Hello World: Create and deploy WebSphere Portal components using Lotus Component Designer

Skill Level: Introductory

Ameet Kulkarni (aakulkar@us.ibm.com)
Staff Software Engineer
IBM

Keri Tuttle
Advisory Software Engineer
IBM

24 Jul 2007

Experience a fast way to create and deploy components for IBM® portal applications. During this tutorial, you will learn how to build a blog component in an easy-to-use, intuitive visual environment. You will then deploy your component and create a portal application. When finished with this tutorial, you will have the practical skills needed for using Lotus® Component Designer to leverage the collaboration features of WebSphere® Portal. This tutorial is part of the Hello World tutorials series on IBM Software Delivery Platform products.

Section 1. Before you start

About this series

The Hello World series is for novice developers who want a high-level, hands-on overview of IBM software products. Each tutorial in the series provides simple exercises and step-by-step instructions to familiarize you with the components and use of a particular product. The exercises only cover the basic concepts, but are enough to get you started.
About this tutorial

This tutorial demonstrates how to create WebSphere Portal applications using Lotus Component Designer without writing any Java code. This tutorial introduces you to the basics of Lotus Component Designer by showing how easy it is to build XML-document-based applications. It also demonstrates how quickly an application can be deployed to WebSphere Portal and Portal Express.

Objectives

In this tutorial:

- Create a blog component using Lotus Component Designer
- Create an XML schema for the component
- Create a page for recording blog content
- Create a page for viewing blog content and navigation options
- Deploy the blog component

Prerequisites

This tutorial is written for script developers -- for example, those with Lotus Domino, Microsoft Visual Basic, or Web development skills. You should have a general familiarity with using the Eclipse IDE before you begin.

System requirements

To run the examples in this tutorial, you need a system with Microsoft® Windows® or Linux®, Lotus Component Designer, and WebSphere Portal installed (see Resources). You also need at least 5 GB of free disk space.
Lotus Component Designer is a visual script-based tool that helps you create new applications that provide a rich user interface and supporting business logic without writing Java code. The intuitive integrated design environment, which is built on Eclipse, allows script developers, like IBM Lotus Domino or Microsoft Visual Basic developers, to use their existing skills and quickly be productive.

Lotus Component Designer provides the ability to drag and drop controls to define the user interface, implement JavaScript for business logic, and easily define application data with a built-in XML schema editor. These capabilities enable developers to build a complete end-to-end application with one tool. This tutorial uses the term component to refer to the application that you build.

What are Lotus Component Designer components?

Lotus Component Designer builds components. These components can incorporate a dedicated database for XML documents, a user interface, and business logic. All of these are defined when you create the component; in fact, they define a self-contained application. This is what you will be doing in this tutorial.

But components can also provide a user interface with business logic on an external data store (RDBMS, Domino), XML file, or Web service. In this case, the component is providing a front end to what may be another application. Another reason to use the term component is that components are the building blocks of the WebSphere Portal composite application framework. In the context of a composite application, components are portlets with features like composite application roles and points of variability.

About the sample application

The sample application you build in this tutorial is a simple blogging application that allows a user to enter rich text blog entries. It can be considered an electronic diary that other people can read. Once a user creates a blog entry, other users will be able to post comments about that entry. The application has a view to see all blog entries that have been created, along with each comment. The sample will also include a button that shows how client-side JavaScript works by using the alert() method.

Section 3. Create a blog component using Lotus Component Designer.
In this section, you create a blog component using the New Component wizard in Lotus Component Designer. Such components can incorporate a dedicated database for XML documents, a user interface, and business logic.

Would you like to see these steps demonstrated for you?

1. Choose **File > New > Component** to start creating the blog component. You should see the New Component window.

2. In the Name field, type `user_initials_Blog`. For example, if your name is Harry Smith, type `HS_Blog`. We'll use that name throughout this tutorial.

3. In the Description field, type **Hands-on Blog component**.

4. Select **Start with blank component**. The New Component window should look like Figure 1.

**Figure 1. New Component window**
5. Click **OK**.

Your new component now displays in the Designer Navigator on the left side of the screen. Click the name of the component to expand the list of design elements contained in the component.

