Webifying an existing Notes application

by Teresa Deane

[Editor's note: This article resides in "Iris Today", the technical Webzine located on the http://www.notes.net Web site produced by Iris Associates, the developers of Domino/Notes.]

Suppose your boss comes to you and asks you to make a great little Notes application work on the Web. Your boss may think that you just need to "flip a switch" in a few places. However, to do it right, there are several issues that you should consider before you even start coding. This article will walk you through some of the decisions you need to make when developing an application for both Notes and Web users. To give you a real-life example, we'll take you on a tour of how we "webified" the Discussion template (discsw46.ntf) using the new Notes Designer for Domino 4.6.

The goal of this article is to simply help you *get started* modifying an existing application for the Web. For more information, please see the *Application Developer's Guide* (which is also available as part of the online Help). In particular, you might want to refer to Appendix D, "*Features to avoid using in Web applications*." For more information on Notes Designer, please see the article "*New Web design features in Notes Designer*," which was published in an earlier issue of *Iris Today*.

This article is organized into three parts:

1. Planning your application
2. Designing forms and views
3. Putting your application to work

Part I: Planning your application

This is a situation where a little planning will go a long way. Before you start coding, you should first define your new Web audience and think about how the application should work for those users. Regardless of how simple the application is, you don't want to waste time porting a feature that wasn't worth porting to begin with!

Step 1. Evaluate each task to see if it's appropriate for Web users, and what type of access is required.

For example, administrative tasks that require Manager access may not be appropriate for your Web audience. Also, keep in mind that there are *two* places in the database's access control list (ACL) that control access from a Web client. The Basics section of the ACL lists the access levels for individual users, groups, and servers. In addition, the Advanced section specifies the maximum level of access allowed from the Web in the "Maximum Internet name & password access" field, which overrides individual levels set in the ACL. The LotusScript actions in your application may require Manager or Designer access that is not allowed from the Web. If you are creating templates, keep in mind that this setting does not get inherited from the template, and that the default setting is Editor.

In the Discussion template, the archiving action is a good example of where we needed to consider the access level required. The archiving action only appears to editors. If users with Editor access try to run the archiving action, they get a message saying that they need at least Designer access. In addition, the WebArchiveSave agent tries to enable the archiving agent, which is also something that requires at least Designer access.

Also, consider if you want tasks available to Web users who are Anonymous (that is, users who have not authenticated their identity). If you do not add an Anonymous entry to the application ACL, all Web browser clients have the Default level of access. The Discussion template has Anonymous set to "No access" because the Interest Profile actions are not available to anonymous users. For examples on how to set up the ACL for a Web database, see the *Application Developer's Guide*.

Step 2. Decide how you will use graphics. (and consider how this might affect performance.)
When designing for the Web, people tend to think of graphics. However, if performance is an issue, you might want to sacrifice the fancy graphics in favor of better performance. (As a side note, Notes now allows you to add alternate text descriptions to graphics so that users who turn off graphics, or use a browser that doesn't support them, can still participate in your application.) The redesign of the Discussion template included a major graphics overhaul. We decided to make the most of the Navigator features in Notes, and to also display Web actions as graphic hotspots.

Step 3. Determine if your user community will be using both Notes and the Web as their clients.
If you have certain members of your user community who only use the Web and certain users who only use Notes, your task is a little easier. You don't have to worry about fields that aren't supported on the Web and vice versa. You can actually come up with a separate form for each user. For example, the Combined Mail template (mailc46.ntf) opens separate forms for a user, based on how he/she is accessing the database (Web or Notes).

