Introduction to Adobe Flex

Alin-Cristian JOIȚA
Faculty of Computer Science for Business Management,
Romanian – American University, Bucharest, Romania

ABSTRACT

Rich Internet applications *(RIA) are web applications that have the features and functionality of traditional desktop applications. RIAs typically transfer the processing necessary for the user interface to the web client but keep the bulk of the data (i.e. maintaining the state of the program, the data etc) back on the application server.*

Adobe Flex *is an umbrella term for a group of technologies initially released in March of 2004 by Macromedia to support the development and deployment of rich Internet applications based on their proprietary Macromedia Flash platform.*

1. INTRODUCTION

The Adobe Flex product line is the most comprehensive solution for delivering RIAs across the enterprise and over the web. Designed to help developers and development organizations meet the challenges presented by RIAs, Flex is already being used by hundreds of organizations to deliver interactive data dashboards, customer and employee self-service applications, online product selectors and configurators, and business-to-business applications.

The Flex product line provides a highly productive programming model (Flex framework), integrated Eclipse-based development tools (Flex Builder), and robust data integration services (Flex Data Services) that enable organizations to rapidly deliver solutions that dramatically improve user productivity and increase online revenues, while integrating with existing applications and websites.

Applications delivered with Flex offer a better experience because they take advantage of the browser and Flash Player runtime. Installed on over 97% of Internet-connected PCs, Flash Player provides a consistent, cross-platform runtime that combines a high-performance virtual machine with integrated support for multilingual text display, printing, data manipulation, motion, and multimedia. On top of these capabilities, the Flex framework layers a rich set of user interface components and design principles that encapsulate best practices in interaction design and usability.

Flex provides client-side service components that enable applications to interact with any remote server via SOAP web services, REST services, or raw HTTP or custom socket-based protocols. For more sophisticated integration needs, Flex Data Services provides additional support for publish/subscribe messaging, real-time streaming data, and direct integration with existing server-side Java objects as well as other enterprise back-end applications including messaging, security, and transaction management.

Finally, Flex provides a highly productive development model that easily integrates with existing processes and is based on standards and best practices that have emerged over the last ten years of Internet development. The Flex development model uses XML for user interface design and layout and an implementation of ECMAScript (that is, JavaScript) for client logic. The Flex Builder integrated development environment (IDE) provides a robust set of coding, debugging, and visual user interface layout tools that shorten the learning curve for new developers and easily integrate with existing source code management systems. In addition, Flex provides integrated support for unit testing and

automated functional testing tools.

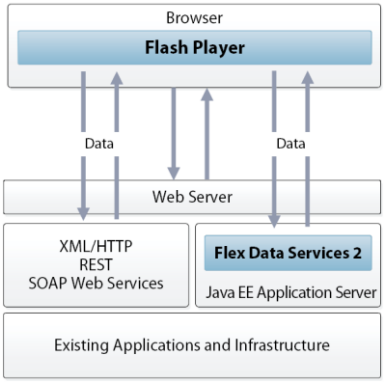
2. FLEX RUNTIME ARHITECTURE

The Flex runtime architecture is closely aligned with the just-in-time deployment model of web applications. The client portion of a Flex application is deployed as a binary file that contains the compiled bytecode for the application. Users then deploy this file to a web server just as they would an HTML file or an image. When the file is requested by a browser, it is downloaded and the bytecode is executed by the Flash Player runtime.

As illustrated in Figure 1, once started, the application can request additional data or content over the network via standard HTTP calls (sometimes referred to as REST services) or through web services (SOAP). Flex clients are server agnostic and can be used in conjunction with any server environment, including standard web servers and common server scripting environments such as JavaServer Pages (JSP), Active Server Pages (ASP), ASP.NET, PHP, and ColdFusion.

Figure 1

If the Flex client application is used in conjunction with Flex Data Services, the application has access to additional services. Flex clients can make direct calls to Java objects as well as subscribe to real-time data feeds, send messages to other clients, and integrate with existing Java Message Service (JMS) messaging systems. The Flex Data Services application runs on the server within the Java web container.



3. FLEX DEVELOPMENT MODEL AND APPLICATION FRAMEWORK

The development process for Flex applications mirrors the process for Java, C#, C++, or other traditional client development languages. Developers write MXML and ActionScript source code using the Flex Builder IDE or a standard text editor. As shown in Figure 2, the source code is then compiled into bytecode by the Flex compiler, resulting in a binary file with the *.swf extension.

