Notes.net Iris Today

Taking your Domino applications to the Web

by
David
DeJean

Level: Intermediate
Works with: Designer 5.0
Updated: 01/02/2001

More and more, being a Notes developer has to do with the development tools you use, rather than the client software your application runs in. The Web browser has taken a place beside the Notes client, and applications originally created for Notes clients must be reworked for browsers.

Release 5 of Domino Designer has made it much easier to develop NSF applications that work well on the Web. And by now there's a significant body of experience with the new features and functions of Release 5—and with the strengths and weaknesses of browsers as a development target.

This article provides a framework for analyzing your Notes application in preparation for redeploying it as a Web application. It then examines the different elements of the application and offers tips and techniques for making your applications shine on the Web.

This article assumes an in-depth understanding of designing applications using Domino Designer as well as knowledge of LotusScript and JavaScript.

Understanding the differences

Moving a Notes application to the Web is not a simple matter of adding some HTML code to forms and views. The division of labor in a Notes application—what the Domino server does and what the Notes client does—is very different from the architecture of a browser-based application. This is because the capabilities of a Notes client are very different from a Web browser, and the protocols the browser and server use to communicate are very different as well.

Release 5 uses JavaScript and Java to translate Notes client functionality to the browser with fuller fidelity, so that Web users can interact with action bars, Rich Text, views, and other Notes features in familiar ways. But there is still no Web functionality equivalent to the Notes Document Object Model (DOM) and Remote Procedure Call (RPC) protocol. A Notes client can send instructions to the server to perform a task and receive back results that are refreshed within the current open form or document. This kind of interaction is nonexistent between a browser and server. All a browser can do is send a request to a server, and all a server can do is send a complete page to a browser.

Limitations of the Web browser and the browser/server architecture are the basis for most of the problems you must solve as you consider whether the functionality of your application can be delivered in a browser at all, and if so, how you might do it. Beyond functionality, there are other issues, as well—performance of both the client and server, security and access, and ease of use, for example.

Getting ready to make the move

Before you can begin reworking an application for the Web, you need to examine what you currently have and what you really need. There are three parts to completing a good analysis of the application.

Re-evaluate basic assumptions

Your review of the application should begin by being as specific as possible about the desired result, and then work its way backwards. Start by rethinking the who-what-where of your application in its new context:

- Who will the users of your application be on the Web? Will they be the same people who use it now in a Notes client? A completely new and different set of users? Or some of both? Will the application run only on the Web, or will it serve a mix of Web and Notes users?
- What will browser users do with the application? Will they need all the functionality of the Notes version or only part of it? Will Web deployment change the level of activity in the application, with resulting impact on server load?
- Where will the Web users be? Will your application be accessible on a corporate intranet, or is the goal to deliver it across the Web itself? What access levels will users need, and are those appropriate for a Web application?

Exercise the application
With the answers to these questions in mind, you can begin to analyze your application. Start by accessing your Notes application in a browser. Navigate through it as completely as possible. Create and save new documents using the forms if you can. Look at the results and note errors. Then go back and compare the application in the two major browsers—exercise the application in the versions of Internet Explorer and Netscape Navigator you would expect your users to use, and note differences between the two. During this hands-on evaluation, you'll begin to spot problems and issues as well as discover some changes that will improve the application in general.

Review the application piece-by-piece
Now you can begin looking at the application piece-by-functional-piece to see:

- Whether the piece *needs* to work on the Web
- Whether it *can* work on the Web
- *How* it would work on the Web

Keep those points in mind as you consider each category:

- **Forms**
  Does each form look acceptable in the browser? Do graphics display properly, fields and field labels align? Are there differences between the browsers? Could problems be solved by creating separate forms or subforms for Notes and Web users? Do the forms contain LotusScript that must be replaced with JavaScript for the browser? Are existing agent actions available on the Web? Do you need to rewrite back-end agents to process query strings or change them to scheduled agents?
- **Views**
  Does each view work on the Web? Do all columns display? Can you select documents, or do you need to provide additional code to create HTML links? Are all the actions your application requires available?
- **Data acquisition**
  Where does the data come from, and how does it get into your application? Is it entered by the users? Is it drawn from other applications or from back-end data sources? Is it environmental data like user names, document IDs, and server and database names?
- **User interface and graphics**
  Does the application's user interface and graphic design meet user expectations for a Web application? Does the application's existing navigation work in a browser, or do you need to add navigational choices to compensate for the loss of Notes client functionality? Does it need anchors on the Web that it didn't need in Notes—a home

page, section pages, help pages?

- **Access and security issues**
  How will you balance user access against system security? Will your application require users to authenticate with the Domino server? What level of access will they require? If your application will run both in a Notes client and a browser, does the security need to be different for the different clients?

If you've considered these questions carefully, you'll have gone a long way toward mapping out what needs to be done to take your Notes application to the Web.

Tips and techniques for forms

Trying out your forms in a browser may be a frustrating experience at first, because you are likely to see error messages rather than anything familiar. If that happens, make a copy of the form and remove fields, formulas, and scripted events from it until the form displays in the browser. Then begin adding them back in, checking to see that each one works in a browser and taking note of those that don't.

There are many parts of a form that might not work in a browser. For example, "Allow values not in list" for keyword fields aren't supported by browsers. (There are workarounds. For example, the *Iris Today* article "**Webifying an existing Notes application**," includes an example of how you can let a browser user add a keyword to a multi-value field.)

Check your application for @functions and @commands that don't work in a browser. Those @functions that don't work fall into three major categories:

- They work only in the Notes client interface. There are no browser counterparts to @DialogBox, @Picklist, and @Prompt, for example. And several advanced mail-handling functions are unique to the Notes client, such as @MailSend and @IsDocBeingMailed.
- The much simpler structure of Web views doesn't support many functions, including @DocChildren, @DocLevel, @DocNumber, @IsCategory, or @Responses.
- Many features associated with Notes access, preferences, and the workstation environment don't carry over to a browser—@Certificate, @IsAgentEnabled, @MailEncryptSavedPreference, and @UserPrivileges, for example.

Most @commands are associated with controlling the Notes client interface, so they don't work in Web applications. The handful that do work on the Web work hard, though, because they're used often. @Command([FileSave]) and @Command([FileCloseWindow]), for example are used frequently together to simulate a Submit button.

You can check to see whether @formulas and properties of other programming constructs work for the Web by looking through the topics listed in the "Overview of features to avoid using in Web applications" topic in **Domino 5 Designer Help**.

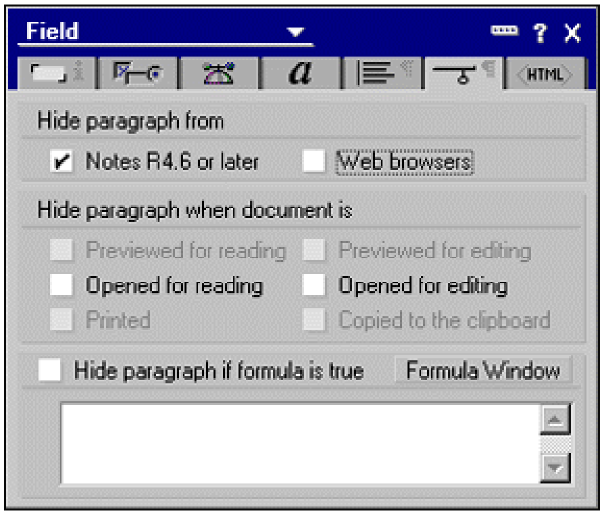
Multiple solutions for multiple clients

As you diagnose and fix problems, you will likely find you need as many solutions as the number of client types you are supporting: if your application is intended to run in both a Notes client and a browser, you may need to code one way for Notes, another way for Internet Explorer, and still another for Netscape Navigator. There are many ways to do this using multiple forms or subforms, selection formulas, and hide-whens that identify the user's client and serve up the appropriate code for it.