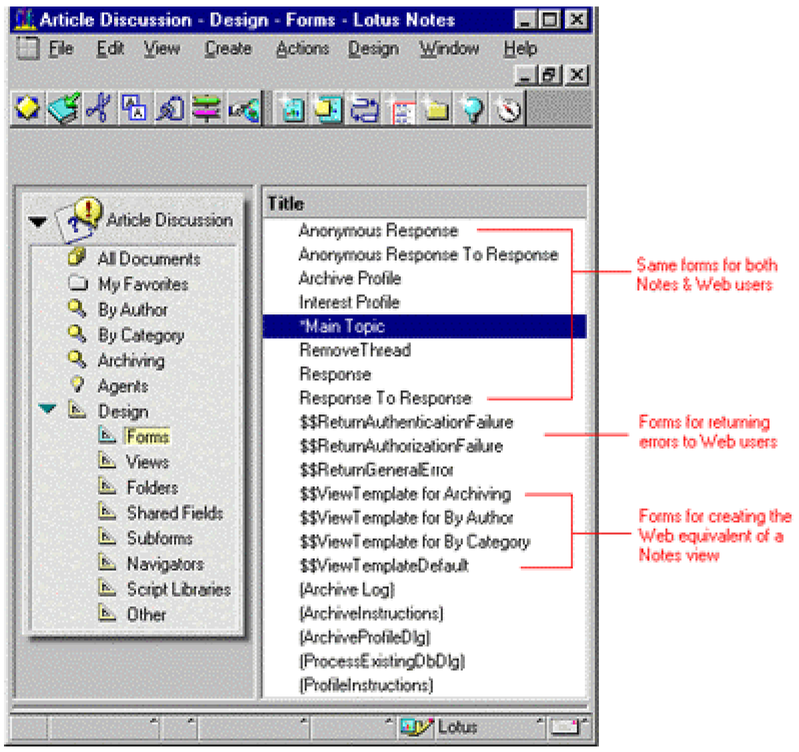
However, if the same user will be modifying documents using both a Web client and a Notes client, your task becomes harder because you must iron out more issues beforehand, like how you will handle fields and data synchronization between the fields. The Discussion template is designed for both Notes users and Web users, with the additional caveat that Notes users can edit a document that they created in a Web browser. This provides more flexibility for the users.

Step 4. Examine your LotusScript code to see how much you can use for the Web.
Ideally, you should try to share as much code as possible. Then, if you need to modify the code in the future, you'll be able to make the changes all in one place. (It will also help you keep the size of your application down.) Only the back-end classes are supported on the Web, so you can have a separate script library for Notes to access for front-end classes.

Part II: Designing forms and views

Now that you have a well-defined application and user audience, let's start designing with forms. Form design is a very important part of any Notes database design, but it is much more involved when designing for the Web.

The following screen shows an overview of the forms we used in the Discussion template.

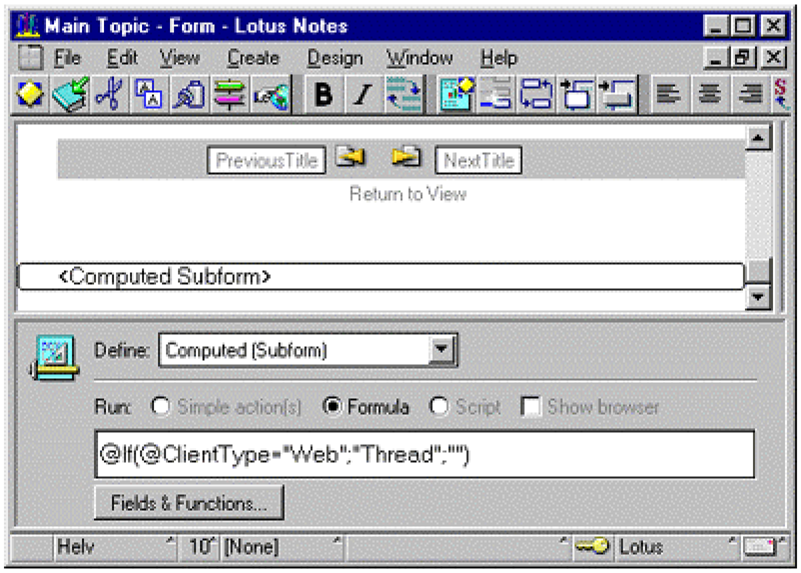


Anonymous Response
Anonymous Response To Response
Archive Profile
Interest Profile
*Main Topic
RemoveThread
Response
Response To Response
$$ReturnAuthenticationFailure
$$ReturnAuthorizationFailure
$$ReturnGeneralError
$$ViewTemplate for Archiving
$$ViewTemplate for By Author
$$ViewTemplate for By Category
$$ViewTemplateDefault
(Archive Log)
(ArchiveInstructions)
(ArchiveProfileDlg)
(ProcessExistingDbDlg)
(ProfileInstructions)

Step 1. Decide if you will have separate forms for the Web, or perhaps the same form with just different subforms for Web users.

