Adobe® Flex™ 2

While few would question the benefits that Internet-based applications have brought to businesses and consumers alike, the actual experience of interacting with many web-based applications leaves much to be desired, especially when compared with the richness and usability of the best desktop applications.

For consumer-oriented applications, such as e-commerce, the web's page-based model and lack of client-side intelligence can make even relatively simple transactions confusing and error prone. As a result, online businesses are losing millions of dollars to abandoned shopping carts or costly customer service calls.

For business applications, the problem is particularly acute. While the web deployment model has allowed IT organizations to reduce the cost of software deployment, it has also created a community of underserved business users that long for a return to the usability and responsiveness of desktop and client/server applications. As a result, businesses are losing millions of dollars per year due to low productivity or poor decisions.

Fortunately, after long focusing on the technical challenges of web-enabling their application infrastructure, forward-looking IT professionals are now turning their attention to design patterns and technologies that can improve the client side of the equation. As a result, we are now seeing widespread deployment of rich Internet applications (RIAs), a new class of applications that combines the responsiveness and interactivity of desktop applications with the broad reach and ease of distribution of the web.

RIAs can drive increased return on investment (ROI) by simplifying and improving the user interaction—enabling users to find information more easily, complete tasks quickly and accurately, and use rich data visualization to make better decisions. Before realizing these benefits, however, IT professionals must navigate through a new set of technologies as well as understand the architectural and developer skill requirements implied by the move toward RIA-style applications.

This technical white paper discusses both the short-term and long-term requirements of RIAs and discusses how the Adobe Flex 2 product family helps IT organizations take advantage of existing skills and infrastructure to efficiently deliver a broad range of RIAs that can scale from simple marketing applications to mission-critical enterprise applications.

The evolution of rich Internet applications

Macromedia (now part of Adobe) introduced the term *rich Internet application* in 2001 to describe a new kind of Internet-based application being delivered by the vanguard of the web development community. RIAs combine best practices in user interaction design—for example, avoiding page refreshes, expanding information in place, and using interactivity and video to guide or train users—with sophisticated use of web-based technologies such as Macromedia® Flash® from Adobe, HTML, and JavaScript to deliver a better user experience.

Since then, the interest in RIAs has exploded. While consumer-facing sites have been the most aggressive adopters of RIA technology, many enterprises are now moving to apply that technology to internal and external business applications as well. For example, SAP has made improvement of the user experience using RIAs a major element of its 2006 product roadmap.

RIAs are more than just "eye candy"; rather, they provide measurable value to the enterprise. According to leading researchers, adoption of RIA technology is accelerating. Forrester Research foresees "a significant swing in 2006 toward the thin client model for enterprise application development and deployment,"[1] while Gartner believes that by 2010 over 60% of new projects will include RIA technology[2].

As enterprises move to develop and deploy RIAs, however, they are finding that delivering on the vision requires two important ingredients:

- A new class of client runtime that can support the range of needs inherent in rich Internet business applications
- Tools and technology that can provide a productive environment for building, maintaining, and managing these applications throughout their lifecycle

The need for a service-oriented client

Over the past five years, IT organizations have made significant investments in modernizing their back-end systems to take advantage of service-oriented architecture (SOA). By exposing core business systems (and the processes they embody) as a set of services, IT organizations hope to become more agile as well as reduce the cost of system maintenance or updates. As a result, many organizations can now make business processes more efficient and implement new business processes that integrate existing systems through web services or an enterprise service bus.

While the move to SOA has steadily improved the efficiency and stability of back-end applications, web browsers—the main client-side application runtime—have not advanced beyond their original role as document browsers. To realize the full benefits of a SOA, developers need a richer set of technical capabilities to modernize the client-side components of their applications. This new set of capabilities will provide a service-oriented client (SOC)—a runtime environment that can deliver not only the enhanced usability promised by RIAs but also reliable and secure connectivity to back-end systems.

While the full set of services required for SOCs will continue to evolve as RIAs become more commonplace, at a minimum organizations should seek the following capabilities:

- **High-performance, cross-platform runtime**—Business applications must handle large amounts of data and support complex client-side business logic and data processing. As a result, the SOC must be able to manipulate large amounts of data in memory and update the user interface without a user-perceivable degradation in performance.
- **Integrated support for text, graphics, animation, and audio/video**—The most usable applications combine multiple modes of presentation to deliver information more effectively. The SOC must provide an integrated programming model that allows developers to control all of these modes within their application.

[1]"The Rise of Rich Internet Applications," Forrester Research, April 10, 2005.

[2]"Rich Internet Applications Are the Next Evolution of the Web," Gartner, May 5, 2005.

- **Enterprise data integration**—The request/response model is sufficient for website browsing, but many applications require optimized high-performance data transfer as well as additional modes of interaction, including publish/subscribe messaging and the ability to push data or alerts from the server to the client.
- **Support for disconnected computing**—While wireless broadband has increased connectivity, business applications need to continue functioning when the network connection is lost temporarily, and many other applications must enable users to work offline and resynchronize their work when they reconnect.
- **Security and reliability**—Before organizations deliver critical applications using RIA technologies, they must be confident that the applications will be available when needed. Moreover, the runtime must ensure that data can be transferred securely, and the client "sandbox" must prevent users or third-party applications from accessing sensitive information without proper authorization.

While only some applications will require all of these capabilities, IT organizations that want to deploy RIA technology as a strategic platform—or even just minimize the number of one-off solutions deployed in their organization—should adopt client technology that can provide the full range of capabilities required by the SOC.

RIA development model

Of course, to adopt RIAs, organizations need technology and tools that make development and delivery efficient. Moreover, the new applications must extend existing investments in skills, processes, applications, and infrastructure.