In formulas, you can use @ClientType to identify the user's client as "Notes" or "Web." In cases where you need to code around differences between the two major browsers, you can use @BrowserInfo to help you determine what the user is running.

The extent of the differences will affect your solution as well. You may be able to resolve problems with different versions of fields or formulas that are specific to the clients. If you need to duplicate several fields, you may find it easier to put them on subforms, or create two versions of the same form. Performance may be a deciding factor: computing a single hide-when formula to display one of two forms is more efficient than computing 30 hide-whens to display variant fields within a single form.

Hiding fields
If you can solve a problem with two versions of a field formula, one that works in Notes and the other that works on the Web, then include both in the form and use the "Hide from Notes/Hide from Web browsers" selection in the Field properties box to display one or the other:



Hide paragraph from
[✓] Notes R4.6 or later
[ ] Web browsers
Hide paragraph when document is
[ ] Previewed for reading
[ ] Previewed for editing
[ ] Opened for reading
[ ] Opened for editing
[ ] Printed
[ ] Copied to the clipboard
[ ] Hide paragraph if formula is true
Formula Window

If you're using the same form for both Notes clients and browsers, the "Hide from Notes/Hide from Web browsers" attribute on an object's properties can be very useful. But don't forget that *hidden* has a different meaning for each client. In Notes, hidden fields are still present in the document and can be used in scripted calculations. Fields hidden from Web browsers, on the other hand, are cut out of the document by Domino before the page is served to the browser. The field contents are not available to be used by JavaScript.

Tip: If you want to make fields invisible in a browser, but keep their contents available to JavaScript, don't use the "Hide from Notes/Hide from Web browsers" and don't put "type=hidden" into the fields' HTML Body Attributes objects. Make sure that:

- Their hide-when properties are set correctly for the Notes client
- "Use JavaScript when generating pages" is set in the database properties
- "Generate HTML for all fields" is selected in the form's properties

Domino will generate the appropriate <type=hidden> tags in the HTML page:

```
<input type="hidden" name="fieldname" value="fieldcontents">
```

Remember that field names and values treated this way are not secure. They can still be seen by any user who clicks on the browser's "View Page

Source" function.

Remember, too, that not all fields can be passed to a browser this way—the password field, for example, "$Updatedby," "$HtmlHead," and any objects that contain NULL characters (which includes users' public keys) because NULLs can't be expressed in HTML.

Use subforms
Subforms can be an easy way to create forms that work both in a Notes client and a browser. Add a computed subform and set its hide-when properties so that it displays properly. The following formula calls one of two computed subforms named NS and IE that include browser-specific code by using the CGI variable for browser type in @BrowserInfo:

```
@If(@BrowserInfo("BrowserType")="Netscape";"NS";"IE")
```

Search for @BrowserInfo in **Domino 5 Designer Help** to see the entire list of properties accessible using this function.

In R5, each subform has its own JS Header, so you can selectively include JavaScript as well as other data types in your forms by using computed subforms.

Remember that all subforms open simultaneously with the main form. You can't display a computed subform on the basis of calculations after the page opens.

Use duplicate forms for the Web and Notes
"Hide from Notes/Hide from Web browsers" will also work for entire forms. Create two versions of the form—one for use in the Notes client, one for the Web—and give them different names but the same alias. Set their hide-when attributes so that one is hidden from Notes clients and the other from Web browsers. Use the alias to open the form, and Domino will open the correct form for the user's client.

LotusScript or JavaScript?
The amount and type of LotusScript in your application may have a major influence on how easily you can migrate it to a browser.

If your application uses LotusScript primarily in the client for processing data as it is entered and saved—to validate the contents of fields or to massage the data's format—you may be able to do the same thing in a browser with JavaScript. If your LotusScript is contained mostly in agents that execute on the server, it may work well for both Notes and Web versions of your application.

In the R5 Notes client, many form, field, and button events can be scripted in either JavaScript or LotusScript (or, to be sure, @formulas). Depending on how your application uses scripted events, this may mean that you can convert some of your scripts to JavaScript and use the same form for both Notes and Web clients. There are differences in the capabilities, however. JavaScript in the Notes client has access only to the data in the currently open form—it lacks the access to front-end and back-end Domino Objects of LotusScript. Your application may work best if you write LotusScript for execution in the Notes client and JavaScript for browsers, and use hide-whens or separate forms to keep the execution straight.

Pages or forms?
The page is a low-impact alternative to a form or document for displaying static information, or for carrying embedded controls for outlines, navigators, views, or folders to the browser. The Domino server performs many fewer calculations when it opens a page than it must do when it opens a form or document, so there is a performance benefit. Pages are saved as database

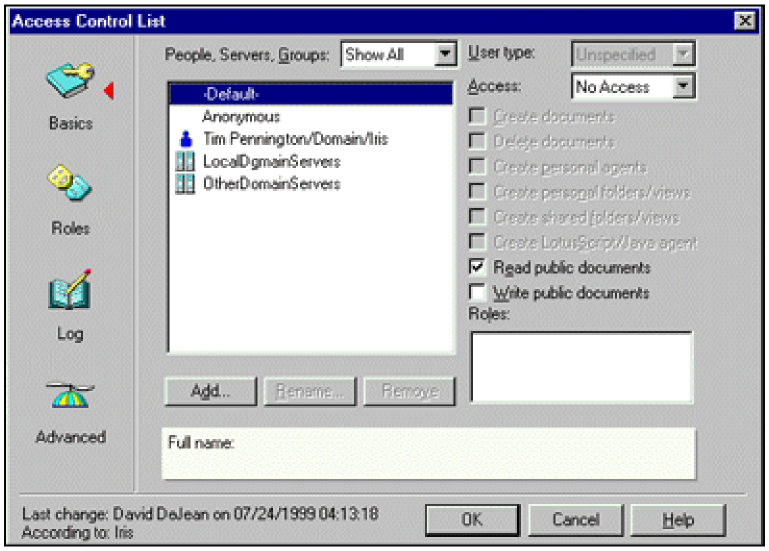
resources and can be referenced in outlines, included in framesets, and used as the application's start page. Pages cannot include fields, but they can include computed text.

Pages require a lower level of access permission than forms—users with Reader access can see pages, while forms require Author access. This difference can make it easier to allow unauthenticated users to view your application and navigate through it without creating the potential for abuse, because common navigation constructions (combining a navigator and a view, for example) that previously had to be embedded in forms can now be embedded in pages.

Another way to reduce the performance hit of opening a form when all you want to do is to display a Web page is to put all the HTML code for the page into a field named HTML. When Domino finds an HTML field on a form, it sends the contents to the browser without performing form checking or calculations. You can also use HTML fields to get large volumes of HTML into Domino databases. Use a form with an HTML field (it can be either text or Rich Text). Copy and paste the HTML code, save the document, and edit it to fix links or path information. The HTML field can be either computed or editable. Use a computed field if you want to compose a single page that can't be edited by users later. Use an editable field if you're using the form to create many documents holding HTML pages.

Replace missing Notes dialog boxes
You should consider designing forms for your application to replace error messages and help dialog boxes that are missing on the Web. Particularly if you require users to authenticate in order to use your application, you should create a $$ReturnAuthenticationFailure form to inform users that authentication failed and give them links they can use to navigate back to familiar territory in the application.

