What's new in the Registration template

by
Mark
Gordon

Level: Advanced
Works with: Designer 5.0
Updated: 02/02/2000

[Editor's note: This article is part of our ongoing series exposing how the Notes.net site is run. This time, we explain the design elements behind our new registration process.]

About six months ago the Notes.net team updated the Notes.net registration application, and since then we've received many requests to post the new template for download. It is **now available in the Iris Sandbox**. Be sure to read the README.TXT file included in the download for instructions on setting up the registration database on your Domino system.

This article describes the design of the new features and explains how they were implemented so you can modify the design for use on your Web site.

You've probably already seen many of these new features if you recently registered, updated your account, or lost your user name or password. If you're not familiar with our changes, read the Notes.net Exposed article **Using the new Notes.net registration features.**

As always, you can still access most of Notes.net as an anonymous user. You need to register if you want to download software or participate in the discussion forums in the Iris Cafe.

This article discusses how we implemented the following new features:

- E-mail confirmation of new registrations
- Registration documents are now used only for initial registration
- More efficient, faster-running agents
- Usability improvements to the registration form
- New "forgotten user name" and "forgotten password" forms
- Account update improvements
- Agent failover

Download components

In the Iris Sandbox, the compressed file **download** contains three files: REGISTRATION.NTF, REGERROR.NSF, and README.TXT.

All of the features described in this article are contained in REGISTRATION.NTF, which is the Notes.net Registration template itself.

REGERROR.NSF contains error messages used in the registration process.

README.TXT includes instructions on setting up REGISTRATION.NSF (the database created from REGISTRATION.NTF) and REGERROR.NSF on your Domino system.

E-mail confirmation of new registrations

Users now get an e-mail confirming their registration. That e-mail lets them know their registration is complete and gives them a record of the user name and password they chose. If that e-mail fails, it returns to a special view in the registration database, which we use to contact the user to get the correct

© Copyright 2000 Iris Associates, Inc.

e-mail address or cancel the user's registration. Also, ensuring we have each user's e-mail address gives us a communication channel back to any user who posts a topic in the forums.

You might ask why it is important to e-mail the user to confirm their registration. Isn't the registration immediate anyway? The user specifies their own user name and password when registering, and shouldn't have to wait for a confirmation e-mail to access their account, right? The answer is, in our case, registration isn't really immediate, or at least it's not guaranteed to be. The reason is that the Notes.net site runs several servers in a cluster. Every time you access the site, you may be accessing a different server, even if you just registered and are coming back to the site a couple of minutes later. And since the servers are not synchronized immediately, there can be a short delay between the time your new registration is done on one server and all the other servers are updated with it.

So we decided to tell applicants when they fill out the registration form that their registration will not be in effect until they receive the confirmation e-mail. This e-mail confirmation has two other benefits. First, since the confirmation e-mail contains the new user name and password, the user has a record of what they registered under. We have already seen a sharp decrease in the number of people who contact us saying they have forgotten their user name or password. Second, if they don't get the confirmation e-mail they can contact us via our **feedback form** so we can check their e-mail address.

The e-mail confirmation is simply a subroutine that is called within the *Process New Registration* agent in the Registration template. If you have downloaded the template, you will see the *sendEmail* subroutine, which is called from the end of the *registerNewUser* subroutine in the agent.

```
Sub sendEmail
    Dim docMemo As New NOTESDOCUMENT(db)
    Dim bodyField As New NOTESRICHTEXTITEM(docMemo, "Body")

    Dim userAddress As String
    Dim firstName As String

    userAddress = doc.GetItemValue("email")(0)
    firstName = doc.GetItemValue("FirstName")(0)

    Call docMemo.ReplaceItemValue("SendTo", userAddress)
    Call docMemo.ReplaceItemValue("Subject","Notes.net Registration
    Confirmation")
    Call docMemo.ReplaceItemValue("From","reg@notes.net")
    Call docMemo.ReplaceItemValue("Form","Memo")
    Call bodyField.APPENDTEXT("Dear " &  firstName & ":")
    Call bodyField.ADDNEWLINE(2)
    Call bodyField.APPENDTEXT("Your Notes.net Registration has been
    successfully processed.")
    Call bodyField.ADDNEWLINE(2)
    Call bodyField.APPENDTEXT("Your username is: " &
    doc.GetItemValue("FullName")(0))
    Call bodyField.ADDNEWLINE(1)
    Call bodyField.APPENDTEXT("Your password is: " &
    doc.GetItemValue("NewPassword")(0))
    Call bodyField.ADDNEWLINE(2)
    Call bodyField.APPENDTEXT("Thanks for using our site!" )
    Call bodyField.ADDNEWLINE(1)
    Call bodyField.APPENDTEXT("Notes.net Team")
    Call bodyField.ADDNEWLINE(1)
    Call bodyField.APPENDTEXT("http://www.notes.net")
    Call docMemo.Send(False)
End Sub
```

© Copyright 2000 Iris Associates, Inc.