With both methods, the user won't find much difference in performance. However, maintenance might be easier if you choose to use one form with separate subform components. Computed subforms for Web/Notes users are much easier to do with Notes Designer because of the new @ClientType function. You no longer need to clutter your application with roles. Also, the hide-when enhancements, "Hide from Notes" and "Hide from Web", make form design much easier, because you can hide specific design elements from one client or the other.

In the design of the Discussion template, we decided to use the same forms for both Notes and Web users. Some design elements, such as fields and hotspots, are hidden either from the Web or from Notes. Also, the "Main Topic" form has one computed subform designed specifically for Web users.


Main Topic - Form - Lotus Notes
File Edit View Create Design Window Help
PreviousTitle
NextTitle
Return to View
<Computed Subform>
Define:

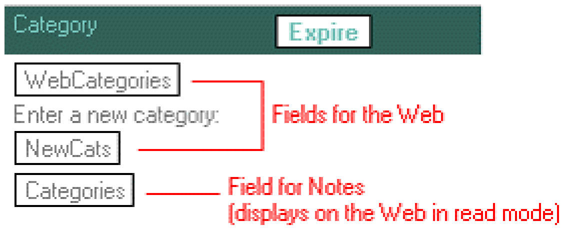
Computed (Subform)

Run: Simple action(s) [✓ Formula] Script Show browser

```
@If(@ClientType="Web";"Thread";"")
```

For an example on how to use separate forms for the Web, please see the Combined Mail template (mailc46.ntf). Basically, you just create two forms with the same alias, and select to hide one from Notes and the other from the Web. Then, the correct form appears automatically in the appropriate environment. (For example, see the forms Memo and (web Memo) in the Combined Mail template.)

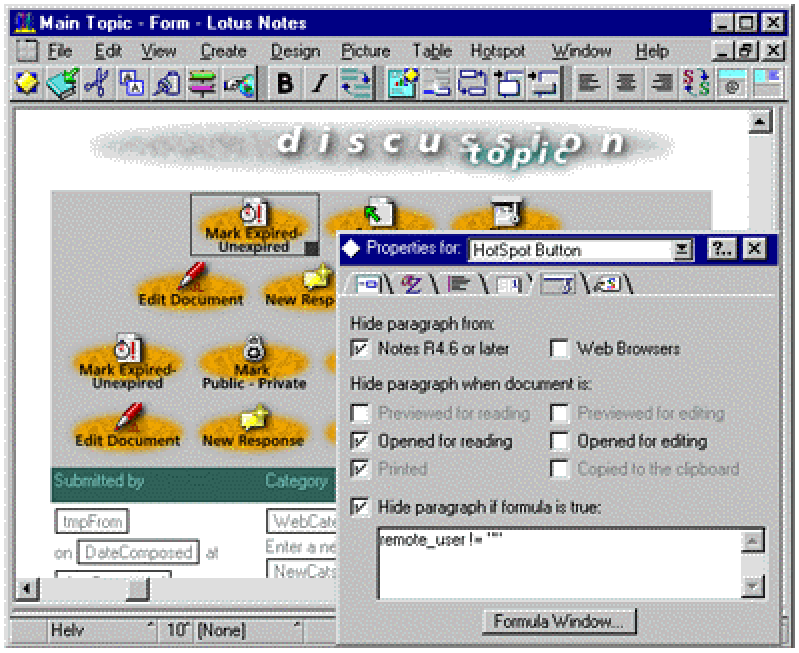
Step 2. Decide how you will design fields for the Web.
Your choice of fields is very different for the Web. For example, the Web doesn't support the concept of a multi-value keyword field that allows users to enter choices not in the list. However, this is a common data type in Notes. We solved this problem in the Discussion template by making it two fields for the Web and one for Notes.



