



***Macromedia MX:
Components and Web
Services***
by Jeremy Allaire

April 2002

Copyright © 2002 Macromedia, Inc. All rights reserved.

The information contained in this document represents the current view of Macromedia on the issue discussed as of the date of publication. Because Macromedia must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Macromedia, and Macromedia cannot guarantee the accuracy of any information presented after the date of publication.

This white paper is for information purposes only. MACROMEDIA MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.

Macromedia may have patents, patent applications, trademark, copyright or other intellectual property rights covering the subject matter of this document. Except as expressly provided in any written license agreement from Macromedia, the furnishing of this document does not give you any license to these patents, trademarks, copyrights or other intellectual property.

{INSERT OUR MARKS IN THE DOCUMENT} are either trademarks or registered trademarks of Macromedia, Inc. in the United States and/or other countries. The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Macromedia, Inc.
600 Townsend Street
San Francisco, CA 94103
415-252-2000

Contents

Challenges in Web Architecture	1
The Emergence of Rich Clients	2
The Emergence of Web Services	3
The Reality of Web Applications Today	5
Web Programming Models Must Evolve.....	5
The Macromedia MX Solution	6
Power of ColdFusion Components	6
Creating a Simple Model	6
ColdFusion Components and Rich Clients	10
ColdFusion Components and Web Services	12
Dreamweaver MX and Web Services	13
Conclusion	14

The convergence of rich clients, web services and the need for a more scalable web application development and deployment model have driven Macromedia to deliver components and web services as part of Macromedia MX. ColdFusion Components are a major shift in how server-side web applications can be built, empowering a wide range of RAD and script-level developers to access the power of object-based component development, rich client/server models and web services without the pain of complex frameworks. The Macromedia MX product family combines ColdFusion Components with the rich client capabilities of the Macromedia Flash Player, the development capabilities of Macromedia Dreamweaver MX and the openness and interoperability of SOAP and XML web services.

Web application architectures are evolving to better meet the needs of companies investing in the Internet. The emergence of rich clients and new devices and the promise of web services are pointing Internet application development technologies in new directions. With the release of the Macromedia MX family of products, Macromedia is introducing critical new technology to help developers take advantage of these trends and overcome limitations in their current approach to web development.

Challenges in Web Architecture

Increasingly, the basic architecture of the web is being challenged as corporations seek to derive more value and return on investment (ROI) out of the Internet. As web technologies have reached a plateau, users are increasingly finding that the primary model of web applications as collections of dynamically generated pages or documents is not meeting the performance or usability requirements for many applications. Applications are often cumbersome to use and perform slowly. They are complex to create because developers need to assemble dozens or even hundreds of files and scripts to run an online system.

While the web continues to deliver significant value as a content and application platform, the Internet end-user experience remains fragmented. People interact with the Internet through a disparate set of clients—HTML and web document browsers, e-mail clients, instant messaging clients, and a variety of streaming media players. The web's architecture was never built to support a deeper blending of these content, application and communication types, despite the increasing need for them to converge.

Web applications today are developed without significant structure. Often they consist of server-side dynamic pages with embedded script, presentation logic and data access logic. As a result, most of the business logic and data associated with these applications are locked up—unable to be shared inside and outside of corporations. In essence, web applications and data are kept in silos, limiting the ability of companies to share their information and business logic.

The Emergence of Rich Clients

In 2002, the Internet application model will change dramatically as designers and developers embrace rich clients, such as the Macromedia Flash Player, in order to improve Internet applications and content.