To be successful within today's enterprise IT organization, RIA technology solutions must:

- **Provide a familiar programming model**—Application developers are constantly under pressure to deliver more with less. The RIA development environment must leverage existing skills and best practices, including the use of object-oriented languages for business logic and tag-based, declarative models for user interface layout.
- **Leverage existing architecture**—Organizations have invested heavily in application server technology and SOAs. Using and complying with this infrastructure is also a requirement for most organizations.
- **Support standard protocols and application programming interfaces (APIs)**—One of the many positive results of the web has been the adoption of a broad spectrum of standards across the entire technology stack. This includes, but is not limited to, industry standards, such as HTML/HTTP(S), XML, Simple Object Access Protocol (SOAP)/web services, cascading style sheets (CSS), and Scalable Vector Graphics (SVG) as well as Java Platform, Enterprise Edition (Java EE)—formerly known as J2EE—and Microsoft .NET. Incorporating these standards, where appropriate, is a requirement for most organizations.
- **Follow common key design patterns**—To increase both quality and modularity, development organizations are increasingly adopting common patterns like model-view-controller (MVC) as standard architectures for their applications. The RIA development models should build on this store of best practices and developer knowledge.
- **Integrate with existing processes**—Development organizations have adopted source code control systems, automated testing suites, and other application lifecycle tools to increase efficiency and quality and reduce maintenance costs. An RIA development solution must fit into these existing processes and integrate with the tools already at use within the organization.
- **Provide rich tooling**—Developers spend a great deal of time coding and debugging application logic. Beyond an efficient development model, an RIA solution must include tools that can facilitate learning, automate common tasks, and reduce the amount of time developers spend finding and fixing bugs. At the same time, the development model must enable developers to continue using their existing editors for core code writing tasks.

Introducing Adobe Flex

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 and Flash Player also provide the robust connectivity required in the SOC. 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.

Flex and the Adobe Engagement Platform

With the combination of Adobe and Macromedia, Adobe has brought together the best-in-class tools, services, and clients to dramatically reduce the cost and complexity of developing engaging web applications. By combining these powerful technologies, Adobe is delivering an industry-defining development platform for creating applications that dramatically improve how businesses engage with people, processes, and information.

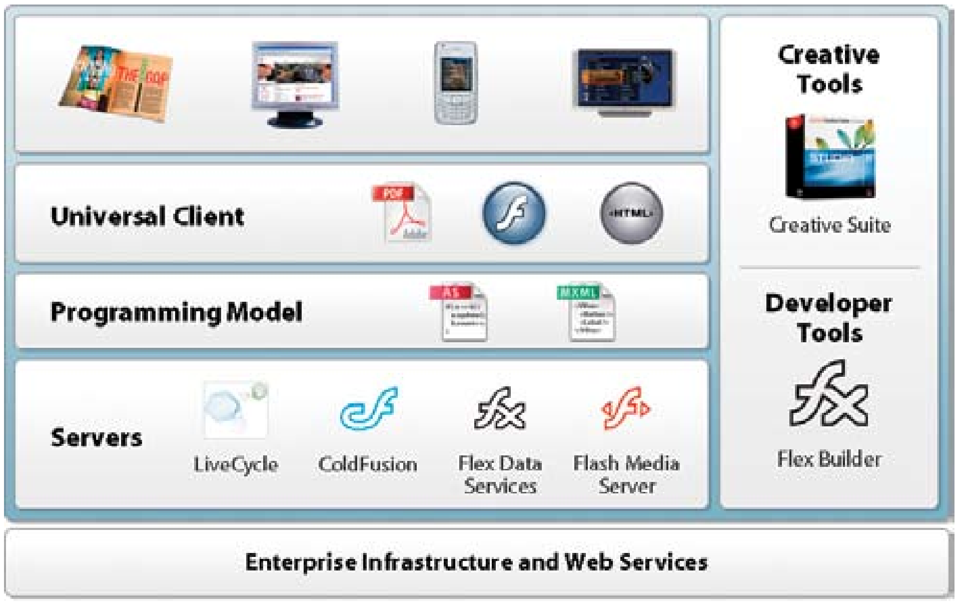


Figure 1: The Adobe Engagement Platform architecture.

The Adobe Engagement Platform comprises:

- **Universal client technology**—By combining the strengths of ubiquitous Flash Player with Adobe Reader® software, HTML, and JavaScript, developers can deliver a predictable, high-quality application experience across browsers, desktops, and devices.
- **Programming model**—The Flex development model (MXML and ActionScript) plays a central role in the platform. By providing a versatile and robust programming model, Flex enables organizations to efficiently deliver RIAs that take advantage of the universal client technology.
- **Development and design tools.**—With products like Adobe Photoshop®, Dreamweaver®, Flash Professional, and Illustrator®, Adobe is a recognized leader in the creative tools market. Through integration with Flex Builder and third-party development tools, Adobe is enabling designers and developers to work together to deliver more engaging experiences.
- **Server framework**—Adobe server technologies build on existing infrastructure standards like Java EE and .NET, while providing services that simplify integration and extend the capabilities available to rich clients. Beyond the services provided by Flex Data Services, Flash Media Server enables applications to integrate two-way audio and video streaming, while Adobe LiveCycle® software provides services for business process management, document generation, and information assurance.

The goal of the Adobe Engagement Platform is to blend the strengths of Adobe technologies and open standards to provide a versatile foundation for extending the reach of information, processes, and services to customers, partners, and employees anytime, anywhere, and in any medium.

By providing a comprehensive yet open solution for RIA development, Flex enables organizations to extend their existing investments in application logic, infrastructure, and SOA while realizing the benefits that RIAs can deliver for end users, customers, or partners.

The following sections provide a more in-depth overview of the Flex product line.

Flex product line overview

As shown in Figure 2, the Flex product family comprises four separate products:

- **Flex Software Development Kit (SDK)**—The core component library, development languages, and compiler for Flex applications
- **Flex Builder IDE**—An Eclipse-based development environment that includes code editors, visual layout tools, project management tools, and an integrated debugger
- **Flex Data Services**—A Java server-based application that enables high-performance data transfer, cross-tier data synchronization and conflict management, and real-time data messaging
- **Flex Charting**—A library of extensible charting components that enables rapid construction of data visualization applications

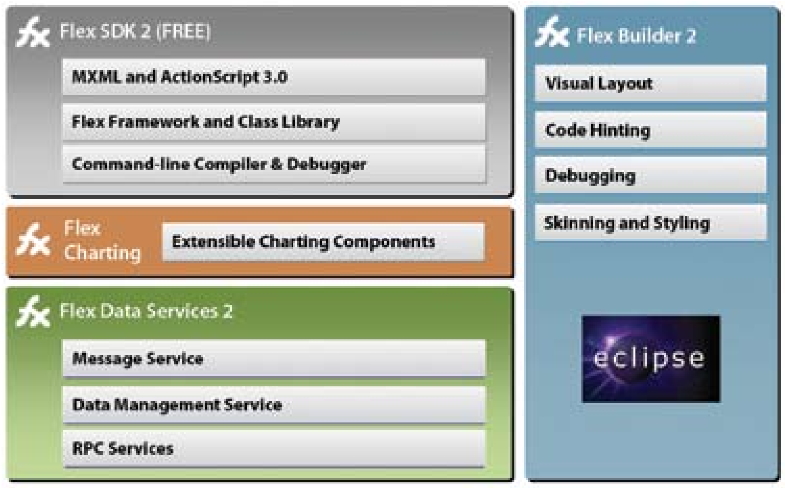


Figure 2: Members of the Flex product line.

