php architect
www.phparch.com

MOBILE APPS

CONNECTING WITH THE IPHONE
Get Your Data Anywhere

ADVANCED MOBILE DEVICE DETECTION WITH TERA-WURFL
Find Your Match

FIREFOX FOR CONTROL FREAKS
See the Whole Picture

GAME DESIGN FOR WEB DEVELOPERS
Get Your Users into the Action

GOVERNMENT AND PHP
Untapped Potential

SAVIOR OF THE UNIVERSE?
WHAT'S NEW IN FLASH BUILDER 4

INSIDE

PLUS

CONTENTS
MARCH 2010

FEATURES

COLUMNS

Download this month's code at:
http://phparch.com/code

WRITE FOR US!

If you want to bring a PHP-related topic to the attention of the professional PHP community, whether it is personal research, company software, or anything else, why not write an article for php|architect? If you would like to contribute, contact us, and one of our editors will be happy to help you hone your idea and turn it into a beautiful article for our magazine. Visit www.phparch.com/writeforus.php or contact our editorial team at write@phparch.com and get started!

SAVIOR OF THE UNIVERSE?

by Marco Tabini

Flash Builder 4—the much anticipated successor to Flex Builder—is finally out. Here's an exclusive look at all the juicy bits that you'll find inside.

REQUIREMENTS

PHP *5.0+*

Other Software:

- Flex Builder 4 - http://labs.adobe.com/technologies/flashbuilder4/

So far, 2010 has not been a particularly happy year for Flash—Apple's sidewise revelation of the iPad's lack of support for Adobe's front-end technology has, depending on who you ask, either signaled its demise or simply highlighted the Fruit Company's increasing arrogance.

I prefer to think that the iPad announcement highlighted a third thing: how little the general public (technical or not) understands how much Flash has grown as a platform since the days of annoying shoot-the-monkey ads and endless intros.

Granted, both the ads and the intros are still around, but if you think that they represent all there is to know and like about Flash, you're missing some rather significant pieces of the puzzle.

Flex Who?

Here at php|architect, we have been using Flash extensively for many years, even though you don't get to see it when you're using our site (with some minor exceptions, like the podcast player). In fact, almost our whole publishing arm is managed entirely using Flash.

Now, before you start thinking about our customer support folks spending their day trying to shoot monkeys in order to access a customer's account, you need to know that we actually use Flash as a platform for building client-side applications (that is, software that runs natively on our computers, as opposed to residing in a browser).

The tool that allows us to do this is called *Flex*—a framework that builds on top of Flash to provide an augmented set of functionality built with the professional developer in mind. Once Flex comes into the picture, gone are timelines, assets and other designer-oriented notions, to be replaced with concepts that a programmer will be more familiar with, such as classes, methods and windows.

I am, of course, oversimplifying: the timeline and its companions are not really gone—they're just being hidden by the framework in such a way that they are no longer a concern.

Free, and Not So Free

The Flex framework is completely free and open source—you can simply visit the Adobe Labs site at http://www.adobe.com/products/flex/ and grab your very own copy.

While building full-fledged applications this way is entirely possible (and, in fact, convenient to set up your very own continuous integration environment), it is not nearly as practical as using Adobe's Flex Builder, which provides a tightly-integrated IDE for creating applications using a much more visual approach.

For the last several years, I have been actively using Flex Builder, both for internal and external clients, and I must say that the only real drawback to it is the fact that it is based on Eclipse, an IDE that I would be perfectly happy to have never encountered in my life.

Despite its Java-ness, Flex Builder is the perfect tool for creating Flex applications: it features a visual environment that lets you lay out your interface without having to go through the typical guesswork that is part and parcel of HTML development (aided, in part, by the fact that Flex's box model appears *not* to have been developed by the inmates of a Guatemalan insane asylum deep in the jungle primeval), and a code editor that comes with all the bells and whistles that we normally associate with an advanced IDE: debugging, code introspection and completion, and so forth.