Create your form and save it with the name $$ReturnAuthenticationFailure. In order to make this form available for public access, you must do two things. First, you must set the Default role to read public documents. You do this in the application's Access Control List:



People, Servers, Groups: Show All
User type: Unspecified
Access: No Access

-Default-
Anonymous
Tim Pennington/Domain/Iris
LocalDomainServers
OtherDomainServers

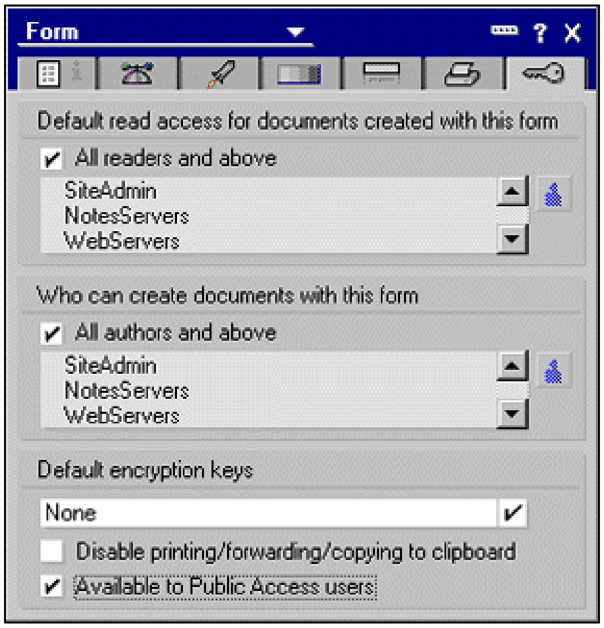
- [ ] Create documents
- [ ] Delete documents
- [ ] Create personal agents
- [ ] Create personal folders/views
- [ ] Create shared folders/views
- [ ] Create LotusScript/Java agent
- [✓] Read public documents
- [ ] Write public documents

Roles:

Full name:

Last change: David DeJean on 07/24/1999 04:13:18
According to: Iris

Second, set the form's security properties to make it available to public users by choosing the last item on the Security tab:



Default read access for documents created with this form

✓ All readers and above

SiteAdmin
NotesServers
WebServers

Who can create documents with this form

✓ All authors and above

SiteAdmin
NotesServers
WebServers

Default encryption keys

None

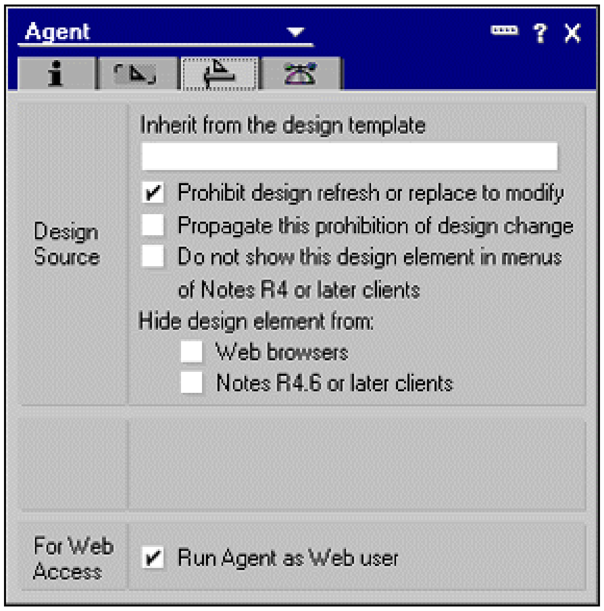
☐ Disable printing/forwarding/copying to clipboard

✓ Available to Public Access users

There are four reserved form names that can be used to create customized error messages for browser users: $$ReturnAuthenticationFailure, $$ReturnAuthorizationFailure, $$ReturnDocumentDeleted, and $$ReturnGeneralError. For more about these fields and how to use them, see "Displaying a customized error message" in **Domino 5 Designer Help**.

Agents and actions
Part of the process of architecting your application is deciding what needs to be done by JavaScript in the browser and what must be done by agents on the server. On the server side, you can run agents from WebQueryOpen and WebQuerySave events. Agents can be run from the Web by @Command([ToolsRunMacro]) or @URLOpen formulas.

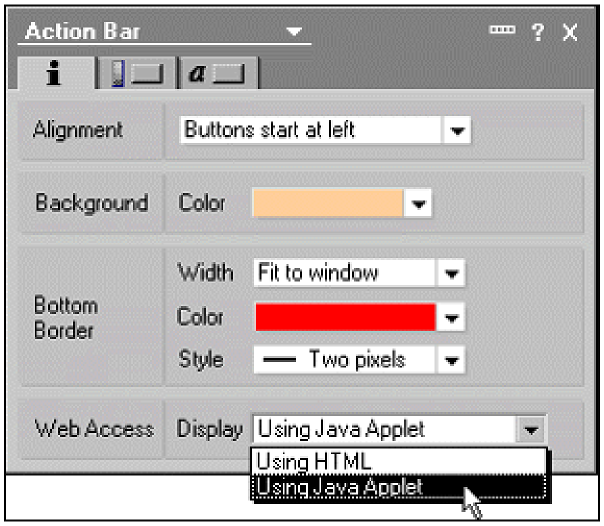
By default, Web agents run under the identity of the agent author (the person who saved the agent). To run agents under the identity of the Web user, open the Agent properties box, click the Design tab, and select "Run Agent as Web user" in the For Web Access section:


Agent
Design Source
Inherit from the design template
☑ Prohibit design refresh or replace to modify
☐ Propagate this prohibition of design change
☐ Do not show this design element in menus of Notes R4 or later clients
Hide design element from:
☐ Web browsers
☐ Notes R4.6 or later clients
For Web Access
☑ Run Agent as Web user

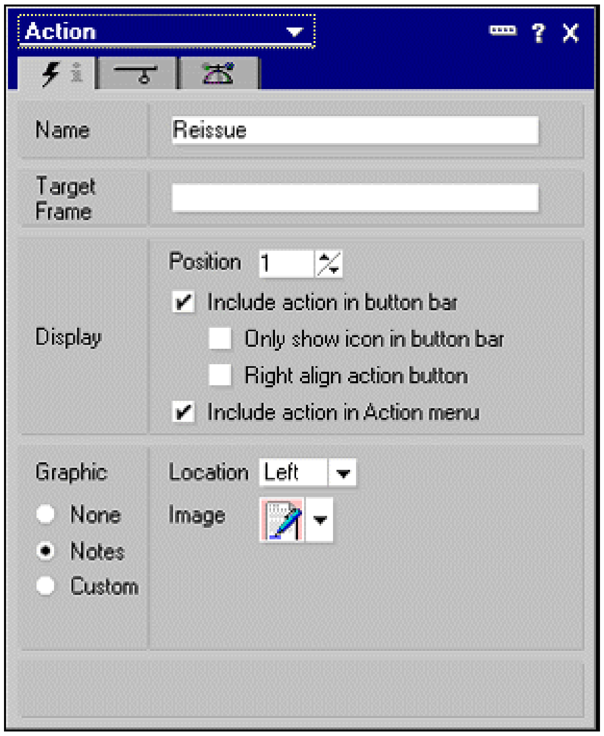
This option can provide more security, because when a Web user tries to run an agent with this property set, Domino prompts the user for name and password and checks this against the invoker's rights in the database ACL.

You can reference a URL or generate an HTML page from within an agent by using Print statements. This form of redirection doesn't work from the WebQueryOpen event, but from a WebQueryClose event, you can use it to deliver a personalized confirmation message to the user or to display the results of the agent execution.