Take a moment to familiarize yourself with the layout of Lotus Component Designer, illustrated in Figure 2. As you start to create elements in Lotus Component Designer, notice that many of these items come into view for you to use while you design your component.

**Figure 2. Lotus Component Designer perspective**
Section 4. Create an XML schema for the component

In this section, learn how to create a XML schema for the blog component. Input fields and controls are bound to schema elements in an XML document that is associated with the page. XML documents are stored in Lotus Component Designer's native XML database.

Would you like to see these steps demonstrated for you?

Show me

1. Click the component name in the Designer Navigator to give it focus. In this example, the component is named HS_Blog.
2. Start creating an XML schema using one of the following:
   - In the Designer Navigator, right-click XML Schemas and select New XML Schema.

3. In the New XML Schema window, type `blogEntry` in the Name field.

4. In the Description field, type `This schema is for the blogEntry page. (You will create this page in the next section.)`

5. In the Component field, select `HS_Blog`.

6. Check `Create in default location`. Your window should look like Figure 3. **Figure 3. New XML Schema**

![New XML Schema Window](image)

7. Click OK.

   The `blogEntry` schema now opens in the schema editor, as shown in Figure 4. **Figure 4. Lotus Component Designer schema editor**
1. In the schema editor, under Root Elements, click element0 (string).

2. Click the Basic tab in the Properties view. In the Specify Name and Type section:
   1. Replace the default text with s_Date for Field name.
   2. Select date from the Field type list.

3. In the schema editor, click blogEntry under Root Elements.

4. In the main menu, choose Schema > Insert Element > As Child.

5. Repeat Steps 1 and 2, replacing the default text in the Field name field with s_Subject. In the Field type list, select string.

6. Repeat Steps 3 and 4 to insert child elements into the blogEntry XML schema. Use Table 1 as a guide for field names and corresponding field types:

<table>
<thead>
<tr>
<th>Field name</th>
<th>Field type</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_content</td>
<td>String</td>
</tr>
</tbody>
</table>
Your schema should look like Figure 5. The order of your schema does not have to match the order in the figure; just be sure that all the elements are listed.

**Figure 5. Schema editor**

![Schema editor screenshot]

7. Save and close the schema editor by using the Save shortcut in the toolbar or by closing the schema editor and clicking **OK** when asked if you want to save your changes. Now you are ready to create a page that uses your new blogEntry XML schema.

---

**Section 5. Create a page for recording blog content**

In this section, create a page where the user can record his or her blog content. A **page** is the basic building block for many applications. Pages comprise controls, like input fields and buttons, as well as other objects, like images.
1. Click the HS_Blog component name in the Designer Navigator to give it focus.
2. Right-click **Pages** in the Designer Navigator, and choose **New Page**.
3. In the New Page window, type `editBlogEntry` in the Name field.
4. In the Description field, type *This page is for creating blog entries and for editing blog entries.*
5. Under Data, select **Do not create a data source**.
6. In the Component field, select **HS_Blog**.
7. Check **Create in default location**. The window should look like Figure 6. *Figure 6. New Page window*
8. Click **OK**.

Now that you have created the editBlogEntry page, specify the XML schema that you created in the previous section of this tutorial as the data source.

1. Open the editBlogEntry page in the page editor, and click inside the page to give it focus.

2. In the Properties view, click the Data tab.

3. In the Data sources section, click **Add** and then select **XML Document**.
Be sure that document is selected in the field.

4. In the Data Source: XML Document section:
   1. Leave the default text in the Name field.
   2. In the Schema list, select blogEntry.
   3. In the Context list, leave the default, /blogEntry.
   4. In the Document action list, select Create document.

5. Leave the Parent ID field blank.
   The editBlogEntry page should look like Figure 7.

**Figure 7. editBlogEntry page view**

5. Choose File > Save to save the editBlogEntry page.

Next, update the editBlogEntry page. Add a table control for formatting purposes as well as some additional controls for data entry.

1. In the Container Controls section of the UI Controls palette, select Table and drag it to the top part of the page editor.
2. In the page editor, select the table and right-click in the top-right cell of the table. Select **Delete Column** from the contextual menu. You should have two rows and one column.

3. From the main menu, choose **Table > Append Row(s)** and enter 3 in the Append Rows window. You should now have one column and five rows in your table.

