

Christian Wenz



Dateien auf DVD

# Flash meets AJAX

**AJAX ist so populär, dass Adobe/Macromedia auch auf den Zug aufspringen will. Die FABridge soll eine Verbindung zwischen Flex-Anwendungen und AJAX herstellen. VISUAL-X wirft einen Blick auf eine erste Vorabversion des Tools.**

Macromedia Flex (beziehungsweise Adobe Flex – bleiben wir einfach bei Flex) ist vom Markt nur äußerst zögerlich angenommen worden. Das hat viele mögliche Gründe, über die in Medien und Weblogs schon zur Genüge philosophiert wurde. Mit Flex 2 scheint sich die Situation etwas zu bessern. Der Beta-Zyklus findet öffentlich statt; nach einer kostenlosen Registrierung steht zum Redaktionsschluss Version Beta 2 von Flex 2 unter [1] zum Download bereit.

Möglicherweise, um noch etwas zusätzliches Interesse zu generieren, hat Adobe ein Produkt namens FABridge veröffentlicht – eine Flex-AJAX-Verbindung. Aktuell gibt es davon eine Vorabversion vom 20. März 2006. Nach eigenen Aussagen von Adobe [2] handelt es sich dabei um eine Version, die noch nicht einmal Alpha-Status erhalten hat. VISUAL-X hat einen Blick darauf geworfen. Da FABridge aber noch am Anfang seiner Entwicklung steht, ist davon auszugehen, dass sich bis zur Finalversion noch viele Details ändern werden.

**Infos.** Eine Möglichkeit, zwischen ActionScript (im Flash-Film) und JavaScript (auf der Website) zu kommunizieren, gibt es bereits seit längerem. Das gilt natürlich auch für reguläre Flash-Filme, in Flex-Anwendungen gibt es die Klasse *ExternalInterface*. Diese stellt eine Schnittstelle zur Verfügung, um von außen auf den Film zuzugreifen und umgekehrt. Das erfordert sowohl ActionScript- als auch JavaScript-Code. Außerdem können nur herkömmliche Datentypen ausgetauscht werden, bei komplexen Objekten erreicht man schnell ein Limit. Dank JSON [3] (siehe dazu auch die Anmerkungen auf Seite 46) ließe sich zwar auch hier eine Lösung finden, doch ganz ohne Aufwand ginge es nicht.

FABridge erlaubt es nun komplett mit JavaScript zu arbeiten. Die „Brücke“ stellt eine JavaScript-Schnittstelle zur Verfügung, um auf ActionScript-Code und -Elemente zuzugreifen. Was das Ganze mit

AJAX zu tun hat, lässt sich in der aktuellen Version nicht erkennen – es könnte der Verdacht entstehen, dass hier auf ein populäres Schlagwort gesetzt wurde, um die Verbreitung anzukurbeln. Flex-AJAX-Bridge klingt einfach etwas schmissiger als Flex-JavaScript-Bridge.

**Installation.** FABridge wird unter [2] zum kostenlosen Download angeboten. Das Archiv selbst ist nicht besonders groß (in der von uns getesteten Version: 590 KB), doch es ist auch nicht alles, was benötigt wird. Um die Software zu verwenden, ist Macromedia Flex 2 notwendig.

Die Installation von Flex 2 beinhaltet unter anderem die Möglichkeit, den Flash Player zu installieren. Das ist insofern wichtig, als dass Flex 2 (und damit auch die FABridge) beide den Flash Player 8.5 benötigen. Auch diesen gibt es vorab als Beta-Version, sowohl innerhalb des Flex-Installers als auch in Form eines Standalone-Installers (einer für Internet Explorer, einer für Mozilla-Browser). Wichtig: Ein Uninstaller wird gleich mitgeliefert.