Browser users don't have a counterpart to the Notes client's menus, so you must provide them with whatever actions they need to perform in your application. The R5 action bar applet allows you to do this within a consistent UI. To create an action bar, in Designer, open the page, form, or view where the action bar will appear. Choose Design - Page/Form/View Properties. In the properties box, change to Action Bar properties and select "Display Using Java Applet," as shown in this screen:



Set the Action properties for each of the actions that should appear on the action bar so that they are included in the bar in the proper order, and if you want, so that they have an icon:



Name: Reissue

Target Frame:

Display: Position 1

- [✓] Include action in button bar
- [ ] Only show icon in button bar
- [ ] Right align action button
- [✓] Include action in Action menu

Graphic: None [✓ Notes] Custom

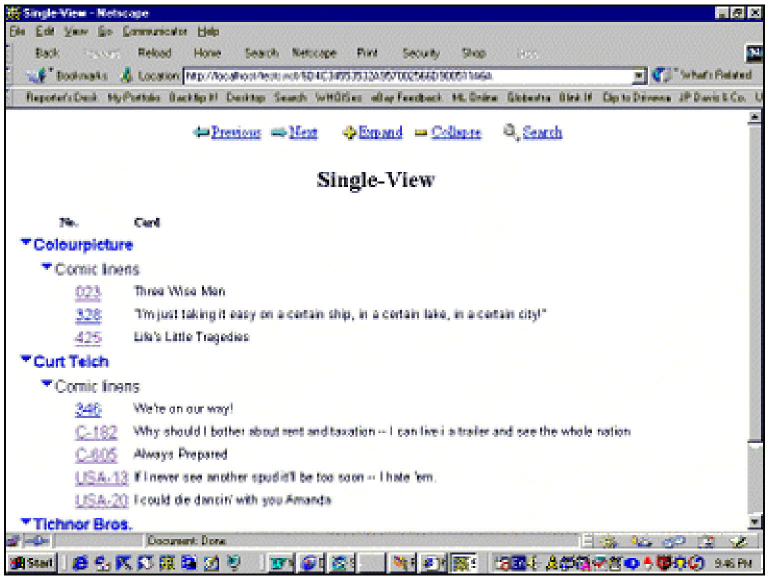
Location: Left

Image

With the applet, Web users can see the same action icons that appear in the Notes client. They can scroll across the action bar if it is wider than the screen. Actions that have drop-down menus in Notes appear as a second row of smaller buttons when the user clicks the main action.

Tips and techniques for views

The default display of a view in a browser is functional, but not very exciting, as this example shows:


Single-View - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location What's Related
Previous Next Expand Collapse Search
Document: Done
Start

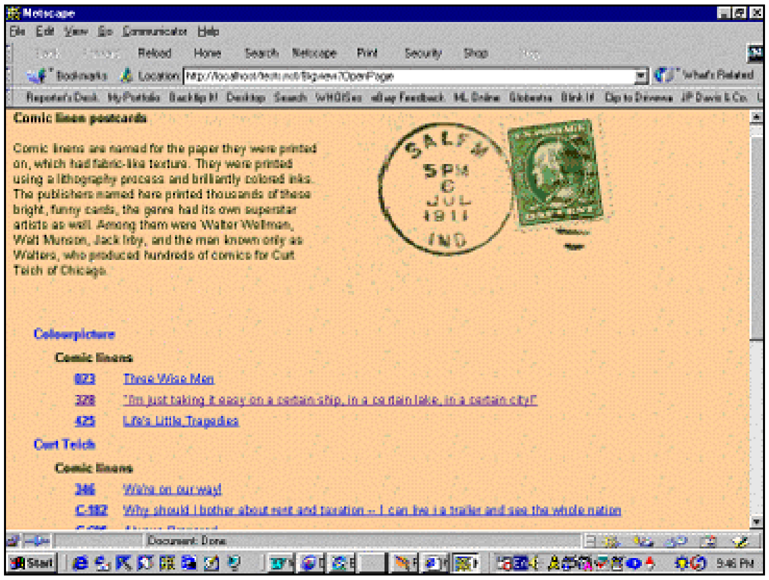
Single-View

No.	Card
▼Colourpicture	
▼Comic linens	
023	Three Wise Men
328	"I'm just taking it easy on a certain ship, in a certain lake, in a certain city!"
425	Life's Little Tragedies
▼Curt Teich	
▼Comic linens	
346	We're on our way!
C-182	Why should I bother about rent and taxation -- I can live i a trailer and see the whole nation
C-605	Always Prepared
USA-13	If I never see another spud it'll be too soon -- I hate 'em.
USA-20	I could die dancin' with you Amanda
▼Tichnor Bros.	

Domino renders the major features of the view faithfully. Users can collapse and expand categorized views, and navigation is provided to move the user to previous and next pages if the view is a long one. There is a link to bring up a search box.

This out-of-the-box interface doesn't have any of the graphical appeal of a well-designed Web application, and it doesn't provide some of the functions most frequently used in Notes applications—the ability for a user to select multiple documents, for example.

Customize views with HTML and JavaScript
Fortunately, views can be customized to help make them more Web-like and more functional. The most basic improvement is to embed a view in a page and write some HTML code. Here's the same view embedded in a page:


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location What's Related
Document: Done
Start
9:46 PM

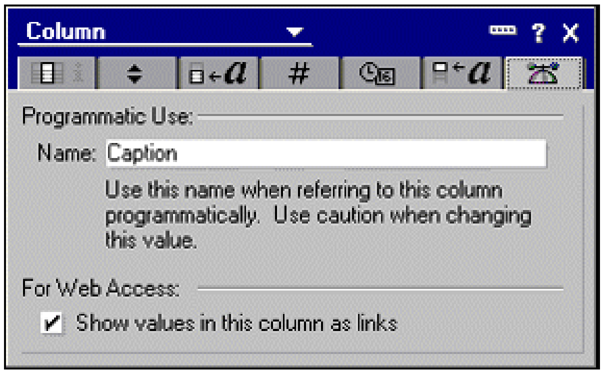
Comic linen postcards

Comic linens are named for the paper they were printed on, which had fabric-like texture. They were printed using a lithography process and brilliantly colored inks. The publishers named here printed thousands of these bright, funny cards, the genre had its own superstar artists as well. Among them were Walter Wellman, Walt Munson, Jack Irby, and the man known only as Walters, who produced hundreds of comics for Curt Teich of Chicago.

Colourpicture
Comic linens
023 Three Wise Men
328 "I'm just taking it easy on a certain ship, in a certain lake, in a certain city!"
425 Life's Little Tragedies
Curt Teich
Comic linens
346 We're on our way!
C-182 Why should I bother about rent and taxation -- I can live in a trailer and see the whole nation

The page lets you add text and graphics, and even a background image file. The generic navigation graphics and links are gone, as well, and you can replace them with your own, or put the page in a frameset and pair it with a navigator or an outline to provide navigation.

The view displayed above has been further customized. The twisties—those graphics that indicate a category can be expanded or collapsed—have been removed, and the contents of the right-hand column have been made into hyperlinks by selecting "Show values in this column as links" on the Advanced tab of the Column properties box:



Programmatic Use:
Name: Caption
Use this name when referring to this column programmatically. Use caution when changing this value.
For Web Access:
[✓] Show values in this column as links

You can change the color of the twisties, or make them disappear completely from individual embedded views by writing a JavaScript function called "imagereplace" into the form that carries the embedded view. To recolor the twisties, create two GIF files, OPEN.gif and CLOSE.gif, and save them in your application as Image Resources. Copy the JavaScript for the imagereplace function into the JS Header object:

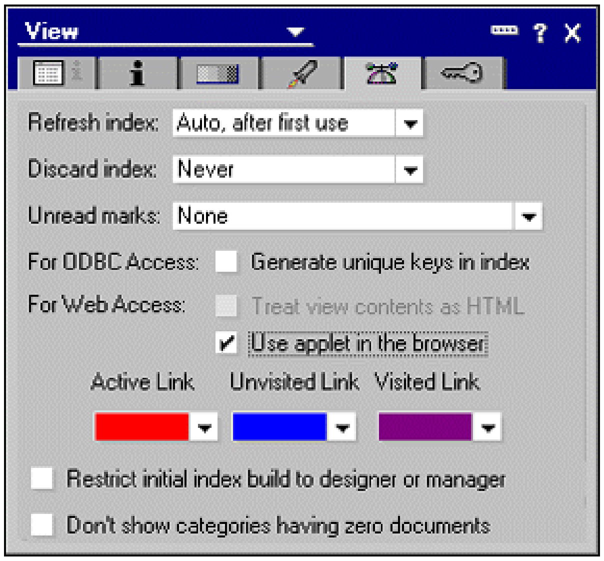
```
function imagereplace()
    {
    for( i=0; i<document.images.length ; i++ )
    {
    if( document.images[i].src.indexOf('expand.gif') != -1)
    {
    document.images[i].src=src =
    '/DATABASE.nsf/OPEN.gif?OpenImageResource'
    } //end if

    if( document.images[i].src.indexOf( 'collapse.gif' ) != -1)
    {
    document.images[i].src= src =
    '/DATABASE.nsf/CLOSE.gif?OpenImageResource'
    } //end if
    } // end for
        }
```

And finally, in the page's onload JavaScript event enter "imagereplace()" to execute the imagereplace function when the page is opened in the user's browser.

