Adobe Flash Lite Social Networking Application Tutorial

Version 1.0; 25 March 2010

Flash

NOKIA

Contents

Change history

25 March 2010	Version 1.0	Initial document release

1 Introduction

This tutorial, which was made possible by the Open Screen Project Fund, is one in a series of seven. Each tutorial in this series is designed around a user experience where Adobe Flash Lite and Nokia devices are a great fit: rich media, personalised experiences, social media, photo sharing, location based information, music, news, video, advertising, and marketing.

The focus of this tutorial is on using the social networking as a mechanism to support an advertising or marketing campaign for a brand or product. This application is designed to be shown as-is or modified easily. By modifying the application it can be used to quickly present a concept within your company or to pitch to outside clients.

An additional feature of the demonstration application is that it can be reskinned easily: The UI elements and the background can be changed in a matter of minutes.

Please check out the other tutorials in this series, as the concepts taught can be used together in a single application. The other tutorials are:

- Event — Displays information about an event and enables the user to send an SMS message containing text event information or call an event telephone number.

- Video Player — Plays an external `*.flv` video file in both landscape and portrait modes.

- Music Player — Plays external `*.mp3` audio files.

- RSS News — Reads and displays RSS feeds.

- Photostream — Displays Flickr photos.

- LBS Coffee Finder — Leverages GPS location and Google Maps to find the nearest coffee shop.

1.1 Prerequisites

These tutorials have been designed for developers with intermediate Flash skills or above.

1.2 Requirements

To modify the applications Adobe Flash CS 3 or CS4 is required. A [S60 5th Edition device by Nokia](#) is recommended to show the demonstrations. Use of a device is more effective than presenting on a computer.

If you don't have a device, you can run the application using the Forum Nokia [Remote Device Access or Virtual Developer Lab](#) services. These services provide access to a range of Nokia devices over the internet.

You can use Adobe Device Central in Flash CS4 also.

1.3   Flash Lite skills taught

This tutorial is designed to teach you the skill to use the following capabilities of Flash Lite on Nokia devices:

- Using a user's Twitter content.

- Timing events, in this case transitions in a slideshow.

- Parsing XML files.

- Scrolling or paging a content window.

1.4   Running the application

The following methods can be used to run the demonstration applications on a S60 5th Edition device:

- Use a Bluetooth connection to send the `*.swf` files to your device. The Flash Lite application can then be run by opening it from the device's messaging application.

- Create a new folder called *trusted* in the *other* folder on any drive of your S60 device. Copy the `*.swf` files to the *trusted* folder. Open the folder in the device's file manager application and tap any `*.swf` file to run it.

- Install the widget (`*.wgz` file) onto your device and then locate the widget in the device's applications folder. Tap the application icon to run the application.

2   About the author

Omega Mobile is a technology-driven design studio that obsesses about mobile and devices. The company is passionate about clean design, simple user interfaces, and targeted experiences. Its specialty is mobile design, user interfaces, user experiences, prototypes, and Flash Lite. Omega Mobile has been producing mobile Flash applications since 1999. For help on a mobile project or for more information, please visit http://www.omegamobile.com or contact Omega Mobile at +1-415-596-6342 or contact2010@omegamobile.com.

3   Open Screen Project Fund

Announced by Nokia and Adobe in 2009, the Open Screen Project Fund awards grants to help developers create exciting new applications and content. Grants are made from a fund of US$10 million and the fund will remain available until December 2010. The fund also provides marketing and educational support for the Open Screen Project, which aims to establish cross-platform runtimes, remove development and distribution barriers, and innovate through industry collaboration. To learn more about the Fund, visit Get Started page of the Fund's website.

4   Application description

The social networking demonstration application is designed to teach you how to make use of the Twitter API in a Flash Lite application. In addition, the application includes code that illustrates how to time events, in this case the transition of a slideshow, and implement a scrolling content window.

4.1 Application screens

The application provides the following screens:

Home	Tweets	Slideshow	About

4.1.1 Home screen

This screen provides a panel containing an image and text appropriate to the tweets and enables the user to access the feature of the application through four buttons:

- Tweets — Opens the tweets screen.

- Slideshow — Opens the slideshow screen.

- About — Opens the about screen.

- Quit icon — Exits the application.

4.1.2 Tweets screen

This screen displays the encoded user's tweets. The list starts with the most recent tweets and a simple scrollbar is provided to navigate the loaded tweets. The screen offers a back icon to return to the home screen.

4.1.3 Slideshow screen

This screen displays the encoded user's tweets individually in an automatic slideshow. The screen offers a back icon to return to the home screen.

4.1.4 About screen

The about screen is designed provide a space to describe the application and the company that created it. It is recommended that information about your company, including your contact details, is used when distributing this application to potential clients. The screen contains a back button, which takes the user back to the home screen.