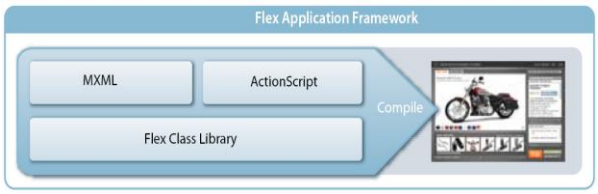


Figure 2

As shown in Figure 2, the Flex application framework consists of MXML, ActionScript 3.0, and the Flex class library. Developers use MXML to declaratively define the application user interface elements and use ActionScript for client logic and procedural control. The Flex class library contains Flex components, layout managers, behaviors, and service components.

With the Flex component-based development model, developers can create applications using prebuilt components, combine prebuilt components into composite components, or create new components by extending the prebuilt components or their base classes. The ability to create custom components is one of the most powerful aspects of Flex development.

Like other enterprise runtime environments, Flash Player provides a rich set of services that developers can use to construct components. These include display APIs for drawing to the screen, manipulating graphics, and controlling audio or video as well as APIs for accessing network resources, parsing data, and performing calculations. Combined with the built-in layout, data binding, and effects classes in the Flex component API, these provide a complete environment for delivering a wide variety of custom applications.

4. MXML: THE FLEX MARKUP LANGUAGE

Like HTML, MXML is a markup language that describes user interfaces that expose content and functionality. Unlike HTML, MXML provides declarative abstractions for client-tier logic and bindings between the user interface and application data. MXML helps maximize developer productivity and application reusability by cleanly separating presentation and business logic.

The following example produces the output shown in Figure 3:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
xmlns:mx="http://www.adobe.com/2006/mxml
" layout="absolute">
        <mx:Form x="397" y="298">
                <mx:FormHeading
label="Form Heading"/>
                <mx:FormItem
label="RadioButton Control">
                        <mx:RadioButton
x="409" y="477" label="Radio"/>
                </mx:FormItem>
                <mx:FormItem
label="CheckBox">
                        <mx:CheckBox
x="409" y="451" label="Checkbox"/>
                </mx:FormItem>
                <mx:FormItem
label="ComboBox">
                        <mx:ComboBox
x="409" y="339"></mx:ComboBox>
                </mx:FormItem>
                <mx:FormItem
label="TextField">
                        <mx:TextInput x="409"
y="421"/>
                </mx:FormItem>
                <mx:FormItem
label="Button">
                        <mx:Button x="409"
y="309" label="Button"/>
                </mx:FormItem>

        </mx:Form>
</mx:Application>
```

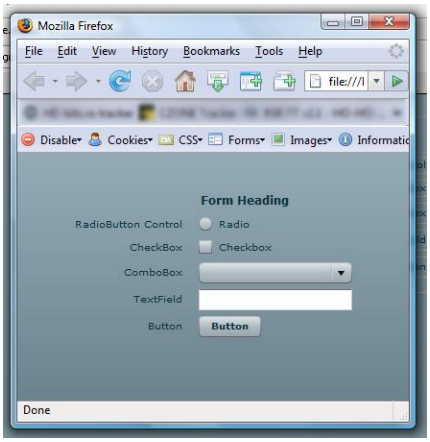


Figure 3

5. ACTIONSCRIPT 3.0

ActionScript is the object-oriented programming language used for Flex development. Like JavaScript, ActionScript 3.0 is an implementation of ECMAScript, the international standardized programming language for scripting. However, because it is an implementation of the latest ECMAScript proposal, ActionScript provides many capabilities not common in the versions of JavaScript supported by most browsers. At development time, ActionScript 3.0 adds support for strong typing, interfaces, delegation, namespaces, error handling, and ECMAScript for XML (E4X).

At runtime, the most significant difference between JavaScript and ActionScript is that ActionScript is just-in-time compiled to native machine code by Flash Player. As a result, it can provide much higher performance and more efficient memory management than interpreted JavaScript.

Flex developers use ActionScript to write client-side logic, such as event listeners and call-back functions, or to define custom components.

ActionScript 3.0 is up to 10 times faster than it's predecessor ActionScript 2.0.

MXML gets translated into ActionScript 3.0 code at compile time. The following would be the AS equivalent of the previous MXML example:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
xmlns:mx="http://www.adobe.com/2006/mxml
" layout="vertical" initialize="start()">
  <mx:Script>
```

```
    <![CDATA[
        import mx.controls.TextInput; import
mx.core.UIComponent;              import
mx.controls.Button;
        import mx.controls.ComboBox; import
mx.controls.CheckBox;             import
mx.containers.FormHeading;
        import mx.controls.RadioButton;
import mx.containers.FormItem;    import
mx.containers.Form;
        private function start():void {
                var form:Form = new Form();
                var fh:FormHeading = new
FormHeading(); fh.label = "Form Heading";
form.addChild(fh);
                var firb:FormItem = new
FormItem();  firb.label = "RadioButton
Control";
                var rb:RadioButton = new
RadioButton();
                rb.x = 409; rb.y = 451;
        rb.label = "radio"; firb.addChild(rb);
        form.addChild(firb);
                var ficb:FormItem = new
FormItem(); ficb.label = "CheckBox";
                var cb:CheckBox = new
CheckBox();
                cb.x = 409; cb.y = 451;
ficb.addChild(cb); form.addChild(ficb);
                var ficmb:FormItem = new
FormItem(); ficmb.label = "ComboBox"
                var cmb:ComboBox = new
ComboBox();
                cmb.x = 409; cmb.y = 339;
ficmb.addChild(cmb); form.addChild(ficmb);
                var fiti:FormItem = new
FormItem(); fiti.label = "TextField";
                var ti:TextInput = new
TextInput() ti.x = 409; ti.y = 421;
fiti.addChild(ti); form.addChild(fiti);
                var fib:FormItem = new
FormItem(); fib.label = "Button";
                var btn:Button = new Button();
                btn.x = 409; btn.y = 309;
btn.label = "Button"; fib.addChild(btn);
form.addChild(fib);
                this.addChild(form);
        }
    ]]>
  </mx:Script>
</mx:Application>
```