Flex, Evolved

In short, Flex Builder has everything you need to turn a web developer into a full-fledged Flash developer without losing their sanity. Thanks to Adobe AIR—a technology that extends Flex into the desktop space—that same developer can just as easily build client-side applications that run natively on a user's computer instead of their browsers.

It is a clear sign of how the lines between web and desktop are blurring—and of how Adobe's strategy in those spaces is evolving—that the two products, Flash and Flex, are rapidly converging into a single platform. Thus, it's no surprise that the next iteration of Flex Builder, which Adobe has just released to the general public, is called *Flash Builder 4* (FB4).

As I will show you throughout the rest of this article, FB4 doesn't just represent the next step in the evolution of Flex, but also the first step in a revolution in the way application development takes place, especially when taken within the larger context of Adobe's suite of creative applications.

The Development Lifecycle

Building applications (web or otherwise) is an exercise that requires the involvement of different types of people with wildly different talents that do not really mix well with one another: designers, artists, developers, and so on.

It's interesting to note that Adobe produces software that caters to many of these groups—many designers I know use Illustrator to build their interfaces (something that, I freely admit, has always puzzled me) and Photoshop to make them look pretty.

Unfortunately, the process doesn't always work as well as it could. For example, Illustrator is a vector software, which makes it difficult to build interfaces on a pixel-by-pixel basis (something that designers seem to relish torturing frontend developers with); similarly, the process of translating a Photoshop file into an HTML document can, and often is, anything but simple.

The biggest hurdle, however, in the process is that it is strictly one-way: a Photoshop design that has been "sliced and diced" into HTML is essentially immutable—even if you use Photoshop's built-in slicing features, *any* change to the HTML makes it essentially impossible to go back and alter the PSD file with any hopes of regenerating the HTML output.

As web applications grow in complexity and provide an increasingly richer user experience, this traditional model falls apart more and more quickly. The demand for application changes keeps growing as we explore the new possibilities of Web 2.0, while our ability to keep up with it has remained flat.

Tool developers have clearly taken note of this problem and have started coming up with solutions—however timidly. Microsoft's Expression Studio, for example, includes *Blend + SketchFlow*, an application that allows interface designers to create mockups and wireframes that can then be fed directly into one of the company's other IDEs to allow developers to wire them to actual functionality.

With the release of Flash Builder 4, Adobe is rapidly moving in this direction as well. FB4 is part of a larger initiative that includes a new product called Flash Catalyst—the latter, currently in beta, allows designers to build interactive user interfaces that can then be fed into FB4 so that the developers can start working on them right away. Unfortunately, the process is currently monodirectional—meaning that you still can't go back from FB4 into Catalyst for changes; however, building applications with the new workflow is a much simpler affair, and it's not farfetched to expect that, as the tools mature, the process will become fully bidirectional.

The Emperor's New Clothes

Compared to its predecessors, it's fairly obvious that a good amount of effort has gone into improving the look and feel of everything from the developer experience to the user interface of the applications generated by FB4. Even the installer, shown in Figure 1, has undergone a complete facelift and now sports a slick, colorful interface.

While the overall interface of the IDE (whose start-

FIGURE 1

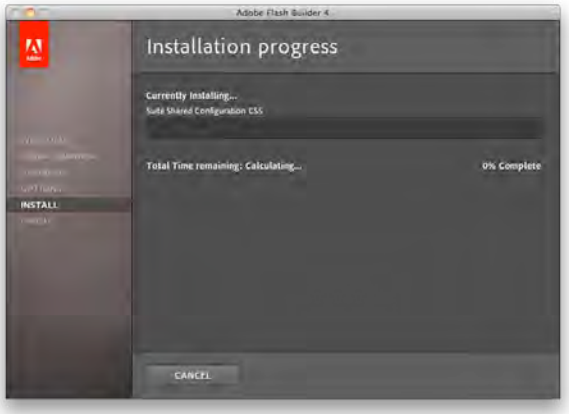
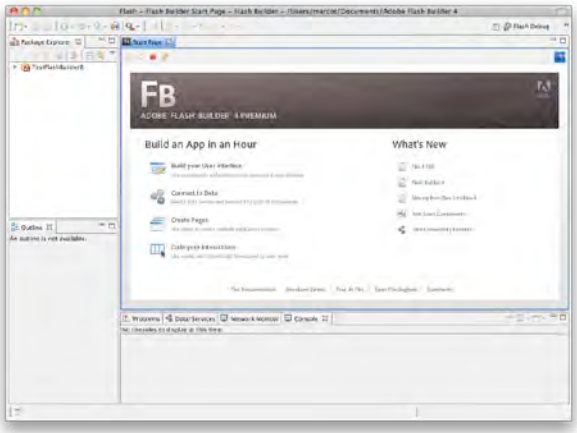