4. In the second row of the table, type Subject:.

5. Place your cursor in the third row and choose **Insert > Core Control > Edit Box**.

6. In the fourth row, type Content:.

7. In the Core Controls section of the UI Controls palette, select **Rich Text** and drag it to the fifth row of the table.

Your page should look like Figure 8.

**Figure 8. editBlogEntry page**
Next, bind the XML schema elements to the fields on the page.

1. In the page editor, select the edit box in the third row to give it focus.
2. In the Properties view, select the Data tab and do the following:
   1. Select **Use data binding**.
   2. In the Data source field, select **document**.
   3. In the Data binding field, select **s_Subject [xs: string]**.

The page should look like Figure 9.

**Figure 9. Data binding**

3. Select the rich text control in the fifth row of the table.
4. In the Properties view, select the Data tab and do the following:
   1. Select **Use data binding**.
   2. In the Data source field, select **document**.
3. In the Data binding field, select **s_Content [xs: string]**.

5. Save and close the editBlogEntry page.

---

Section 6. Create a page for viewing blog entries

In this section, create a page and add a view control to it that displays the saved blog entries.

Would you like to see these steps demonstrated for you?

Show me

1. Select the HS_Blog component in the Designer Navigator to give it focus.

2. Right-click **Pages** in the Designer Navigator and select **New Page**.

3. In the New Page window, type **viewBlogEntry** in the Name field.

4. In the Description field, type **This page is for viewing a list of blog entries.**

5. In the Data section, select **Do not create a data source**.

6. In the Component list, select **HS_Blog**.

7. Check **Create in default location**. Your screen should look like Figure 10.

**Figure 10. New viewBlogEntry page**
8. Click **OK**.

Now that you have created the viewBlogEntry page, add controls and a view to the page.

1. In the Container Controls section of the UI Controls palette, click **Table** and then drag it to the top portion of the viewBlogEntry page in the page editor.

2. From the main menu, select **Table > Append Row(s)** and type 2 in the Append Rows window, then click **OK**. You should now have two columns
and four rows.

3. In the Core Controls section of the UI Controls palette, click **Image** and drag it to the first column cell/first row cell of the table.

4. In the Select Image window, shown in Figure 11, click **Add Image**.

![Figure 11. Select Image window](image)

5. In the Import Image window, click **Browse**. Select a graphic from your filesystem (we'll use bloglogo.gif in this example), and then click **OK** to import the image.

6. In the Select Image window, select **bloglogo.gif** and click **OK**.

7. Next to the image in the first row, type **Designer Hands-On Blog**, as shown in Figure 12.

![Figure 12. Add image to page](image)
Now add a view control to the page. This view will list all the blog entries that are created by a user.

1. In the Container Controls section of the UI Controls palette, click View and drag it to the fourth row and first column of the table. The Select View Query window opens:

**Figure 13. Select View Query window**
2. In the Select Query for View window:
   1. In the Type of query list, select **New View Query**.
   2. In the Name field, type **blogView**.
   3. In the Outline Information section, put a checkmark next to **blogEntry**.
   4. Click **OK**.
   5. Choose **File > Save**, but do not close the viewBlogEntry page.

Your view control in the page editor should now look like Figure 14.

**Figure 14. View control on page**
Next, update the view control to include the columns of data that you want to see.

Would you like to see these steps demonstrated for you?

Show me

1. Inside the view control, select the second row.

2. Choose View > Append Column. Repeat this step until you have three columns in the view.

3. Select the top row of the first column (Column1). This is the view column header.

4. In the Properties view, click the View Column Header tab.

5. In the Label field, change the default text to Author. Do not change the
column headings for the second and third columns. Your screen should now look like Figure 15.

**Figure 15. View column data binding**

6. In the page editor, click the Column1 view column.

7. In the Properties view, click the Data tab.

8. In the View Query Information section, verify that Column1 is selected in the View query column field, then click **Edit**. The Edit View Query Column window opens:

**Figure 16. Edit View Query Column window**
9. In the Column name field, delete **Column1** and type **Author**.

10. Leave **String** in the Data type field.

11. Select **blogEntry**.

12. In the Data binding tree structure, select **blogEntry/s_Author[xs:string]**.

13. Click **OK**.

14. With the Author column still selected, click on the View Column tab in the Properties view.

15. On the right-hand side under Column Display, select **Check box**.

16. In the page editor, select the second column view.

17. In the Properties view, click the Data tab and then click **Add**. Your screen should look like Figure 17.

**Figure 17. View column data binding**
18. Use the table below to complete the New View Query Column window, and then click **OK**.