6. FLEX CLASS LIBRARY

Flex includes a rich class library that contains Flex components (containers and controls), data binding, behaviors, and other features.

Beyond providing a set of built-in capabilities (described in the following subsections), Flex components follow a consistent cross-platform experience model based on user interface design best practices. As a result, developers can deliver professional-looking applications that delight users without the active involvement of a graphic designer.

Where a custom look and feel is desired, designers can easily customize components through an extensive set of CSS-based styles. In addition, users can create custom skins using industry-standard tools such as Photoshop, Illustrator, and Flash Professional. As with built-in styles, custom skin properties are set using CSS properties.

6.1. VISUAL COMPONENTS

The component-based model eases the creation of Flex applications. Developers can use the prebuilt components included with Flex, extend components to add new properties and methods, and create new components.

The Flex class library supplies two types of visual components: containers and controls. When developers build an application using Flex, they describe its user interface with controls and containers. Controls are user interface components that handle user interactions and display data that users can manipulate directly through that control. Examples of controls are the DataGrid and the TreeControl. A container defines a region of the Flash Player drawing surface and controls the layout for everything in the container, including other containers and controls. Examples of containers are a data entry Form container, a Box, and a Grid.

Flex components are extremely flexible and provide developers with a great deal of control over the component’s appearance, how the component reacts to user interactions, the font and font size of any text included in the component, the size of the component in the

application, and more. Flex components support the following characteristics:

- **Events—**Application or user actions that require a component response
- **Behaviors**—Visible or audible changes to the component triggered by an application or user action.
- **Skins—**Symbols that control a component's appearance
- **Styles—**Set of characteristics, such as font, font size, and text alignment
- **Size—**Height and width of a component (all components have a default size)

Developers can control these characteristics at development time through MXML or CSS, or at runtime through the component's ActionScript API, including creating or destroying instances of a component based on application data or user interaction.

6.2. SERVICE COMPONENTS

The Flex service components and underlying Flash Player enable applications to access data from a wide variety of resources. The Flex class library includes built-in classes for calling SOAP-based web services and for loading XML or other data via HTTP. Developers can also take advantage of custom protocols by leveraging support for binary sockets in Flash Player or by loading data from the host browser. Using Flex Data Services, developers can also make remote API calls to Java objects or subscribe to real-time message queues and data services.

Once retrieved, data in a Flex application can be managed as a typed variable, an array of objects, as native XML (using E4X), or as an instance of the Collection class. The Collection class simplifies development of data-driven applications by automatically keeping track of changes to the data so that they can be sent to the remote server when the application is ready to synchronize.

Flex also provides a mechanism for binding data objects to visual controls so that the user interface is automatically updated when the underlying data is changed, either as a result of logic running on the client or of changes sent from a remote server. Data binding can be set up declaratively in MXML or programmatically in ActionScript.

6.3. FLEX BEHAVIORS

The Flex class library also provides prebuilt behaviors that enable developers to easily add motion and sound to their application to give users context for their actions. For example, a developer can use behaviors to cause a dialog box to bounce slightly when it receives focus or animate a user selected item to illustrate the transition from a master view to a detail view.