To remove the twisties, make a single small transparent GIF file and change the imagereplace code to replace the collapse.gif and expand.gif files with your file. This leaves the twistie functionality in place, but makes it invisible.

Use the view applet
The view applet, new in R5, gives browser users many of the features of views in the Notes client, including column resizing, collapse and expand, multiple document selection, and vertical scrolling. To enable a view to use the Java applet display in a browser, open the View properties box, and on the Advanced tab, select "Use applet in a browser" in the For Web Access section:

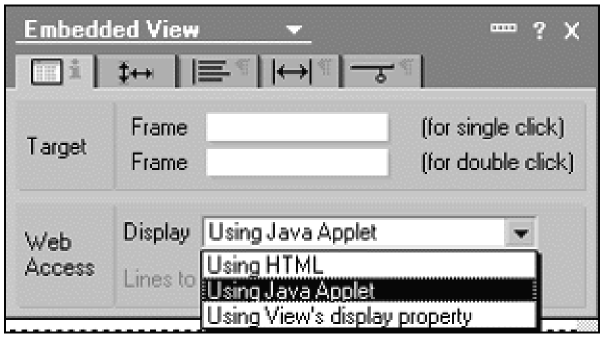


Refresh index: Auto, after first use
Discard index: Never
Unread marks: None
For ODBC Access: ☐ Generate unique keys in index
For Web Access: ☐ Treat view contents as HTML
✓ Use applet in the browser

Active Link	Unvisited Link	Visited Link
🟥	🟦	🟪

☐ Restrict initial index build to designer or manager
☐ Don't show categories having zero documents

You are not limited to just one presentation of the view in your application. If you embed the view in multiple forms or pages, you can set the properties of that object to follow the view's setting and display using the Java applet in one instance or override it and display the view as HTML in another. To set the embedded view's display properties, in Designer, open the page in which the view is embedded, and then edit the Embedded View properties:

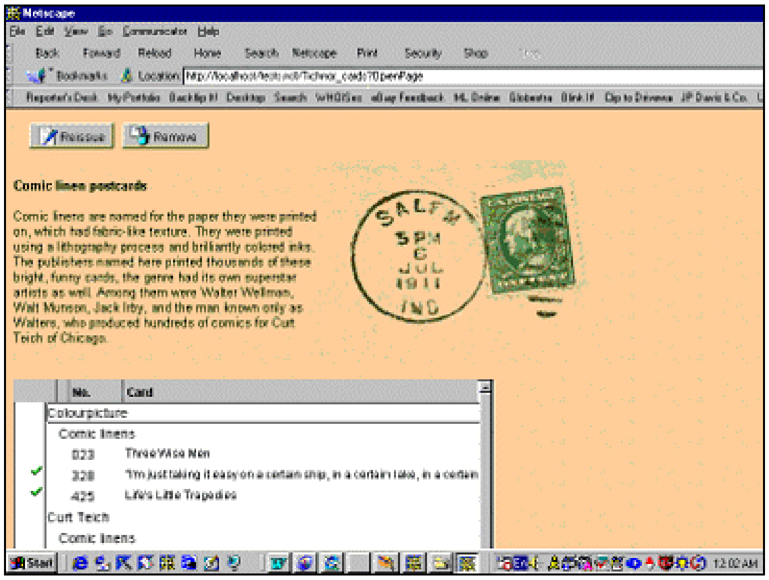

Embedded View

Target	Frame		(for single click)
	Frame		(for double click)
Web Access	Display	Using Java Applet	

The view applet supports multiple document selection. This allows you to move applications to the Web that require users to select a group of documents for processing by an agent. If you embed a view in a page, you can create hotspots or buttons that submit the list of selected document IDs for processing by an agent on the server. The programming required to do this depends on LiveConnect, a capability built into Internet Explorer, Netscape Navigator, and the Notes client. LiveConnect allows JavaScript to interact with components, such as Java applets, that are written in other languages. The view applet, like the other Domino Java applets, has an API that can be addressed by JavaScript in your application. For an example of the code you would use to create a list of the UNIDs of documents selected in the view applet and pass this list to a server agent, see **Getting a handle to documents selected in a view applet**, one of the **JavaScript lessons** from BinaryTree Inc. on the **Lotus Technology Learning Center**.

(There are HTML solutions to selecting multiple documents, too. See the *Iris Today* article "**Combining forms and views for friendlier Web applications (Part 2)**" for an ingenious implementation of checkboxes or list boxes in views.)

Create an action bar
If your application uses cascading menus for actions, you'll want to enable the action bar applet in the Web view or page. This applet puts your application's actions into a row of buttons across the top of the HTML page, as shown in the screen below. If there are submenus of actions, the applet displays them as a row of smaller buttons when a top-level choice is clicked. See the **Agents and actions** section under "Tips and techniques for forms" above for more about enabling the action bar applet.


Comic linen postcards

Comic linens are named for the paper they were printed on, which had fabric-like texture. They were printed using a lithography process and brilliantly colored inks. The publishers named here printed thousands of these bright, funny cards, the genre had its own superstar artists as well. Among them were Walter Wellman, Walt Munson, Jack Irby, and the man known only as Walters, who produced hundreds of comics for Curt Teich of Chicago.

	No.	Card
	Colourpicture	
	Comic linens	
	023	Three Wise Men
✓	328	"I'm just taking it easy on a certain ship, in a certain lake, in a certain
✓	425	Life's Little Tragedies
	Curt Teich	
	Comic linens	

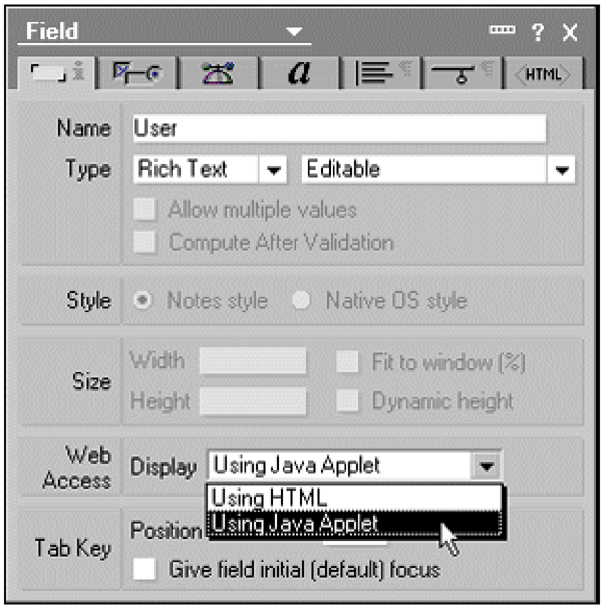
Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location
Reissue
Remove
SALEM
5 PM
6
JUL
1911
IND
Start
12:02 AM

Tips and techniques for data acquisition

Web browsers impose limitations on all but the most basic kind of fill-in-the-blank data collection from users. Data types are restricted, and things that Notes developers and users take for granted, like Rich Text and file uploads, require much more attention.

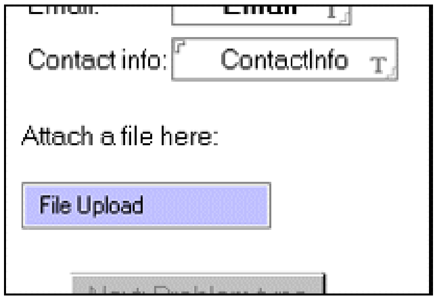
Release 5 adds many new features to make data acquisition in a browser work more like a Notes client. If you have a Rich Text field in a form, you can specify that when that form is edited from the Web, a Java editor applet be called to support some of the familiar Rich Text formatting conventions.