Notice the *From* field (highlighted in red above) on the mail memo is set to *reg@notes.net*, and the agent is signed by a user ID called *Reg/Web*. This is a special user ID we created for use in this process, and it ensures that any replies to this memo and delivery failures get returned to the mail-in registration database.

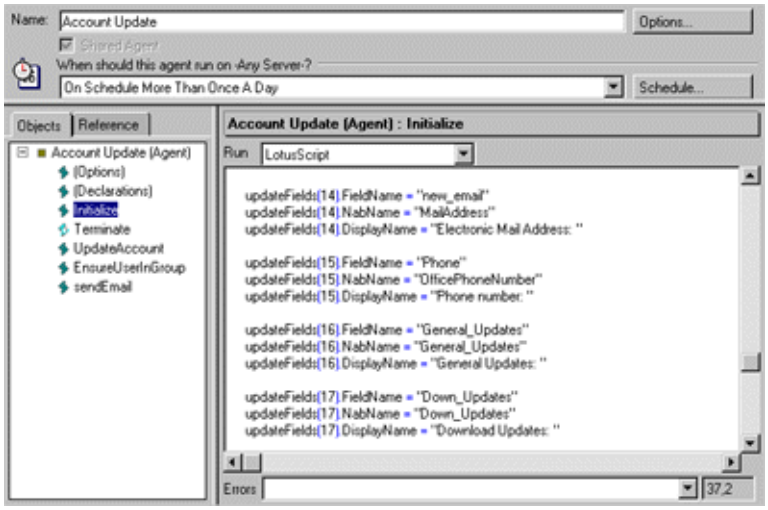
The mail notices that are delivered to the registration database show up in the *(Mail)* view, and the Notes.net administrators monitor that view for replies and delivery failures. The selection formula for the (Mail) view is:

```
SELECT (@LowerCase(Form) = "memo") | (Form = "NonDelivery Report") |
@IsResponseDoc
```

Registration documents are now used only for initial registration

In the old registration template, the registration document the new user filled out was stored in the database and used as a place to make updates to the user's account information. Updates to the registration documents had to be synchronized with Person documents in the address book, because Domino security runs from the Domino Directory (a.k.a address book). Now, all registration information is maintained in Person documents. Registration documents capture the initial information for the registration agent to process, but once that document is processed it is no longer necessary for anything but history.

The revised version of the account update function includes the *Account Update* agent, which takes each field updated on the *Account Update* form and updates it in the Person document. If you look at the agent's Initialize event, you see each of the fields defined to be synchronized with the Person document, including some custom fields not normally stored in a Person document, such as information about what type of mail notification the user wants to receive when the Notes.net site is updated:



More efficient, faster-running agents

In the new template, the agents are much more efficiently designed, with the result being a ten-fold improvement in their performance. The two primary application design changes that resulted in faster performance were:

1. Using view lookups and full-text searches instead of the *Search* method to look for a user in the address book, and
2. Skipping the call to the address book update DLL (Dynamic Link Library).

The *Search* method is very slow. Since it uses no index, each time it is called it must search through the entire database looking for documents that meet

© Copyright 2000 Iris Associates, Inc.

the selection formula passed to it. This is analogous to rebuilding a server index from scratch.

The following line of code is from the new *validateUser* subroutine in the *CommonRoutines* script library:

```
'Since the first column in $LDAPCN is sorted off the FullName field than any
duplicate usernames will be found
'by using the following line of code.
Set collection =
view.Getalldocumentsbykey(Lcase(doc.GetItemValue("FullName")(0)), True)
```

In contrast, the old version of the application used the much slower *Search* method of the NotesDatabase class. This is from the *GetPersonDocument* subroutine of the *RequestUtilities* script library:

```
search$ = "(Form = ""Person"" | Form = ""Request"") &
@LowerCase(FullName) = """ + Lcase$(personName) + """"
Set coll = nab.Search(search$, Nothing, 0)
```

Chances are you never noticed the performance impact of the *Search* method if you used the older registration database on a smaller address book. But if you have an address book the size of the Notes.net address book you'll appreciate the improvement!

