



**Manage your  
Web site visitor  
information using  
a registry created  
with LDAP.**

# Using LDAP

## for an E-business User Registry

by Pat Fleming


**W**hen you are designing e-business applications, you must decide how and where to manage an application's user information. One management method is via a repository called a user registry. In this article, I will investigate using the Lightweight Directory Access Protocol (LDAP) directory included with OS/400 V4R3 (and later) for this purpose. I will show you that introducing a JavaBean as an accessor to an LDAP server enables Java Server Pages (JSPs) and servlets to use a user registry. I will also explore the use of the XML standards to structure data that is interchanged between Web site components.

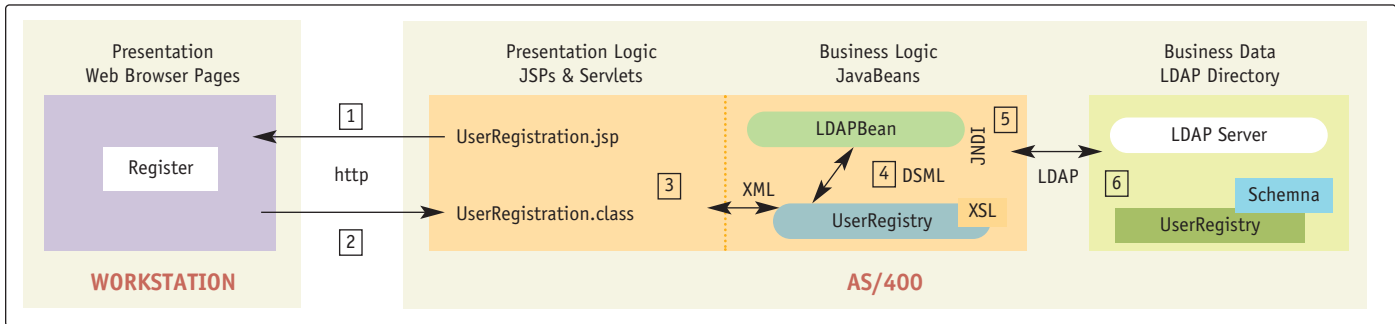
### **What Is a User Registry?**

A user registry is simply a place where information about users is kept. The information may be collected via Web-based applications or internal corporate applications. This information may be used by applications for authentication, marketing purposes, or corporate telephone books. The user registry data is kept in a repository that may be implemented using text files in a file system, tables in a database, or in a directory, as I'll discuss.

### **A Web Site Example**

For my example, I'll discuss a fictional Web site for a local pet store site that provides information about pets available for adoption by local customers. Site visitors register before viewing the information about available animals. The

**WEB  BONUS!** Find a sidebar for this article at [www.midrangecomputing.com/mc](http://www.midrangecomputing.com/mc).



**Figure 1: Upgrade your Web site user registry using XML, Java applications, and an LDAP directory.**

Web site's implementation registers visitors by using a Web browser-based form and a server-side Web application to store this information in a user registry. The user registry's repository is an LDAP directory, which is a directory in the DB2 UDB for iSeries 400 (DB2 UDB) database that is accessible using LDAP.

Implementing the user registry in the LDAP directory on the iSeries or AS/400 provides the following potential options for the Web site's use of the user registry:

- HTTP servers, such as the HTTP server for iSeries 400 and now the **Apache** HTTP Web server, can be configured to use an LDAP directory when authenticating Web users. By configuring the HTTP server to use the LDAP directory-based user registry, visitors can be restricted to selected Web pages.
- Web application servers, such as the **IBM** WebSphere Application Server for iSeries 400, can be configured to use an LDAP directory when authenticating Web users. By configuring a Web application server to use the LDAP directory-based user registry, visitors can be restricted to specific functions or operations provided by the Web site.
- Since LDAP directories are hierarchical, you can maintain multiple user registries in the same LDAP directory, but they are in different directory partitions or branches of the hierarchy. You can also organize the structure of each user registry based on information entered by the visitor.
- OS/400 directory services provides LDAP standard utilities for extracting information from the directory. Both workstation- and OS/400-based utilities can query for information about Web site visitors for marketing purposes.
- Since most Web browsers are LDAP-enabled, you can view the contents of the user registry directly from a browser. For example, the following URL would display the contents of the pet store's user registry stored in the LDAP directory on the system called myiSeries400:

```
Ldap://myseries400:389/cn=Visitors,ou=UserRegistry,o=PetStore??sub
```

- Changes to the visitor registration form require corresponding changes to the user registry's schema or data definition. Later in this

article, you'll see how easy it is to make these changes to the LDAP directory. Changes can be made by either editing the LDAP server's schema files in OS/400 V4R3 and V4R4—or, starting in OS/400 V4R5, changes can be made while the LDAP server is running, either by a command line utility or by using a GUI.

### Implementation Overview

In this section I'll look at some of the technologies used in the implementation of the Web site and show how they fit together in the implementation. The implementation is based on the following guidelines:

- Provide an introduction to various technologies used to implement server-side Web applications. A partial implementation is provided showing the use of HTML, JSPs, servlets, JavaBeans, LDAP, and available OS/400 technologies working together to implement a simple Web site.
- Provide a separation of various components. This separation allows for reuse of components, ease of implementation using specialized skills, and faster enhancements. The implementation closely follows the well-known Model-View-Controller (MVC) framework. As its name suggests, it is made up of three components.
- The model is responsible for the underlying data and transactions that can be associated with it. This is the business logic. In the example implementation, the model is composed of JSPs and servlets.
- The view is responsible for displaying the data. In the example implementation, the view is the Web browser.
- The controller is responsible for decoupling the interactions between the model and the outside world. This is the interaction control and need have no knowledge of how the View works. In the example implementation, the controller is composed of JavaBeans, Java Naming and Directory Interface (JNDI), and an LDAP server.
- Provide a means for data interchange between components. The implementation uses XML, HTTP, and LDAP for data interchange between various components.
- Figure 1 shows both the components of the implementation and a simple function flow.