Here's what rich clients promise to do:

- **Provide an efficient, high-performance runtime for executing code, content and communications.** The assumption here is that the end-user experience of HTML-based web applications suffers from a variety of performance-related challenges, ranging from the request-response page rendering model to the need to dynamically generate large blobs of text for transmission of simple data, the lack of client-side data storage, the inability to invoke and use remote business logic easily, and even the basic graphics model of HTML. Next-generation rich clients should combine code, content and communications logic into an extremely high-performance runtime model.
- **Integrate content, communications and application interfaces into a common environment.** The end-user experience of the Internet today is fragmented into the HTML browser for textual content and basic application interfaces, multiple messaging clients for communications, and multiple media players for handling audio, video and other forms of media. Rich clients need to provide deep integration for all of these types of interaction, enabling the development of applications that fulfill the promise of integrated media, communications and application logic.
- **Provide powerful and extensible object models for interactivity.** While web browsers have progressed much in terms of their support for interactivity through the Document Object Model (DOM) and JavaScript or DHTML, they still lack the richness needed for building serious applications. Rich clients need to provide a powerful, object-oriented model for applications and events, integrating user interface, communications and system-level services into a common object model that can be used by developers of varying skill levels.
- **Enable rapid application development through components and re-use.** Rich clients should support powerful component-based development, enabling both third-party and corporate developers to create and reuse visual components easily to accelerate development and give junior developers access to complex functionality without difficulty. These components should easily integrate into the design-time environment for easy development, and mirror components and services hosted on application servers.
- **Enable the use of web services and data services provided by application servers.** The promise of rich clients is also the ability to separate presentation logic and user interfaces cleanly from application logic hosted on the network. Rich clients should provide a model for easily using remote services provided by back-end components, whether hosted in an application server or accessed as XML web services.

- **Embrace connected and disconnected clients.** While many users are accustomed to working online in a web browser, the reality is that most applications would benefit from offline usability on occasionally connected devices such as PDAs and laptops. Likewise, many applications require constant connections and need support for two-way, notification-based communications. Rich clients should enable these types of applications to be built and deployed easily.
- **Enable easy deployment on multiple platforms and devices.** Internet applications are all about reach. The promise of the Internet is one of content and applications anywhere, no matter the computer platform or device. Rich clients must embrace and support all popular desktop operating systems, as well as the broadest range of emerging device platforms such as smart phones, PDAs, set-top boxes, game consoles and new Internet appliances.

At the center of Macromedia's strategy for next-generation rich clients is the Macromedia Flash Player. Designers and developers around the world are already pushing the limits of expression, creativity and application richness with the Macromedia Flash platform. Over the past years, Macromedia Flash has evolved from a powerful rich media technology to a powerful application development and deployment environment. We're taking it even further with Macromedia Flash MX, which aims to make the Macromedia Flash Player the rich client of choice for Internet applications in 2002.

While Macromedia Flash MX offers tremendous opportunities to companies seeking to increase the value of their Internet investments, it also involves the introduction of a new architecture for Internet applications that is a hybrid of client/server and the web. Rich clients clearly define a presentation layer that is similar to a client on a desktop computer, which in turn requires server-side code for business logic and data access, or access to remote services somewhere else on the Internet. This new model creates a significant challenge to the current model for how server-side logic and data are exposed to Internet applications.

The Emergence of Web Services

While rich clients help to address some of the limitations in the web architecture from an end-user perspective, web services will usher in a major shift in how applications share their behavior or logic and data with other applications.

Web services provide a set of protocols that allow applications to expose their functionality and data to other applications over the Internet. They essentially are software components in the network. Like prior protocols for distributed computing—such as DCOM, DCE, CORBA and Java RMI—web services provide a framework for remotely invoking an object and exchanging data.

Web services are actually quite simple. They provide a language and platform-independent syntax for exchanging complex data using messages. The internals of web services are implemented using XML, making it easy for any platform to support this technology.

Fundamentally, web services are comprised of two key standards: Simple Object Access Protocol (SOAP) and Web Services Description Language (WSDL). SOAP is the protocol used for sending messages and data between applications. WSDL provides syntax to describe the functionality of a web service. This could be used be a tool to generate the code necessary to access that web service, or provide code-hinting tools to a developer using that service. Both of these specifications are broadly supported by industry leaders including IBM, Microsoft, Sun, BEA, Oracle and others—as well as Macromedia. There are many other standards that complement or extend these basic technologies.