Anybody who implemented the registration database in Domino 4.6 or earlier will be familiar with the call to the *nnotes* DLL, because you had to ensure the proper call was made depending on whether you were running Domino on an Intel or Unix server. The DLL call, which was invoked when the *EnsureUserInNAB* subroutine was called, is no longer necessary in R5. If you refresh the view indexes of the ($LDAPCN) and ($ServerAccess) views, the registration is immediate. Skipping the DLL calls also makes the agent run faster.

The problem for the Notes.net site, since it runs several serves in a cluster, is that the registration is immediate only on the server handling the registration.

Here is the *registerNewUser* subroutine of the *Process new registration* agent, which does the updates on the views that make a new registration immediate:

```
'If person document is succesfully created
'complete realtime update by refreshing $LDAPCN
'and $ServerAccess views.
If (ret = True) Then

    Dim nabFullNameView As NotesView
    Dim nabServerAccessView As NotesView

    Set nabFullNameView = dbNAB.GetView("($LDAPCN)")
    Set nabServerAccessView = dbNAB.GetView("($ServerAccess)")

    Call nabFullNameView.Refresh
    Call nabServerAccessView.Refresh

    'Closing session commits person document to NAB.
    Call s.Close
```

Usability improvements to the registration form

You have probably experienced a problem like this on more than one Web site: you fill out a long, complicated form. You click submit, but then get an error message, usually because you left out a mandatory field. The message tells you to press the back button to correct it, but when you do, you get a blank version of the form, with all your data missing.

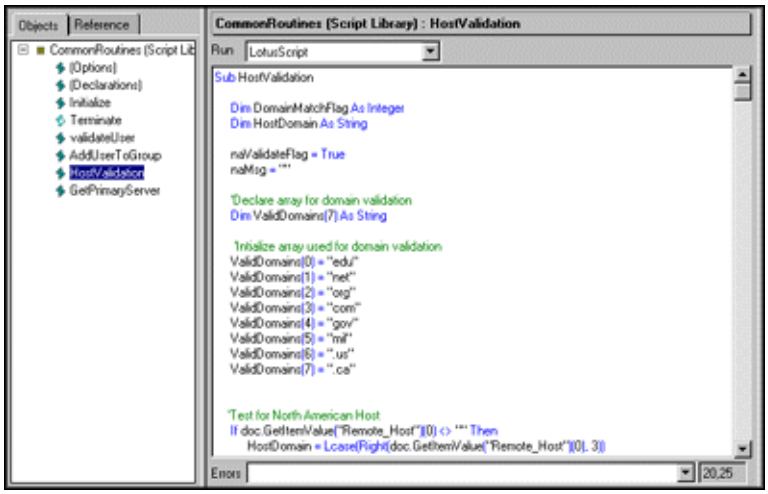
© Copyright 2000 Iris Associates, Inc.

This was a common problem for us as well, and we have fixed it with two techniques. First, we now use JavaScript for field validation, so where possible you can correct errors before you even submit the form. Second, where data must be validated at the server, we split the form into a couple of smaller forms, so that the data that must be validated at the server (in this case the user name you choose) is validated on the first form. With this validation scheme, you will know right away if the user name you choose has already been taken, *before* you spend time filling out additional details such as address and phone numbers.

The end result of our improvements is that no user will ever lose more than a couple of fields of data due to a validation error/back button scenario.

The first step in the multi-part form, after an initial license agreement page, is this form that asks two questions to determine whether you can qualify for a North American registration (North American users can download the North American versions of Notes and Domino, which contain more powerful encryption code that the U.S. government does not allow to be exported):

Also, before this form even comes up, the domain suffix of your Internet Service Provider (such as .com, .edu, .net, .org) is checked to see if it is what the U.S. government says is a valid indicator that you are visiting Notes.net from within North America. This happens via the WebQueryOpen event of the form, which calls the *(Regna1Open)* agent, which makes a call to the *HostValidation* subroutine from the *CommonRoutines* script library:



If the form opens successfully, meaning your host validation passed, and you

© Copyright 2000 Iris Associates, Inc.

answer the two questions correctly, you then go to Step 2, which asks you to specify a user name and password. The user name field cannot be checked with JavaScript, but must be checked against the address book to see if a user with that user name already exists. If there is a duplicate, you are given a link that takes you back to the form, and you are asked to choose a different name.

