

Using Adobe Flex in JSR-286 Portlets

This article shall show you how the Adobe Flex SDK can be used in a Portal environment to enhance the user interface for a Portlet. It has also previously been possible to use Flex with JSR-168 Portlets and alike [FlexPortal], but with the new Portlet 2.0 specification (JSR-286) some things have improved, such as the support for asynchronous requests, public render parameters and events. Support for the JSR-286 specification has been available in Websphere Portal since version 6.1. It will be shown what is possible with the technology combination Flex/Portlets and where the limits are. The article builds around a Flex demo Portlet example, which is available for download [below](#).

A short introduction to Flex

Adobe Flex SDK [FlexProductHome], meanwhile in version 3.0, is an open source user interface oriented framework that allows the creation of binary SWF files from XML markup and ActionScript. As opposed to the traditional more visual Flash approach, mainly used by people with a graphical design background, this approach targets at classical programmers. An experienced Java programmer will find it easy to get acquainted with Adobe's Eclipse-based, commercial development environment, called Flex Builder. A compiler called `mxmlc` is run to compile the MXML markup and the ActionScripts into a single SWF file. This SWF file will then be rendered on the client side by the Flash Player plug-in available in most web browsers.

The compilation can either be performed at development time or on-the-fly on the server. For the demo the first approach has been chosen, since it is the default procedure and requires no additional products installed on the server.

Data Access

In order to embed Flex into web applications, this technology depends on interfaces for data exchange, which are available both on the client and on the server side [FlexHelp]:

a) server side data exchange

- ◆ pure HTTP services (GET / POST)
- ◆ SOAP / XML Web Services
- ◆ Remoting: direct access to server-side Java objects

The latter employs a proprietary binary protocol on top of HTTP, which has some performance advantage. It is only available when you have a license for a special server product sold by Adobe. For our purpose, the pure asynchronous HTTP service requests are the best choice, since it allows us not only to stay within standard protocols but also to stay within the portal context.

It is worth noting that Flex has no means of accessing databases directly.

b) client side data exchange

- ◆ initialization through parameter-passing
- ◆ an Ajax bridge, which allows both, the calling of and listening to JavaScript commands outside the Flash container

The example accompanying this article has on the client-side only made use of initialization through parameter-passing for keeping the example simple.

Flex versus JavaScript

So one or the other reader might be asking what is better - JavaScript or Flex? The answer is that both technologies have their advantages and disadvantages and in the end it will depend on a project's concrete requirements and the developers' skills and/or preferences. And, as outlined in the previous section, the decision needn't be excluding one or the other: both technologies can co-exist and interact with each other.

Nevertheless, let me state a brief comparison of the technologies here:

- Traditionally JavaScript has been hard to handle because of its very limited type safety. Until now, no JavaScript development environment was therefore able to give generic editing support for the objects and methods defined in JavaScript libraries. Flex, on the other hand, is a framework that provides a reliable type system. This is one of the reasons, why there is a powerful IDE available for Flex, namely the Eclipse-based Flex Builder. But, just recently, also editing support for JavaScript has reached a new level by the new Eclipse JavaScript Development Toolkit (JSDT) [JSDT-Primer]. JSDT properly models the JavaScript language in the background and provides type management and class inference for JavaScript libraries by an inference engine.
- Uneven Web browser compatibility is another drawback of JavaScript. This can be encapsulated by JavaScript libraries, but it needs to be explicitly taken care of. Flex leverages the widely available Flash Player plug-in, which behaves the same way in any browser and on any platform.
- Drag and drop between two Flash files embedded on the same web page is not possible (e.g. between two Flex Portlets). In comparison to a UI based on JavaScript this is a disadvantage.

The CMS Flex Demo Portlets

Now let's have a look at the demo example provided with this article. Below you can see a screen shot of the application.

Flex Portlet

Flex Portlet Demo Edit Mode

Heading:

Article Body

This **formatted text** is

- displayed here in pure *html* and
- can be edited in **edit mode** with *flex*

Verdana 10

B *I* U http://

When you click on *Save & View*, your current version is being stored in your portlet preferences.

This Portlet is listening to the *content/itemFromHistorySelected* event. Whenever new content is being saved, a *newContent/itemCreated* event will be thrown.