Web services are radical in their implications because for the first time there will be a standard way to connect any application in the world to another. While HTML and HTTP have unlocked content and data to the world, web services will unlock applications and business logic to the world. This change promises to change the economy of software, as more and more companies can rely on services in the network to assemble and create business value.

Web services are also a natural counterpart to rich clients. If rich clients provide a model for a clear and powerful presentation layer that runs on client computers and devices, web services help provide a model for logic and data to feed them.

Macromedia is deeply committed to providing a complete platform for developing and deploying web services. With the release of Dreamweaver MX and ColdFusion MX, Macromedia is introducing a web services engine that supports SOAP and WSDL, and runs in the core J2EE app server container, as well as a set of integrated development tools for building and using web services. The ColdFusion MX web services engine is built on the Apache Axis project, formed by an alliance between Macromedia, IBM and the Apache community to deliver an open-source web services platform. This web services engine will enable application developers to expose their back-end functionality easily as web services, and also to use web services provided by others.

To complement this web services runtime, Dreamweaver MX includes development tools for building and using web services.

Like rich clients, web services require significant changes in how applications are created. Logic and data need to be separated into services that can be used by other programs—without even knowing what that end-program is doing.

The Reality of Web Applications Today

The constraints of the web's architecture have, in some respects, driven web application developers into a corner. Both the page-based development model and the wide availability of rapid application development (RAD) scripting languages such as CFML, JavaScript, VBScript, PHP and others, have contributed to developers creating applications where the presentation logic, business logic and data are all intermingled in collections of server scripts and templates.

This has been a double-edged sword. On the one hand, developing these thin-client web applications has introduced rapid application development to huge numbers of people. Learning HTML, simple scripting and databases has unleashed a wave of applications and dynamic sites into corporations around the world. On the other hand, even moderately complex business systems built with this approach often result in extremely complex code that is difficult to integrate with other systems. These applications do not scale as well, and make team development and code reuse nearly impossible.

Over the past few years, a number of vendors have tried to introduce more sophisticated models for web applications. These involved writing business logic components using standards such as Enterprise JavaBeans and COM+. While these approaches did not attempt to solve the problem of creating web services or exposing functionality to rich clients, they did attempt to introduce more manageable and scalable models for development and deployment of web applications.

For the most part, these approaches have failed to gain traction with developers. Some of this is because these frameworks were very complex, often requiring sophisticated object-oriented system programmers to implement them. Another reason is that the RAD imperative still pervades IT organizations—where rapid development and quick ROI count more than hard-core architecture. Generally, however, these approaches have failed because they do not meet the skills and complexity threshold of the vast majority of web application developers in the market today.

While these models may not be as widely used as had been originally anticipated, the architectures are sound and the need to take advantage of those at a higher-level of abstraction appears to be critical to the next wave of Internet applications.

Web Programming Models Must Evolve

All of these trends and forces are driving us towards a shift in web programming models. We believe that next-generation Rich Internet Applications require a solution that:

- Supports an easy separation of presentation from logic and data.
- Makes it easy to expose server-side logic and data to rich clients and devices.
- Enables rapid creation and use of web services.

- Supports a richer, more scalable development and deployment model.
- Meets the “Keep It Simple, Stupid” requirement and works for the mass majority of developers.

The Macromedia MX Solution

With the introduction of the Macromedia MX family of products, we’re introducing a powerful programming model and runtime to create components and web services, and a range of new development tools to build and use these components.

