Options for portalizing Domino applications

Lotus software

http://www-136.ibm.com/developerworks/lotus/

Table of contents

If you're viewing this document online, you can click any of the topics below to link directly to that section.

Section 1. Introduction

Should I take this tutorial?

This tutorial is for developers and managers who want to gain a better understanding of some of the issues involved in exposing a Domino application to a portal audience. It discusses some of the technical options, and their pros and cons. Most of the tutorial deals with a means for creating portlets, or portalized applications, with no code, but some Java coding is shown as an example.

The tutorial does, however, assume some familiarity with administering WebSphere Portal. For the necessary skills, see Resources on page 33 at the end of the tutorial.

What is this tutorial about?

This tutorial discusses several different options for portalizing a Domino application. These options include:

- Standard Lotus portlets
- The Portlet Builder for Domino (a component of Websphere Portal Application Integrator)
- The Bowstreet Portlet Builder
- Using the Domino Portlet API
- Using a Domino Web service

This tutorial is an overview of these topics and is intended to give you an idea of both the options for portalizing and their pros and cons. The Resources on page 33 direct you to in-depth tutorials on each of these topics.

Tools

Each section of the tutorial describes the integration of a simple Lotus Domino application with WebSphere Portal. You don't have to perform this integration to understand the concepts involved, but if you choose to, make sure you have installed and fully configured the following software. Links for downloads are available in Resources on page33 .

Install and configure these components on the WebSphere Portal server machine:

- IBM WebSphere Portal v5.0
- Lightweight Directory Access Protocol (LDAP) provided via Domino
- Lotus Collaborative Components
- Lotus Collaboration Center

The following components should be installed and configured on a separate machine (or machines):

- IBM Lotus Domino R5, 6.0 and/or 6.5 Server
- IBM Lotus Instant Messaging and Web Conferencing (Sametime) v3.0 or later
- IBM Lotus Team Workplace (QuickPlace) v3.0.1 or later
- IBM Lotus Discovery Server v2.0 or later
- IBM Lotus Domino Document Manager (Domino.Doc) v3.0 or later

To allow all servers to work together and to eliminate the need for users to log in more than once per browser session, Multi-Server Single Sign On (SSO) between WebSphere and Domino must be configured on all servers.

About the authors

Kenneth Adams, a Studio B (www.studiob.com) author, has been in the Information Technology field for over 17 years and has been working with Lotus technologies for over 8 years. Ken is a dual Principal Certified Lotus Professional (PCLP), certified in both Lotus application development and systems engineering/administration, a CLP in Collaborative Solutions Administration and is certified in Lotus Domino Messaging Administration/Migration.

His development experience includes Domino/Notes Designer, LotusScript, HTML, JavaScript, CSS and others. Ken also has a solid foundation in LAN/WAN technology having been network certified with a Master CNE (Certified Novell Engineer) and having many years experience in networking.

Ken has worked with Domino & Notes since R2, and with practically every Lotus product including Sametime, QuickPlace, Discovery Server, Domino.Doc, Enterprise Integrator and more. Ken was a contributing author on the books "Special Edition: Using Lotus Domino/Notes R5" and "Professional Developer's Guide to Domino." He can be reached at: *kadams@kennethadams.com*.

Nicholas Chase, a Studio B (www.studiob.com) author, has been involved in Web site development for companies such as Lucent Technologies, Sun Microsystems, Oracle, and the Tampa Bay Buccaneers. Nick has been a high school physics teacher, a low-level radioactive waste facility manager, an online science fiction magazine editor, a multimedia engineer, and an Oracle instructor. More recently, he was the Chief Technology Officer of an interactive communications firm in Clearwater, Florida, and is the author of four books on Web and XML development, including "XML Primer Plus" (Sams). He loves to hear from readers and can be reached at *nicholas@nicholaschase.com*.

Section 2. Overview

Why portalize?

The first question to ask about portalizing a Domino application is: "What is portalizing anyway, and why should I do it?"

The answer lies in the usefulness of the information held in your Domino applications. In many ways, a portal shouldn't be strange to you. One of the advantages of Domino is the ability to keep disparate types of information, such as e-mails, documents, and calendars, in one place for easy retrieval. A portal such as WebSphere Portal does the same thing, but with an even wider range of information.

Using *portlets*, or miniature portal applications, a portal enables a user to access information from many different sources at the same time via the browser.

This architecture provides a number of advantages for the administrator. First, you can keep all of your access control in one place. As long as permissions are set properly on the portal, users will have access to the appropriate material. In many portal installations, users are also able to customize their portal experience, which makes it even easier for users to use your data.

So rather than having two competing systems, by integrating your Domino applications with the portal system you can truly achieve one-stop information shopping.

But how can you do it? The answer depends on what you want to achieve, and what you're willing to do in order to achieve it.

Built-in Lotus portlets

The simplest way to bring Domino information into a portal is to use the built-in portlets that come with WebSphere Portal. These portlets are designed to read information from, and write information to, Domino databases and they require absolutely no coding.

The advantage to these portlets is their simplicity. All a user (or administrator) has to do to use them is enter the appropriate authentication and connection information. Of course, because of their simplicity, they're limited, so you may find it convenient to move to the next level of complexity.

Portlet Builder for Domino

One step up from the built-in portlet is the Portlet Builder for Domino, which is a subset of the WebSphere Portal Application Integrator (WPAI) technology.

The Portlet Builder is also a no-code method for accessing Domino information from within WebSphere Portal, but it offers a bit more flexibility. Using Portlet Builder, administrators create new portlets by specifying the appropriate database, view, and so on to be displayed.

Because the Portlet Builder for Domino uses a base portlet to create new portlets, it can be used by nonprogrammers. However, customization of the resulting display is limited.