FIGURE 2

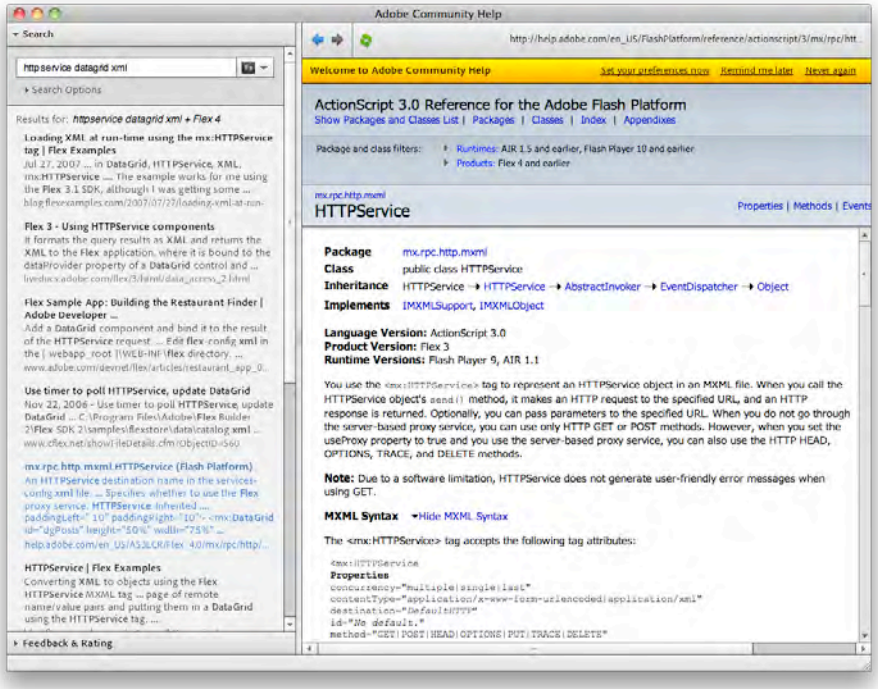


up screen you can see in Figure 2) retains the classic Eclipse look, a number of things have changed. First of all, FB4 does away with the integrated help of old, replaces it with the standard Adobe Help application (Figure 3), which boasts a much better look and seems to be better integrated with the company's online live documentation and community sites.

More importantly, FB4 seems to be much more responsive than its predecessor—although I haven't run any scientific benchmarks, everything from visual design to compilation is much snappier than it used to be in Flex Builder 3. I'm not sure if this is due to improved coding on Adobe's part or the use of a better JVM (Java Virtual Machine)—but it's definitely a welcome change.

Finally, the entire UI component library has been retooled, and a new visual theme, called Spark, has been introduced. The differences between Spark and the older Halo theme are quite substantial—as you can see in Figure 4, the interface looks much cleaner and less visually intensive than its predecessor.

FIGURE 3

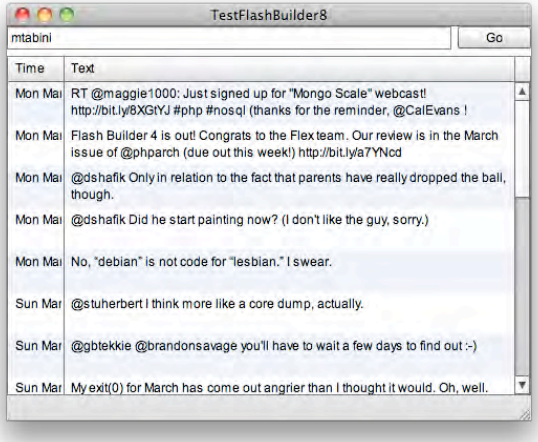


Life with Flex

Like always, Flex is set up primarily to provide a rapid development environment for applications that are, at their core, data-centric thin clients—essentially, the controller and view part of a model-view-controller architecture in which the model resides somewhere reachable via a network operation.