Power of ColdFusion Components

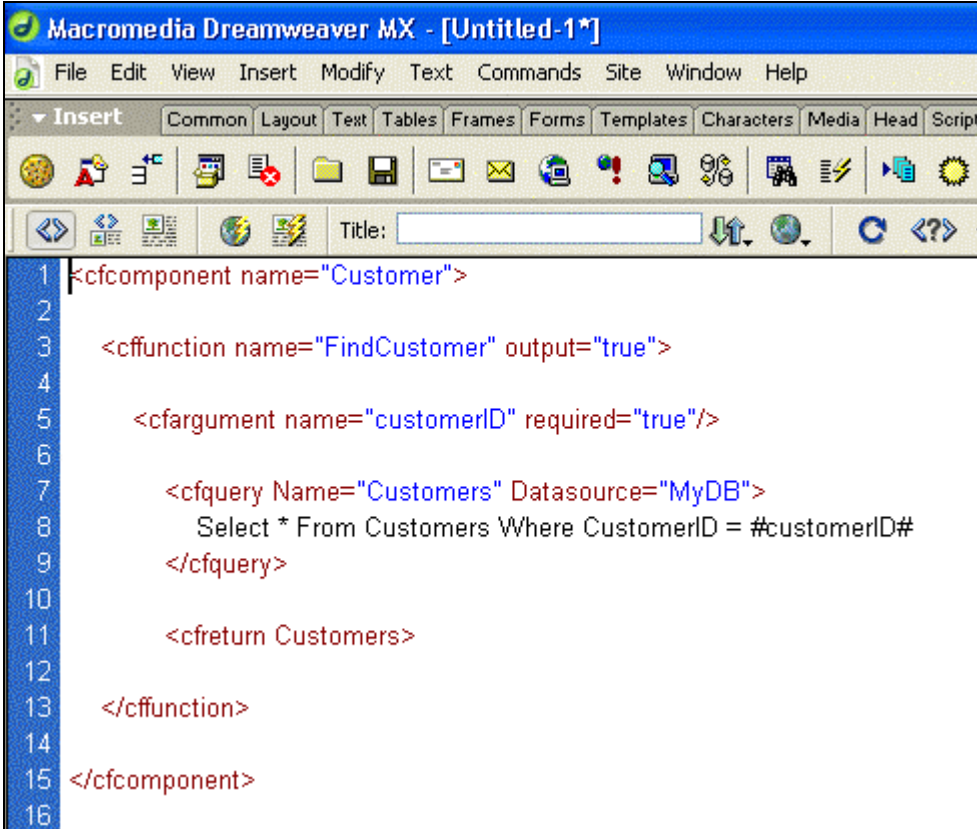
Over the past year Macromedia has been working on a new technology aimed at enabling application developers to create component-based applications that can be used from rich clients as web services, and enable a more scalable development and deployment model. ColdFusion Components (CFCs) provide a simple yet powerful model for the rapid development of web components.

Creating a Simple Model

CFCs allow developers who aren’t necessarily sophisticated programmers to create object-based components easily. Using a tag-based syntax to define and encapsulate a component’s behavior, this model is immediately approachable to developers familiar with scripting languages like CFML, JavaScript, PHP and others.

The goal of ColdFusion Components was not to introduce a deeply complex framework with unlimited power but, instead, to yield some of the power of object-based component development to an audience and architecture that desperately needed it—web applications.

The following figure shows a simple example of a component that accesses a customer database (Figure 1). This simple example provides a component with one method or function, called FindCustomer, which queries the customer database to find the customer information.

A screenshot of the Macromedia Dreamweaver MX software interface. The title bar reads "Macromedia Dreamweaver MX - [Untitled-1*]". The menu bar includes "File", "Edit", "View", "Insert", "Modify", "Text", "Commands", "Site", "Window", and "Help". The "Insert" menu is expanded, showing sub-menus: "Common", "Layout", "Text", "Tables", "Frames", "Forms", "Templates", "Characters", "Media", "Head", and "Script". The main workspace contains ColdFusion code for a component named "Customer". The code is as follows:

```
1 <cfcomponent name="Customer">
2
3   <cffunction name="FindCustomer" output="true">
4
5     <cfargument name="customerID" required="true"/>
6
7     <cfquery Name="Customers" Datasource="MyDB">
8       Select * From Customers Where CustomerID = #customerID#
9     </cfquery>
10
11     <cfreturn Customers>
12
13   </cffunction>
14
15 </cfcomponent>
16
```

Figure 1: Simple example of a component accessing a customer database.