4.2　Code

The bulk of the code (see Example 1) dealing with the tweets is used by the tweets screen. The code is found in the DATA and TWEETS block.

```
/*
Gets the feed of the given user and parses the data into an array.
*/
function getTwitterInfo ()
{
    tweet_xml = new XML();
    tweet_xml.ignoreWhite = true;

    tweet_xml.onLoad = function(success)
    {
        if (success)
        {
            tweets = new Array();
            var rootNodeTweet:XMLNode = tweet_xml.firstChild;
            for(var i = 0; i <rootNodeTweet.childNodes.length; i++ )
            {
                var tweetDate =
rootNodeTweet.childNodes[i].childNodes[0].firstChild;
                var tweetText =
rootNodeTweet.childNodes[i].childNodes[2].firstChild;
                var textToUse = cleanHtmlText(tweetText.toString());
                tweets.push([tweetDate, textToUse]);
            }
            showTweetsInit();
        }
        else
        {
            trace("*ERROR: Unable to load XML!!*");
        }
    }
    tweet_xml.load("http://twitter.com/statuses/user_timeline.xml?screen_
name=" +TWITTER_USER_NAME + "&count=" + NUM_TWEETS_TO_LOAD);
}


/*
Cleans the text.
*/
function cleanHtmlText (textToUse)
{
    while(textToUse.indexOf("&amp;") != -1)
    {
        var locOfString = textToUse.indexOf("&amp;");
        var firstPart = textToUse.substring(0,locOfString - 1 );
        var secondPart = textToUse.substring(locOfString+5);
        textToUse = firstPart + "&" + secondPart;
    }
    htmlTextBox.htmlText = true;
    htmlTextBox.htmlText = textToUse;
    var textToReturn = htmlTextBox.text;

    return textToReturn;
}


/*
Loads the xml of the user information. when it is done, advances to the
home screen
*/
function getUserXML()
{
    user_xml = new XML();
    user_xml.ignoreWhite = true;
```

```
        user_xml.onLoad = function(success)
        {
            if(success){
                if(user_xml.firstChild != null)
                {
                    userInfo = new Array();
                    var rootNodeUser:XMLNode = user_xml.firstChild;
                    for(var i = 0; i< rootNodeUser.childNodes.length; i++)
                    {
                        var currentValue = rootNodeUser.childNodes[i].firstChild;
                        switch(rootNodeUser.childNodes[i].nodeName)
                        {
                            case "profile_image_url":
                            var userPhotoURL = currentValue;
                            userInfo[USER_PHOTO_URL] = userPhotoURL;
                            break;
                            //----------------------
                            case "name":
                            var userName = currentValue;
                            userInfo[USER_NAME] = userName;
                            break;
                            //----------------------
                            case "screen_name":
                            var userScreenName = currentValue;
                            userInfo[USER_SCREEN_NAME] = userScreenName;
                            break;
                            //----------------------
                            case "description":
                            var userDescription = currentValue;
                            userInfo[USER_DESCRIPTION] = userDescription;
                            break;
                            //----------------------
                            case "location":
                            var userLocation = currentValue;
                            userInfo[USER_LOCATION] = userLocation;
                            break;

                        }
                    }
                    switchScreen("Home");
                    showUserPhoto();
                }
            }
            else
            {
                trace("*ERROR: Unable to load XML!!*");
            }
        }

        user_xml.load("http://twitter.com/users/show/" + TWITTER_USER_NAME +
".xml");
        }


        /*
        Displays the user information in the home screen. if it is the first
        load, it downloads the user icon from the network and stores it in the
        icon mc.
        */
        function showUserPhoto()
        {
            if(isFirstUserLoad)
            {
                var urlToUse = userInfo[USER_PHOTO_URL].toString();
                var cutPlace = urlToUse.indexOf("normal");
                var newUrl = urlToUse.substring(0, cutPlace);
                var fileTypePlace = urlToUse.lastIndexOf(".");
```

```
            var fileType = urlToUse.substring(fileTypePlace);
            newUrl = newUrl + "bigger" + fileType;
            icon.loadMovie(newUrl);
            spinnerOff();
            isFirstUserLoad= false;
        }

    userName.text = userInfo[USER_SCREEN_NAME];

    var userDescription = userInfo[USER_DESCRIPTION];
    if(userDescription == null)
    {
        userDescription = "";
    }
    userDesc.text = userDescription;
    icon._visible = true;
}


/*
Starts the slideshow interval
*/
function initSlideshow()
{
    curSlideNum = -1;
    maxSlideNum = numberOfTweets;
    changeSlideshowText();
    slideshow_interval = setInterval(cycleSlideshow, TIME_SLIDESHOW);
}


/*
Starts the fading animation of the slideshow
*/
function cycleSlideshow()
{
    slideshow.gotoAndPlay("on");
}


/*
Clears the slideshow interval when slideshow is stopped
*/
function deleteSlideshowInterval()
{
    if(slideshow_interval)
    {
        clearInterval(slideshow_interval);
    }
}


/*
Changes the text of in the slideshow when the fade is at alpha 0
*/
function changeSlideshowText()
{
    curSlideNum ++;
    if(curSlideNum > maxSlideNum-1)
    {
        curSlideNum = 0;
    }
    slideshow.textCont.tweetTxt.text = tweets[curSlideNum][1];
}


/*
Called when twitter feed data is loaded
```

```
*/
function showTweetsInit()
{
    spinnerOff();
    isTweetsLoaded = true;
    numberOfTweets = tweets.length;
    if(currentScreen == "Tweets")
    {
        initTweetsScreen();
    }
    else if (currentScreen == "Slideshow")
    {
        initSlideshow();
    }
}


/*
Compiles all the tweets into one text box
*/
function initTweetsScreen()
{
    var textToShow = "";
    for(var i = 0; i< numberOfTweets; i++)
    {
        var textToUse = tweets[i][1];
        textToShow += textToUse;
        if(i != numberOfTweets - 1)
        {
            textToShow += "<br>---------------------<br>";
        }
    }
    htmlTextBox.htmlText = true;
    htmlTextBox.htmlText = textToShow;
    var cleanText = htmlTextBox.text;
    tweetFeed.text = cleanText;

    if(tweetFeed.maxscroll == 1)
    {
        sb._visible = false;
    }
    scrollDistance = SCROLLBAR_MAX_Y - SCROLLBAR_MIN_Y;
    scrollUnits = scrollDistance/tweetFeed.maxscroll;
}


/*
Makes the scrollbar functional
*/
function adjustTweetsFeed()
{
    var curYPos = sb._y;
    var adjustedYPos = curYPos - SCROLLBAR_MIN_Y;
    var scrollToShow = Math.round(adjustedYPos/scrollUnits);
    tweetFeed.scroll = scrollToShow;
```