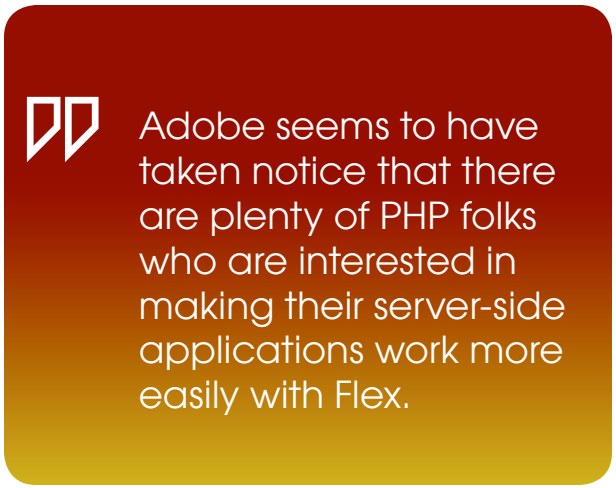
This is not to say that you cannot create complex applications that do not depend on a remote

FIGURE 4



backend of some sort—but, rather, that a number of facilities in FB4 are designed to facilitate the creation of software whose business layer resides somewhere else.

Whereas previous versions of FB were aimed primarily at the Java crowd, Adobe seems to have taken notice that there are plenty of PHP folks who are interested in making their server-side applications work more easily with Flex. The entire process



of connecting to an external service layer has been greatly simplified to the point where all you need to do is point FB4 to your server's web root and choose a file whose functionality you want to be able to access.

FB will essentially take over from that point. Flash uses a protocol called AMF to marshal and serialize data and function calls between servers and clients. If your server does not include an AMF-compliant library, it will even offer to download and install a copy of **Zend_AMF** for you, which, although part of Zend Framework, can be used independently from the framework as a standalone library (Figure 5). Once **Zend_AMF** is installed, you can browse your PHP code and have FB create a proxy ActionScript3 class that you can use to transparently call your server-side functions from your Flex projects.

If AMF is not your thing, FB4 features a number of other wizards that are designed to interface with traditional web services that run SOAP or that return XML or JSON. In these last two cases, the system makes a number of assumptions on how your service layer is set up—primarily, it expects it to follow general REST principles.

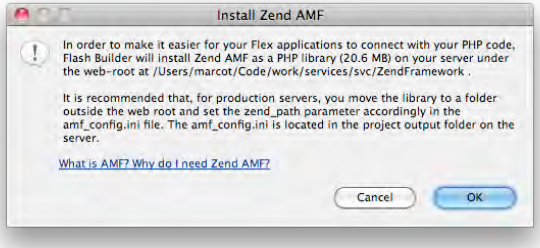
The proxy classes that FB's wizards create do, of course, have some limitations—most significantly, the fact that they need to be generated from scratch every time the server-side functionality changes. This is not necessarily an issue, as long as your team follows proper refactoring principles.

If, like me, you prefer to create your own infrastructure, Flex still provides some excellent facilities for data manipulation—from the E4X XML-handling functionality built into AS3 to the JSON add-on that is available as part of Adobe's **as3corelib** project at http://code.google.com/p/as3corelib/. It is, in a way, a shame that JSON—which has become the lightweight format of choice for web services—is not yet built into the language or framework themselves; on