The Bowstreet Portlet Builder

The Bowstreet Portlet Builder takes a different approach to building portlets. Instead of creating or configuring a portlet from within WebSphere Portal, the Bowstreet Portlet Builder is an integrated development environment that works with an application such as WebSphere Studio Application Developer or Eclipse.

The Bowstreet Portlet Builder doesn't create a portlet, it creates a *model* within the Domino server. A generic Bowstreet portlet looks at the model to determine what to display.

The Bowstreet Portlet Builder provides a means for pulling Domino information into WebSphere Portal with a minimum of programming, but still provides you with flexibility in what is displayed.

Using the Domino portlet API

You may choose to access Domino directly from within your Java portlets. For this, WebSphere Portal provides both a Java API and a custom JSP tag library. Together, they allow you to connect to a Domino server, access information, and make changes to the Domino database, all within the Java portlet. The main advantage of this method is that it enables you to completely integrate Domino data with other resources or programming, all within a single portlet.

Using a Domino Web service

Another means for accessing Domino information within WebSphere Portal is to create a Web service that outputs the data from Domino, and then create a Web services client that requests and interprets the information from that service.

One advantage to this approach is that it completely decouples the portlet from Domino; the portlet doesn't have to include any form of a Notes client in order to access the information. It also means that your application can be designed without worrying about where the data is coming from. Also, WebSphere Studio Application Developer makes it relatively simple to create a Web services client based on the Web Services Description Language (WSDL) file that describes the Domino Web service, thus making any changes to the Web service interface relatively uncomplicated.

Before we get that complex, however, let's start by looking at the simplest way to use Domino data within a WebSphere Portal.

Section 3. Built-in Lotus portlets

Overview

Many of the tools that ship with WebSphere Portal can be used to quickly integrate Lotus products. You don't need to be an experienced Lotus developer or administrator, or an experienced WebSphere developer or administrator to use these tools. They require only skills in WebSphere Portal configuration and an understanding of the products with which they integrate.

These tools include Lotus Domino Web Access portlets, Notes portlets, Lotus Domino Document Manager portlets, and collaboration-related portlets.

Lotus Notes portlets

The Domino Web Access portlet, previously named iNotes, is the advanced Web user interface (UI) to Domino Personal Information Manager (PIM). Through a single portlet you can provide access to the Welcome, Mail, Calendar, To Do List, Contacts, and Notebook functions of Domino Web Access.

The Lotus Notes View portlet can display the contents of any view in a Lotus Domino/Notes database. The portal can provide access to multiple views of multiple databases. Once added to the portlet, the views are selected via a pull-down list.

The Lotus Notes Discussion and Lotus TeamRoom portlets are used to display Domino/Notes databases built with the Discussion Database and the TeamRoom Template. They work exactly like the standard Lotus Notes View portlet.

The Lotus Notes Mail, My Lotus Notes Mail, My Lotus Notes Calendar, and the My Lotus Notes To Do portlets provide a simpler view of Notes Personal Information Manager (PIM) features. The user interface (UI) has more of a portal-based look and feel, rather than a Domino Web Access look and feel. These portlets automatically determine the currently authenticated user's mail server and database, so they require no configuration by the administrator or user. You simply add the portlets to portal pages and you're ready to go.

The Quick Appointment and the Quick e-Mailportlets

The Quick Appointment Version 5.0 portlet allows the user to create a simple meeting document in the user's Notes mail database on a Domino server. The Quick Appointment portlet stores appointment information in the Notes mail database specified for the current user of the portal in that user's Person document in the Domino Directory.

The Quick e-Mail portlet allows the user to send a simple e-mail message using either a Lotus Domino or SMTP server. It is configured by the portal administrator with mail server information. The user can edit no properties except for entering a name and password to use with the portlet. This portlet sends messages through Domino or Internet mail servers that are set up to work with the portal. If the portlet uses a SMTP outgoing Internet mail server instead, you specify the portlet's configuration parameters to connect to that server, as well as parameters for an incoming POP3 or IMAP Internet mail server to verify users' return addresses. The Quick e-Mail portlet does not support single sign-on (SSO), but a user can specify a name and password to persist for the browser session and avoid repeated authentication requests. If the portlet is configured to use a Domino server, the user must have the appropriate access to the Domino server that has been set up for use with Lotus Collaborative Components.

Lotus Domino Document Manager (Domino.Doc) and Web Page portlets

The Lotus Domino Document Manager portlet provides access to the Domino Document Manager (Domino.Doc) system. It provides a simple-to-use interface to the document Libraries, File rooms, File cabinets, and documents in the system. You configure the portlet by specifying the server and database filename of the library database.

The Web Page portlet is used to display any Web page. It is simply an IFrame used to display any URL within the Portal UI. This portlet is useful as a quick way to provide portal access to a Domino Web-based application, without almost no effort. Simply add this portlet to a page, and configure it with the URL to your Domino application.

Lotus Team Workplace (QuickPlace) portlets

The Lotus QuickPlace Version 5.0 portlet allows the user to select from a list of up to six QuickPlaces and display and use the selected QuickPlace in a separate browser window.

The Inline QuickPlace Version 5.0 portlet allows the user to display and use a QuickPlace within the portlet without launching a separate browser window.