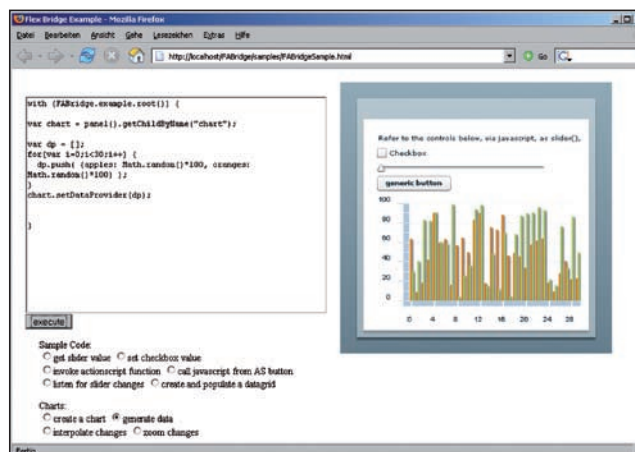
Dann endlich kann auch FABridge zum Einsatz kommen. Das ZIP-Archiv enthält im Wesentlichen zwei Ordner: SRC und SAMPLES. Im ersten finden Sie den eigentlichen Code für FABridge, während

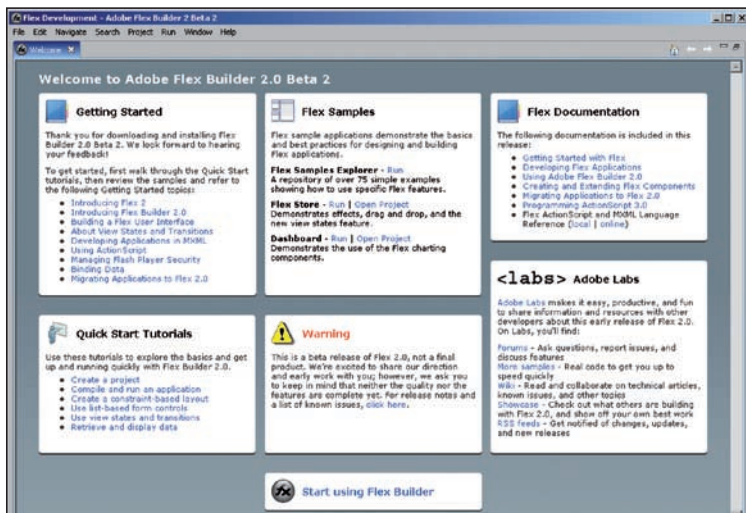
sich im zweiten eine Demo-Anwendung verbirgt, die einige der aktuellen Möglichkeiten von FABridge aufzeigt. Am besten, Sie kopieren den SAMPLES-Ordner auf den Webserver. Zudem dürften unter anderem die folgenden Dateien interessant sein:

- *FABridgeSample.html*: Eine recht komplexe Anwendung mit vielen Beispielen
- *SimpleSample.html*: Ein etwas einfacheres Beispiel
- *srcview/index.html*: Browserbasierte Quellcodeansicht der Beispieldateien

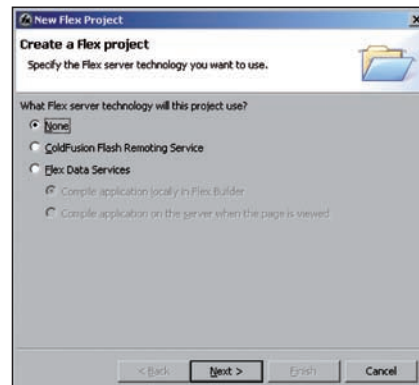
**Anwendung.** FABridge soll an einem kleinen Beispiel vorgeführt werden. Legen Sie dazu in Flex 2 ein neues Projekt an (Projekttyp *None* genügt). Es kann ein paar Sekunden dauern, bis das Projekt initialisiert wurde – sprich: ein paar Dateien angelegt worden sind, inklusive einer leeren MXML-Beschreibung. Auf einem deutschen Windows-System beginnt hier leider schon der Ärger. Beispielsweise wird als Standardordner für das Projekt *C:\Dokumente und Einstellungen\<Benutzername>\My Documents\Flex\<Projektname>* gewählt. *Dokumente und Einstellungen* wurde also lokalisiert, nicht jedoch *My Documents*, der auf einem deutschsprachigen Windows *Eigene Dateien* heißt. Zudem gibt es beim

Beispiel: Die Anwendung demonstriert einige der Features.