To enable a field for the Java editor applet, set its Web Access property to "Display Using Java Applet:"


Field

Name: User
Type: Rich Text, Editable
Allow multiple values
Compute After Validation
Style: (● Notes style) Native OS style
Size: Width, Height; Fit to window (%), Dynamic height
Web Access: Display: Using Java Applet
Using HTML
Using Java Applet
Tab Key: Position
Give field initial (default) focus

Many of the things Notes users use Rich Text fields for, like pasting in graphics or uploading files, are not supported by the applet—there is a separate file upload control you must include in your form if that's something you want to allow users to do. To insert a file upload control into a form, choose Create - Embedded Element - File Upload Control. The control will appear in the form like this:



Intermediate results
One of the most profound differences between applications developed for Notes clients and those developed for a Web browser is the use of intermediate results. The interaction between Web browsers and a Web server is severely limited: a browser requests a URL from the server, and the server returns the requested URL to the browser. The requested URL may be an HTML page, or it may be a server action. This is how HTML forms work: the client sends a query string that includes variable names and values and a URL on the server that identifies a script that the server executes to process the contents of the query string.

Notes forms, in contrast, are much more flexible and allow for intermediate results. Your application may use an @DbLookup formula to let the user choose a value from a list, then use that value in other computations in the open form. This is not possible in a Web browser. You can approximate the same result by having the user select the document containing the value, and then passing this document ID to a server agent that looks up the value, performs the computations, and writes a new HTML page, which is sent to the user.

CGI variables
Notes makes a great deal of environmental data easily available to the developer. If you want to identify the user you include @UserName in a formula. Other information about the current application environment is equally easy to access using @Author, @DbName, and @ServerName. Similar information is available to applications on the Web, but you must take an extra step to acquire it.

Every request from a Web browser is accompanied by a set of CGI (Common Gateway Interface) variables that pass information about the user's environment, including such data as the user's name, the browser type, and the user's Internet Protocol (IP) address. Domino treats CGI variable names as reserved names—when it encounters a field on a form with the name of a CGI variable, Domino copies the field value from the CGI environment and places it in the field. In order to use a CGI variable, you must have a named field for it on your form.

One very basic use for this information is to smooth over a difference between the Notes client and a browser. In Notes, @DbName returns not only the name of the current database, but the name of its server, as well. On the Web, however, @DbName returns only the name of the database. If you're trying to compute a URL, you have to find the server name somewhere else. The place to look is a CGI variable called, appropriately, server_name. An easy shortcut is to add a field to your form called server_name.

This can be especially useful if you build and test your application on a development server, and then move it to a production server. Using CGI variables eliminates much of the hardwiring of server and database names that must be corrected by hand to put an application into production.

The complete list of data available as CGI variables is a long one. It is available in **Domino 5 Designer Help** under the heading "Table of CGI Variables."

Tips and techniques for the user interface and graphics

As you move your application from the Notes environment to the Web, you will face two challenges in the user interface design. On the one hand, the browser's lack of counterparts for many Notes features and functions may force you to lower your expectations for ease of use, if not actual reduced functionality. But on the other hand, you have to meet Web users' expectations for graphical design and visual impact if you want your application to be taken seriously. This will probably require work on your application's graphics and navigation, in particular.

The good news is that Release 5 supports most of the advanced features of current browsers, so the more HTML and JavaScript you know, the more Web-like your application can be. If you've been impressed by Web sites with layered interfaces built on Dynamic HTML, for example, see the *Iris Today* article "**Teach Domino New Web Tricks with DHTML**," which shows you how to use DHTML to create a layered interface in a Domino application.

But you aren't required to know DHTML to turn a Notes application into a good Web application. You just have to master some basics and use them intelligently—and take advantage of Release 5's ability to provide some visual appeal.

Use tables to control placement

Tables are the best way to control the placement of elements in a Web page. If you have any experience with HTML, you'll appreciate the problems of getting elements such as fields, text, and graphics to line up properly. Part of translating your application to a browser will probably involve putting these elements into tables on your forms to make alignment simpler.

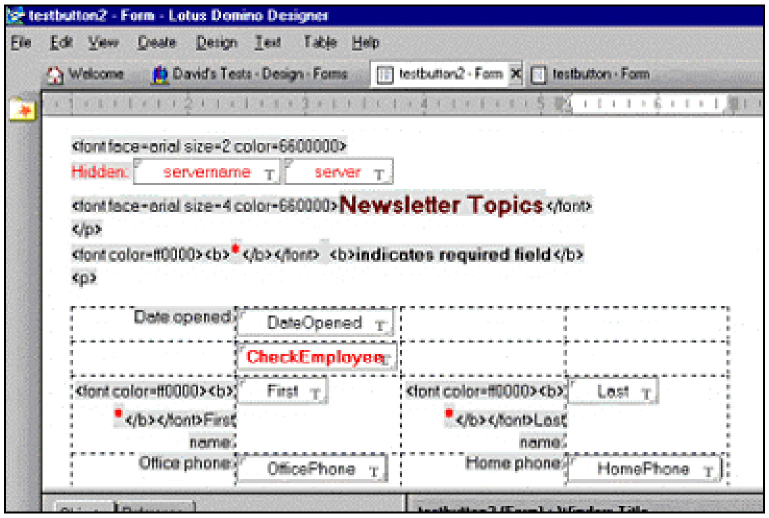
One problem with using tables to control page layout occurs if you place large Rich Text fields in table cells that are deeper than a single printed page. If you print the Web page, cell content is truncated at the end of the first page. One solution is to create a separate form for printing the document without tables and giving the user a button in the table to switch to the form without tables.

Tip: Be careful about moving back and forth between versions of Domino Designer. Once you have edited a form containing a table in R5 Designer, don't re-edit it in an earlier Designer version or the table can be corrupted.

Validate user input data

If your application does sophisticated input validation using LotusScript, much of it can probably be transformed into JavaScript to work in a browser as long as it uses data already in the document. If validation depends on external data—lookups into other documents or databases, or the acquisition of document UNIDs, for example—you will have to move such actions to server agents.

Simple validation is still easy to do with @functions and @commands. Here is an example that checks a form to make sure that five fields have been filled in: first and last names, a category, an item name, and a description. This screen shows the top part of the form open in Designer:


testbutton2 - Form - Lotus Domino Designer
File Edit View Create Design Text Table Help
Welcome
David's Tests - Design - Forms
testbutton2 - Form
testbutton - Form

```
<font face=arial size=2 color=6600000>
Hidden: servername T server T
<font face=arial size=4 color=660000>Newsletter Topics</font>
</p>
<font color=ff0000><b>*</b></font> <b>indicates required field</b>
<p>
```

Date opened:	DateOpened T		
	CheckEmployee T		
<font color=ff0000><b>*</b></font>First name:	First T	<font color=ff0000><b>*</b></font>Last name:	Last T
Office phone:	OfficePhone T	Home phone:	HomePhone T

The field named CheckEmployee, in red above, is computed for display. You should give it its own name as a value and set it to hide when "CheckEmployee=""". Each data entry field has a similar "check" field.