Flash Player

No discussion of the Flex product would be complete without mentioning Flash Player. Flex applications are deployed as compiled bytecode that is executed within the Flash Player runtime. Installed on over 97% of Internet-connected PCs, Flash Player provides a consistent runtime environment that works across browsers and operating systems, enabling Flex applications to deliver much greater cross-platform compatibility than other RIA technologies.

Flash Player provides a unique combination of capabilities that developers can use in their applications, including rich text rendering, powerful graphics APIs, an animation engine, and an integrated audio/video codec. These capabilities not only contribute to the rich out-of-the-box look and feel displayed in Flex applications, but they are also available to developers who need to construct custom components. By taking advantage of the drawing APIs, for instance, developers can quickly build a custom component that provides a unique visualization needed by their application.

The ActionScript 3 virtual machine provides a high-performance client-side runtime for application code. When Flex applications are loaded, the just-in-time compiler translates the cross-platform Flash bytecode into machine code, enabling rapid client-side data processing and efficient memory management. This type of performance is particularly important for business applications, in which the requirements for client-side data sorting and business rule execution are more significant.

Flex runtime architecture

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 3, 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®.

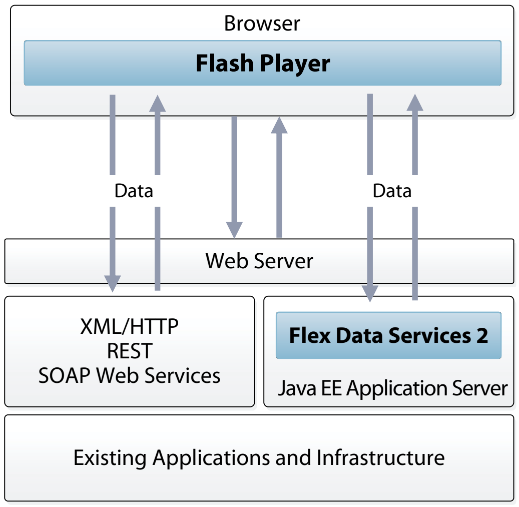


Figure 3: Flex runtime architecture.

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.

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 4, the source code is then compiled into bytecode by the Flex compiler, resulting in a binary file with the *.swf extension.

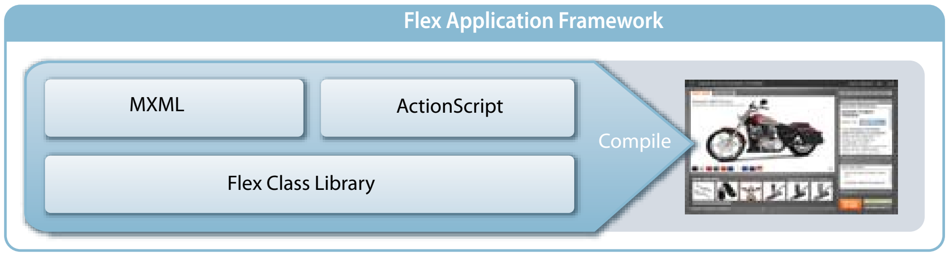


Figure 4: The Flex framework comprises MXML, ActionScript, and the Flex class library.

As shown in Figure 4, 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.

MXML: The Flex markup language

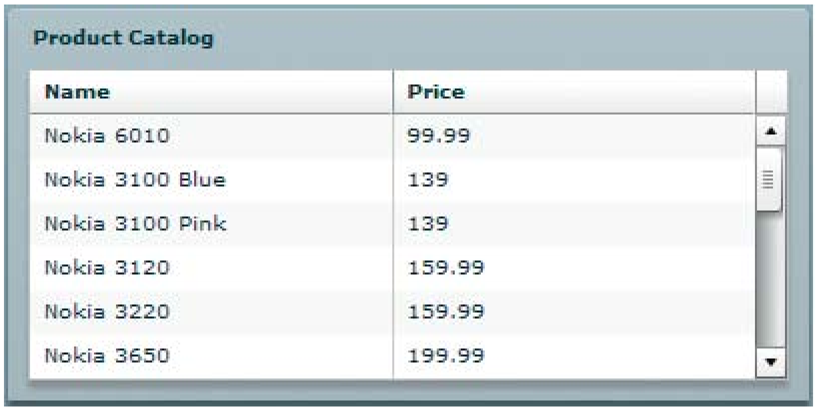
Like HTML, MXML is a markup language that describes user interfaces that expose content and functionality. Unlike HTML, MXML provides declarative abstractions for client-tier logic and bindings between the user interface and application data. MXML helps maximize developer productivity and application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

The following code listing uses MXML to define the user interface for a product catalog application. This example uses the built-in Panel and DataGrid components as well as an instance of the WebService class, which sets up a connection between the client application and the catalog web service.

```
<?xml version="1.0" encoding="utf-8"?>

<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
creationComplete="ws.getList()">
   <mx:Style source="main.css"/>
   <mx:WebService id="ws" wsdl="http://mysite.com/services/CatalogWS?wsdl" >
      <mx:operation name="getList"/>
   </mx:WebService>
   <mx:Panel title="Product Catalog">
      <mx:DataGrid width="100%" height="100%"
      dataProvider="{ws.getList.lastResult}">
         <mx:columns>
            <mx:DataGridColumn dataField="name" headerText="Name"/>
            <mx:DataGridColumn dataField="price" headerText="Price"/>
         </mx:columns>
      </mx:DataGrid>
   </mx:Panel>
</mx:Application>
```

Figure 5 shows the resulting application, including the data retrieved from the web service.