Dreamweaver MX comes with a powerful coding environment, wizards for building CFCs, and a built-in ColdFusion Component Browser (also called the Create Component dialog box), which can introspect and browse components on a ColdFusion MX Server (see Figure 2). Once created, you can use a component from a variety of contexts, which ultimately demonstrates the power of CFCs.

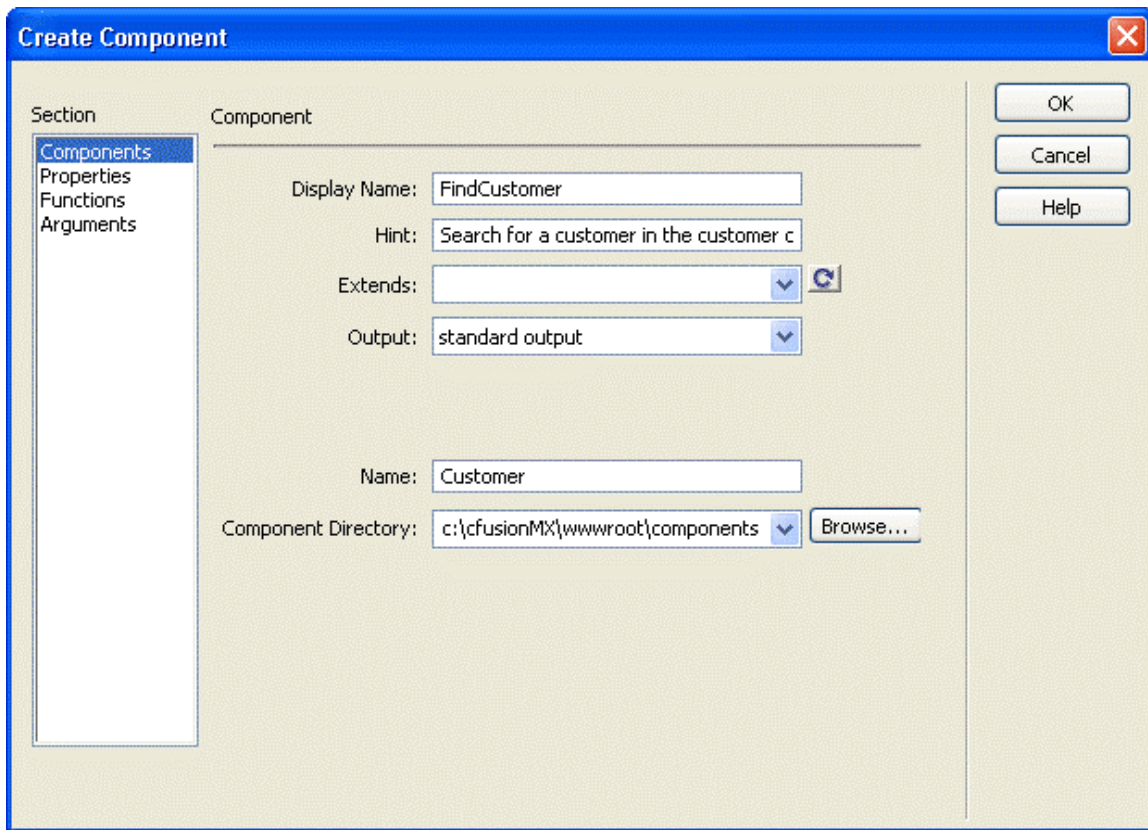


Figure 2: The ColdFusion Component Browser (also called the Create Component dialog box).

Within a local application running on the server, CFCs enable developers to package functionality in a reusable way, enabling better structured code, easier team development, and more maintainable systems. They also provide a means for developers to encapsulate complex functionality for use by other developers. While the example in Figure 1 is trivial, the component method could have included a very complex set of processes for accessing customer data.