The My Lotus Team Workplaces portlet that comes with WebSphere Portal featuring Collaboration Center lets users find, work in, and request new team workplaces as well as view workplace details. This portlet complements the People Finder portlet and the Lotus Web Conferencing portlet. As with the other Collaboration Center portlets, people links are visible in the My Lotus Team Workplaces portlet to help make employee interaction fast and easy, and to help improve personal and organizational productivity. With the My Lotus Team Workplaces portlet, users have a central location from which to see and visit the online workplaces to which they belong. Users can quickly find workplaces to see summary views of activity and recent events, get more detailed information about the workplace, and work in the place. People links in team workplaces provide a way for place members to interact with each other. Users with Edit permission to this portlet can edit the display properties to set their preferences. Administrators install, configure, and deploy the My Lotus Team Workplaces portlet in the portal as they do other portlets.

Lotus Instant Message and Web Conferencing (Sametime) portlets

The Lotus Sametime Connect Version 5.0 portlet lets the user launch the Lotus Sametime Connect client software from a link.

The Sametime Contact List Version 5.0 portlet displays an inline list of people that the user has compiled and with whom the user wants to chat online or include in online meetings.

The Sametime Who Is Here Version 5.0 portlet displays an inline list of people in the page or virtual page with whom the user is able to chat.

The Lotus Web Conferencing portlet that comes with WebSphere Portal featuring Collaboration Center lets users find, attend, and schedule e-meetings as well as view meeting details. The Web Conferencing portlet is provided by the portlet application, `LotusWebConferencing.war`. This portlet complements the People Finder portlet and the My Lotus Team Workplaces portlet. As with the other Collaboration Center portlets, people links are visible in the Web Conferencing portlet to help make employee interaction fast and easy, and to help improve personal and organizational productivity.

People Finder portlet

The People Finder portlet is the primary application of Collaboration Center. It provides quick search and advanced search options for locating people and information about people. Once found, a person is visible to other users as a person link that indicates online presence and displays a menu of instant messaging and other options. Clicking a person link displays the Person Record, including the person's business card information followed by administrator-defined fields organized in sections (for example, Contact Information, Current Job, and Background). The portlet also provides a view of the person's place in the organizational context. In the Organization View, users can display the person's reporting structure -- the employees the person manages and the person's management chain.

Taken together, the built-in portlets discussed in this section provide a way for you to quickly leverage your Domino installation into your WebSphere Portal installation. However, if they don't provide enough flexibility for you, you'll need to look at other options.

Section 4. The Portlet Builder for Domino

Portlet Builder for Domino

The Portlet Builder for Domino allows users to create new portlets that can access and manipulate data in any Lotus Domino database, without requiring Domino or WebSphere programming skills. You can select views and forms that you wish to provide access to, set the basic formatting for the views and forms, and add other portlet functionality through a browser interface.

Portlet Builder creates and adds a new portlet to the WebSphere Portal server that can be used like any other portlet. You can even export it to be used in another WebSphere Portal server.

Installation and technical requirements

In WebSphere Portal v5.0, Portlet Builder is installed by default. Some configuration of your Domino server will be required. Portlet Builder for Domino uses IIOP over HTTP to communicate with each Domino server, so HTTP and DIIOP server tasks will need to be loaded and you will need to enable database browsing via HTTP.

You can find details in the WebSphere Portal InfoCenter documentation, in the paper titled: *IBM Application Portlet Builder* (see Resources on page 33).

Creating a portlet

Using the Portlet Builder for Domino to create a new portlet involves a series of steps:

1. Create a new portal page and add the Portlet Builder for Domino portlet to it. You will use this portlet to create new portlets and to edit existing portlets.
2. From within the Portlet Builder for Domino portlet, click the New Portlet button.
3. In the Portlet name field, fill in a name that will appear in portlet lists.
4. In the Domino Server Name field, fill in the hostname of the Domino server, such as `portal.kennethadams.com`. You can also use the server's IP address.

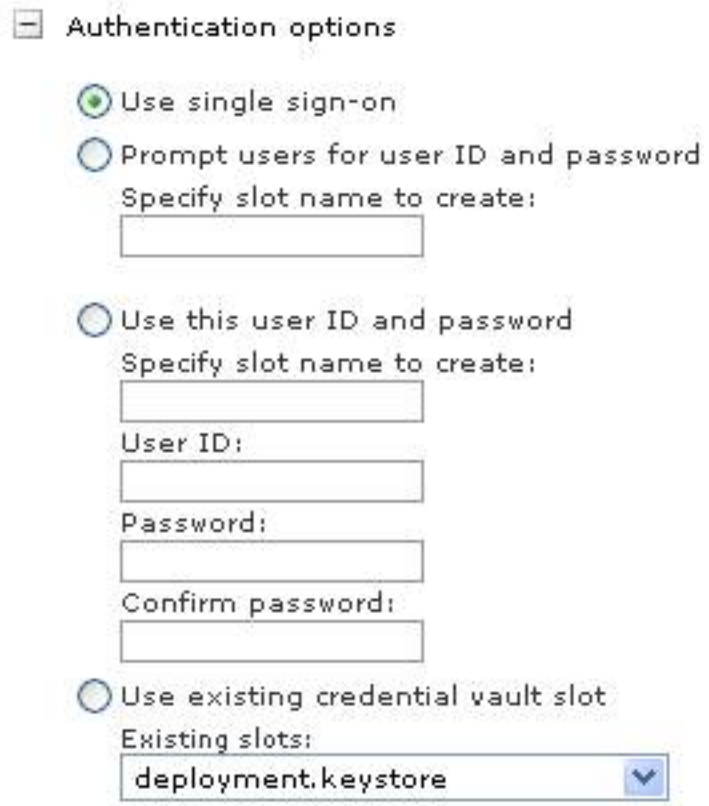
5. Click the Connect to server button, which displays the available databases in the Domino Database Name list.
6. Select a database from the Domino Database Name list, and click the Retrieve forms and views button. A list of forms and views from the selected database will be displayed.

