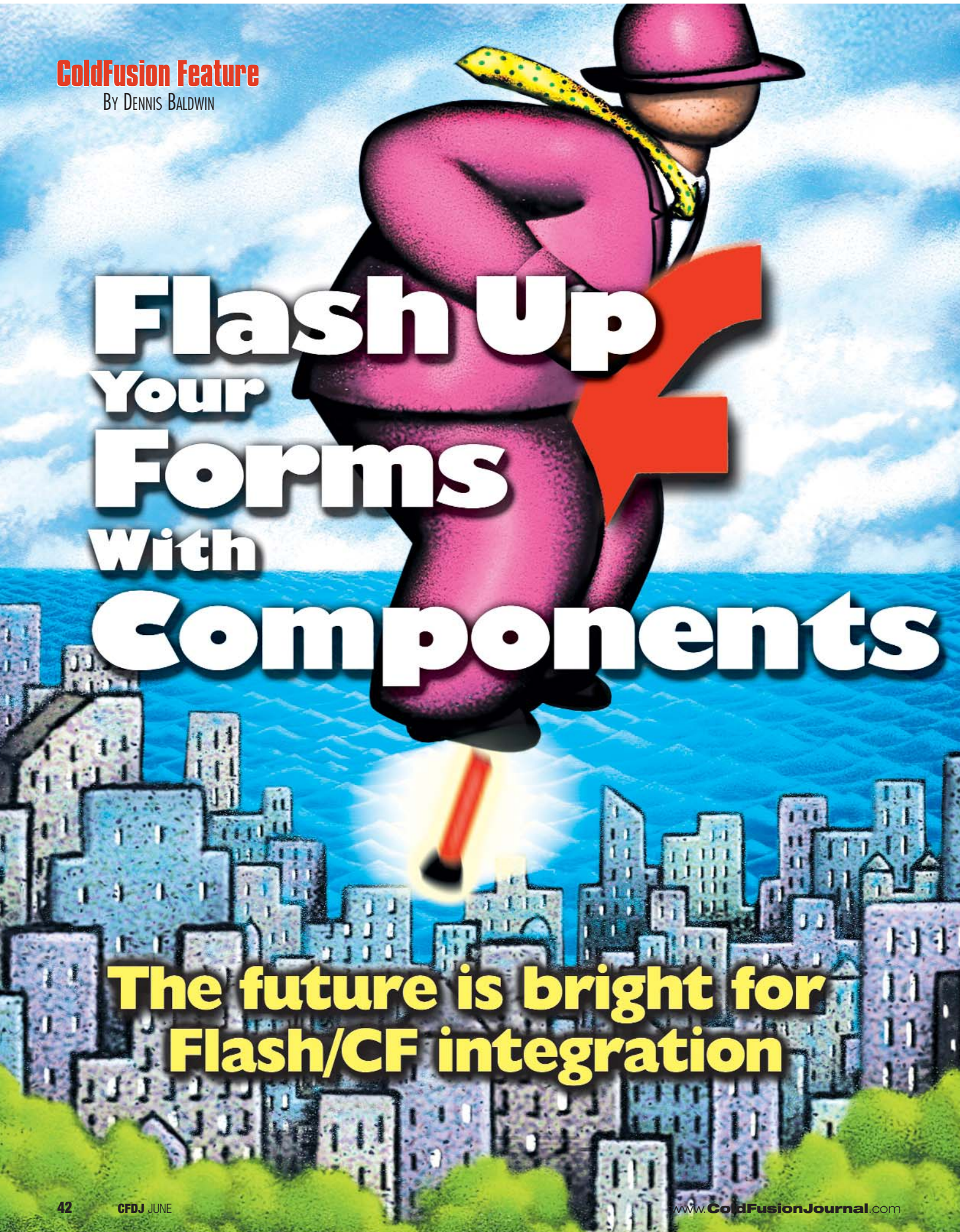


**ColdFusion Feature**

By DENNIS BALDWIN



# Flash Up Your Forms With Components

**The future is bright for  
Flash/CF integration**

# F LASH MX HAS HIT THE STREETS, and some amazing applications are already under development.

Macromedia has gone above and beyond by providing us with this fully functional Web development tool. Now with drag-and-drop components such as scrollbars, list boxes, calendars, and much more, developers can harness the power of reusable code. This will dramatically enhance the development process as well as cut down the time it takes to maintain and debug Flash applications.