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column name</td>
<td>Subject</td>
</tr>
<tr>
<td>Data type</td>
<td>String</td>
</tr>
<tr>
<td>Schemas</td>
<td>blogEntry</td>
</tr>
<tr>
<td>Data binding</td>
<td>/blogEntry/s_Subject[xs:string]</td>
</tr>
</tbody>
</table>

19. Repeat Steps 14 through 16 for the third view column, using the information in the table below.

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column name</td>
<td>Date</td>
</tr>
<tr>
<td>Data type</td>
<td>Date</td>
</tr>
<tr>
<td>Schemas</td>
<td>blogEntry</td>
</tr>
<tr>
<td>Data binding</td>
<td>/blogEntry/s_Date [xs:date]</td>
</tr>
</tbody>
</table>
Change the labels for the Subject and Date columns, if they are not labeled properly.

1. Click the Subject column header in the top row to give it focus.
2. In the Properties view, click the View Column Header tab.
3. In the Label field, type Subject.
4. Select the Date column header in the top row to give it focus. Repeat Steps 1 through 3, but this time type Date in the Label field.

To create a link to open the blog document:

1. In the page editor, select the Subject column.
2. In the Properties view, click on the View Column tab.
3. On the right side of the view, put a check in the box for Show values in this column as links.
4. For the Document open mode, select the Read-only radio button.
5. Save and close the viewBlogEntry page.

Your screen should now look like Figure 18.

**Figure 18. View column document open mode**
Section 7. Add navigation options

In this section, you'll add buttons and a script. This allows you to move through the user interface you created for the blog component.

Would you like to see these steps demonstrated for you?

Show me

1. In the Designer Navigator, expand **Pages** and double-click **editBlogEntry** to open the page in the page editor.
2. Add four buttons to the first row of the table:
   1. Drag and drop a button control from the UI Controls palette into the first row.
   2. Press the right arrow key once and then the spacebar key twice.
   3. Drag and drop another button control into that space.
   4. Repeat the previous three steps until you have four buttons in a row.
   Your page should look like Figure 19.

Figure 19. Add buttons to a page

3. Click the first button on the left to give it focus.
4. In the Properties view, click the Button tab.
5. In the Label field, type Edit.
6. With the first button still in focus in the page editor, click the Events tab.
7. In the Events view, click the onclick action and then select **Simple Actions**.

8. Click **Add Action** to open the Add Simple Action window, shown in Figure 20.

**Figure 20. Add a simple action**

9. In the Category list, select **Document**.

10. In the Action list, select **Change Document Mode**.

11. In the Document mode drop-down, select **Toggle**.

12. Click **OK**.

13. In the Events property panel, select the Full Update radio button on the bottom-right side.

14. Select the second button on the page. The Events view should still be open.

15. Select **Simple Actions** and then click **Add Action**.
16. In the Add Simple Action window, in the Category list, select **All**. In the Action list, select **Execute Script**.

17. In the Script window, type the following code:

```
var curDate=new Date();
if(document.isNewDocument())
{
    document.setDateValue("/blogEntry/s_Date",curDate);
    document.setStringValue("/blogEntry/s_Author",context.getUser().getCommonName());
}
```

18. Click **OK**.

19. In the Events view, click **Add Action**.

20. In the Category list, select **Basic**.

21. In the Action list, select **Open Page**.

22. In the Name of page to open list, select **viewBlogEntry**.

23. In the Target document list, select **Open document**. Leave the Document ID field empty.

24. Click **OK**. Figure 21 shows what the second button should look like in the Events view. When this button is clicked at runtime, it will set the date and send the user value to the schema for saving. Then it will open the viewBlogEntry page.

**Figure 21. Events view**
25. With the second button in focus in the page editor, click the Button tab in the Properties view.