7. Check the views and forms you would like to include in the portlet and click the Next button.

8. Click the OK button to create the portlet.

Authentication options

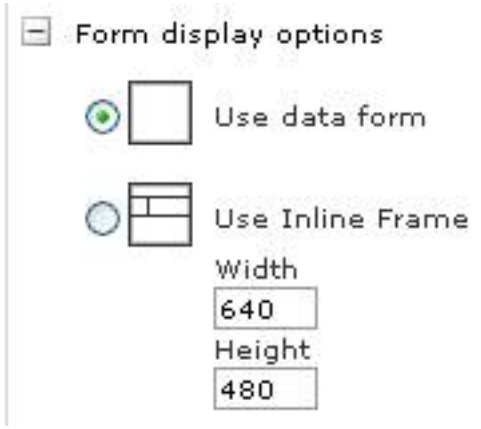
The Authentication options allow you to specify whether the portlet will use SSO, prompt the user for credentials, specify a user for the portlet, or use the credential vault. The credential vault enables you to provide authenticated access to a resource such as a Domino database without giving users the authentication information.



Form display options

When documents are selected from a view, they can be displayed in two ways:

- `Use data form` displays the document using a simple form layout inside the portlet window. These forms have limited display capabilities and usually require additional configuration to be acceptable.
- `Use Inline Frame` displays the document inside an IFrame using the original Domino form design. The document is rendered by the regular Domino HTTP engine, so it has no limitations and provides complete control over the look of the form.



Using the new portlet

Your newly created portlet works just like any other portlet, so to use it, simply add it to a new or existing portal page.

A portlet created with Portlet Builder for Domino allows the user to select from the views (which you specified when building it) from a pull-down list. The portlet also provides navigation icons, icons for opening, editing, and deleting documents, and so on.

Advantages and disadvantages

Some of the advantages of using the Portlet Builder for Domino over conventional Domino development tools and methods include:

- No Domino or Web development skills are required. Only Portal configuration skills are needed.
- No changes to existing Domino databases are required.
- Portlets can be enabled for Click to Action, which enables portlets to talk to each other.
- Portlets can have integrated presence awareness via Sametime integration.
- Support for offline browsing is available.
- Support for mobile browsers that support HTML and WML is available.
- Domino Portlets can be used and reused anywhere in the portal, just like any other portlet.
- Portlets are reusable by exporting to WAR files.

Some of the disadvantages of using the Portlet Builder for Domino over conventional Domino development tools and methods include:

- There is much less control over the look of an application.
- Portlet Builder for Domino is dependent on DIIOP running on the Domino server.
- Rich text handling is limited. Rich text fields are rendered as plain text, if using the form capabilities of the portlet. Although, it is possible to configure a portlet to allow Domino to render the document in an IFrame to get around this limitation.

For a tutorial dealing specifically with the Portlet Builder, see Resources on page33 .

Section 5. The Bowstreet Portlet Builder

The Bowstreet Portlet Builder

The Portlet Builder for Domino (discussed in the previous section) is a tool specifically designed to provide a way to access Domino data from a portlet without any need for programming, but it provides a limited amount of flexibility (none beyond the functionality expressly included). The Bowstreet Portlet Factory, on the other hand, provides the ability to create a portlet without knowing how to code in Java, HTML, and so on, but also provides the flexibility to get into the code if desired.

Whereas the Portlet Builder is itself a portlet, the Bowstreet Portlet Factory is an IDE that is integrated into WebSphere Studio Application Developer or Eclipse. You create your new portlet by creating a Bowstreet *model*, then pointing a new Bowstreet portlet at that model. The steps are as follows:

1. Create the model.
2. Copy the generic Bowstreet portlet.
3. Point the copy at the Bowstreet model.
4. Put the new portlet on a portal page and use it.

Building the model

The first step in creating a new model is, of course, to install the Bowstreet Portlet Factory into WebSphere Application Developer or Eclipse. (For more information on installation, see Resources on page33 .)

Next, create a new model by doing the following:

1. Open WebSphere Studio and choose New Model from the Bowstreet menu.
2. Create the new model by choosing Empty base model. Click OK.
3. Enter a name in the Save New Model dialog box and click OK.
4. Click the Add Builder Call icon (in the right-hand corner of the Builder Call List) and choose the Domino View and Form Builder from the Collaboration category in the Builder Palette. Click OK.
5. Populate the Builder Call. To see a list of accessible views, click Get databases and views.
6. Click OK to save the information.
7. Click the Run Model icon to see the results of your handiwork.

Now it's time to turn the model into a portlet.

Running the model as a portlet

The Bowstreet Portlet Factory doesn't create a portlet. Instead, it creates a model that can be run by a copy of the generic Bowstreet portlet. To create the new instance, perform the following actions:

1. Log in to WebSphere Portal as an administrator and choose Portal Administration - Manage Portlets.
2. Select the Bowstreet Portlet and click Copy. Name the new copy and click OK.
3. Select the new copy and click Modify Parameters.
4. Enter the name of the model you created in the previous panel and click Save.
5. Set the title, locale, and so on and save the changes.
6. Click Activate to make the new portlet available to the portlet.
7. Add the new portlet to a page to view it.

Options for customizing the portlet

One of the advantages of using a product like the Bowstreet Portlet Factory over the Portlet Builder for Domino is the ability to completely control the appearance of the data that it produces. To do this, you have two options. You can either directly edit the HTML (or CSS, or JSP) that Bowstreet puts out, or you can use the advanced section of the Bowstreet Builder.

You can also modify the model itself, performing such actions as modifying the format of fields output within the Domino view.

Advantages and disadvantages

Advantages of using Bowstreet Portlet Builder include:

- It gives users complete control over the appearance of the information output by the portlet.

