescreen view. When the screen size changes, the onresize() event is fired, and you can then decide the view you want to show, depending on the current screen size. In practice, there are two options. One is to switch the display attribute of div element to none to hide the element and to block to show the element. Another way is to write a value ‘view hidded’ for the ‘class’-attribute of div-element, like this:

```
<div id="homeScreenView" class="view hidden">
```

This may be done with following JavaScript™ code:

```
document.getElementById("homeScreenView").setAttribute("class", "view hidden");
```

And to show a view in HTML happens by writing value ‘view’ for the ‘class’-attribute, like this:

```
<div id="homeScreenView" class="view">
```

This may be done with following JavaScript™ code:

```
document.getElementById("homeScreenView").setAttribute("class", "view");
```

Here are the things needed to code using JavaScript™, when windowResized-event is fired:

```
// Assign event handler to the onresize event
window.onresize = windowResized;

// Called when the window is resized
function windowResized() {
  // Add code to do following:
  //  1) detect screen size
  //  2) switch to correct view and
  //  3) set active style sheet
}
```

A full example of how to detect an onResize()-event can be seen at Detecting orientation changes in Symbian Web Runtime. Reacting to the changes in screen size in Symbian Web Runtime explains how to react to the changed screen size. You may also see Web Developer's LIbrary for a real world example. In these examples, pay attention to the homescreen size and the style sheet enabling for the homescreen.

(C) Copyright Nokia 2013. All rights reserved.

Note that some devices do not support the onresize function. As a solution for this known issue, a timer can be started to poll for changes in the widget size.

```
 var timer = null;
 window.onload = init;

// Does initialisation tasks for the widget
function init(){
   // Call the tick function at one second intervals
   timer = setInterval("tick()", 1000);
}

// Called when the timer's interval elapses
function tick() {
   // Call the windowResized function to achieve the same effect as if it were
   // triggered by WRT
   windowResized();
}
```

Tip: It is recommended that you define the layout for the homescreen view and full screen view in a separate CSS file. For example, it is a good idea to choose a smaller font size for text shown in the homescreen view.

The following example shows what the CSS for the homescreen view could look like:

```
#mini_view {
  width: 308px;
  height: 85px;
  background: white;
  overflow: hidden;
}

#mini_view th {
  color: #fff;
  font-size: 12px;
  font-weight: bold;
  background: #7fcce5;
  padding: 4px 6px;
  text-align: left;
}
...
```

Homescreen content

The homescreen view shows a subset of the widget's most important information. When designing content for the homescreen, you should limit the amount of data displayed. Placing too much content in the homescreen view, results in a less-than-optimal widget. Avoid content-free decoration — the best graphics are useful and important. Users should be conscious of the substance of the widget, rather than its graphic design. Also display information of the widget, which is easy to understand.

Avoid using complicated images and 3D graphics in homescreen views. Smaller images in terms of both pixel size and color depth conserve memory and load faster. A color depth of 16 bits offers a good compromise between looks and size.

Animation is not recommended because it consumes a lot of battery. The following section "Updating the homescreen content" offers tips on how to update homescreen content and make it more appealing for device users. If you do create a homescreen view with updateable content, use words, numbers, and images together to illustrate the widget's most important data. The AccuWidget below displays the most important data (name and Celsius-degree information for the city), together with a supportive image showing the weather type, in a small space.

Homescreen view with a small image and supportive text and data

Note also that visualization and animation rules for widgets in general can be found at Designing Widgets.

Updating the homescreen content

The homescreen does not offer any interactivity with the user. The only interaction with a homescreen portion of a widget is launching the full view of the widget. However, dynamic content can be introduced to the homescreen view by updating the view, for example by timer. Basically, the homescreen view should show a subset of the widget's most important information. This might be, for example, the local forecast from a weather widget, or upcoming appointments from a calendar widget. It is recommended to remove any continuously updating media, such as video and flash animation.

What you can do is implement a timer to update important data on the homescreen. A timer event can, for example, rotate news headlines or change

(C) Copyright Nokia 2013. All rights reserved.

weather forecast data displayed in the homescreen. Keep in mind, however, that timers are not fired consistently because timers are suspended when widget is not shown or screen saver / key lock is activated Archived:JavaScript timers not triggered with widgets running in the background in BrowserNG/7.1 (Known Issue). It is not recommended to use update intervals longer than the screen saver / key lock time-out value (defaults to 1 minute) as timer only advance when the widget is shown. When fetching the data, that is supposed to be shown on the homescreen, from the server it is advised to determine the need for update based on timestamp of the last retrieved data. You could for example inspect the timestamp when homescreen view is shown and retrieve the data only if needed.