26. In the Label field, type **OK**.

27. On the right-hand side, under **Options**, in the Button type list, select **Submit**.

28. Select the third button in the page editor.

29. Click the Button tab in the Properties view.

30. In the Label field, type **Cancel**.

31. Click the Events tab to open the Events view.

32. Select **Simple Actions** and then click **Add Action**.

33. In the Add Simple Action window, in the Category list, select **Basic**. In the Action field, select **Open Page**.

34. In the Name of page to open field, select **viewBlogEntry**.
35. In the Target document field, select **Open document**.

36. Save and close the editBlogEntry page.

You'll update the fourth button later. First, add the finishing touches to the viewBlogEntry page.

Would you like to see these steps demonstrated for you?

[Show me]

1. Open the viewBlogEntry page.

2. Drag and drop a button control into the second row of the table.

3. In the Properties view, select the Button tab and type **Post** in the Label field. (This button will be used to post new blog entries.)

4. Click the Events tab to open the Events view.

5. Click **onclick** and then select **Simple Actions**.

6. Click **Add Action** and then use the information in the table below to complete the Add Simple Action window.

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>All</td>
</tr>
<tr>
<td>Action</td>
<td>Open Page</td>
</tr>
<tr>
<td>Name of page to open</td>
<td>editBlogEntry</td>
</tr>
<tr>
<td>Target document</td>
<td>New document</td>
</tr>
</tbody>
</table>

7. Click **OK**.

8. In the page editor, press the spacebar key two times to remove focus from the post button.

9. Drag and drop another button control into the new space.

10. In the Properties view, select the Button tab and type **Delete** in the Label field. This button will be used to delete blog entries.

11. Click the Events tab to open the Events view.
12. Click `onclick` and then select **Simple Actions**.

13. Click **Add Action** and then use the information in Figure 22 and the table to complete the Add Simple Action window.

**Figure 22. Simple action**

![Add Simple Action window](image)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>Document</td>
</tr>
<tr>
<td>Action</td>
<td>Delete Selected Documents</td>
</tr>
<tr>
<td>View name</td>
<td>viewPanel</td>
</tr>
<tr>
<td>Confirmation text</td>
<td>Do you want to delete the selected documents?</td>
</tr>
</tbody>
</table>

14. Save and close the viewBlogEntry page.

Next, add JavaScript to the last button on the editBlogEntry page. This button demonstrates how you can use JavaScript on the client side to alert the user or perform other actions. Any text can be used for alert text; this example uses "Test
JavaScript.

Would you like to see these steps demonstrated for you?

Show me

1. Open the editBlogEntry page from the Designer Navigator.
2. In the page editor, select the fourth button, called Label.
3. In the Properties view, select the Button tab.
4. In the Label field type, click me.
5. Click the Events tab.
6. On the left side of the tab, select the onClick event.
7. On the right side of the tab, select the radio button for Script Editor.
8. Select the Web Client tab.
9. Enter the following text in the Script editor field:

```javascript
alert("Test JavaScript")
```

Your screen should look like Figure 23.

**Figure 23. Add JavaScript to the Events view**
10. Select the No Update radio button.

11. Save and close the editBlogEntry page.

Next, change the default page for your blog component.

Would you like to see these steps demonstrated for you?

Show me

1. In the Designer Navigator, double-click **Component Options**.

2. In the Editor panel, click on the Portlets tab.

3. In the Default page drop-down, select **viewBlogEntry**. The panel should look like Figure 24.

Figure 24. Edit component portlet property
4. Click the X in the tab to close the window. Click Yes to save the changes.

Section 8. Deploy the blog component

In this section, you learn to take advantage of the one-click deployment feature. This deploys your component to the server and creates the supporting XML document store all in one step.

1. In the Designer Navigator, right-click the component name and select Deploy in the contextual menu; you'll see the window shown in Figure 25. If a deployment profile already exists, skip to Step 7. (Note that your server setting for Portal could be different from what you'll see in this example. The settings shown assume that you have Portal installed on your local computer.)

Figure 25. Deploy Component window
2. Click **Add** to create a new deployment profile. The following window opens:

**Figure 26. Deploy profile**
3. Type the name of the deployment profile. Use any name you’d like.

4. If you do not have Portal installed on your local system, enter your server information in the Server name or address field.

5. In the User name field, type **designer**.

6. In the Password field, type **password**.

7. Click **OK**.

8. In the Deploy window, click **OK**.
On the lower-left part of the screen, you should see a message that the component has been successfully deployed.