- It provides an integrated development environment with advanced functionality.
- Models can be used not only as portlets, but also as Web services and standalone Web applications.
- Portlets can be changed on the fly by updating the model.

Disadvantages include:

- A more complex interface requires a higher skill level.
- Users must be familiar with the structure of the original Domino application.

For a tutorial dealing specifically with the Bowstreet Portlet Builder, see Resources on page33 .

Section 6. Using the Domino Portlet API

Domino and Java

Now let's take a brief look at creating a portlet that uses the Domino Java API to extract information from a Domino database and display it within a portlet. In this case, you are basically building a small Java application that accesses the Domino database; it's almost a coincidence that the application is running within a portal.

For many of you, this may be a departure from simply creating and running LotusScript, but the benefits include the ability to run your application outside the Notes environment (in this case, in a portal) and the ability to include information from other sources.

For an extensive look at using Domino with Java and the Portal API, see the Resources on page33 .

Create the portlet

The easiest way to create a portlet is to use WebSphere Studio Application Developer and the Portal Toolkit. Start by installing WebSphere Studio Application Developer 5.0.0 (*not* 5.1), and then use the Update Manager (under Help) to upgrade to version 5.0.1. You can then install the toolkit by shutting down WSAD and double-clicking the Portal Toolkit 5.0 installer. (See Resources on page 33 for links to download locations and more information.)

Once you have both WSAD and the Portal Toolkit installed, perform the following steps:

1. Choose New - Other - Portlet Development - Portlet Application Project and name the application.
2. Choose Create Basic Portlet and click Next.
3. Keep the default portlet settings and give the portlet a name. Click Finish.

Double-click the Java class for the portlet to open it in the source editor. The portlet will be named according to the name of the project. For example, if the name of the project is `DominoProject`, the file will be `Java Source/dominoproject/DominoProjectPortlet.java`.

Replace the code in the file with the code below:

```
package dominoproject;

import org.apache.jetspeed.portlet.*;
import org.apache.jetspeed.portlets.*;

public class DominoProjectPortlet extends PortletAdapter {

  public void doView (PortletRequest request, PortletResponse response)
     throws PortletException, java.io.IOException
  {
     response.getWriter().print("<h1>Hello, Domino!</h1>");
  }

}
```

Save the portlet and export it as a WAR file by right-clicking the project and choosing Export - WAR file. Install the new portlet application and add the portlet to a page to test it. (See Resources on page 33 for more information on installing an existing portlet.)

Preparing the environment

In this process, you will be asking your portlet to connect to a Domino database using its own protocol, DIIOP, so you will need to ensure two things. First, you will need to make certain that DIIOP is running on the appropriate Domino server. To do that, open the notes.ini file (you'll find it on the Domino server in the `Lotus/Domino` directory) and make sure that the `ServerTasks` property includes DIIOP, as in:

```
ServerTasks=Update,Replica,Router,AMgr,AdminP,CalConn,Sched,HTTP,LDAP,DIIOP
```

After making the change, restart Domino Server using the Services control panel.

Second, make sure that the portlet has access to the classes normally used by the Notes client; these are the classes that WebSphere Portal uses to access the Domino database. Some versions of WebSphere Portal come preconfigured with these classes. For others, you will need to add them to your project. To do that, you will need to include the appropriate JAR file in the `lib` directory of the portlet project. For a local Domino database, you will need the `NOTES.jar` file (located in the `Lotus/Domino` directory). But, since Domino generally can't run on the same machine as WebSphere Portal, you are more likely looking at using `NCSO.jar` (located in the `Lotus/Domino/Data/domino/java` directory).

Once you've determined the appropriate file, import it into the `lib` directory of the project by right-clicking the Web Content - WEB-INF - lib directory in the Navigator pane of WSAD and choosing Import - File System. Choose the appropriate file and click Finish. Now you're ready to connect to the Domino database.

Connect to Domino

To connect to the Domino database, use the `NotesFactory` class to create a session by passing the server, username, and password to the `createSession()` method as follows:

```
package dominoproject;

import lotus.domino.*;
import org.apache.jetspeed.portlet.*;
import org.apache.jetspeed.portlets.*;

public class DominoProjectPortlet extends PortletAdapter {

  public void doView (PortletRequest request, PortletResponse response)
     throws PortletException, java.io.IOException
  {
    Session session = null;
     try {
         String server = "chaosMagnet";
         String user = "nchase";
         String pwd = "swordfish";
         String dbname = "rantsandraves.nsf";
         session = NotesFactory.createSession(server, user, pwd);
         Database db = session.getDatabase("", dbname);
     } catch (Exception e) {
         //Error handling
     }
  }

}
```

The server name can be a local host name, the fully qualified domain name (such as `portal.kennethadams.com`) or the IP address.

Once you connect to the database, you can retrieve information.

Reading a view

Once you have a connection to the database, you can select and read from a specific `View` by passing the name of the view to the `Database` object's `getView()` method as follows:

```
package dominoproject;

import lotus.domino.*;
import org.apache.jetspeed.portlet.*;
import org.apache.jetspeed.portlets.*;

public class DominoProjectPortlet extends PortletAdapter {

  public void doView (PortletRequest request, PortletResponse response)
     throws PortletException, java.io.IOException
  {
    Session session = null;
    try {
        String server = "chaosMagnet";
        String user = "nchase";
        String pwd = "swordfish";
        String dbname = "rantsandraves.nsf";
        session = NotesFactory.createSession(server, user, pwd);
        Database db = session.getDatabase("", dbname);

        View view = db.getView("All Documents");
         ViewNavigator viewNavigator = view.createViewNav();
         ViewEntry entry = viewNavigator.getFirst();
         while (entry != null) {
            if (entry.isDocument()) {
                Document thisDocument = entry.getDocument();
                Vector items = thisDocument.getItems();
                Enumeration itemEnum = items.elements();
                while (itemEnum.hasMoreElements()) {
                   Item thisItem = (Item)itemEnum.nextElement();
                   response.getWriter().print(thisItem.getName()+"="
                                              +thisItem.getValueString()+"
");
                }
                response.getWriter().print("
_____________________________________________________________________
");
            }
            entry = viewNavigator.getNext();
        }

    } catch (Exception e) {
        //Error handling
    }
  }

}
```

The Domino Java API enables you to loop through a `View` to examine the `Items` that make up the information for a `Document`.