Product Catalog

Name	Price
Nokia 6010	99.99
Nokia 3100 Blue	139
Nokia 3100 Pink	139
Nokia 3120	159.99
Nokia 3220	159.99
Nokia 3650	199.99

Figure 5: A product catalog created with MXML and populated via a web service.

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 types for the client application. For example, the following code shows the definition of the Customer class.

```
Customer.as
    public class Customer {
               private var _ id:int;
               private var _ firstName:String;
               private var _ lastName:String;
               private var _ mobilePhone:String;
               private var _ officePhone:String;
               ...
           public function get id ():int {
           return _ id;
           }
...
    }
```

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.

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.

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 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 (see "Flex Data Services" for more detail).

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.

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.

Other developer tools

Adobe works with third-party application development lifecycle solution providers to help ensure Flex works with standard tools and processes. Adobe also supports several open source projects that facilitate Flex development. Some key initiatives include:

- **Functional testing**—As part of the Flex 2 release, Adobe is working with Mercury Systems to enable users to create and execute automated testing scripts from within Mercury QuickTest Professional.
- **Unit testing**—Modeled on the JUnit test framework, Flex Unit is an open source library that enables developers to create automated unit tests for ActionScript code.
- **Performance and load testing**—Adobe has collaborated with Mercury and Borland (formerly Segue) to integrate Flex with their industry-leading load testing tools.
- **Cairngorm**—Cairngorm is an architectural framework designed and maintained by the Adobe Developer Center. Available as a free open source project, Cairngorm enhances Flex development by providing a standard architecture and methodology for handling user gestures on the client and mapping them to business logic and server interactions through a centralized client controller.

For more information on any of these projects, visit the Flex Developer Center at *www.adobe.com/devnet/flex*.

Flex Data Services

Flex Data Services extends the capabilities of the Flex client framework by providing additional services for managing data transfer and integrating with existing applications and infrastructure.

As illustrated in Figure 6, Flex Data Services fits into an organization's existing deployment environment. It is implemented as a Java web application and can be deployed on standard Java application servers, including IBM WebSphere, BEA WebLogic, Adobe JRun, JBoss, Tomcat, and others.

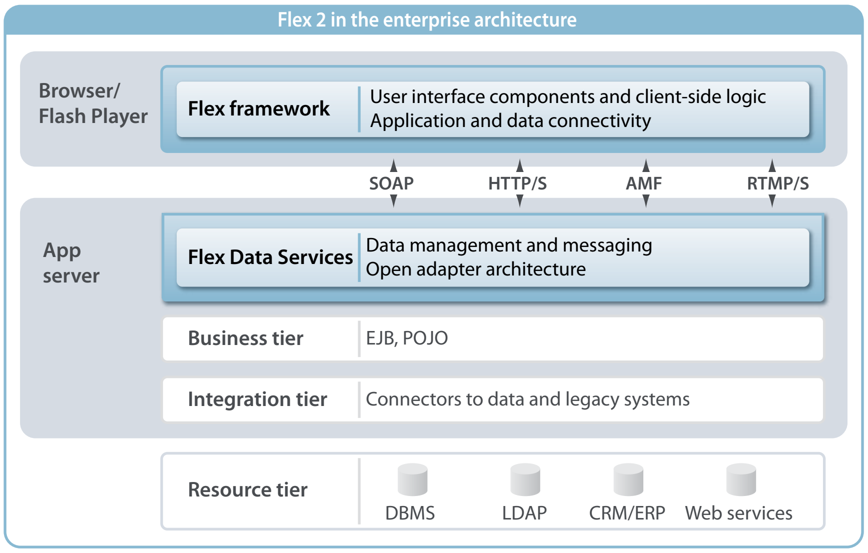

Flex 2 in the enterprise architecture
Browser/ Flash Player
Flex framework
User interface components and client-side logic
Application and data connectivity
SOAP
HTTP/S
AMF
RTMP/S
App server
Flex Data Services
Data management and messaging
Open adapter architecture
Business tier
EJB, POJO
Integration tier
Connectors to data and legacy systems
Resource tier
DBMS
LDAP
CRM/ERP
Web services

Figure 6: Flex is designed to integrate cleanly with the existing infrastructure.

The services provided by Flex Data Services integrate with the existing security profiles defined within the Java application server. Flex Data Services can be deployed using standard deployment tools provided with the server and can integrate with application server clustering features to enable highly available applications. In addition, applications built with Flex Data Services can access existing server-side session data and application logic using standard Java APIs.

Figure 7 shows a high-level overview of the services provided by Flex Data Services. When working with Flex Data Services, developers define a set of "destinations" using XML configuration files. These definitions are used by the built-in service adapters provided as part of the Flex Data Services application. These include low-level adapters to connect to Java objects (data access objects), JMS topics/queues, or ColdFusion components (CFCs) as well as higher level adapters for common persistence solutions such as Hibernate, Enterprise JavaBeans (EJB), and Spring. The Flex Data Services adapter architecture is open and customizable, allowing connectivity to any back-end data system or application.

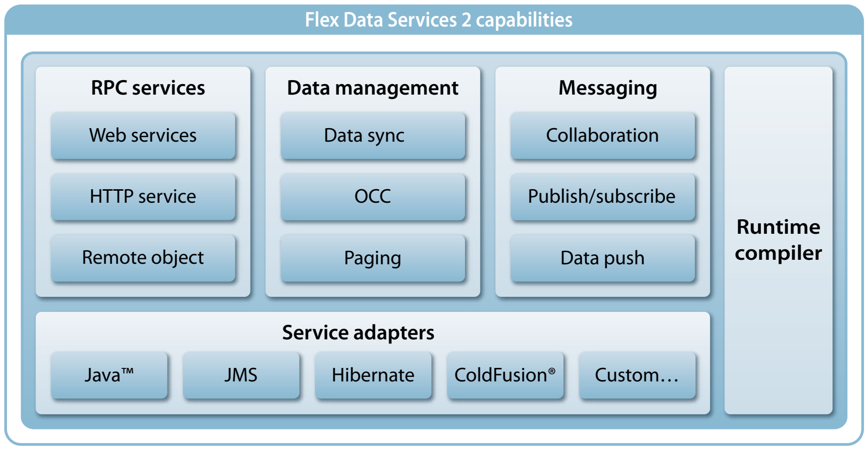


Figure 7: Flex Data Services 2 capabilities.