One way to use components is in a declarative way by using the simple `<cfinvoke>` tag:

```
<cfinvoke component="Customer"
  method="FindCustomer"
  customerID="form.customerID"
  result="CustomerInfo" />
<cfoutput>
  #CustomerInfo.FirstName#
  #CustomerInfo.LastName#<br>
</cfoutput>
```

This example illustrates how to invoke a component in a declarative manner, passing in the method and arguments needed to get some result and then outputting that result to the page. Developers can drag and drop CFC methods from within the Dreamweaver MX Components Window (sometimes referred to as the CFC Browser) into their code view to automatically generate the appropriate code (see Figure 3).

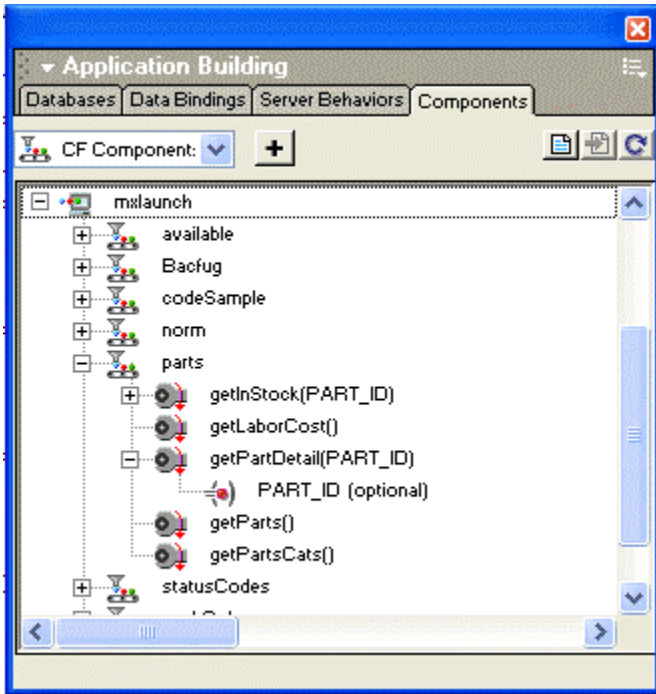


Figure 3: Dragging and dropping CFC methods from within the CFC Browser (also called the Components Window).

In addition, because CFCs are defined in an object-based syntax, developers who are comfortable with object-scripting using CFScript or Server-Side ActionScript (SSA) can also access the component as an object:

```
<cfscript>
Customer = CreateObject("component", "customer");
    Customer.FindCustomer(form.customerID,
    customerInfo);
</cfscript>
<cfoutput>
    #CustomerInfo.FirstName#
    #CustomerInfo.LastName#<br>
</cfoutput>
```

This example creates an instance of the customer component in a local object called "customer," calls the "findCustomer" method on that component with the required arguments and then outputs the results into an HTML page.

ColdFusion Components are built on a new set of CFML tags for describing a component and its internal structure. These new CFML tags are dynamically compiled into Java classes and executed in the hosting J2EE application server. This compilation takes place the first time the component is invoked; successive invocations are executed against the compiled Java classes. This model gives developers the power of the J2EE platform with the simplicity of a rapid development and scripting model such as ColdFusion.

ColdFusion Components and Rich Clients

With the emergence of rich client technologies, such as the Macromedia Flash Player, CFCs become a powerful technology for easily exposing your server-side assets to these clients. One of the most powerful capabilities of CFCs is their ability to use a component either locally as a logical component in an application (such as the example in the previous section) or by a remote client, such as the Macromedia Flash Player.

Macromedia Flash MX introduces a new client/server application development model for delivering Rich Internet Applications called Flash Remoting. This relies on a services-based programming model where the client accesses and uses services hosted in the network on application servers or on remote web services. To deliver this, we created a new highly optimized binary protocol for accessing objects and exchanging data between the Macromedia Flash Player and remote services. This HTTP-based protocol is called the Action Message Format (AMF).