Section 9. Add the blog component to a Portal Template page

The blog component is ready and has been deployed to Portal. You will now learn how to add the component (or a portlet) to the template page. In this section, you will need to be aware of the setting you used for your deployment profile. If you are using a remote Portal server, please contact your system administrator for the correct URL for the server. If you have Portal installed on your local system, you will use localhost or the IP address in the URL.

Would you like to see these steps demonstrated for you?

Show me

1. Launch your Internet browser.

2. Enter the appropriate URL for Portal's home page. This should be something like http://%ServerName:10038/wps/portal. The server name could be an IP address; as noted, you should contact your system administrator if you are unsure.

3. Log in to WebSphere Portal using wpsadmin as both the username and password.

4. In the Product Links section, click Templates, highlighted in Figure 27.

Figure 27. Portal templates
5. Click the Template Library tab, shown in Figure 28.  
*Figure 28. Template library*

6. On the Template page, click **New**. To create a new template:
   
   1. In the Application template name field, type **HS_Blog**.
   
   2. In the Category field, select **Composite Applications**.
   
   3. Optionally, in the Description field, type a description of the application. (The only time users will see the description is when they look at the properties of the template.)
   
   4. In the Starting point field, select **Portal Blank Template**. Your screen should look like Figure 29.  
*Figure 29. New Application template*
5. Click OK. The template should appear in the list of available templates.

7. Click the arrow next to the HS_Blog template to display the pop-up menu in Figure 30.

Figure 30. Pop-up window

[Diagram showing the pop-up window with options such as Edit Template Properties, Edit Template Layout, Edit Template Parameters, Manage Roles, Export Template, Delete Template, Assign Template Permissions]
8. Select **Edit Template Layout**.

9. You should see a Blank Page link under the title. On the right side of the Application Layout page, click the Edit Page Layout button.

10. On the Edit Layout page, click the Add Portlets button, shown in Figure 31.

   **Figure 31. Add a portlet**

11. Search for and select your portlet, HS_Blog, as shown in Figure 32.

   **Figure 32. Select the portlet**

12. Click **OK**, then select **Done** to save the layout.

13. Click **Save** to save the template.

14. Click the Application Library tab, shown in Figure 33.

   **Figure 33. Application Library tab**
15. Click **New**.

16. On the Applications page, shown in Figure 34, type the name **HS_Blog** and select the template that you just created, then click **OK**.

**Figure 34. Select the template**

17. A message will indicate that your application has been created, as shown in Figure 35.

**Figure 35. Application created**
18. Click the HS_Blog link to open your application.

Figure 36 shows what your blog component will look like in the Portal application. Once you have explored your new blog component, take a moment to make a couple of changes to your blog in Lotus Component Designer. Once you have made your changes, save the blog component and use the steps from the previous section in this tutorial to deploy your component again." You will notice that it takes no time at all to update and change a component with rapid deployment. When the component is finished deploying, navigate back to the application in which you created the blog and view the changes you made to the component.

**Figure 36. Blog application in action**

---

**Section 10. Conclusion**

In this tutorial, you learned how to build a blog component in an easy-to-use, intuitive visual environment. You also learned how to deploy your component and create a portal application. Finally, you acquired practical skills for using Designer and quickly leveraging the collaboration features of Portal. You will now be able to use the advanced features of Lotus Component Designer in the User Interface, like external
data, advanced view queries, schemas, and JavaScript programming to build more complex components.
Resources

Learn

• Refer to the Lotus Component Designer product documentation for more information.

• Check out the Lotus Component Designer product page.

• Find out more about WebSphere Portal Express.

Get products and technologies

• Download a trial version of WebSphere Portal Express.

Discuss

• Participate in the discussion forum for this content.

About the authors

Ameet Kulkarni
Ameet Kulkarni is a staff software engineer working at IBM Lotus in Westford, Massachusetts. He has worked for IBM Lotus on various features of the Lotus Component Designer product, including user interface development, tooling for Designer, support for external data stores like Domino, and overall release engineering aspects of the product.

Keri Tuttle
Keri Tuttle joined Lotus/IBM in 2000 and currently works as the accessibility and automation lead for Lotus Component Designer. She has also worked for QE on Workplace Templates and Team Spaces.