A behavior is a combination of a trigger paired with an effect. A trigger is an action, such as a mouse click on a component, or a component becoming visible. These are typically exposed as events.

An effect is a visible change to the component occurring over a period of time, measured in milliseconds. Examples of built-in Flex effects are fade, move, resize, or pause. Developers can define their own effects using ActionScript or composite multiple built-in effects together to meet their application needs. Effects can be applied to individual components or containers.

7. FLEX BUILDER

Flex Builder is the Adobe IDE for Flex development. It is built on the open source Eclipse tools platform and can be used either as a standalone product or as a set of plug-ins within an existing Eclipse installation.

Flex development can be done with any text editor, but Flex Builder enables developers to learn Flex quickly and continue working productively by providing a rich set of code editors, a drag-and-drop user interface assembly, and a powerful interactive debugger.

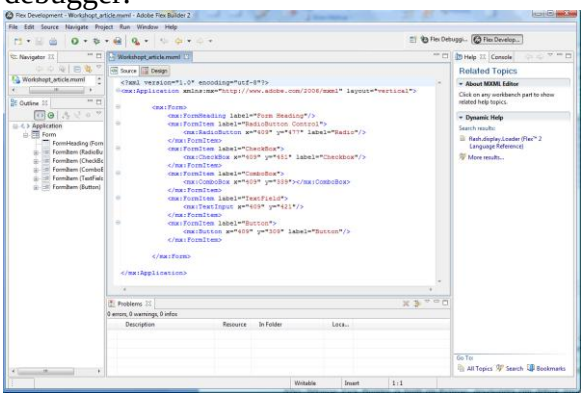


Figure 4

Flex Builder provides built-in code editors for MXML, ActionScript, and CSS. In addition to code hinting for built-in Flex tags and classes, Flex Builder provides statement completion and type checking for custom classes and libraries. The built-in incremental compiler also flags syntax errors and type mismatches as developers work, enabling them to quickly fix mistakes and move on, rather than spending valuable time trying to hunt down problems after the fact.

The Flex Builder design view enables developers to quickly assemble and preview Flex application interfaces. Developers can add custom or built-in components by dragging them from the component view and then take advantage of snapping and alignment tools to arrange them in the user interface. They can also make changes directly in the code and quickly switch to design view for a high-fidelity preview of the compiled application. Flex Builder supports all of the layout models available in MXML, including the box model, absolute positioning, and constraint-based layout.

Flex Builder also makes it easier to customize the appearance of an application. Property editors enable developers to quickly set the most commonly used properties and preview the results in design view. In addition, users can easily import graphical assets created in professional design tools such as Flash or Photoshop for use as icons or skins in Flex applications.

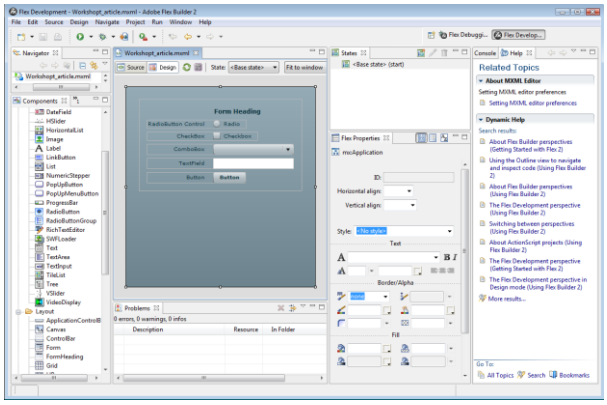


Figure 5

8. INTERACTIVE DEBUGGING

The Flex Builder integrated debugger enables developers to quickly track down and resolve problems in their applications. The debugging perspective allows them to set breakpoints, inspect variables and expressions, change values, and monitor trace messages. Applications can be debugged in standalone Flash Player or in any browser that has Debug Flash Player installed, including remote machines running a different operating system.

9. CONCLUSION

Flex is a very powerful tool for developing RIAs. Its rich library is full of specialized components that speed up the process of RIA building. It is also very flexible. Developers can take full advantage of the powerful ActionScript 3.0 language to build custom components from scratch, extend existing components or just modify them to suit their needs.

Flex applications run smooth on user machines because of the optimized Flash Player 9 environment which uses a “garbage collector” to free up system resources as they are no longer needed.

Flex offers endless possibilities for building fast and complex Rich Internet Applications.

REFERENCES

[1] Adobe Products Overview http://www.adobe.com/products/flex/

[2] FLEX, http://www.flex.org/

[3] Wikipedia http://en.wikipedia.org/wiki/Adobe_Flex/