Start: Die Startseite von Flex 2 (Beta 2).



Per Assistent: Legen Sie ein „leeres“ Flex-Projekt an.

Verarbeiten von Flex-Projekten Schwierigkeiten, da etwa lokalisierte deutsche Ressourcen gesucht, aber nicht gefunden werden.

Die derzeitige Beta 2 ist nur auf Englisch erhältlich. Im Zweifel hilft also nur, ein englisches Betriebssystem einzusetzen.

Als nächstes müssen Sie dem Projekt die Komponenten von FABridge hinzufügen. Geben Sie dazu bei den Projekteigenschaften unter FLEX BUILD PATH den SRC-Ordner von FABridge als *Classpath* an, wohin auch immer Sie ihn entpackt haben – oder kopieren Sie FABridge di-

rekt ins Projekt. Dann können Sie in der MXML-Datei, innerhalb von `<mx:Application>`, folgendes Element hinzufügen:

```
<fab:FABridge xmlns:fab="bridge.*" />
```

Die Anwendung ist dann in der Lage, FABridge zu nutzen.

Als simple Beispielanwendung verwenden wir ein Texteingabefeld, eine Schaltfläche und ein Label. Wenn ein Benutzer etwas in das Textfeld eingibt und auf die Schaltfläche klickt, soll der Text im Label erscheinen. Das MXML ist dabei denkbar einfach:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" xmlns="*"
                 layout="absolute">
  <fab:FABridge xmlns:fab="bridge.*" />
  <mx:TextInput x="5" y="25" id="TextInput1" />
  <mx:Button x="175" y="25" id="Button1" />
  <mx:Label x="5" y="70" id="Label1" />
</mx:Application>
```

Per JavaScript soll nun eine HTML-Seite darauf zugreifen. Im Ordner HTML-TEMPLATES des Projekts befindet sich eine vorbereitete HTML-Datei. Um den entsprechenden JavaScript-Code unterzubringen, ist es an dieser Stelle allerdings notwendig, selbst eine HTML-Datei zu erstellen.

Der erste Schritt besteht darin, die JavaScript-Komponente von FABridge zu laden. Abhängig von dem Pfad, in dem sich diese befindet, wird sie ungefähr wie folgt in die HTML-Seite integriert:

```
<script language="JavaScript" type="text/javascript"
src="../../src/bridge/FABridge.js" ></script>
```

Sobald der Flex-Film geladen wurde, kann FABridge initialisiert werden. Dazu gibt es zwei Möglichkeiten. Löst der Benutzer – etwa durch eine Schaltfläche – die Initialisierung aus, sieht das wie folgt aus:

```
var flex;
function init() {
  flex = FABridge.flash.root();
}
...
<input type="button" value="Flex initialisieren"
       onclick="init();" />
```

Wird eine Callback-Funktion eingesetzt, kann sich die FABridge-Komponente alternativ auch selbst auslösen:

### Listing 1

#### Die HTML-Seite mit JavaScript-Code, der mit Flex spricht (via FABridge)

```
<html>
<head>
<title>FABridge</title>
</style>
<script language="JavaScript" type="text/
      javascript"
      src="../../src/bridge/FABridge.js" ></script>
<script language="JavaScript" type="text/
      javascript">

// 
var flex;