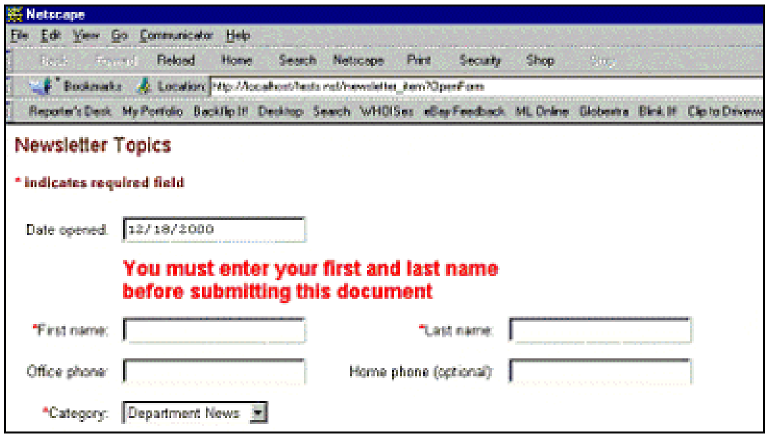
The work of validation is done by the Submit button formula:

```
FIELD CheckEmployee := @If(First = "" | Last = "";"<font face=arial size=4
color=ff0000><b>You must enter your first and last name<br>before
submitting this document";"");
FIELD CheckCategory := @If(Category = "";"<font face=arial size=4
color=ff0000><b>You must choose a category<br>for this item";"");
FIELD CheckItem := @If(First = "" | Last = "";"<font face=arial size=4
color=ff0000><b>You must enter a a name<br>for this item";"");
FIELD CheckDescription := @If(First = "" | Last = "";"<font face=arial size=4
color=ff0000><b>You must enter a description<br>for this item";"");

NoProblems :=
CheckEmployee+CheckCategory+CheckItem+CheckDescription;

@If(NoProblems = "";
@Do(@Command([FileSave]);
@Command([FileCloseWindow]));
@Command([ViewRefreshFields]))
```

If any of the data fields is empty, its check field is assigned an error message. A temporary variable, NoProblems, aggregates the check fields. If there are no empty fields, NoProblems has no length and the @Do command saves the file and closes the window. If any field is empty, the @Do instead executes the @Command([ViewRefreshFields]) condition, which makes the error messages visible:


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://localhost/tests.nsf/newsletter_item?OpenForm
Reporter's Desk My Portfolio Backflip It! Desktop Search WHOISes eBay Feedback ML Online Globester Blink It! Clip to Drivewa

Newsletter Topics

* indicates required field

Date opened: 12/18/2000

You must enter your first and last name before submitting this document

*First name: *Last name:

Office phone: Home phone (optional):

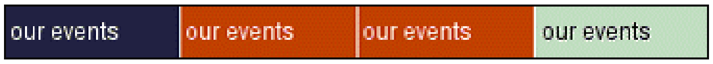
*Category: Department News

This minimal validation lacks the elegance of scripting solutions that allow you to place focus on the field that lacks input, but it does allow you to serve the needs of the users for feedback—and yields better, more complete data with little effort on your part.

Graphics improvements in R5
Release 5 makes major improvements in the handling of graphics. One of the most important is native support for the two most widely used Web file formats, JPG and GIF, as Image Resources. When you import a file into a Domino R5 application, it is stored in its format rather than converted to a proprietary format (which is what happened in previous versions). You no longer need to worry about color palettes and unexpected color shifts. However, this doesn't work if you paste the image, only if you import it. For more information on graphics, see the *Iris Today* article "**The Graphic Truth about Notes**."

The image resource is a great improvement for managing graphics within your application. Use the image resource to save an image just once in an application and then use it on many forms or pages. Image resources replicate with the application so that even if there are multiple copies of your application in service, as long as they all replicate, you can update icons or logos in all their occurrences by replacing just one copy.

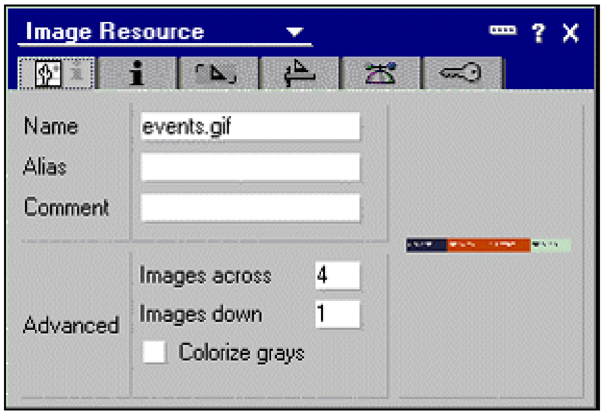
Easy rollover buttons
The image resource is a good example of an easy way to use Release 5 to make your application more Web-like. Domino can automatically turn properly prepared image resource files into rollover buttons that change state when you mouse over them. Use a graphics editor to assemble the various states of the image into a single graphic with one pixel of separation between the images:



Domino will recognize up to four states for the graphic:
- The image in the first position on the left is treated as the normal image.
- Second position is the image displayed on mouseover.
- Third position is the image when selected.
- Fourth position is the image when clicked.

This order is fixed, but you can use only two or three states by inserting an image multiple times to fill the array. The example above uses the same image for the mouseover and selected states, for example, so that the clicked state can display a different image. If you want only normal and mouseover, you can leave the last two positions empty.

Next, import your multi-state graphic into your application as an image resource, and indicate how many images it contains in its properties box:



Name events.gif
Alias
Comment
Advanced
Images across 4
Images down 1
Colorize grays

Domino automatically uses this information to make the image display properly both in the Notes client and on the Web. (The entry for "Images down" is used for multiple sizes of bookmark icons. For more on image resources, see **Domino 5 Designer Help**.)

Tip: If you add images to forms or pages by writing HTML image tags, you can improve performance by including the graphic's dimensions:

"<IMG SRC=myimage.gif width=264 height=128>"

(Domino does this automatically for image resources.) You can also make your application more user-friendly by adding alternate text that appears when an image can't or won't be displayed. In HTML, write it into the image tag:

"<IMG SRC=myimage.gif ALT="Photograph of new manufacturing facility">"

For image resources, add text in the Alternate Text field on the Information tab of the Picture properties box.

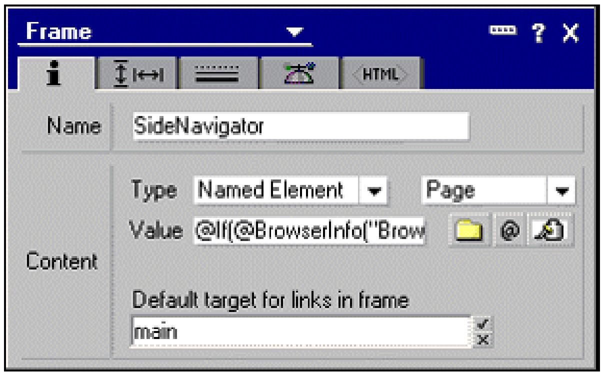
Navigation
Web applications typically need more elaborate navigation built into them than Notes applications because browsers lack the built-in navigation provided by the Notes menus. Release 5 adds several features that you can use.

The problem with any Web application is that you need to separate its navigation as much as possible from individual documents so that when its navigational structure changes, as it inevitably will, the changes can be made by modifying as few documents as possible. On pure HTTP servers, this can be done with server-side includes or .ASP pages to combine a panel of navigational links into documents as they are served to users. In a Domino application, a subform can often perform the same role. Or you may embed an outline in a page or form.

Outlines are a new Release 5 feature that you can use to create flexible, high-level navigation across the areas of an application or a Web site. Outline entries can be programmed with commands such as hide-whens so that a single outline can appear differently in different areas of the application. An outline's appearance can also be tailored in each instance of its use. A Java applet is available to make outlines work in a browser as they do in a Notes client. For more information, see the *Iris Today* article, "**Domino Designer R5: The Outline Designer**."

Frames
Framesets are often used to place a navigation pane next to content. Domino Designer R5 makes framesets easier to create with a graphical frameset designer. You can size your frames by dragging them, or set frame attributes such as size, scrolling, border colors and width, and frame spacing in the Frame properties box.

You can specify the contents of each frame in advance, or have it computed when the frameset is opened. The following Frame properties box shows part of a formula that computes the name of the navigator to display in the frame. Formulas can control frame content based on client type or CGI variables including user name, so you can write a display formula based on user roles.



Name SideNavigator

Content
Type Named Element Page
Value @If(@BrowserInfo("Brow
Default target for links in frame
main

You can set a frameset to launch automatically when a database, form, or page opens. For more on framesets, see the *Iris Today* article "**Domino Designer R5: Framesets**."

Framesets and navigators are an easy way to develop consistent navigation for a site, but remember that users often find framesets confusing because it is often hard to reverse direction and navigate back along a path you have followed in a frameset. Use them with care.

