Lotus Developer Domain
The Technical Resource for Lotus software
LDD Today

Level: Intermediate
Works with: QuickPlace
Updated: 01-May-2002

by Ian Connor
and Tara Hall

Right out of the box, QuickPlace provides forms to help your organization get started. Sometimes, however, a standard QuickPlace form may not suit the specialized needs of your organization. You may need to customize forms to gather registration or customer contact information or to create surveys. Maybe you want to build a documentation library where users can submit and view or download files that QuickPlace doesn't support? Using an imported HTML file, the QuickPlace ActiveX controls, and some JavaScript, you can build a custom form that allows users to attach files that can be displayed or downloaded.

This article shows you how to create a custom form from an HTML file. We include a QuickPlace ActiveX control in the file to create a field where you can attach Adobe PDF files. Then we use JavaScript to build the URLs needed to download or to display the attached PDF file. We also show you how to embed a third-party ActiveX control that displays AutoCAD DWF files. Finally, we show you how to upload the file into QuickPlace.

This article assumes knowledge of HTML, JavaScript, and ActiveX controls. You should also know how to create at least simple or Microsoft Office forms in QuickPlace.

Creating custom forms

QuickPlace offers three options for form creation: simple forms, HTML forms, and Microsoft Office forms. To create simple forms in QuickPlace, you only need a Web browser.

As you might expect, simple forms are easy to create. However, they offer a limited number of field types and provide limited programmatic control. For instance, you can't add code to validate required fields on a simple form.

To create a more sophisticated form, you can import an existing HTML or Microsoft Office file into QuickPlace. Microsoft Office forms allow you to use Word, Excel, and other Microsoft Office templates within QuickPlace. Creating custom forms from existing files is as simple as uploading them into QuickPlace.

To create a custom form from an HTML file, you need:

- Manager access to the QuickPlace where you want to create the form
- The HTML file on which the form will be based (The **HTML file examples** used in this article can be downloaded from the **Sandbox**.)

HTML file basics

To create your HTML file to upload, you must know which HTML tags are supported.

To build a QuickPlace form from an existing HTML file, you can include the <FORM> tag in the file. However, this tag is optional. If you leave it out, QuickPlace will still generate the form. When QuickPlace parses the supported form tags that are in the HTML file, it automatically adds the <FORM> tags to display the form in a Web browser.

Here are the supported HTML form tags:

- <INPUT> with the supported type attribute values "text," "password," "checkbox," "radio," "file," and "hidden"
- <TEXTAREA>
- <SELECT>

When you upload an HTML file into QuickPlace, the server maps the HTML tags to the appropriate field type in the Notes document where data is stored. For instance, the following tag:

```
<INPUT type="text" name="firstfield">
```

creates a text field that is mapped to a Notes text field. To map a tag name to a Notes field, QuickPlace adds prefixes to the Notes field name. You must keep all field names unique.

Just as in the Notes client, when users open a QuickPlace page in edit mode, they can enter text into the text field. When users open a page in read mode, only the value of the text field appears.

Embedding QuickPlace ActiveX controls

The <QuickPlaceControl> tag looks and acts like an HTML tag. It embeds the QuickPlace ActiveX controls in an HTML file. There are two QuickPlace ActiveX controls:

- The Rich Text control creates a text field in which users can format text. Regular text fields created with the <INPUT> tag or <TEXTAREA> don't allow users to format their text.
- The Attachment control creates a field in which users can attach a file.

The <QuickPlaceControl> tag has two attributes:

- The type attribute specifies the type of control. The values are "richtext" or "attachment."
- The name attribute specifies the field name. This attribute applies only to the Rich Text control.

To embed a control, you include the <QuickPlaceControl> tag in the HTML file where you want the control to appear. You must place the tag somewhere between the <BODY> tags.

For example, to create a field where users can attach a file, and provide them with directions to do so, you could include this HTML between the <BODY> tags of your HTML file:

```
PDF Title: <br>
<INPUT type="text" id=h_Name name=h_Name>
<P>
...
Drag your PDF file or attach it into this control:

<br>
<QuickPlaceControl type="attachment">
```

In addition to the two QuickPlace ActiveX controls, you can specify any other ActiveX controls or Java applets in your HTML file. QuickPlace supports any ActiveX control or Java applet; however, your Web browser may not. (Later in this article, we show you an example of embedding a third-party ActiveX control that displays attached files.)

Using JavaScript to access the attached file

You can use JavaScript in the HTML file to validate fields, to write HTML, or to perform any number of tasks that you'd normally use JavaScript for. In our example, we use JavaScript to display or to download the file attached to the QuickPlace Attachment control.

Note: Not every file can be displayed. To view a PDF file, you need the Adobe Acrobat Reader plug-in for your Web browser. Otherwise, the Web browser won't recognize the file type and will initiate a file download.

When you create a page based on the custom form in QuickPlace, attach the PDF file, and then publish the page, the JavaScript code displays the attached file. QuickPlace generates a URL that follows this convention:

/QuickPlace/directory/databasename.nsf/h_viewname/UNID?OpenDocument&arguments

This URL brings you to the page, but it doesn't allow you to access the attached file. To access the attached file,

you can use JavaScript to remove the ?OpenDocument URL command and to append /$FILE/file_name to the remaining URL. The /$FILE/file_name command accesses the specified file. (You don't need the ?OpenElement URL command to access the file.)