Using ColdFusion Components, when a developer simply marks the access attribute in a function or method on their component as “remote,” that component can immediately be used as an object within ActionScript. For example, in a rich client application you might have a Find Customer button. When clicked, it calls the server to find customer data and returns the information as a recordset that can be bound to a specific display area. With Macromedia Flash, this becomes possible without refreshing the screen and requires only a very small interaction with the remote server. To Macromedia Flash, the remote component is treated just like a local ActionScript object. For example, the following ActionScript code could be used to handle a button click that calls a remote server:

```
Function onClick(FindButton) {  
    Customer.FindCustomer(customerID);  
}
```

In this example, when the FindButton is clicked an object called “Customer” is invoked using the FindCustomer method passing in the value of a form field. Under the hood, the Customer object is a proxy to the Customer component on the server (this is established in a setup script at the beginning of your Macromedia Flash application), and the framework handles marshalling the data between ActionScript and ColdFusion on the server, as well as all of the network communications. On the server, the Flash Remoting service included in ColdFusion MX handles the request and dispatches the request to the appropriate component code, returning the result as native ActionScript.

That same component can be accessed and used from Macromedia Flash whether the application is running on a full multimedia desktop computer, or on a Flash-enabled PDA, set-top box, smart phone, game console or other Internet appliance.

Additionally, with Macromedia Flash MX we’ve introduced a powerful new component model on the client for both visual and logical user interface components. Developers can create Macromedia Flash Components that provide well-defined methods, properties and events. Other developers can drag and drop these visual components into their Macromedia Flash applications.

CFCs provide a great server-side complement to these Macromedia Flash Components, where a developer could create a powerful client/server component with both a visual and logical aspect. For example, a company could create an employee directory finder that includes a Macromedia Flash Component for the user interface and a ColdFusion Component for the actual logic that accesses the corporate directory through Lightweight Directory Access Protocol (LDAP) or some other means (see Figure 4).

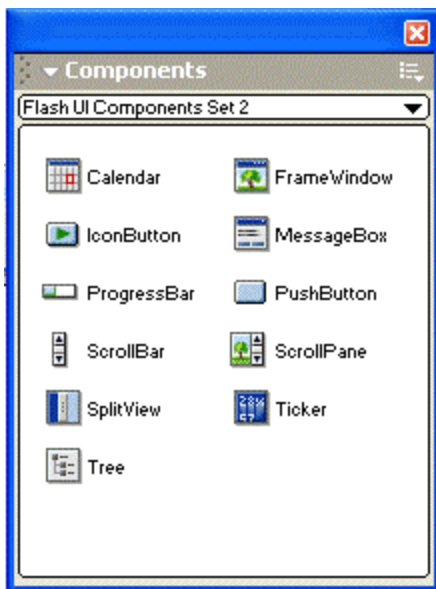


Figure 4: *Employee directory finder that includes a Macromedia Flash Component for the UI and a CFC for the logic.*

ColdFusion Components and Web Services

Just as CFCs can be easily exposed as Macromedia Flash services, they can also be exposed as SOAP-accessible web services. In fact, the native model for Remote CFCs, or accessing and using CFCs over the network—even over an intranet or LAN—is SOAP and web services.

Using the simple example of the customer search application in the previous section, we could change the code as follows:

```
<cfcomponent name="Customer">
  <cffunction name="FindCustomer" output="true"
    access="remote">
    <cfargument name="customerID" required="true"/>
    <cfquery Name="Customers" Datasource="MyDB">
      Select * From Customers Where CustomerID =
        #customerID#
    </cfquery>
    <cfreturn Customers>
  </cffunction>
</cfcomponent>
```

Here we've specified that the findCustomer function can be accessed remotely by SOAP or Macromedia Flash.

Web services use WSDL to describe a web service interface or the component capabilities. Rather than hand-code these files and manually deploy them, use CFCs to generate their WSDL definitions dynamically through a URL argument to the component. For example, the following URL would return the WSDL definition for the Customer component:

<http://localhost/components/customer.cfc?wsdl>

Any client or web service consumer can access that URL to get a definition of which functions of that component are available to call.