LISTING 1

```
<?xml version="1.0" encoding="utf-8"?>
<s:WindowedApplication xmlns:fx="http://ns.adobe.com/mxml/2009"
      xmlns:s="library://ns.adobe.com/flex/spark"
      xmlns:mx="library://ns.adobe.com/flex/mx">
   <fx:Declarations>
      <s:HTTPService id="twitterService" url=
"http://api.twitter.com/1/statuses/user_timeline/{username.text}.xml"
useProxy="false" />
   </fx:Declarations>
   <s:VGroup width="100%" height="100%" left="0" top="0">
      <s:HGroup width="100%">
         <s:TextInput width="100%" id="username" text="mtabini" />
         <s:Button label="Go" click="twitterService.send()" />
      </s:HGroup>
      <mx:DataGrid
         dataProvider="{twitterService.lastResult.statuses.status}"
         width="100%" height="100%">
         <mx:columns>
            <mx:DataGridColumn headerText="Time"
               dataField="created_at" width="100" />
            <mx:DataGridColumn headerText="Text" dataField="text"
               wordwrap="true" width="800" />
         </mx:columns>
      </mx:DataGrid>
   </s:VGroup>
</s:WindowedApplication>
```

FIGURE 5



the other hand, acquiring and installing `as3corelib` is as easy as downloading it from Google Code—and that gives you complete access to its source code.

All this functionality, of course, dovetails with Flex's tightly integrated object model, so that even complex interface interactions can be managed by just wiring the right components together, thus freeing you to focus on the actual functionality provided by your application. Take, for example, the code in Listing 1, which when dropped into an AIR application template, will give you a fully-functional (albeit rather bland) Twitter client capable of reading any user's timeline. If the resulting application is, perhaps, not that impressive, the fact that it was built without writing a single line of ActionScript code never fails to amaze me.

Something in the AIR

FB4 incorporates version two of Adobe's AIR platform, which allows you to create Flex-based desktop applications. As far as I'm concerned, AIR is the true star of the entire Flex platform. Even if you think that HTML5 is the way to go on the web, it's hard to argue with the fact that AIR makes it essentially trivial for a web developer who is reasonably familiar with JavaScript to become a full-fledged purveyor of desktop applications.

AIR is usually criticized for two reasons: its inability to accurately replicate native user interfaces and the limitations imposed on it by its sandbox model.

On the former point, I personally think that people who claim that AIR applications don't look anything like their native counterparts are missing the point that they are not *supposed to*. Although it's possible to create a reasonably credible approximation, the whole point of AIR is to provide a cross-platform development environment. Therefore, the differences in user interface appearance and behaviors between operating systems would make it challenging—to say the least—to create an application capable of working across multiple platforms.

On the sandbox front, however, I concur that some of the design decisions have been more difficult to understand. For example, an application built with AIR 1.x could not instantiate any other application on the system. This has now been corrected (to a degree) in AIR 2, where it is possible to invoke the default handler for a given file type. You can also package your AIR applications with a native executable and exchange information between the two.

Finally, AIR 2 comes with two interesting new features: a much-enhanced networking functionality (including the capability of exchanging data using UDP and of creating inbound sockets), and direct access to audio input sources, such as your microphone. Perhaps, this will finally make it possible to realize my dream of a podcast client that doesn't require pagan sacrifices in order to work (I *am* running out of goats and chickens).

The Verdict

There is much to like in FB4—not least, the fact that Adobe is obviously taking Flex's prospects very seriously and that they are finally treating PHP developers as first-class citizens.

The combination of new features and increased support for PHP make Flex and AIR an interesting platform for developing your applications on the web and, especially, on the desktop. Flash is clearly not the best solution to *every* problem that requires a dynamic user interface, but Adobe is very quickly expanding its reach well beyond the much-hated ads and video players.

While the functionality provided by AIR 2 and FB4 are a huge step forward from what was previously available, there still are areas where there could be improvements—for example, having full access to locally-captured video streams, or screen sharing. Overall, however, Flex and AIR are as strong a platform as they have ever been—and every indication is that they will only get better with time.

MARCO TABINI is Keeper of Keys and Garbage Collector Extraordinaire for MTA (phparchitect's parent company) and Blue Parabola, LLC, where his job consists mainly of yelling at people over the phone (and, sometimes, over the Internet). Despite twenty years in the IT business, he still harbors the secret wish of someday becoming an internationally famous science fiction author.