However, network connection may not always be available, in which case you should display an offline-icon on the homescreen and keep rotating old content, if available. See section Informing the widget of online/offline status to better handle situations, when a network connection is not available.

Considering performance and battery consumption

When a user adds a widget to the homescreen, it launches immediately and continues to run until it is removed from the homescreen. Even if the widget is in mini mode, it might reserve the memory and consume the battery as much as if it was running in full screen mode. That is why battery lifetime optimization is apparent when creating homescreen enabled widgets. Below are some suggestions that will help balance power usage.

Loading graphics

Usually a WRT widget loads all graphics at the same time that it is loading the HTML code. However, if the widget uses noticeably large images, they could be lazy-constructed separately when switched to full screen view. Lazy construction means that the graphics are loaded just before they are needed. When returning to the homescreen view, the graphics should be released from memory.

In practice this means:

1) Switching to a full-screen view would trigger the windowResized event as before, leading to screen size detection and changing the view and active style sheet. In addition, the HTML code containing the large bits of graphics should be created dynamically using JavaScript™ code. Code should add, for example, a new img tag to the correct place in the DOM tree.

```
var parentNode = document.getElementById('parentid');
var newimg=document.createElement('img');
newimg.src="your.img.src";
parentNode.appendChild(newimg)
```

2) Switching to the home-screen view would trigger the windowResized event as before, leading to screen size detection and changing the view and active style sheet. In addition, the objects in the DOM tree containing the large images should be removed by using the delete operator in JavaScript™ code. Calling the delete releases the memory reserved when the graphics were loaded in the full-screen view. You can also simply remove the Child from the DOM tree.

```
list_row = document.getElementById('record' + id);
list_row.parentNode.removeChild(list_row); // or: delete list_row;
```

Note that this pattern should not be used automatically in order to save battery. The JavaScript garbage collection in the WRT environment may not always work efficiently. This can lead to even worse scenarios, when you merely try to save some memory. New image nodes are dynamically created every time the full screen is enabled, but the old nodes are not deleted from memory (only the references to them). In addition, if the DOM tree is large, searching of the dynamic parts might get slow. See High performance Widgets: Optimize your JavaScript.

Using S60 platform services

In previous chapter the loading of graphics was done with lazy construction when full screen view is shown. The same pattern should be used with platform services, which consume a lot of battery. These may be for example Location Service API, Media Management Service API of Messaging Service API. See full list of Platform Services.

Make JavaScript™ code and CSS external

For better performance, create external files out of style sheets and JavaScript™ code. Using external files generally produces faster pages because these files are cached by the browser. Inline JavaScript™ code and CSS get downloaded every time the HTML document is requested. However, modularization should not lead to a large number of different CSS and JavaScript™ files (for example, by every view mode or screen), because linking to them causes extra HTTP requests to render the page. For more details, see High performance Widgets: Combine your JavaScripts and CSS in external Files.

Using timers

Using timers is one way to update the homescreen content. However it should be kept in mind that using timers is easily very power-consuming, especially with short timer values. As a general tip to save battery power, do not use timers with minimum intervals for long periods of time. Start with longer times (for example several minutes), and decrease the time only if it is necessary for your application. Also keep in mind, that it is a bad idea to let any other timer continue running while in the homescreen. Turn off other timers when switching to the homescreen view, and turn them on again when activating the full screen. Long-running frequent timers also may prevent the processor from entering the low-power sleep mode and then increase power consumption in the long run.

Instead of timers, you might want to consider using some other programming paradigm, like asynchronous notification-based services.

Using device back-light and screen saver

The back-light of the mobile device screen consumes energy. The back-light and screen saver settings work automatically according to user-set preferences. Do not override the default settings unless absolutely necessary.

It would also be worth considering to try releasing any unneeded resources, when entering idle mode (meaning screen-saver is activated). For example network connection should be closed when idle mode initiates. For example the onBlur-event of JavaScript to catch the lost focus from an object might be useful.

Creating network connections

Implement a reasonable schedule for retrieving data via network connections. For example, retrieve news data once every hour. Optimize the amount of data transferred.

(C) Copyright Nokia 2013. All rights reserved.

Informing the widget of online/offline status

The homescreen widget usually needs to access the network to show relevant dynamic data in the homescreen. Sometimes, however, the user may have denied network connection of homescreen widgets or the device is set to offline mode. Online/offline API is needed to detect such a situation and widgets should implement it to ensure the best possible user experience.

Currently this feature is supported in all recent devices supporting homescreen widgets and Nokia N97 software releases 11.0.021 and Nokia N97 Mini software release 10.0.020 onwards.

Handling online/offline in WRTwidget

Useful links

- Developing homescreen widgets
- JavaScript Performance Best Practices
- AccuWidget
- RouteWidget
- All Nokia's Web tools

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.