No longer “just an animation tool,” as it has been labeled in the past, the new Flash has given rise to more full-blown Web applications that utilize Flash as the front end with a powerful application server like ColdFusion as the back end. Long gone (I’m hoping) are the days of useless intros, so let’s make a conscious effort to start pushing Flash MX application development to its limits!

In Flash5 we were introduced to an incredible scripting environment that just keeps getting better. ActionScript allows the developer to harness the power of modular, object-oriented code that can control all aspects of a Flash movie. Reusable code is a staple in today’s development projects and will save time and frustration in the long run. We’ll take a look at a Flash application that utilizes components, gathers user input, and passes this information to be handled by CF. Let’s get started.

## Example Application

The purpose of this application is to become familiar with some new Flash MX components along with a few of their methods. Not only that, we’ll see how simple it is to send information back and forth between Flash and CF. Since forms are used on practically every Web site, this will be a good example of how components can be used to enhance development. Flash MX has several built-in components that we’ll get to know a little better.

The Macromedia development team has provided us with some handy form components that mimic HTML-based form elements. You’ll need to download and install the Flash UI Components Set 2 from the Macromedia Exchange (<http://dynamic.macromedia.com/bin/MM/exchange/main.jsp?product=flash>). Installing these components requires the Extension Manager, which can also be downloaded from the exchange.

The Flash form will gather information from the user and insert it into the database. The key to handling information provided by the user resides in the ActionScript code. Almost every event has a corresponding method, or handler, that gathers the data and places it into a data object, which is then sent to CF upon submission.



## Digging Into the Code

Open “form\_components fla” (this and other files referenced in this article are downloadable at [www.flashcfm.com/cfdj/form\\_components.zip](http://www.flashcfm.com/cfdj/form_components.zip)) and you’ll see the code in frame one of the AS layer. This is all the code necessary to run the application. The components were dragged and dropped onto the stage and assigned the corresponding methods listed in the AS code. The init method is the key to kick-starting the application by setting focus to the “first\_name” field, creating the data objects, initializing a few variables, and loading the states list from CF (see Listing 1).

The data object is simply a container that will gather data from the user and send it to the server once the form is submitted. The receive object will accept a response from the server and display whether the transfer was successful or not. The LoadVars object, new to Flash MX, adds increased functionality when sending and loading data, similar to the XML object. As with loading the states list from CF, an event handler that will fire once the data has finished loading can be specified. In our application the statesLoaded method is called and the data is handled accordingly.

```
function statesLoaded(success) {  
    if(success) {  
        var states = this.state.split(",");  
        state_mc.setDataProvider(states);  
    } else {  
        trace("error loading data");  
    }  
}
```

If the load is successful, the states are split into an array and used to populate the combo box component, (see Figure 1), which has been assigned an instance name of “state\_mc”. The setDataProvider method is available to all combo box components and provides an efficient way of populating the combo box with data.

To learn more about the methods of Flash MX components, be sure to view the ActionScript reference panel (Window > Reference). This panel comes in very handy when using components and learning how to interact with them.

The CF code in `get_states.cfm` is a simple database query that pulls the states from a table and places them in a comma-delimited list. Be sure to take a look at the states table in the `users.mdb` database. If a problem occurs when loading the states from CF, an error message is output to the debug window. It's a good idea to use a fully qualified URL, set in the "base\_href" variable, so the CF data can be parsed and loaded into Flash within the IDE. If a relative URL is used, your CF code will be loaded into Flash and won't be very helpful for testing and debugging your application. Otherwise you'll have to publish and test your movie through the browser every time, which can be an extremely monotonous process.

The application has been initialized – what next? We need to start gathering data from the user and building our data object to send to CF. The input text fields such as `first_name`, `last_name`, `city`, and `e-mail` simply sit on the stage and wait for user input. These field values will be set in our data object once the send button is pressed; we'll look into this shortly.

The calendar component is a powerful one and in our application serves as a means to let users specify their birth date. The calendar carries an instance name of `birthDate_mc` and is assigned a change handler of `getBirthDate`. Each time a selection is made on the calendar the `getBirthDate` handler is called (see Listing 2).

When `getBirthDate` is called, we grab the date value using the `getSelectedItem` method. This value is set in our "birth\_date" variable, and we use a little ActionScript to take care of the rest. Once we have the selected date, we need to format it in a manner compatible with CF. To do this we simply take advantage of the Flash Date object and create a new date from "birth\_date". Now that we've formatted "birth\_date" into a valid date object, we can call different methods of this object. Once these methods are called we can build our date string in the format of `YYYY-MM-DD`. After it's been built we set it in our `dataObj`, which will be sent to the server.