The Domino Java API, combined with the Portlet API, enables you to create fully functional Domino applications within the portlet environment.

Advantages and disadvantages

The major advantage of using the Domino Java API within a portlet is that you are no longer restricted to the functionality envisioned by the builders of a tool. You are also unrestricted in terms of combining both Domino and non-Domino information in a single portlet, because it's all controlled by the Java that you write.

One major disadvantage for Domino developers, though, is the potential need to learn a new language. Given the current direction of the Lotus products, however, it may be worth the time and effort.

Section 7. Using a Domino Web service

Domino, Portal, and Web services

This tutorial has moved farther away from using Notes to access the Domino server, but even in the last section when you were building a Java-based application, you were still using the Notes libraries. It was still, at its heart, a variation of the Notes client.

Now it's time to make a clean break.

This section of the tutorial discusses the idea of accessing information in a Domino database as a Web service. Web services, in case you've been hiding from them for the last couple of years, are applications that can be called from a separate application. The client makes a request, usually over the Internet using HTTP, and the application returns a result. Now of course, there have been applications doing this for years, but the protocols that an application follows typically qualify it as a Web service. Messages are sent and received in a specific format (usually SOAP) and can be described using a specific format (usually WSDL).

In this case, I'm going to create a situation in which the Domino server has an agent that listens for these requests, returning the appropriate information. I then create a portlet that acts as a client, sending the request and processing the returned information.

Soap messages

Before I go any further, it would help to have a basic understanding of what a SOAP message looks like. It is, in essence, an XML document with an Envelope that contains a payload of information. It's impossible to cover the entire topic of XML and namespaces in just one panel, but here's a basic document, with the information you're looking for in bold:

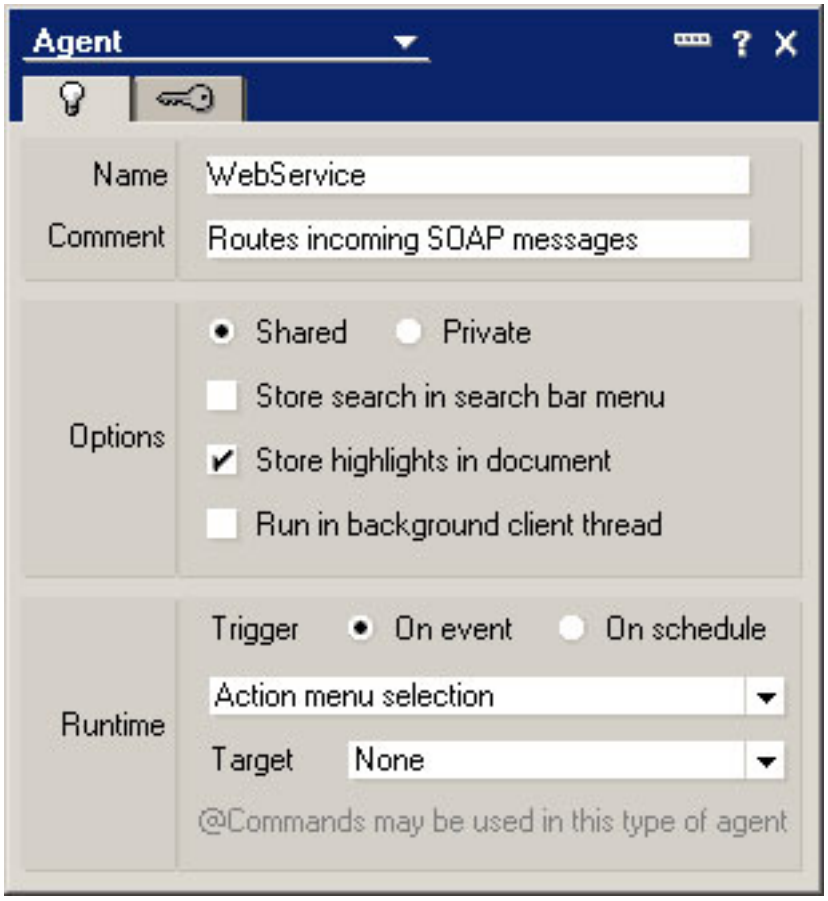
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
    <m:actionResponse xmlns:m="uri:tempuri.org"
           SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        Response for the client
    </m:actionResponse>
```

```
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Create the Web service

The first step in this process is to create the Web service. To do that, open the appropriate database in Domino Designer and take the following steps:

1. Expand Shared Code and click Agents.
2. Click the New Agent button and set the agent as Shared, with an On event trigger, and no target. It should be an Action menu selection.



