Archived:How to package Flash content in a 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 article is believed to be still valid for the original topic scope.

This article shows how to package Flash content in a Symbian Web Runtime widget. This provides the convenience of an icon for launching the content from the device UI, and is much simpler than packaging the application as a Symbian SIS file.

Introduction

27 Jul 2008

One of the biggest issues in delivering Flash Lite applications is that the user needs to open the Flash Lite Player or use the File manager or Gallery to open the file. The Flash Lite content is not treated as other applications. It does not have an icon and the application's shortcut cannot be added to any quick access menus.

Another disadvantage is that any external resources need to be delivered separately, which is tedious for the user. In S60 this can be tackled by creating a SIS file that simply extracts the contents to the correct folder, however, the .sis file requires must be signed and the target folder may vary between S60 platform versions.

Why not just use a .sis then?

There are commercial and freeware solutions and guides on how to do this in Symbian C++, but systems run only on a Windows PC and you need to know at least the basics of C++ coding. Many Flash developers work with Apple Macintosh computers, and the S60 SDK and tools are not compatible with Macintosh.

Widget to the rescue!

With the S60 Web Runtime developing small applications is fairly simple and you do not need to learn Symbian C++. All you need is basic knowledge of HTML, CSS, and JavaScript. The contents are zipped and the package is renamed with the.wgz extension. This package does not need to be signed and the devices supporting the format recognize it as an installable application. The package contents are copied to a private folder and an icon for starting the widget is placed in the applications folder.

Since Web Runtime is based on the S60 browser, everything supported by the browser is also supported by WRT - also Flash Lite. This means that a Flash Lite application can be packaged inside a Widget.

O.K. already! Show me how.

Option 1. Embedded Flash

Get the example widget

Before getting started, familiarize yourself with Web Runtime. The best place to start is the Widget website. You should be able to create a working package by following these instructions, but if you want to know more about widgets, check out the Web site.

- Download the Media:MyFlashWidget.zip

- Rename the file to FlashWidget.wgz.
  - The .wgz file is just a renamed .zip file.
- Extract the contents to a folder.
  - Maintain the folder structure
  - The folder can be in any folder.

Modify the icon

You can also use the icon I created. It is only for demo purposes, however, and must not be used for commercially distributed

(C) Copyright Nokia 2014. All rights reserved.

Flash widgets. The icon must be in PNG format. SVG icons are currently not supported.

Modify the info.plist

Open the `info.plist` file and edit the key values marked with green in the following example. The `DisplayName` is the title of the application and the `AllowNetworkAccess` key defines if the widget can access the Internet. If your application does not need network access, you can set it to "false" and the user will not be asked for permission when launching the Widget. More information about the info.plist file can be found from the Widget Web site

```
<key>DisplayName</key>
<string>MyFlashWidget</string>
<key>Identifier</key>
<string>textcom.forum.widgets.MyFlashWidget</string>
<key>Version</key>
<string>1.0</string>
<key>MainHTML</key>
<string>main.html</string>
<key>AllowNetworkAccess</key>
<true/>
```

The HTML and JavaScript

Open the file `Main.html` file in a text or html editor. You may want to modify the following three values:

CSS file reference

Using an external CSS file is recommended. In this case the file is insignificant, since we are creating full-screen Flash content. The main purpose of the CSS file is, in this case, to color the background black and make sure the full-screen Flash Lite content is positioned correctly. For more information on how to work with HTML and CSS in the widget scope, see Widget Web site.

```
body {
    background: rgb(0,0,0);
    position: absolute;
    left: 0px;
    top: 0px;
    width: 100%;
    height: 100%;
    margin: 0px;
    overflow: hidden;
}
```

JavaScript

According to the Widget documentation, the preferred coding convention is to use separate .js files. In this project JavaScript is used only to set the navigation mode. If you want to use more widget functionalities, see the Widget website.

Setting tab navigation mode

```
<SCRIPT LANGUAGE="JAVASCRIPT" TYPE="TEXT/JAVASCRIPT">
<!--
 widget.setNavigationEnabled(false);
//-->
</SCRIPT>
```


(C) Copyright Nokia 2014. All rights reserved.

The above piece of code embeds a single JavaScript command to the Main.html page: `widget.setNavigationEnabled(false);` This tells the Web Runtime environment to use tab-based navigation. If you want to use pointer navigation instead, change the value to ´´true´´, as in the following example:

Setting pointer navigation mode

```
<SCRIPT LANGUAGE="JAVASCRIPT" TYPE="TEXT/JAVASCRIPT">
<!--
 widget.setNavigationEnabled(true);
//-->
</SCRIPT>
```

The example file has only an animation, but tests have proven that both tab and pointer navigation work in the Flash Lite application. Feel free to experiment to find out what works best for you.

Embedding the .swf

The following piece of code is basically the code generated by Adobe Flash Professional. That is why the embedding is done both ways. If you use the `main.html` file provided in the example, you need to make sure you change the values to match your .swf. The values concerned are mainly the `src` and `movie` parameters and the `allowScriptAccess` directive.