Once the appropriate destinations have been configured, the Flex developer can access them through a set of client APIs, either in MXML or ActionScript.

Flex Message Service

All of the data services mentioned previously use the Flex Data Services server message bus for high-performance data communication and reliability between the Flex client and the server. All service calls are routed through the message service, including remote procedure call (RPC) calls. As a result, calls made from within Flex applications are automatically queued when the network is unavailable, providing a higher level of reliability and the foundation for offline applications. Similar to a JMS-based messaging system, the Flex Message Service provides developers with complete control over quality of service, including configuration parameters for reliable message delivery and time to live, load balancing and clustering, and the ability to set up multiple failover channels per destination.

The Flex Message Service is optimized for communication between Flex clients and Flex Data Services servers. It is not designed to replace existing messaging systems but rather to extend those systems to allow thin clients to participate in existing enterprise messaging applications. The message service integrates with existing messaging systems through a set of adapters. A JMS adapter is available out of the box. Third-party vendors and developers can also develop their own adapters for messaging systems that don't support JMS.

The addition of messaging to Flex applications enables whole new classes of web applications. Real-time data feeds that integrate with JMS or other messaging technologies can provide highly accurate stock prices for a rich trader desktop or monitoring dashboard, and asynchronous communications can be pushed to a Flex application without a client request. The Flex Message Service supports collaborative applications that include peer-to-peer chat, gaming, or in-context co-browsing. All of these applications can be deployed using a thin client and the web deployment model.

RPC Services

Flex RPC Services include the Remoting Service and proxies for managing HTTP and SOAP requests. The Remoting Service provides native connectivity between Flex clients and remote server-side Java objects. As a result, Flex applications can easily integrate with existing application logic, including the Java session object. The Remoting Service handles data marshaling between Java and ActionScript automatically and integrates with existing security profiles to handle user authentication and authorization. In addition, because data is transferred in a binary format, the Remoting Service can significantly increase performance in applications where large amounts of data need to be transferred from client to server.

The Proxy Service fulfills two functions. First, it enables communication between the Flex client and domains it cannot access directly, due to security restrictions.[3] By proxying requests from the application's domain, the Proxy Service enables developers to integrate multiple services with a single Flex application as well as reduce the opportunity for malicious discovery of security credentials used to access applications exposed as web or HTTP services. In addition, because services are accessed via developer-configured names, the Proxy Service also provides a layer of abstraction between the Flex client and the back-end service, enabling the underlying implementation or Uniform Resource Identifier (URI) to change without requiring modifications to the Flex client.

Data Management Service

The Data Management Service is designed to address one of the unique challenges posed by RIAs. In traditional page-centric HTML applications, the client is mostly used as a data capture and display device. The client doesn't own a version of the data, except perhaps to edit a particular record, and even in that case, changes are quickly sent to the server for verification. The data synchronization (or persistence) process is limited to synchronizing data that exists in middle-tier applications with data that is stored in a back-end data system (RDBMS, mainframe, and so forth).

With RIAs, however, the application client can own its own copy of the data. With a more complete data model, applications become more responsive since functions like sorting or filtering can be performed locally. However, having a local copy of the data introduces several issues that must be addressed. The application must synchronize and manage data conflicts between the client and the server, especially when application data is constantly changing, as well as optimize data download and bandwidth utilization in applications with large data sets.

The Data Management Service greatly facilitates the process of synchronizing or persisting data between the client tier and the middle tier. It can also integrate with existing persistence solutions (such as Hibernate) to provide an end-to-end persistence solution. The Data Management Service represents an evolution towards data orientation in which the programming model is centered on data and objects instead of focusing on method invocations, dramatically improving developer productivity.

Using the DataService API, developers can automatically synchronize data changes (creates, reads, updates, and deletes) between all clients using the managed data object as well as with the back-end data system via the Flex Data Services server. The Data Management Service also offers developers the ability to manually control when and how data updates are sent from client to server. If any conflicts arise, the Data Management Service raises an exception, and the developer can use the conflict resolution API to handle it appropriately, allowing client updates to overwrite server updates and vice versa.

The high-level data management API dramatically reduces the amount of code that developers have to write, debug, and maintain. Not only does it relieve developers of complex synchronization code, but data objects using the Data Management Service are automatically enabled to use data paging to maximize client-side application performance and optimize bandwidth utilization. As a result, developers can focus their efforts on implementing business logic, rather than on low-level data marshaling or cursor management.

Moreover, because the Data Management Service uses the underlying message service, changes can also be pushed to other clients that are subscribed to the same destination. This can be extremely useful for applications in which decisions depend on having the most up-to-date copy of the data, such as a customer relationship management or inventory tracking application. The Data Management Service can also be used to create collaboration-enabled applications in which multiple clients work simultaneously on a common data set.

[3]Both Flash Player and the browser's JavaScript runtime restrict the ability of downloaded code to access network resources. The sandbox only allows code to access resources on the same domain from which it was loaded, thereby preventing malicious code from accessing a user's internal network resources. For more information on the Flash Player sandbox security model, visit the Adobe Security Center at *www.adobe.com/devnet/security*.

Runtime compilers

Flex Data Services also provides a runtime compiler. As with JSPs, an MXML file can be compiled dynamically when its URI is requested. While primarily employed for iterative development, this capability can also be used to dynamically generate application code at runtime and provide a just-in-time compiled version of the application. As with other dynamically compiled pages, like JSP, the initial compilation of an MXML page can take a few seconds. However, the resulting bytecode is cached by the server so that subsequent requests do not require recompilation (provided the underlying MXML file has not changed).