3. Create the Initialize method for the agent. The idea is to retrieve the necessary information from the SOAP message. How you do it depends on the language you choose to use. If you're using Java, you might manipulate the XML document directly. In his tutorial, *Building Web Services using Lotus Domino 6* (see Resources on page 33), Jeff Gunther used LotusScript to parse the file and send back a response:

```
Sub Initialize

        Dim session As New NotesSession

        Set doc = session.DocumentContext

        'set the output content type to XML
        Print "Content-Type: text/xml"

        'get incoming SOAP message
        incoming = doc.GetItemValue("Request_content")(0)

        'log incoming message
        LogMessage(incoming)

        'remove all whitespace from incoming SOAP message
        incoming = RemoveWhitespace(Fulltrim(incoming))

        'On Error Resume Next
        On Error Goto Errhandle

        bodyPos= Instr(1,incoming,|<SOAP-ENV:Body>|)+15

        'parse out method signature
        methodSigPos= Instr(bodyPos,incoming,|<|)+1
        methodSigEnd=Instr(bodyPos,incoming,| |)
        methodSignature = Mid(incoming,methodSigPos,(methodSigEnd-methodSigPos))

        'parse out method name
        methodPos= Instr(bodyPos,incoming,|:|)+1
        methodEnd=Instr(methodPos,incoming,| |)
        methodName = Mid(incoming,methodPos,(methodEnd-methodPos))

        'parse out namespace
        nameSpacePos= Instr(methodEnd,incoming,|uri:|)+4
        nameSpaceEnd=Instr(nameSpacePos,incoming,|"|)
        nameSpace=Mid(incoming,nameSpacePos,(nameSpaceEnd-nameSpacePos))

        'Build a string containing the script library, function, and
            'incoming soap message
        'The nameSpace variable contains the script library to load
        callString  = |Use | & |"| & nameSpace & |"| & |
        response  = | & methodName & |(incoming)|

        'execute the function
        Execute callString

        'create response
        strTmp = |<?xml version="1.0" encoding="UTF-8" standalone="no"?>| &_
        |<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">| & _
        |<SOAP-ENV:Body>| & _
        |<m:| & methodName & "Response"  & | xmlns:m="uri:| & nameSpace & |"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">| & response & _
        |</m:| & methodName & |Response>| & _
```

```
        |</SOAP-ENV:Body>| & _
        |</SOAP-ENV:Envelope>|

        'send back SOAP response
        Print strTmp

        Exit Sub

Errhandle:
       'Print "Error" & Str(Err) & ": " & Error$
       Exit Sub


End Sub
```

Define the WSDL

In order to enable WSAD to easily generate a client for this Web service, you will have to create a way for it to know what the client should send and what it should expect to receive. To do that, you will need to create a Web Services Definition Language (WSDL) file. In the case of the previously described Web service, that file might look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions
    xmlns="http://schemas.xmlsoap.org/wsdl/" name="ISBNSearch"
    targetNamespace="http://www.your-company.com/ISBNSearch.wsdl"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://www.your-company.com/ISBNSearch.wsdl"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsd1="http://www.your-company.com/ISBNSearch.xsd1">
    <types>
        <xsd:schema
            targetNamespace="http://www.your-company.com/ISBNSearch.xsd1"
            xmlns="http://schemas.xmlsoap.org/wsdl/"
            xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"> </xsd:schema>
    </types>
    <message name="ISBNSearch">
        <part name="isbn" type="xsd:string"/>
    </message>
    <message name="ISBNSearchResponse">
        <part name="title" type="xsd:string"/>
        <part name="isbn" type="xsd:string"/>
        <part name="price" type="xsd:string"/>
        <part name="authors" type="xsd:string"/>
        <part name="publisher" type="xsd:string"/>
        <part name="copies" type="xsd:string"/>
    </message>
    <portType name="ISBNSearchPortType">
```