The more complex part comes when you have someone editing the same document between Notes and the Web client. In that case, you have to synchronize the values and clear out the appropriate fields so that every user sees the correct values. To do this, we added a WebQueryOpen agent to the main topic form to synchronize the field values and an input translation formula to add new values to the list from the Web. Even simple things like you can't have a carriage return in a text field on the Web like you can in Notes meant that we needed to add special case profile strings in the Interest Profile form. The benefits of rich text that you don't get on the Web are far too many to mention, but the general idea is to think through everything that could be entered into a field before you decide the data type.

Step 3. Decide what kinds of actions you will include on the form.
When thinking about actions, remember that Web users do not have access to the Notes menu commands. For example, they have no way of getting into edit mode, creating a document, switching views or forwarding documents. You should run through a user scenario in Notes and then run through the same scenario on the Web to identify key actions that need to be available to the Web user. In the Discussion template, we opted against using the canned action bar that Domino provides and created our own action

bar of graphic hotspots that uses the "Hide from Notes" attribute. You will find several action bars on each form. Each action bar has a specific hide-when attribute that applies to it. For example, we have some actions that are not available to non-authenticated users in edit mode.



Properties for: HotSpot Button

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:

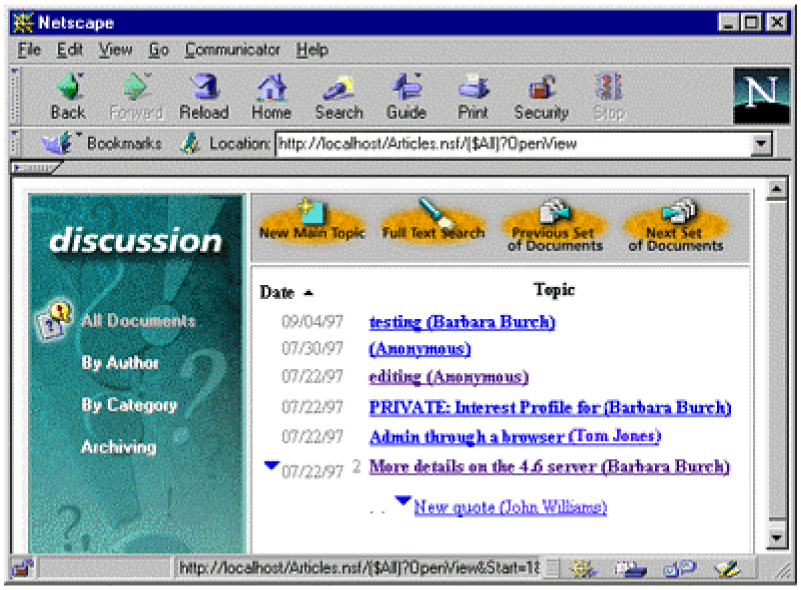
```
remote_user != ""
```

To Web-enable all buttons in a database as well as certain @commands, select the database property "Web access: Use JavaScript when generating pages." Without this property set, Domino recognizes only the first button in a document and treats it by default as a Submit button that closes and saves the document.

Be aware that when you select this "Use JavaScript" property, Domino displays all buttons, actions, and hotspots -- even those that contain @commands and @functions that aren't supported for Web applications.

Step 4. Decide how you will deal with your Notes views.
Although your Notes views will appear OK on the Web, you will lose the attractive, three-paned layout of the Notes interface. You can translate this layout to the Web by using passthru HTML to designate frames. However, another option is that you can embed views, folders, or navigators on a form to create a "frame" effect that helps Web users navigate through your application. The way this works is that you create a form called "$$ViewTemplate for *ViewName*." Then, when a user tries to open the view called *ViewName,* Domino displays the view template instead. If you don't use a $$ViewTemplate form, you may not get the desired display of the view.