The following JavaScript code snippet shows how to change the URL command by first checking for a PDF file attachment, and then appending the file name to the new URL. You can download the **full code sample** from the Sandbox.

```
<script language="JavaScript" type="text/javascript">
<!--
      var sExt = ".pdf";
      var sAttachmentName;
      var sURL

            var iAttachmentNamesLength = h_AttachmentNames.length;
            var sAttachmentName = h_AttachmentNames.toLowerCase();

                  // Get the file name from the list
                  var iExtEnd = sAttachmentName.indexOf(sExt);

                  if (iExtEnd == -1)
                  {
                  // There is no file here
                  sAttachmentName = "";
                  }
                  else
                  {
                  sAttachmentName = h_AttachmentNames.substring(1,iExtEnd+sExt.length);

                  // Rewrite the file path to append /$FILE/ and the file name
                  var iExtStart = sAttachmentName.lastIndexOf("");
                  if ( iExtStart != 0)
            sAttachmentName = sAttachmentName.substring(iExtStart+1, sAttachmentName.length);
                  }

                  var iCmdStart = PATH_INFO_DECODED.indexOf('?');
                  if (iCmdStart == -1)
                  {
                  sURL = PATH_INFO_DECODED
                  }
                  else
                  {
                  if (PATH_INFO_DECODED.charAt(iCmdStart-1) == '/')
                     {
                     sURL = PATH_INFO_DECODED.substring(0,iCmdStart-1);
                     }
                     else
                     {
                     sURL = PATH_INFO_DECODED.substring(0,iCmdStart);
                     }
                  }

      var sFileName = sURL + "/$FILE/" + sAttachmentName;
      if (sAttachmentName != "")
```

When you embed JavaScript in an HTML file, it must appear between the <BODY> tags. QuickPlace overwrites any JavaScript code that you embed between the <HEAD> tags with custom code.

Embedding other ActiveX controls

As previously mentioned, you can embed any ActiveX control or Java applet in your HTML file. To do so, you must know the parameters of the control or applet.

In our PDF file example, we rely on the Adobe Acrobat Reader Web browser plug-in to display the file. However,

some file types have no browser plug-in. If an ActiveX control is available to display a file, you can embed that control in the HTML file and write JavaScript code that passes the attached file name to the control for display.

The following example embeds the WHIP! control from AutoDesk in an HTML file. The WHIP! control is a viewer for displaying DWF (drawing web format) files. Using the QuickPlace Attachment control, you can attach a DWF file to a page. We use JavaScript to access the DWF file that the WHIP! control will display after you publish the page. You can download the **WHIP! control** for free from the AutoDesk Web site.

```
<object id="linkrods" classid="clsid:B2BE75F3-9197-11CF-ABF4-08000996E931"
codebase=" ftp://ftp.autodesk.com/pub/whip/english/whip.cab#version=4,0,42,102 "
width=400 height=300>
<param name="Filename" value="linkrods.dwf">
</object>
```

As with the QuickPlace ActiveX controls, we place the code between the <BODY> tags where we want to the control to appear.

The following JavaScript code snippet passes the DWF file name to the WHIP! control by rewriting the <OBJECT> tag with the DWF file name. You can download the **full code sample** from the Sandbox.

```
    // Pass the new URL to the WHIP! control
    {
    document.write("<OBJECT style='LEFT: 0px; TOP: 0px'
    codeBase=ftp://ftp.autodesk.com/pub/whip/english/whip.cab#version=4,0,42,102 ");
    document.write("      classid=clsid:B2BE75F3-9197-11CF-ABF4-08000996E931 width=100% height=600
    VIEWASTEXT>");
    document.write("<PARAM NAME='Filename' VALUE='");
    document.write(sFileName);
    document.write("'>");
    document.write("<embed name=thanks src='");
    document.write(sFileName);
    document.writeln("' pluginspage='http://www.autodesk.com/whip' width=300 height=300></OBJECT>");

    }
    else
    // If no DWF file is available, display the following message in the browser
    {
    document.writeln("No supported file has been attached");
    }

//-->
</script>
```

Uploading the HTML file to QuickPlace

After you complete the HTML file with the embedded ActiveX control and JavaScript, you can upload the file to QuickPlace. Remember, you must have manager access to create forms in QuickPlace.

1. Find the room in which you want to create the custom form and click Customize or Room Options in the side bar, depending on which room you've chosen.
2. Click Form and then click New Form.
3. Select Imported HTML Form and then click Next to see the following screen:

4. On this screen, you can browse for the HTML file or drag the file from another area, such as your desktop, onto the Attachment field.
5. Click Done.

Once you have created the form, you can edit it either by replacing the HTML file with a new or modified file or by double-clicking the HTML file attachment to launch your default Web editor to modify the file.

Conclusion

You can use the examples in this article to create custom forms that display a variety of attachments that extend QuickPlace to the applications used in your organization. Take advantage of the QuickPlace Attachment and Rich Text controls to provide functionality that HTML alone can't provide. These techniques work with any file that has an ActiveX control for viewing.

ABOUT IAN CONNOR

Ian Connor is a maintenance software engineer with IBM, working with the external support team to fix bugs from escalated support incidents. Ian also works on QuickPlace incidents and spent some time last year working directly for the QuickPlace team on the upload control, programmability, web services, and Notes integration. Ian enjoys skiing, travelling, and spending time with his wife. His current personal goals include understanding emptiness.