```
        <operation name="ISBNSearch">
            <input message="tns:ISBNSearch"/>
            <output message="tns:ISBNSearchResponse"/>
        </operation>
    </portType>
    <binding name="ISBNSearchBinding" type="tns:ISBNSearchPortType">
        <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="ISBNSearch">
            <soap:operation
                soapAction="capeconnect:ISBNSearch:ISBNSearchPortType#ISBNSearch"/>
            <input>
                <soap:body
                    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    namespace="http://www.your-company.com/ISBNSearch/binding"
                    use="encoded"/>
            </input>
            <output>
                <soap:body
                    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    namespace="http://www.your-company.com/ISBNSearch/binding"
                    use="encoded"/>
            </output>
        </operation>
    </binding>
    <service name="ISBNSearch">
        <port binding="tns:ISBNSearchBinding" name="ISBNSearchPort">
            <soap:address
                location="http://192.168.1.2/bookcatalog.nsf/PortalWebService"/>
        </port>
    </service>
</definitions>
```

Now you can use that WSDL file to create the client, as long as it's available at a specific URL. Jeff suggests adding it as a page within the Domino database, so you can call it with a URL such as:
`http://<someserver>/bookcatalog.nsf/WSDL?OpenPage`

Create the Web services client

Creating a Web services client from within WebSphere Studio Application Developer involves creating a proxy that translates method calls into SOAP messages. The portlet will then send its instructions to the proxy, which sends and receives the message and returns the appropriate data. To create the client portlet, execute the following steps (If you didn't create a project for the previous section, follow the instructions in Create the portlet on page 21 to create a new portlet application.):

1. Select New - Other - Web Service - Web Service Client - Next.
2. Select the portlet application project. Make sure that Generate a proxy is

selected. You can also have the wizard create a test application, if you like. Click Next.

3. Enter the location or URL for the WSDL file. Click Finish.
4. The wizard will now create the `PortalWebServicePortlet` and `PortalWebServicePortletBean` classes. The `PortalWebServicePortlet` is the portlet that accesses the Web service.

Advantages and disadvantages

If your development experience is solely in Domino (or even mostly in Domino) this might seem like a strange way of doing business. In fact, this paradigm shift is one of the disadvantages of using this method of accessing the Domino database. There are, however, significant advantages. For one thing, the task of creating the client is eased by its automatic generation based on the WSDL file, so changes to the underlying Domino implementation aren't a problem. You also have the advantage of decoupling the client from the server. The portlet application doesn't care where the client is getting the information, so you can completely change the server, and as long as you update the client, the application will still work.

As far as disadvantages, you must also consider both performance and security. If you are accessing a remote Domino server over which you have no control, you should consider negotiating service-level agreements just as you would in any other implementation in which you interface with another organization. You are also vulnerable to Internet problems.

Securitywise, Domino developers are used to basing authentication on ID files, but implementing such a system via Web services is difficult, if not impossible. Standards such as Digital Signatures do provide a measure of security, but they are well beyond the scope of this tutorial.

Section 8. Summary

One last word about other options

It would be difficult to create a tutorial that covered all of the ways that a Domino application can be integrated with WebSphere Portal. When you come right down to it, that's limited only by the programmer's skill and imagination. This tutorial attempts to give you a brief glimpse at some of the more common methods -- see Resources on page 33 for more details on these techniques -- but let's take a moment to touch on some other methods.

In addition to directly accessing Domino using Java classes, a developer can use the Domino custom JSP tags to create JSP pages that don't require programming. (Be careful if you use this method, however, as some of the tags aren't designed for a portlet environment.) You could also use Domino's XML capabilities, in conjunction with the built-in XML reader portlets, to access Domino data for read-only applications.

There are other ways to use Domino data in a way more similar to the Notes experience. IBM is currently working on the Domino Web Application Portlet, which will access an existing Domino application, integrating it with the portal's styles and so on. (At the time of this writing, the portlet is in beta. It's expected to be released in 2004.)

And of course there's always the `iframe` method of incorporating your Domino data into the portal. An `iframe` is, in some ways, like a browser window within the browser window in which you can simply run the Web version of your Domino database. This has the advantage of being nearly painless from a development perspective, but doesn't provide a way to integrate any of your Domino data with non-Domino information.

Options for portalizing Domino applications summary

Domino provides an environment in which disparate resources can be pulled together and accessed by a single client, Lotus Notes. WebSphere Portal provides an environment in which disparate resources can be pulled together and accessed by a single client, the browser. This tutorial discussed methods of pulling Lotus Domino information into the portal environment, in order to provide the best of both worlds. It discussed the following methods:

- Using built-in Lotus Portlets
- Using the Portlet Builder for Domino

- Using the Bowstreet Portlet Factory
- Using the Domino Portlet API
- Accessing Domino as a Web service

Resources

For an IBM Redbook describing all of the options for integrating existing Domino applications into the IBM WebSphere Portal, see: *Portalizing Domino Applications for WebSphere Portal*.

For more information on Portlet Builder, see the paper, *IBM Application Portlet Builder*, available from the *WebSphere Portal InfoCenter*.

For more information on WebSphere Studio Application Developer, check out *WebSphere Studio section*.

For more information on Lotus, see the *Lotus section on developerWorks* (http://www-136.ibm.com/developerworks/lotus/) . *Lotus documentation* is also available online.

IBM also has numerous tutorials and articles to guide you on your way, including:

- *Building Web services into a portal*
- *Build custom portlets for Domino: Portlet Builder for Domino simplifies development*
- *Building Lotus Domino Portlets: Bowstreet Portlet Factory for WebSphere streamlines the process*.
- *Building Web services using Lotus Domino 6*
- *Turn your Lotus apps into Web services*
- *Develop portlets that use Web services to obtain data from remote systems*
- *Jetspeed, Part 1: Developing portlets*
- *Jetspeed, Part 2: Advanced portlet technology*
- *WebSphere Portal V4 programming, Part 1: Portlet application programming*
- *WebSphere Portal V4 programming, Part 2: Portlet application programming*
- *The making of MetroSphere, Part 13: Install an existing portlet*
- *Making of MetroSphere, Part 20: Creating portlets: Implementing and deploying the MetroSphere.com blogging system explains how to create a portlet application*
- *Making of MetroSphere, Part 6: Get started with WebSphere Portal-Express* (tutorial) details the skills necessary for administering portlet applications

and creating portal pages
- *The making of MetroSphere, Part 13: Install an existing portlet*

Specifications to check out:

- *Web Services Description Language (WSDL)* (http://www.w3.org/TR/wsdl)
- *SOAP Version 1.2 Part 0: Primer*

Software to download

- *WebSphere Studio Application Developer*
- *Lotus Domino Server*
- *Lotus Domino Designer*
- *Portlet Factory for Domino*
- *Bowstreet Portlet Factory for WebSphere*
- *Portal Toolkit for WebSphere Portal* (http://www.ibm.com/websphere/portal/toolkit)

You can also find a number of other portlets in the *Portlet Catalog*.

For more from Kenneth Adams, see the books *Special Edition: Using Lotus Domino/Notes R5* and *Professional Developer's Guide to Domino*. Ken was a contributing author on both.

Also see *XML Primer Plus*, one of four books on Web and XML development written by Nick Chase.

Feedback

Colophon

This tutorial was written entirely in XML, using the developerWorks Toot-O-Matic tutorial generator. The open source Toot-O-Matic tool is an XSLT stylesheet and several XSLT extension functions that convert an XML file into a number of HTML pages, a zip file, JPEG heading graphics, and two PDF files. Our ability to generate multiple text and binary formats from a single source file illustrates the power and flexibility of XML. (It also saves our production team a great deal of time and effort.)

For more information about the Toot-O-Matic, visit www-106.ibm.com/developerworks/xml/library/x-toot/ .