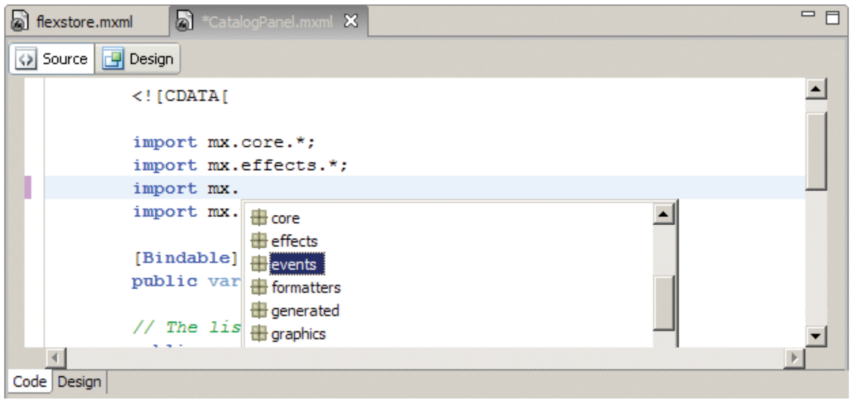
Flex Builder

Flex Builder is the Adobe IDE for Flex development. It is built on the open source Eclipse tools platform[4] and can be used either as a standalone product or as a set of plug-ins within an existing Eclipse installation.

As discussed previously, 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.

Code editing

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.



```
        <![CDATA[

        import mx.core.*;
        import mx.effects.*;
        import mx.
        import mx.

        [Bindable]
        public var

        // The lis
```

Figure 8: Flex Builder source editor.

Visual layout and skinning

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.

[4]Eclipse is an open source platform for developing tools. The project is managed by the Eclipse Foundation, an international consortium of software vendors (including Adobe) that contribute to the Eclipse projects and set the direction of the core platform. Besides being the most popular Java IDE on the market, the Eclipse tools platform is also used as the foundation for products from IBM, SAP, Sybase, and Actuate, among others.

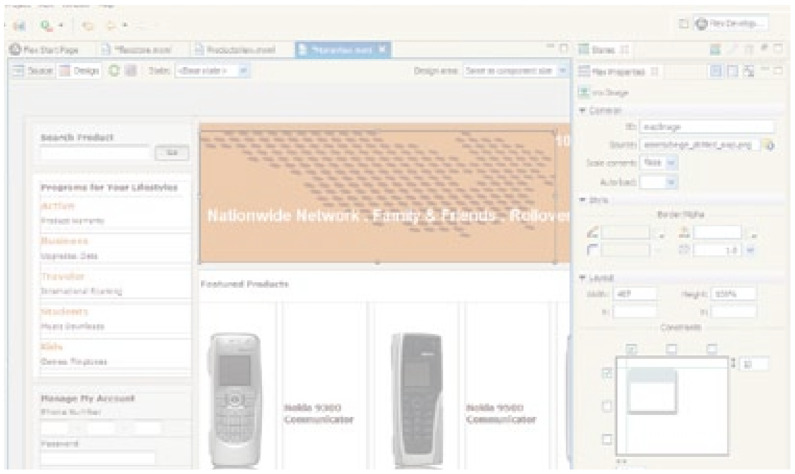


Figure 9: Flex Builder design view.

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.

Also, because Flex Builder is built on Eclipse, developers can debug Java and Flex applications side by side using the Eclipse Java debugger together with the Flex ActionScript debugger.

Extensibility

Because Flex Builder is built on the Eclipse platform, developers can easily take advantage of hundreds of commercial and open source plug-ins available for Eclipse to provide additional functionality. For example, plug-ins are available for all of the leading source control systems, as well as common programming languages like C#, C++, and Java, and many of the lifecycle tool vendors are now delivering tools as Eclipse plug-ins.

Flex Builder also provides an extensibility API, enabling organizations to build custom actions that automate team tasks (such as integrating with an automated test suite) or to build custom plug-ins. Developers can share plug-ins for Flex Builder through the Flex Developer Center at *www.adobe.com/devnet/flex*.

Flex target applications

While countless types of applications can be enhanced through RIA technology, Flex is particularly well suited for applications with a distinct set of characteristics, including those that:

- Automate a multistep process, particularly where the steps are nonlinear or recursive
- Combine graphical or multimedia content with data and application logic, particularly where users must interact with data or media locally
- Require server push or access to real-time streaming data, such as operational data or stock quote information
- Must operate in a disconnected fashion, for either short or extended periods of time
- Can benefit from complex client-side validation, such as logic that depends on previous user entries or sophisticated validation logic
- Involve large data sets, particularly where client-side data manipulation is important

While many applications embody some or all of these characteristics, they manifest themselves most frequently in the following use cases.

Interactive data visualization

Interactive dashboards are designed to allow users to quickly assess a situation through data visualization, enabling better decision-making and more rapid responses to business change. The extensible charting components available with Flex enable developers to assemble everything from conventional pie and bar charts to highly customized data visualization solutions. Moreover, because the charts are drawn locally using Flash Player's built-in vector graphics rendering features, users can quickly customize the chart or filter the data set to investigate a trend or drill down into a particular detail.

For example, SAP is using Flex as the user interface technology for a new line of analytics applications being delivered to customers in 2006. These applications provide a rich visual interface with SAP Business Warehouse data, enabling business users to analyze trends, monitor operational status, and make better decisions.

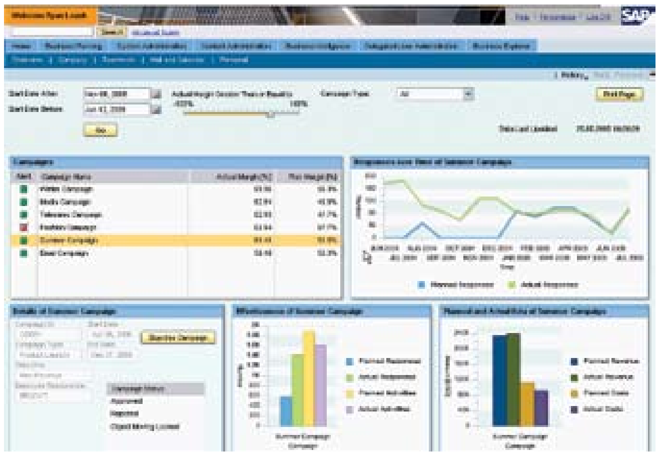


Figure 10: An example of an SAP analytics dashboard.