At runtime when the ColdFusion Web Services Engine encounters an incoming SOAP request, it already knows that the request is for a CFC and dispatches the request with any input arguments to the CFC. It then returns the results to the client with SOAP-encoded data automatically.

Dreamweaver MX and Web Services

With the introduction of Dreamweaver MX, Macromedia ushers in a set of powerful tools for enabling designers and developers to integrate components and web services into their Internet applications. In addition to the tools described above for building web services using ColdFusion Components, Dreamweaver MX includes tools for using web services.

The Dreamweaver MX web services browser allows developers to add web services easily using WSDL URLs, or find available web services in UDDI repositories. Using this basic information, Dreamweaver MX creates a view of the available interfaces or APIs on any web services, and automatically generates the appropriate client-side SOAP libraries using ColdFusion MX, ASP.NET or JSP (see Figure 5).

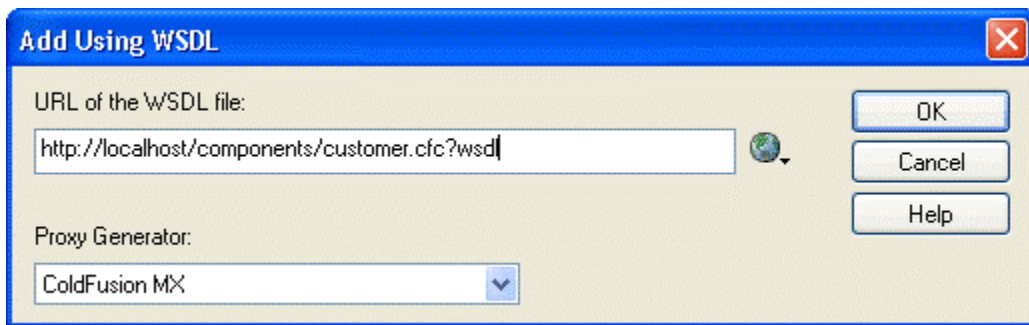


Figure 5: Adding web services with the Dreamweaver MX web services browser (also called the Web Services Chooser dialog box).

Once a developer has defined a web service using Dreamweaver MX, he or she can easily drag and drop the web service methods into the page and automatically get simple client code for using that web service (see Figure 6).

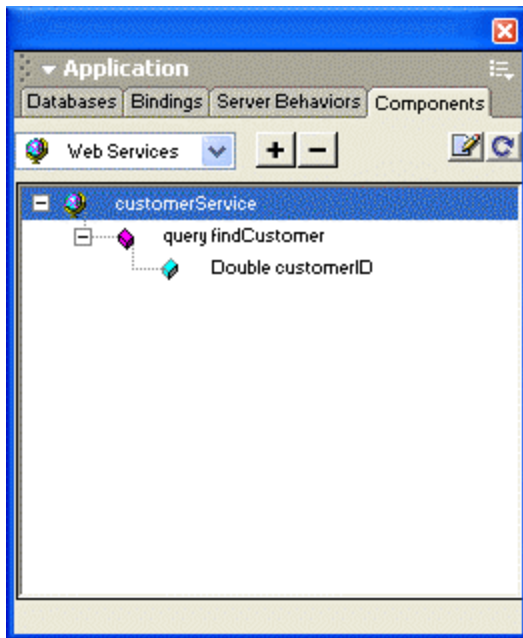


Figure 6: *Dragging and dropping web service methods into the page to get simple client code for using the web service.*

Conclusion

The convergence of rich clients, web services and the need for a more scalable web application development and deployment model have driven Macromedia to deliver components and web services as part of Macromedia MX. ColdFusion Components are a major shift in how server-side web applications can be built, empowering a wide range of RAD and script-level developers to access the power of object-based component development, rich client/server models and web services without the pain of complex frameworks. The Macromedia MX product family combines CFCs with the rich client capabilities of the Macromedia Flash Player, the development capabilities of Dreamweaver MX and the openness and interoperability of SOAP and XML web services.