We'll look at the next three components at the same time since they access a similar method. The state (combo box), gender (radio button), and notify (checkbox) components use the `getValue` method to specify what selection the user has made. The methods in Listing 3 are assigned as change handlers to set the state, gender, and notification variables in our data object. To specify a

change handler, select the component on the stage and go to the parameters tab in the properties panel (see Figure 2).

The comments field, a multiline text field that is set to wrap, is assigned an instance name of "comments". The purpose of the instance name is to provide a way for the scrollbar component to reference the text field. Once the scrollbar is dragged onto the stage it needs to be assigned a target text field. As with all components, this is specified in the properties panel. After assigning this target (in our case it's "comments"), the scrollbar is ready to perform. Any text that goes beyond the viewable area will trigger the scrollbar, which has very advanced functionality. Play around with it and see if it doesn't remind you of a typical Windows scrollbar.

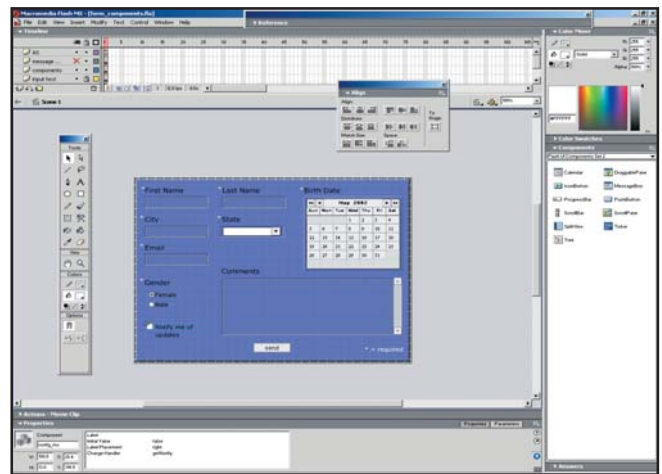


FIGURE 2: Properties panel with parameters set

So our user has completely filled out the form and clicks the send button – now what? The send button is actually a push-button component and is assigned the `sendData` method (see Listing 4) as its click handler.

Once the method is called, we assign our input text values to the data object. Since the data object was constructed as a `LoadVars` object, we have access to the `sendAndLoad` method, new to Flash MX. This method takes `dataObj` and sends it to the specified CF template, in our case "data\_insert.cfm", to be handled by the server. The data is received by CF as name value pairs and by default is sent via the "POST" method. We can access these variables in CF as `#form.first_name#`, `#form.last_name#`, and so on.

The second parameter of the `sendAndLoad` method is a result function that can be used to handle the data sent back from CF.

The receive object was initialized at the beginning of the application when "init()" was called. This object waits for the response from CF and calls the "receiveHandler()" method when loaded (see Listing 5).

If the user's information is recorded successfully, we display the message box component (also part of the Flash UI Components Set 2) with the corresponding message. The send push button is disabled, via the `setEnabled` method, to prevent the user from accidentally resubmitting the form. If there's an error, we display the message clip with an error message. The user can acknowledge the error, correct the form, and submit it again.

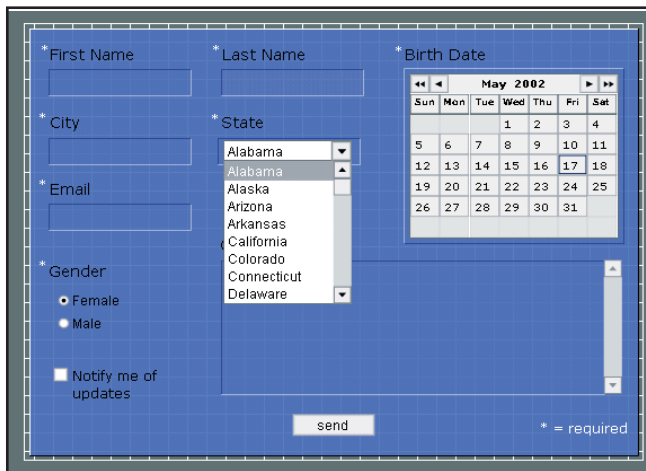


FIGURE 1: Application with the states drop-down fully populated

## Summing Up

Flash and ColdFusion integration may seem daunting at first, but with a little time and effort the knowledge you gain from building this basic form can be leveraged into building full-blown, integrated applications. Components are your friend, and they're here to save you time, money, and frustration. To develop this form from scratch, using components and writing the necessary code, could take from four to five hours. Without these components you'd be looking at three to four days to build a basic calendar...and you'd just be getting started! Take some time and get to know the component methods and how to access them. Learn how Flash and ColdFusion talk to each other

and challenge yourself to build on this application. Try adding e-mail functionality where the user receives an e-mail after the form is submitted. The sky really is the limit when building integrated applications, so enjoy.