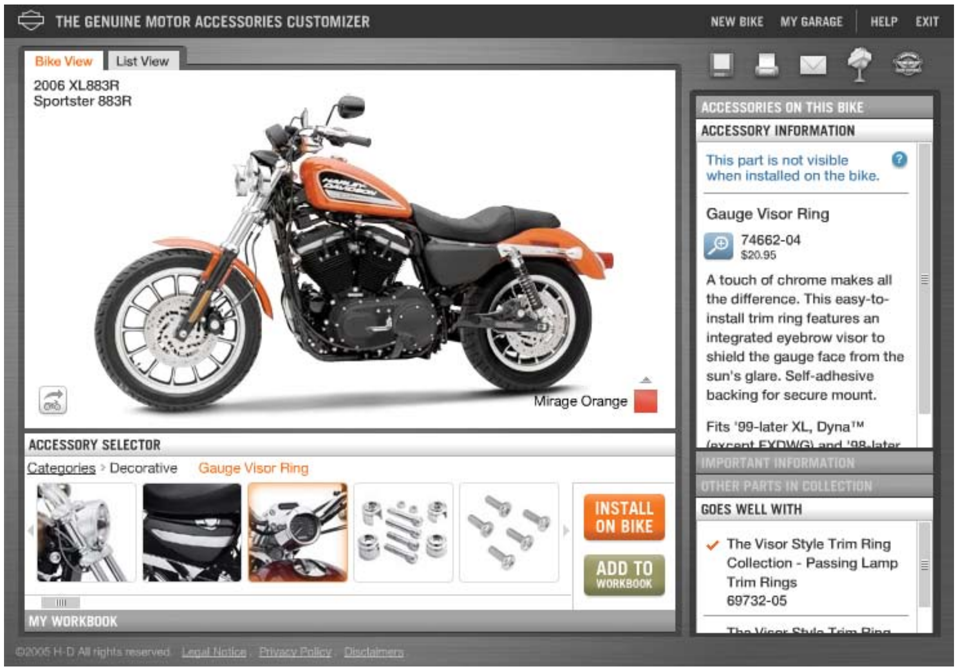
When linked with the data service architecture, Flex can enable a new class of data visualization solution. Rich, multidimensional charts can be bound to real-time data feeds so users always see the latest information. In addition, developers can use the Flex Message Service to enable real-time collaboration between users analyzing the same data set.

Product configuration and selection

E-commerce continues to grow in both the business-to-consumer and business-to-business sectors. However, as the number and complexity of products being sold online increase, customers need additional tools to help them quickly locate the product or service they want and configure it to their particular needs.

Flex based applications can help users sort through hundreds or even thousands of available products by enabling client-side data filtering and by providing immediate visual feedback based on their input. In addition, with native support for graphics rendering and manipulation, Flex and Flash Player enable organizations to deliver product configurators that show users a visual representation of the options they have chosen.

For example, Harley Davidson is using Flex to allow customers to configure their own motorcycle online. As users select a model, options, and colors, they can immediately preview the bike's appearance and the total cost of their selections.



This part is not visible when installed on the bike.

Gauge Visor Ring

74662-04
$20.95

A touch of chrome makes all the difference. This easy-to-install trim ring features an integrated eyebrow visor to shield the gauge face from the sun's glare. Self-adhesive backing for secure mount.

Fits '99-later XL, Dyna™ (except FXDWG) and '98-later

✓ The Visor Style Trim Ring Collection - Passing Lamp Trim Rings 69732-05

Figure 11: Harley Davidson motorcycle configurator built in Flex.

Process integration

As businesses expose more of their core business processes through services, there are more opportunities to deliver composite applications that present information or tasks in a role-specific or task-specific user interface. By combining native support for SOA with the ability to deliver a desktop-like experience in the browser, Flex is particularly well suited to these process integration applications.

An RIA created using Flex can combine data from multiple back-end systems into a single, integrated user interface that automates employee "onboarding" or helps customer service representatives navigate through information stored in disparate customer databases.

For example, Centive has used Flex as the user interface for its sales incentive management system, giving salespeople a cross-platform, easy-to-use application that allows them to easily track commissions through a set of rich visualizations.

Guided self-service

Guided self-service applications give users relevant information in the context of the transactions they are trying to complete. They can also provide in-context prompting through audio and video integration, allowing for just-in-time delivery of multimedia help or real-time collaboration between customers and representatives.

For example, a guided self-service application can walk a customer through processes such as bill pay or new account enrollment in an online banking application, new service selection for cable or phone customers, and quote generation. This increases the effectiveness and ROI in online customer self-service infrastructure and reduces the need for customers to engage a call center representative to answer a question or solve a problem. When customers need help, they can click a button to view multimedia help in the context of the application or contact a representative through a chat conversation. They can then watch the representative's mouse driving the application onscreen and solving the problem immediately. These customer interactions not only save money on customer service operations, but they also increase revenues due to lower process abandonment rates.

Internally, guided self-service applications can help employees sign up for or modify their benefits enrollments. They can also help customer service representatives navigate through complex service offerings or the latest specials they can present to customers.

Flex and other technologies

Flex and Ajax

Ajax is an acronym that stands for Asynchronous JavaScript and XML. The term was coined to describe the use of browser technologies to deliver RIAs. Ajax is associated with a set of application design patterns as well as a variety of open source projects and commercial products.

Flex and Ajax both promote an architecture that enables applications to take greater advantage of the client runtime to provide richer application functionality. As a result, the approaches used to expose business logic to Ajax clients (web services, REST APIs, and so on) are similar to those used in Flex applications.

While it is possible to develop rich clients using only client-side JavaScript and HTML (as many Ajax vendors promote), Flash Player provides additional capabilities not available in HTML, including a high-performance, just-in-time compiled execution engine, integrated APIs for graphics manipulation and vector drawing, and the robust, real-time messaging and integration services provided by Flex Data Services.