Only once you get past this step do you get to Step 3, which is a longer form where you enter details about your address, phone numbers, etc. At that point the mandatory field validation can be done with JavaScript, and your chances of losing information are much lower.

To lead the user through and among the forms for Step 1, Step 2, and Step 3, we use a QuerySave agent to ensure they have met validation criteria. For example, here is the agent *(Regna1Submit*), which is called when the Step 1 form is submitted. It takes the user to the *Regna2* form (the Step 2 form) if they have answered the North American questions correctly:

Note: This code references the REGERROR.NSF database.

```
Sub Initialize
    Set doc = s.DocumentContext

    msgFlag = False

    If (doc.GetItemValue("CitzenNA")(0) = "No") Then
        'Error Message
        Print "[/regerror.nsf/notices/nacitzen?OpenDocument]"
        Exit Sub
    End If

    If (doc.GetItemValue("UseNA")(0) = "No") Then
        'Error Message
        Print "[/regerror.nsf/notices/naexport?OpenDocument]"
        Exit Sub
    End If

    If (doc.GetItemValue("UseNA")(0) = "No") And
    (doc.GetItemValue("CitzenNA")(0) = "No")Then
    'Error Message
        Print "[/regerror.nsf/notices/naboth?OpenDocument]"
        Exit Sub
    Else
    'Redirect to next registration page.
        Print "[/registration.nsf/regna2]"
        End If
End Sub
```

New "forgotten user name" and "forgotten password" forms

Users who forget either their user name or their password can now fill out a form and have the information sent to their e-mail address. This was a fairly simple feature to implement, and it both saves a tremendous amount of site administrator time and alleviates a lot of frustration for users.

Both forms are very similar. Here is the forgotten user name form, as it looks from the Web:

© Copyright 2000 Iris Associates, Inc.

When the user enters their e-mail address, it is stored in a forgotten user name document, which is processed by the *Forgot Username* agent. The *GetUsername* subroutine finds the Person document that matches the e-mail address entered.

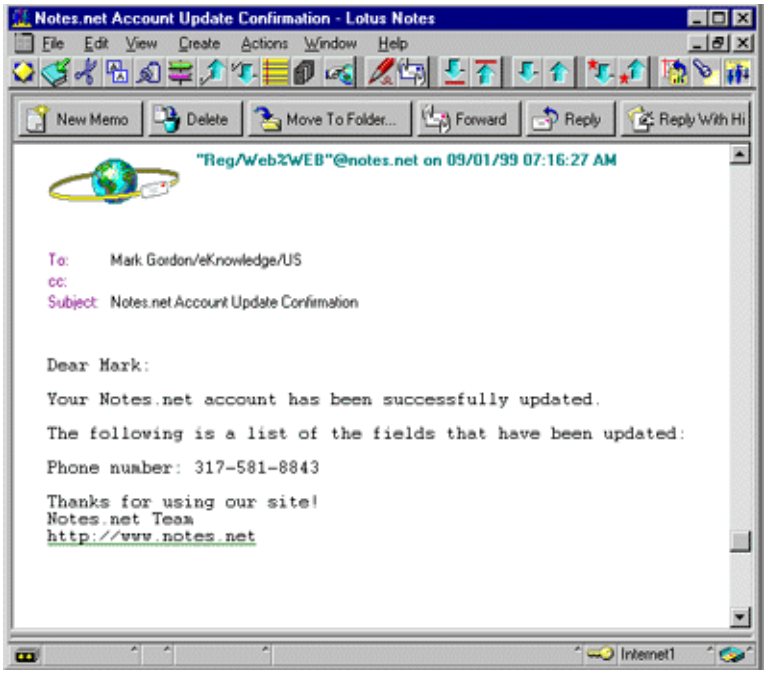
```
Sub GetUsername

    Set emailView = dbNab.GetView("($MailAddress)")
    Set emailMatchCollection =
    emailView.Getalldocumentsbykey(Lcase(doc.GetItemValue("email")(0)),
    True)

    For y = 1 To emailMatchCollection.count
        Set docPerson = emailMatchCollection.GetNthDocument(y)
        Redim Preserve userNames(y) As String
        userNames(y) = docPerson.GetItemValue("FullName")(0)
    Next

    Call sendEmail

    'Set status to processed, indicating the update was successfully made.
    Call doc.ReplaceItemValue("status","Processed")
    Call doc.Save(False, False)
End Sub
```

The *sendEmail* subroutine then puts the user name into a mail memo and sends it to the e-mail address the user specified.