Example 1: The code employed by the social networking application's tweets screen.

5 Skinning

5.1 Introduction

The application illustrates a number of methods that can be used to skin UI elements:

- Buttons, panels, the header, and scrollbars are skinned using preferences set in code.

- The background graphic can be changed by replacing the graphic.

- The text and the graphics used within the application's panels can be changes in code.

5.2 Example skins

The application has been build using three example skins:

- Skin 1

- Skin 2

- Skin 3

5.3 Component skinning using preferences

The `setPrefs` function (see Example 2) is used to define the application's main skinning preferences. These preferences define:

- Colour and alpha levels for the buttons, panel, and header.

- The Twitter user name.

- The number of tweets to load.

- The time between transitions in the slideshow.

```
/*
Here the colors for the ui elements are declared. Also, the twitter user
name and the number of tweets to display are set here.
*/
function setPrefs()
{
    TWITTER_USER_NAME = "WeLoveSoda";
    NUM_TWEETS_TO_LOAD = 10;
    TIME_SLIDESHOW = 5000;

    SCROLLBAR_MAX_Y = 498;
    SCROLLBAR_MIN_Y = 141;

    /**/
    //SKIN 1
    COLOR_BUTTON_UP_STROKE = 0x391C16;
    COLOR_BUTTON_UP_TEXT = 0xFFE066;
    COLOR_BUTTON_UP_OVERLAY = 0x7D4E18;
    COLOR_BUTTON_UP_BG = 0x663300;
    COLOR_BUTTON_UP_STROKE_ALPHA = 100;
    COLOR_BUTTON_UP_BG_ALPHA = 70;
    COLOR_BUTTON_UP_OVERLAY_ALPHA = 70;

    COLOR_BUTTON_DOWN_STROKE = 0x663300;
    COLOR_BUTTON_DOWN_TEXT = 0xFFE066;
    COLOR_BUTTON_DOWN_OVERLAY = 0x7D4E18;
    COLOR_BUTTON_DOWN_BG = 0x7D4E18;
    COLOR_BUTTON_DOWN_STROKE_ALPHA = 100;
    COLOR_BUTTON_DOWN_BG_ALPHA = 100;
    COLOR_BUTTON_DOWN_OVERLAY_ALPHA = 100;

    COLOR_PANEL_STROKE = 0x391C16;
    COLOR_PANEL_BG = 0xFFFFCC;
    COLOR_PANEL_BG_ALPHA = 80;
```

```
COLOR_PANEL_STROKE_ALPHA = 100;

COLOR_HEADER_BG = 0x391C16;
COLOR_HEADER_BG_ALPHA = 70;
COLOR_HEADER_STROKE = 0x391C16;
COLOR_HEADER_TEXT = 0xFFDF65;
COLOR_HEADER_ICON = 0xFFDF65;

COLOR_SCROLLBAR_STROKE = 0x000000;
COLOR_SCROLLBAR_BG = 0xB36F16;
…//MORE CODE
}
```

Example 2: The code used to set the skinning preferences for the social networking application.