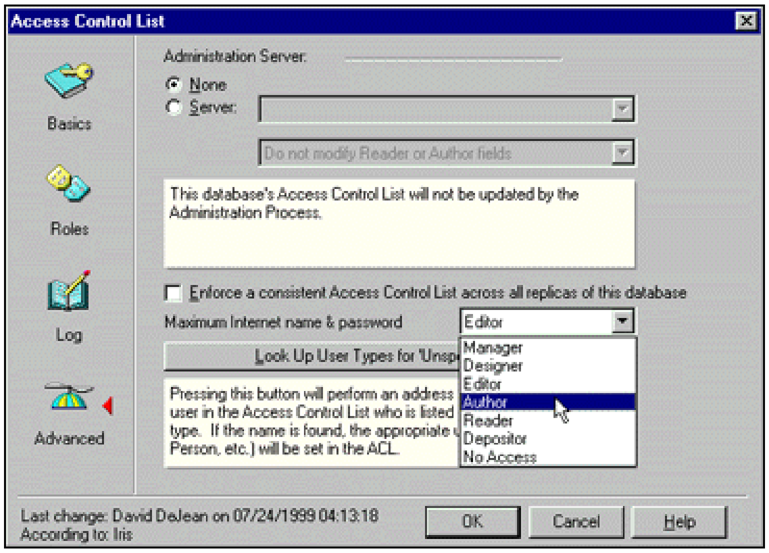
Tips and techniques about access and security
Setting access to your application is often a thorny problem. On the one hand, allowing anonymous access is easiest. Forcing users to authenticate may become an administration problem, because every authenticating user must have a record in the server's Domino Directory. But on the other hand, users must have at least Author access to create new documents in a Domino application. Many Notes administrators view setting the Anonymous role to Author access as an invitation for abuse. Their solution is to set Anonymous access for most Notes databases to "no access," set the Default role to Author, and force users to authenticate.

If your application is intended for corporate users on an intranet, this isn't a problem—most corporate Notes applications are probably intended for use by authenticated users. But many Web applications are intended to permit anonymous, unauthenticated access.

Restricting Anonymous users to "no access" has some problems in addition to the administrative burden. One of them is the need to create some Public Access forms that are visible to all users. We saw this in the discussion of the $$ReturnAuthenticationFailure form. If you require Web users to authenticate but their authentication fails, they won't see an error message unless you create one and set it for Public Access.

If your application is aimed at giving Web users read-only access to information, it can work well with Anonymous access set to Reader. Just remember that this may limit users' ability not only to create, edit, and delete documents, but to run agents that create, edit, and delete documents.

If you would like to limit users to one level of access from the Web, while allowing them their customary access from a Notes client, you can use the "Maximum Internet name & password" setting on the Advanced tab of the ACL. It's a drop-down list, as shown in the screen below:

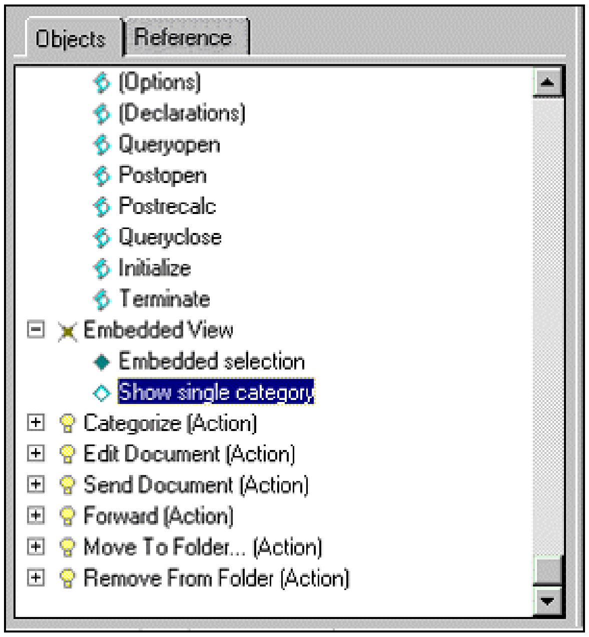

Access Control List
Basics
Roles
Log
Advanced
Administration Server:
None
Server:
Do not modify Reader or Author fields
This database's Access Control List will not be updated by the Administration Process.
Enforce a consistent Access Control List across all replicas of this database
Maximum Internet name & password
Editor
Manager
Designer
Editor
Author
Reader
Depositor
No Access
Look Up User Types for 'Unsp
Pressing this button will perform an address user in the Access Control List who is listed type. If the name is found, the appropriate Person, etc.) will be set in the ACL.
Last change: David DeJean on 07/24/1999 04:13:18
According to: Iris
OK
Cancel
Help

If you set the access level for your application to Reader here, for example, all Web users will be restricted to Reader access, even though they might have Author or Editor access when they access the application from a Notes client.

For maximum security, use Readers fields
The best way to restrict access to documents in your application is to place restrictions in the form that creates the documents. You can do this in two ways—by editing the settings on the Security tab of the Form properties box, or by placing a Readers field on the form itself. Enter the names of individuals, groups, and roles that should be able to see the documents, and nobody else will be able to open the documents or even see them listed in views.

The choice of whether to use property settings or fields is up to you. Both have advantages. Values entered in a Readers field are combined with the properties list. It is easier to use the properties setting as you create or edit the form, because the selection dialog box does a lookup in the Domino Directory. If you use a Readers field, you can compute its value when the document is created, and use Domino administration to modify or update some or all of the Readers fields in a database at once.

Another option is to limit what's displayed in the views in your application. This works only with embedded views; you can restrict the displayed contents of the view to a single category by opening the page or form that holds the embedded view in Designer and selecting "Show single category" on the Objects tab of the Programmers pane, as this screen detail shows:



Then in the Script area of the Programmers pane, enter the name of the category or write a selection formula that evaluates to a category name. You can use this technique to display documents only to their authors or only to members of specified groups. For example, the following formula displays the appropriate category to users who are members of one of four groups—Admin, Sales, HR, or IS:

```
@If(@Contains(@UserRoles;"Admin");"Admin";
@If(@Contains(@UserRoles;"Sales");"Sales";
@If(@Contains(@UserRoles;"HR");"HR";
@If(@Contains(@UserRoles;"IS");"IS";
""))))
```

One final fillip: if the formula evaluates to an asterisk, all categories are displayed. This means that you can show the entire view to some users and restrict it for others, or by using @ClientType, you can create a view that will display only a single category in a browser, but its entire contents in a Notes client:

```
@If(@ClientType = "Web";"Newsflash";"*")
```

Help is at hand

Most of the challenges you face in converting Notes applications to Web applications involve finding ways to make things Notes does work in a browser. If you're a Notes developer who's just beginning to work on applications for the Web, places to start looking for answers are close at hand:

- The **Domino 5 Designer Help** is a great source, not only of information but of examples that can show you how things are done.
- You can check out the **Iris Sandbox**, a library of downloadable applications and code samples on Notes.net.
- Another good source of tutorials and code is the **Lotus Technology Learning Center**. The Learning Center provides self-paced tutorials that emphasize hands-on work. For example, the Killer Web Apps 2 offering from BinaryTree includes content on CSS, DHTML, HTML, JavaScript, XML, and more.
- Don't forget the **Notes/Domino Gold Release Forum** on Notes.net.

Many of the tips and ideas in this article first appeared there. Generally, if you can't find the answer in the forum, all you have to do is ask the question.

- And *Iris Today* has a number of relevant articles; check out the archives or perform a full-text search for the topic that interests you. If you're just moving to Release 5 from a previous version of Notes, **Domino Designer R5 Technical Overview** and **Domino R5 Technical Overview**, and the articles they are linked to, are good places to start.
- For more on R5 best practices, you can download the **Lotus Notes and Domino Application Development Best Practices Guide**.

This article, along with these resources, should take you a long way toward the successful migration of your Notes application to the Web.