Also, because Flash Player is integrated with the browser runtime environment, developers can easily deliver applications that combine user interface logic written in JavaScript with components or entire applications written in Flex. For example, Google is taking advantage of both JavaScript and Flash as part of its Google Finance site. As shown in Figure 12, the application uses common Ajax techniques to update news stories and highlight news items, while the interactive chart takes advantage of the Flash Player runtime to draw a rich data visualization and show the connections between breaking news and changing stock price.

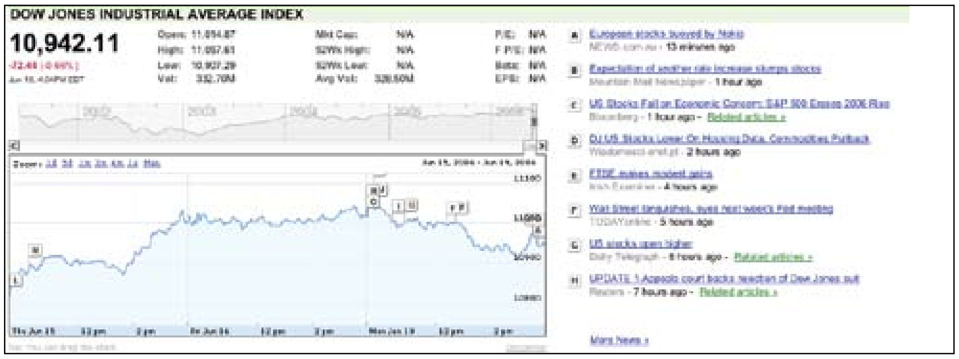


Figure 12: Google Finance combines the strengths of Ajax and the Flash runtime.

Basic interoperability between the Flash Player virtual machine and the browser's JavaScript engine and document object model is provided through the Flash Player External API, which enables bidirectional communication between JavaScript and ActionScript. To further facilitate this type of development, Adobe has released the Flex-Ajax Bridge (FABridge) library, which automatically exposes the public data and methods within a Flex application to the JavaScript engine and vice versa. This enables developers to easily integrate Flex applications with existing sites as well as to deliver new applications that combine Ajax libraries such as Yahoo Widgets with applications or components created in Flex.

For more information on the FABridge, visit Adobe Labs (*http://labs.adobe.com/wiki/index.php/Flex_Framework:FABridge*).

Flex and portals

Portals facilitate the aggregation of web content and applications in an integrated user interface. While many portlets are rendered in HTML, developers can use Flex to render the user interface of portlets and greatly improve the user experience within a portal. In addition, using Flex Data Services, developers can integrate the Flex application with existing security profiles and business logic running in the portal environment or within other applications running at the server tier.

JSR 168 created a common method to expose web applications as portlets so that they can be easily aggregated into a portal. Much of the pain of aggregating disparate web applications lies in dealing with application state. Flex applications avoid this pain because the state of a Flex application primarily lives on the client. This also simplifies the process of exposing Flex applications as remote portlets.

To integrate Flex with a portal environment, developers will generally create a JSP, HTML, and/or Web Services for Remote Portlets (WSRP) wrapper that includes the compiled Flex application. When the portlet is requested, the compiled Flex application is delivered to the user and included in the portal page. Either the Flex application can run as a small window within a page that contains multiple portlets, or it can be launched as a full-screen application once the user selects it from the navigation bar.

The original portlet spec (JSR 168) was primarily about aggregation, but it failed to address issues like interportlet communication, which becomes particularly relevant to rich client applications built with Flex or Ajax. While there are no issues with using the existing WSRP standard for remote Flex portlets or with using existing vendor-specific interportlet communication APIs inside Flex applications, JSR 286 was begun to standardize interportlet communication and to remedy other shortcomings of JSR 168. Adobe is participating in JSR 286 as an expert group member to help ensure better support for client-side state and asynchronous requests.

For more information on using Flex in a portal environment, visit *www.adobe.com/devnet/flex/articles/flex_portals.html.*

Flex and the Microsoft .NET platform

Flex applications can deliver a rich user interface for back-end systems implemented on Microsoft's .NET platform. Since application clients built in Flex are server agnostic, they can communicate with web services or HTTP services built with ASP.NET or C# just as easily as they can with systems implemented in Java, ColdFusion, or other technologies.

Additionally, while Flex Data Services is implemented in Java, it can be deployed with a .NET environment. As illustrated in Figure 13, the destinations exposed by Flex Data Services can be composed from multiple services built in .NET technologies.

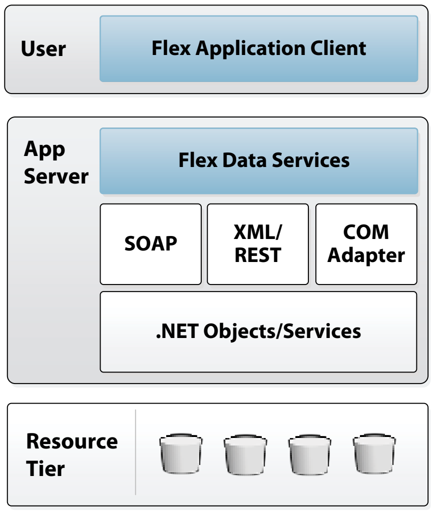


Figure 13: Flex in a .NET environment.

Moreover, using third-party adapters such as J-Integra for .NET, developers can easily expose .NET objects or APIs as services within a Flex environment.

By providing a cross-platform runtime environment for rich clients, Adobe Flex complements the Microsoft .NET architecture, enabling customers with heavy investments in Microsoft systems to reach beyond those platforms and deliver applications that provide a high degree of performance and usability, regardless of the operating system on the client.

Additional resources

Developer resources

In addition to extensive product documentation, Adobe provides a rich set of developer resources to support Flex development. These are accessible from:

• Flex Developer Center: *www.adobe.com/devnet/flex*

• Flex Support Center: *www.adobe.com/support/flex*

• Adobe Online Forums: *www.adobe.com/support/forums*

Security

For detailed information on security in Flex applications, consult the Security Topic Center at *www.adobe.com/devnet/security*.

Case studies and sample applications

To read about companies that have deployed Flex applications, visit *www.adobe.com/products/flex/customers*.

Better by Adobe.™

Adobe®

Adobe Systems Incorporated
345 Park Avenue
San Jose, CA 95110-2704
USA
www.adobe.com

95006794 6/06