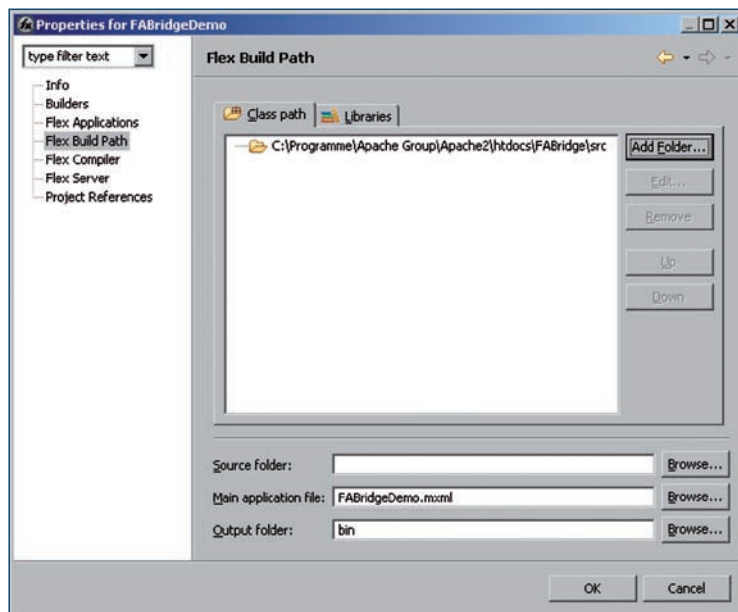
function init() {
  initializeFlex();
  setButtonValue();
  addButtonEventListener();
}

function initializeFlex() {
  flex = FABridge.flash.root();
}

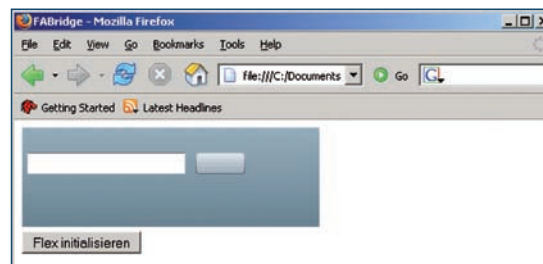
function setButtonValue() {
  flex.Button1.setLabel("Text anzeigen");
}

function addButtonEventListener() {
  flex.Button1.addEventListener("click",
      buttonCallback);
}

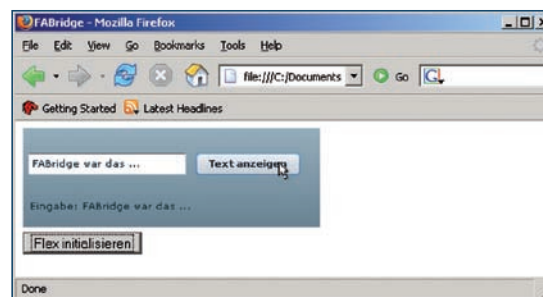
function buttonCallback() {
  flex.Label1.setText("Eingabe: " + flex.
      TextInput1.text())
}
] ]&gt;
&lt;/script&gt;
&lt;/head&gt;
&lt;body&gt;
&lt;object id="flexfilm" classid="clsid:D27CDB6E-
      AE6D-11cf-96B8-444535400000" codebase=
      'http://download.macromedia.com/pub/
      shockwave/cabs/flash/swflash.cab#version=
      8,5,0,0' width="300" height="100"&gt;
  &lt;param name="src" value="FABridgeDemo.swf" /&gt;
  &lt;embed name="flexfilm" pluginspage=
      "http://www.macromedia.com/go/getflashplayer"
      src="FABridgeDemo.swf" width="300"
      height="100" /&gt;
&lt;/object&gt;
&lt;form&gt;
  &lt;input type="button" value="Flex initialisieren"
       onclick="init();" /&gt;
&lt;/form&gt;
&lt;/body&gt;
&lt;/html&gt;</pre></div>
<div data-bbox="71 944 191 961" data-label="Page-Footer"><p>76 VISUAL-X ■ Vol. 13</p></div>
<div data-bbox="817 947 904 960" data-label="Page-Footer"><p>www.visualxmag.de</p></div>
```



**Classpath:** So findet Flex die FABridge-Komponenten.



Vorher: Die Anwendung nach dem Laden.



Nachher: Dank JavaScript-Code tut sich etwas.