Not sure what this is? Have a look at the online help of this portlet. You can also look at the [Flex source file](#)

Flex Cms History Portlet

This Portlet displays your editing history. Double-clicking on an item in the list will throw a *content/itemFromHistorySelected* event.

Version	Date	Title
1	2008-09-16T12:	Welc
2	2008-09-16T13:	Welc

This list is generated by listening to the *newContent/itemCreated* event and is being stored in the user's session (PORTLET_SCOPE).

Not sure what this is? Have a look at the online help of this portlet. You can also look at the [Flex source file](#)

The application is pretty simple. In view mode of the cms detail Portlet, you have a blue display HTML area where you can view the contents generated in edit mode in Flex Rich Text Editor. Upon clicking “save & view” new content is being stored into the user’s preferences and a `newContentItem` event is being thrown. The cms history Portlet is listening to such events and stores each content item in a list in the Portlet session. The list is being displayed in a sortable Flex `DataGrid`. Double-clicking on an item in that list will in turn throw a `contentItemFromHistorySelected` event, which is consumed by the cms detail Portlet and stored in the user’s preferences.

Noteworthy

In addition to the above, it is worth mentioning the following issues:

- Since you cannot throw events from within a `serveResource` method (see [JSR-286], pp. 105), you will have to do full page submits using action URLs in order to send and receive events. This will interrupt the smooth user experience of Flash.
- In order to pass objects between the Java server code and the Flash client it

is a good idea to serialize the objects into its XML representation. For the flex demo portlet the Java API for XML Binding (JAXB), now an integral part of Java 6 SE, has been used for this task.

- Before JSR-286, it has been proposed to use plain Servlets or XML Web Services instead of the new resource serving mechanism for loading content to an asynchronously requesting UI client ([DojoPortal], pp. 10). This meant that the portal's context wasn't accessible. Now when using the `serveResource()` method, full access is given to a user's environment such as the persisted preferences of a user and security constraints can be enforced by the portal server.

Download

The sample code [download](#) consists of a ready-to-deploy portlet application as Web Archive (WAR), containing two portlets:

- a Flex cms detail portlet, and
- a Flex cms history portlet.

The Source code for both Java and Flex are included (Java source located at /WEB-INF/src, Flex source at /flex-files).

When deploying to Websphere Portal, you will have to wire the two cms content events by manual configuration, once you have put the two portlets on a common portal page, since, until now, there is no auto-wiring provided:

Page Customizer Content Appearance Locks Wires

Portlet Wiring Tool

Wires are connections between portlets. The portlet wiring tool allows you to view, add, and delete wires. To add a new wire between two portlets, use the controls in the table below to specify the wire details and click Add Wire.

Wires for page: **Flex Demo**

Source portlet	Sending	Target page	Switch page	Target portlet	Receiving	Wire Type
Flex Portlet	publish.{http://ibm.com/demo/flexportlet/content}newContentItemCreated	Flex Demo		Flex Cms History Portlet	process.{http://ibm.com/demo/flexportlet/content}newContentItemCreated	Public
Flex Cms History Portlet	publish.{http://ibm.com/demo/flexportlet/content}contentItemFromHistorySelected	Flex Demo		Flex Portlet	process.{http://ibm.com/demo/flexportlet/content}contentItemFromHistorySelected	Public
Select One ...		Flex Demo (flex-demo)				Public

References

- [FlexPortal] Coenraets, Christophe (2004). Using Macromedia Flex in a Portal Environment.
http://www.adobe.com/devnet/flex/articles/flex_portals.html
- [FlexProductHome] Adobe Corporation (2008). Flex Product Home Page.
<http://www.adobe.com/products/flex/>
- [FlexHelp] Adobe Corporation (2008). Adobe Flex 3 Help.
<http://livedocs.adobe.com/flex/3/html/help.html>
- [JSDT-Primer] Childs, Bradley (2008). Meet the JavaScript Development Toolkit.
<http://www.ibm.com/developerworks/library/os-eclipse-jsdt/index.html>
- [JSR-286] Hepper, Stefan et al. (2008). Java Portlet Specification, Version 2.0.
<http://jcp.org/en/jsr/detail?id=286>
- [DojoPortal] Bishop, Karl; Philips, Doug (2007). Using the Dojo Toolkit with WebSphere Portal. http://download.boulder.ibm.com/ibmdl/pub/software/dw/wes/pdf/0711_bishop-Dojo-and-WP6.pdf