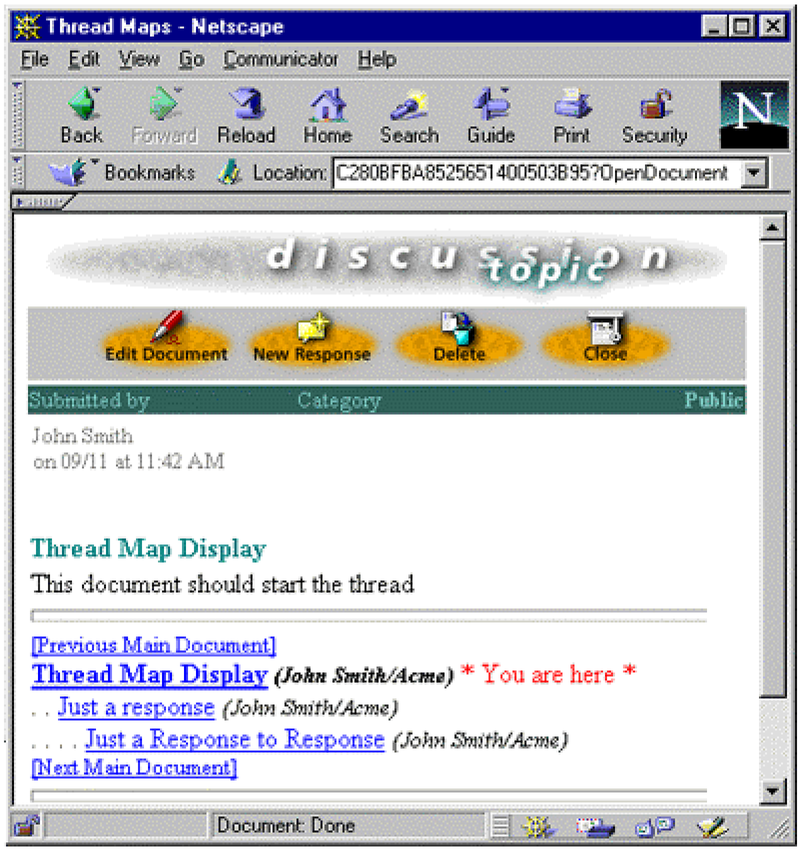
In the Discussion template, we used $$ViewTemplate forms to create this form layout for a view. Because we wanted the same look and feel for Notes users as well as Web users, we used an embedded navigator to help the Web user move between the different views in the database. Above the view is the action bar replacement for the Web. In order to align everything correctly, we placed all the elements in a table. The same navigators are used for Notes as for the Web. The hotspots contain formulas based on the client type to determine whether a view or another navigator is opened.   We also changed the sort order for the documents in the view (again for Web users). Users will see documents sorted from newest to oldest, but they can also click on the column header to sort from oldest to newest.


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://localhost/Articles.nsf/($All)?OpenView
discussion
All Documents
By Author
By Category
Archiving
New Main Topic
Full Text Search
Previous Set of Documents
Next Set of Documents

Date	Topic
09/04/97	testing (Barbara Burch)
07/30/97	(Anonymous)
07/22/97	editing (Anonymous)
07/22/97	PRIVATE: Interest Profile for (Barbara Burch)
07/22/97	Admin through a browser (Tom Jones)
07/22/97 2	More details on the 4.6 server (Barbara Burch)
	. . New quote (John Williams)

http://localhost/Articles.nsf/($All)?OpenView&Start=1

Step 5. Decide how Web users will move between individual documents.
In addition to using an embedded navigator and views (or passthru HTML to designate frames), your users may need a way to move between individual documents. In the Discussion template, we offer a thread map to Web users. On each discussion topic or response, if the client is a Web client, a list of responses in the current thread will appear. The user can navigate anywhere in the thread by clicking on one of the documents in the thread map. However, the thread map requires a separate subform, an additional ThreadMap field on the response documents and a background "Update Thread Maps" agent that runs on new or modified documents.