```
FABridge.addInitializationCallback("flash", init);
```

Die globale Variable *flex* enthält jetzt einen Verweis auf die Flex-Anwendung im Browser. Für den Zugriff gelten die folgenden Regeln:

- Über die Methode (!) *flex.Name()* greifen Sie auf das Flex-Element *Name* zu.
- Über die Methode (!) *flex.Name1().name2()* greifen Sie lesend auf die Eigenschaft *name2* des Flex-Elements *Name1* zu.
- Mit der Methode *flex.Name1().setName2* („neuer Wert“) greifen Sie schreibend auf die Eigenschaft *name2* des Flex-Elements *Name1* zu. Beachten Sie, dass das erste Zeichen der Eigenschaft hier groß geschrieben werden muss!

### More Info

von Tobias Hauser, Armin Kappler und Christian Wenz

### Einstieg in ActionScript

Verlag: Galileo Press  
ISBN: 3898427749  
Preis: 24,90  
Umfang: 416 Seiten,  
mit CD & Quickfinder  
& Referenz



In der Beispielanwendung sollte zunächst die Schaltfläche beschriftet werden. Die dazugehörige Eigenschaft von Button heißt *Label*, was zu folgendem Code führt:

```
flex.Button1().setLabel("Text anzeigen");
```

Analog sieht der Code aus, um den Wert aus dem Eingabefeld in das Label zu schreiben; Sie sehen hier auch den Unterschied zwischen Lese- und Schreibzugriff.

```
function buttonCallback() {
    flex.Label1().setText("Eingabe: " + flex.TextInput1
        ().text())
}
```

Bleibt nur noch ein Problem: Wie erreicht man, dass die Funktion *buttonCallback()* beim Klick auf die Schaltfläche ausgeführt wird? Eben dazu gibt es die Methode *addEventListener()*:

```
flex.Button1().addEventListener("click",
    buttonCallback);
```

Das war es auch schon! Sie müssen nur noch den von Flex generierten SWF-Film einbinden und darauf achten, dass die Benutzer Version 8.5 des Flash Players installiert haben. Listing 1 enthält den kompletten Code für die HTML-Datei; auf der Heft-DVD finden Sie zusätzlich den fertigen SWF-Film sowie die MXML-Datei.

**Fazit.** Obwohl FABridge mit AJAX anscheinend eher wenig zu tun hat, liefert die

Vorabversion bereits einen guten Eindruck davon, was möglich ist. Und zugegeben, die Idee, mit JavaScript auf eine Flex-Anwendung zugreifen zu können, ist verführerisch. Nur schade, dass es auf deutschen Systemen noch Schwierigkeiten gibt – was aber nicht der Fehler von FABridge ist, sondern an der Beta 2 von Flex 2 liegt. Und wer AJAX vermisst, sei auf [4] verwiesen; dort kündigt Adobe die Bibliothek Ajax Client for Flex Data Services (ACFDS) an, die auf die Flex Data Services zugreifen kann. Downloads gibt es leider noch keine, aber wir bleiben am Ball.

### Links & Quellen

- [1] [www.macromedia.com/go/labs\\_flex2\\_downloads](http://www.macromedia.com/go/labs_flex2_downloads)
- [2] [labs.macromedia.com/wiki/index.php/Flex\\_Framework:FABridge](http://labs.macromedia.com/wiki/index.php/Flex_Framework:FABridge)
- [3] [json.org](http://json.org)
- [4] [labs.macromedia.com/wiki/index.php/Flex\\_Enterprise\\_Services:Ajax\\_Client](http://labs.macromedia.com/wiki/index.php/Flex_Enterprise_Services:Ajax_Client)

### Autor Christian Wenz

Christian Wenz ist Autor, Trainer und Berater mit Schwerpunkt Webtechnologien, schreibt für diverse Fachmagazine und spricht auf Konferenzen im In- und Ausland. Er bloggt unter



[www.hauser-wenz.de/blog/](http://www.hauser-wenz.de/blog/)