### About the Author

Dennis Baldwin is the lead developer for Eternal Media, a Web/multimedia firm that provides technology solutions for ministries and nonprofit organizations ([www.eternal-media.com](http://www.eternal-media.com)). He also maintains a couple of online resources for Flash and ColdFusion developers at [www.flashcfm.com](http://www.flashcfm.com) and [www.devnx.com](http://www.devnx.com).



DENNIS@FLASHCFM.COM

#### Listing 1

```
// method to initialize the application
function init() {
    // url where the application is running - change this to
    // suit your needs
    base_href = "http://localhost/form_components/";
    // set focus to the first_name fields
    Selection.setFocus("first_name");
    // initialize the dataObj which will hold the data passed
    // to CF via the sendData method
    dataObj = new LoadVars();
    // initialize the receive object that will handle the
    // response from the server
    receiveData = new LoadVars();
    // once the response comes back the receive handler will
    // be triggered
    receiveData.onLoad = receiveHandler;
    // set some default variables in dataObj
    dataObj.notify = false;
    dataObj.gender = "female";
    // create the loadVars object that will handle loading
    // the states from get_states.cfm
    var statesObj = new LoadVars();
    // once the states are loaded call the statesLoaded method
    statesObj.onLoad = statesHandler;
    // load the states
    statesObj.load(base_href + "get_states.cfm");
}
```

#### Listing 2

```
// method to handle the birth date once it's selected
function getBirthDate() {
    // call the getSelectedItem method of the calendar component
    // given the instance name of birthDate_mc
    birth_date = birthDate_mc.getSelectedItem();
    // create a valid date object from the selected date
    birth_date = new Date(birth_date);
    // set the birth year
    birth_year = birth_date.getFullYear();
    // set the birth month, add 1 since the month is zero indexed
    birth_month = birth_date.getMonth() + 1;
    // set the birth day
    birth_day = birth_date.getDate();
    // format the birth_date as YYYY-MM-DD so we can pass this
    // to CF
    birth_date = birth_year + "-" + birth_month + "-" +
    birth_day;
    // set the birth_date variable in dataObj
    dataObj.birth_date = birth_date;
}
```

#### Listing 3

```
// method to set the selected state, from the combo box,
// in dataObj
function getState() {
```

```
dataObj.state = state_mc.getValue();
}

// method to set the selected gender, from the radio buttons,
// in dataObj
function getGender() {
    dataObj.gender = gender_radio.getValue();
}

// method to set the notify flag, from the checkbox, in
// dataObj
function getNotify() {
    dataObj.notify = notify_mc.getValue();
}
```

#### Listing 4

```
// method to send the data to CF
function sendData() {
    // set input text values into our data object
    dataObj.first_name = first_name;
    dataObj.last_name = last_name;
    dataObj.city = city;
    dataObj.email = email
    // since the comments field is assigned an instance name
    // we access the text of the field via comments.text
    dataObj.comments = comments.txt;
    dataObj.sendAndLoad(base_href + "data_insert.cfm",
    receiveData);
}
```

#### Listing 5

```
// handles the response from the server after the informa-
// tion has been submitted
function receiveHandler(success) {
    // if the transfer was successful display a success mes-
    // sage
    if(success) {
        statusMessage_mc.setMessage("Thank you for filling out our
        form. Your information has been recorded successfully.");
        statusMessage_mc._visible = 1;
        // disable the send clip so they can't send the informa-
        // tion again
        send_mc.setEnabled(false);
        // if the transfer was not successful display an error
        // message
    } else {
        statusMessage_mc.setMessage("Please be sure to fill out
        all required fields.");
        statusMessage_mc._visible = 1;
    }
}
```

CODE LISTING

))))))))))))))

The code listing for this article is also located at

[www.sys-con.com/coldfusion/sourcec.cfm](http://www.sys-con.com/coldfusion/sourcec.cfm)