Account update improvements

Users can now can update any part of their registration information, including their user name. Before, they could update only their password. All of this information is stored in the address book when the user registers, and although the user is filling out a new account update form to update their information (we don't want to give users direct access to the edit their person document in the address book), the existing information is retrieved from their Person document, and an agent called *AccountUpdate* runs periodically against account update documents. Not only does this agent update the Person document with the fields changed in the account update form, it also tracks which fields have been changed and sends the user a confirming e-mail, describing which fields have been changed:

© Copyright 2000 Iris Associates, Inc.



The *UpdateAccount* subroutine in the agent loops through all the relevant fields with the following loop, determining which ones have changed:

```
...
Erase modifiedFields
Redim modifiedFields(0)

'Retrieve field values from update form and update person document.
Forall i In updateFields
    If (doc.GetItemValue(i.FieldName)(0) <>
    docPerson.GetItemValue(i.NabName)(0)) And
    (doc.GetItemValue(i.FieldName)(0) <> "") Then
        Call docPerson.ReplaceItemValue(i.NabName,
        doc.GetItemValue(i.FieldName)(0))
        Redim Preserve modifiedFields(x)
        Redim Preserve modifiedFieldValues(x)
        modifiedFields(x).FieldDisplay = i.DisplayName
        modifiedFields(x).FieldValue =
        Cstr(doc.GetItemValue(i.FieldName)(0))
        x = x +1
    End If
End Forall
...
```

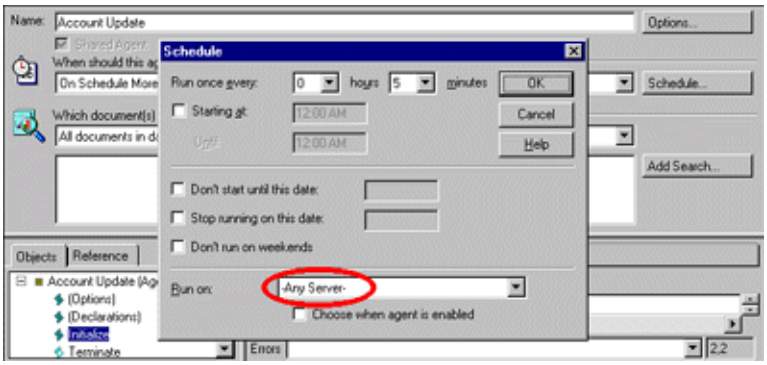
The changed fields are updated in the Person document, and this same list of updated fields is used in the mail memo that is sent to the user in the *sendEmail* subroutine.

Agent failover

The Notes.net server cluster is designed to keep the Notes.net site operating smoothly in the event one of the servers goes down. But much of the registration processing happens via server-based agents, so if the server on which the agents were scheduled to run went down, so would much of the functionality of the application -- the ability to process new applications, update users' accounts, and mail users their forgotten user names and passwords would all be on hold. Enter a new R5 feature, *run on Any Server,* which makes it possible to implement agent failover so that the agents are no longer tied to just one server.

© Copyright 2000 Iris Associates, Inc.

If you open up any of the scheduled agents in design mode, you will notice that they are scheduled to run on *Any Server*. This is a new option with R5.



Look at the first part of the code for any of them, and you will see logic to check to see whether execution is taking place on the current *primary* server. If not, execution is stopped:

```
'Make sure agent runs on primary server only!
Call getPrimaryServer
If (primaryServer <> sName.common)Then
    Exit Sub
End If
```

The *getPrimaryServer* function looks for the primary server in a view called *Server StatusView*, which shows which server is currently set to primary. The primary server is nothing more than the server in the cluster which is currently designated as the one on which agents should run. A separate group of agents -- *Auto-Failover, Ping Server,* and *Delete Pings* -- makes sure that if the primary server goes down, another in the cluster becomes designated *primary* in its place.

Conclusion

Now you've seen several ways in which the user registration template has been improved. Users can now avoid the most common frustrations they had in the past registering for an account and updating their accounts. And administrators have a much improved registration template in terms of performance and manageability. Look for more updates to come!

What do you think about this article?

Register Here!

About this Site | Feedback | Privacy
Iris Home | Lotus Home | IBM Home
© Copyright Iris Associates Inc.