Thread Maps - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security
Bookmarks Location: C280BFBA8525651400503B95?OpenDocument
discussion topic
Edit Document
New Response
Delete
Close
Document: Done

Submitted by	Category	Public
John Smith on 09/11 at 11:42 AM		

Thread Map Display

This document should start the thread

[Previous Main Document]
Thread Map Display *(John Smith/Acme)* * You are here *
. . Just a response *(John Smith/Acme)*
. . . . Just a Response to Response *(John Smith/Acme)*
[Next Main Document]

Step 6. Decide what other miscellaneous forms are necessary for your Web users.
One final area to consider when designing forms is that Web users don't have access to standard Notes dialogs, such as error dialogs or help dialogs. The new $$Return pages allow you to design customized pages for returning errors to Web users. For example, we used the $$ReturnAuthenticationFailure form to display an error message to Web users when there is an authentication failure. Along with displaying the authentication error message, it gives the user a set of links to get back to the database. Also, since Notes users can access help dialogs from the Profile forms, we designed corresponding help forms for Web users.

Part III: Putting your application to work

Now, you need to get down to the details of making your application work.

Step 1. Decide how your agents will work.
As with any Notes/Web application, agents do most of the work. For example, supporting the concept of Interest Profiles on the Web was no easy task. It required five agents for complete functionality with Notes. Most of the agents might use just a few lines of code, but expect the number of agents in your application to grow. The number of agents that are in your application should not affect performance. Obviously, adding a WebQueryOpen agent to a form will certainly make it take longer to open the document (because Domino must run the agent prior to opening the document). The key is to make the agent perform a few key tasks, and if you need to do complex processing, use a background agent. In some cases, we were able to use existing code in a PostOpen event of a form directly in a WebQueryOpen agent, so the work was minimized. When doing this, make sure that you have the correct document context for the Web agent.

More specifically, you can get to the existing back-end document from a WebQueryOpen or WebQuerySave agent, or by using @Command([ToolsRunMacro]) from a document action by using the NotesSession DocumentContext property. However, if you are executing a Web agent from a hotspot on the form via @URLOpen, you don't have access to the fields on the document (other than CGI fields such as remote_user) . Generally speaking, you should use @URLOpen when you need to pass an argument to the agent or if you are deleting the current document. The way to get the real back-end document from the @URLOpen context is to pass the UNID of the document as an argument to the agent, as shown in the following call:

```
 UNID:=@Text(@DocumentUniqueID );
@URLOpen("/"+@ReplaceSubstring(@Subset(@DbName; -1);" ";"+")+"/WebExpire?OpenAgent&UNID="+UNID)
```

Then, access the UNID from within the agent:

```
set note = session.documentcontext
OriginalUNID = Mid(note.Query_String(0), Instr(note.Query_String(0), "&UNID=")+6, 32)
```

Step 2. Decide how you will deal with Web user names.
The management of Web user names was a challenge as many of the tasks in the Discussion template surrounded the user name. For example, when editing an Interest Profile from the view template, we needed access to what would be the equivalent of the @UserName command in Notes. To solve that problem, we added an editable field with the default value of @UserName and the HTML attribute of "hidden" to every $$ViewTemplate form and just accessed this field to find the correct Interest Profile. You always have access to the CGI variable remote_user from the context document, but that only returns what the user authenticated as. For example, the user may authenticate as *jsmith*, but you want the user's full Notes name, which is John Smith. You could take the remote user and look it up in the Public Address Book to find the correct Notes name, but that would be a performance hit. @UserName returns the correct Notes name, even when the user enters their short name.

Conclusion

These steps should help you get started. Because each application is unique, you may have additional issues when you begin webifying your application. But, we think you'll learn that Notes Designer for Domino 4.6 makes life much easier for application developers who struggle with designing for the Web and Notes.

ABOUT THE AUTHOR

Teresa Deane has been with Iris and Lotus for the past 10 years; at Iris she designed the original template for Web navigation in the Notes Web Navigator and has just completed a number of templates for the new Notes Designer and the Notes 4.6 client's PerWeb template for enabling Internet Explorer to be used as an integrated browser component in Notes.