5.4 Content skinning

The text and picture in the main panel of the following screens can be changed by modifying the FLA file:

- Home screen.

- About screen.

If you plan to modify and distribute this application, it is recommended that you replace the content in the about screen.

5.5 Button text

The text used on any button can be changed by updating the code on the movie clip that contains the button. You will see code similar to this on each movie clip that contains a button:

```
onClipEvent(load){
   txt.btnText.text = "BUTTON TEXT";
}
```

Replace BUTTON TEXT with the text you would like displayed on the button.

5.6 Backgrounds

You can choose a different background by:

- Going into the bg_mc movie clip on the main timeline.

- Guiding out the images you don't want.

- Turn off the guides on the image you do want.

You can also import a new background image and place it in the bg_mc movie clip.

6 Packaging in a widget

There are three methods that can be employed to distribute Flash Lite content for installation on Nokia devices:

- The Flash Lite files can be distributed in a ZIP file that the user unpacks and then copies the content to their device. The user then runs the application by opening the main *.swf file from their device's file browser application.

- To pack the *.swf file and content into a Symbian Installation System (SIS) file. This requires use of the Symbian build tools and signing of the SIS file with a self-create certificate. The SIS file is

then installed by the user and the application run from an icon in the device's application menu or from a shortcut on the device's home screen.

- Embed the Flash Lite content within a Web Runtime (WRT) widget. The widget is then installed by the user and the application run from an icon in the device's application menu or from a shortcut on the device's home screen. The widget may also be created to include a read-only home screen view.

Embedding Flash Lite within a widget is the recommended approach to distributing Flash Lite content. This method offers the user a simple installation method, access to the content in the device menu or home screen and a simple packaging method (zipping the widgets content into a file with the extension `*.wgz`) that does not require signing.

In addition, widgets enable you to combine Flash with content created with standard web technologies (such as HTML, CSS, and the JavaScript™ language). As many websites use both HTML and Flash, by using Flash Lite within a widget a mobile optimised version of a rich internet experience can be made available for mobile devices. As such, this combination of technologies provides an attractive way for media and content partners, social networking companies, and other internet based businesses to optimise their web property for consumption on mobile devices by leveraging their existing web development skills.

> **Note:** The WRT security model prevents Flash Lite content from making use of any device integration features, such as the ability to read contacts or determine a device's location. If Flash Lite content within a widget needs access to such information, it can be obtained using the JavaScript™ APIs and the data passed to the Flash Lite content using the `externalInterfaces` class.

To package the demonstration application into a widget, using the files in the `social_networking_demo_widget` folder included in the tutorial package, take the following steps:

- In `main.html`, update *Widget Page Title* to reflect your application name.

- If desired, create a graphic of 312 x 85 pixels to represent your widget in home screen. In `main.html`, update `widget_homescreen_image.jpg` to refer to the new graphic.

- Publish your `*.swf` file (or take the one that came in this demo folder) and, if necessary, in `main.html` update `widget_name.swf` to refer to the new `*.swf` file.

- In `info.plist`, update *Widget Name* to reflect the name of the widget you would like to be displayed in the device's menu. You can also update the graphic that's displayed for the icon by replacing or updating `icon.png`.

- In `info.plist`, update `com.demos.widget_name` with the path on the device you want to use for the application data.

- Zip the widget folder and change the extension of the resulting file from `*.zip` to `*.wgz`.

For more information on packaging Flash Lite in a widget see the document *Packing Flash Lite content in a WRT widget* included in the tutorial package.

7 Further areas to explore

The application could be extended to enable the posting of tweets. To do this you will need to use OAuth, an authentication protocol, with Twitter. For more information on OAuth, see the [OAuth FAQ](#) on the Twitter website.

8   Additional resources

The How to load and parse XML files article from the Forum Nokia Flash Lite Wiki relates to the code discussed in this tutorial.

Ina addition, you will file useful information in the Twitter API Wiki and the Forum Nokia Flash Lite: Live XML Data Integration Example.

You can also find extensive information on using Flash Lite in Nokia devices in the Flash Lite Developer's Library. There are additional documents and code examples available from the Documentation section of the Forum Nokia website.

You can also find a wealth of information on the Flash Lite on Nokia Devices discussion board.

9   Evaluate this resource

Please spare a moment to help us improve documentation quality and recognise the resources you find most valuable, by rating this resource.