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" width="238" height="318"
id="MyFlash" align="middle">
  <param name="allowScriptAccess" value="sameDomain" />
  <param name="movie" value="MyFlash.swf" />
  <param name="loop" value="false" />
  <param name="menu" value="false" />
  <param name="quality" value="high" />
  <param name="wmode" value="opaque" />
  <param name="bgcolor" value="#ffffff" />
  <embed src="MyFlash.swf" loop="false" menu="false" quality="high" wmode="opaque"
bgcolor="#ffffff" width="238" height="318" name="Finish" align="middle"
allowScriptAccess="sameDomain" type="application/x-shockwave-flash"
pluginspage="http://get.adobe.com/flashplayer/otherversions/" />
 </object>
```

Flash Lite application

If you do not have a sample Flash Lite application you can use for testing, now is a good time to create one, or edit one you have created earlier. I have the stage one pixel smaller on all sides to avoid anti-aliasing. In this case the values are set to 238 x 318 pixels.

Tip: *Since the new devices supporting with Web Runtime also support Flash Lite 3, you can use that too to create the application!*

Considering different screen sizes

Since you may not know the screen sizes of the devices beforehand, you can use some Javascript in your WRT widget to detect the screen size and use that information to optimize the SWF for the available real estate through variables, or by using multiple SWFs. You can also deal with this issue in the .swf file, as explained in the article Dynamic Layout control for Flash Lite.

Option 2. Direct play

Instead of embedding the .swf file into an html page, you can replace the main.html file with your .swf file. This means that you need only three files:

- A manifest file (info.plist) with the .swf file as the value for the MainHTML key.

(C) Copyright Nokia 2014. All rights reserved.

```
<key>MainHTML</key>
 <string>myFlash.swf</string>
```

- The .swf file you named in the info.plist file.
- The **`icon.png`** file (optional).

The .swf is launched in the Flash Lite player and when you exit it, the application asks if you want to save it. It falls also then back to an "empty widget" screen (plain white), but this is an easy way to package your .swf and get an icon for it if you don't mind the blank screen you get after you quit or exit the swf.

NOTE: If you do not want to end with a thank you note but exit directly instead, you need to edit the Flash file and disable the "Exit" command, which is by default in the right softkey. See the following example:

```
fscommand2("FullScreen", true); //To hide the softkey labels
fscommand2("SetSoftKeys", "right dummy", "left dummy"); //Replace the dummy values, if
you like to use Flash Player's softkey labels instead of creating your own.
```

If you do this, inform the user that the only way to exit your application is to press the end key. This way there will be no blank screen since the player was started on the WRT engine and therefore the whole package dies.

Option 3. Meta-refresh method and thank you page

The third option is to use the meta-refresh tag in the <head> section of the main .html file. You still get a blank page but this time you can edit the contents. You can, for example, use the page to direct the user to your home page and give some extra information, or you can simply say "cheers". In the following example, the 0 before the URL means that there is no timeout, which makes the functionality more stable in some devices.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
 <title>SWF Launcher</title>
 <META http-equiv="REFRESH" content="0; url=MyFlash.swf">
 <script type='text/JavaScript'>
  function killWRT(){
   var WRT = window.open("", "_top");
   WRT.close();
  }
 </script>
</head>
 <body>
  <div align="center">
   <br><br>
   You can find more information at <a href="http://www.developer.nokia.com/">Nokia
Developer</a>.
   <br>
   <input type="button" value="Close this window" onclick="killWRT()" />
   <br><br>
   <b>Thank you for choosing our Flash Lite application.
   <br><br>
   - the team -</b>
  </div>
 </body>
</html>
```


(C) Copyright Nokia 2014. All rights reserved.

NOTE: You can also disable the softkeys and close the application with the end key, as shown in Option 2. Direct play.

Creating the widget package

When you have made all the necessary modifications or created all the needed files, you are ready to test your widget. As mentioned earlier, the wgz file is a renamed ZIP package, so all you need to do is to use your favorite method of zipping a file and rename it. You must include **the folder** where your .html, .css, .swf, etc. files are in the zip package.

Testing the widget

The emulator in the Widget SDK does not support Flash Lite. You can still test the validity of the widget package with the emulator, but you will not see the Flash content. If you do not have a device that supports the S60 Web Runtime, you must use the Remote Device Access service.

The installation method is always the same. Use Bluetooth, memory card, or any other method to transfer the .wgz package to your device and run it. The device or emulator will start the installation process. When the installation has finished, you will find an icon for your Flash widget in the Applications folder.

Getting help

If you find something wrong in this article, post your issue on the *Comment* and I will try to address it in a timely manner. If you need help in developing the widget part of your package, post your question to theWeb Runtime discussion board. If you need Nokia-specific Flash Lite help, you can use the Flash Lite discussion board or in more general issues, use Adobe's service.

Have fun ;o)

--Risalmin 16:15, 5 December 2007 (EET)

(C) Copyright Nokia 2014. All rights reserved.