In the sections below I'll discuss the components, technologies, and the flow.

### Component Overview

On the workstation, the Web browser is the only component present. Since this is a server-side implementation, no data or HTML files are stored on the workstation. For the server-side implementation, the following components, listed by technology, are implemented on the iSeries 400:

- **JSPs**—The UserRegistration JSP provides the presentation logic by building the HTML form used for visitor registration. JSPs are text files stored in the AS/400 Integrated File System (AS/400 IFS) and used by the IBM WebSphere Application Server when a Web browser references a URL that contains the name of the JSP.
- **Servlets**—The UserRegistration servlet provides the presentation logic for processing the registration form's data. The servlet is invoked by the IBM WebSphere Application Server when the form is submitted because the form's action URL references the servlet.

UserRegistration separates the presentation and business logic. The servlet transposes form data into XML and invokes the UserRegistry Bean.

- **JavaBeans**—Implementation consists of two JavaBeans, UserRegistry and LDAPBean, that implement the business logic. Together, these JavaBeans implement the accessor to the user registry data stored in the LDAP directory. Since JavaBeans provide access to the user registry, the Web site's JSP and servlet developers are not involved with working with an LDAP directory. At this time, the UserRegistry Bean is invoked only by the UserRegistration servlet, but in the future, expect additional JSPs and servlets to use this Bean to access the user registry.

IBM's LDAPBean can be used to access LDAP directories using JNDI. JNDI consists of **Sun Microsystems'** API and IBM's JNDI service provider. Use LDAPBean as is, and implement all user registration specifics in the UserRegistration Bean.

- **LDAP**—For Web site visitor information, instead of using HTTP session information stored as cookies on the workstation, use an LDAP directory for centralized management.

```

dn:cn=schema
changetype: modify
add: attributetypes
attributetypes: ( animalInterests-oid NAME 'animalInterests' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
ibmattributetypes: ( animalInterests-oid ACCESS-CLASS normal LENGTH 256 )

Dn:cn=schema
Changetype:modify
add: objectclasses
objectclasses: ( AFVisitor-oid NAME 'AFVisitor' SUP top AUXILIARY MAY ( animalInterests ) )

```

**Figure 2: Personalize your Web site with visitor-specific information using LDIF files to make schema modifications.**

Use of an LDAP directory allows more control and options for using the information about Web site visitors, as discussed earlier.

Two of the LDAP components, JNDI and the LDAP server, are provided with OS/400 and implement the data management of the business logic. JNDI interacts with the LDAP server using LDAP (protocol). The LDAP server's data repository is the local OS/400 UDB DB2 database.

### Implementation Flow

Before diving into the implementation details, I'll take a look at the simple flow implemented so far for this pet store's Web site by following the steps shown in Figure 1.

## User registry data is kept in a repository that may be in a directory.

1. The JSP UserRegistration was invoked by a URL reference to the JSP. This reference was probably from a New Visitor link on the Web site's home page. This link was processed by the browser, which passed the reference to the OS/400 HTTP Web server, which then passed the reference to the OS/400 application server, IBM WebSphere Application Server, which then invoked the JSP. The JSP built a simple form allowing users to register information about themselves and then returned the form to the Web browser.
2. When the visitor clicks on the form's Submit button, the data is sent to the UserRegistration servlet. UserRegistration extracts the form data and structures it into an XML document using an XML schema specific to the Web site. Refer to my sidebar "Configuring LDAP as an E-business Registry: Additional Implementation Details" at [www.midrangecomputing.com/mc](http://www.midrangecomputing.com/mc) for more information on how this is accomplished. For the pet store Web site, user information is first converted in an XML document using this XML schema. This provides for structured data to be interchanged between the server-side components of the Web site.
3. The Web site is implemented using the UserRegistry Bean, which provides a set of

simple operations for JSPs and servlets to use when accessing the user registry. The UserRegistration servlet invokes the add method of UserRegistry Bean to add the visitor's information to the user registry. The visitor's information is then passed as a structured Directory Services Markup Language (DSML) document.

4. The UserRegistry Bean's add method parses the DSML document and invokes the LDAPBean requesting a directory entry to be created.
5. LDAPBean uses JNDI to request the LDAP server to add a new user to the user registry. LDAPBean hides many of the details of using JNDI such as connecting to the

LDAP server, formatting input data, and handling responses.

6. The LDAP server creates the directory entry by using DB2 UDB interfaces to update the OS/400 database.

### Updating the LDAP Schema

An LDAP server uses an LDAP schema to identify the attributes and object classes allowed when creating directory entries. In support of the user registry, you'll need to enhance the default LDAP schema shipped with OS/400 directory services so that additional attributes can be used when creating directory entries for the users. Most of the attributes needed for visitor information are already defined and contained in the object class iNetOrgPerson, an object class defined by an Internet standard and shipped with most LDAP servers.

To manage the visitor information in the OS/400's LDAP directory, add one more attribute, animalInterests, and the additional object class, AFVisitor, containing this new attribute. Methods for modifying the schema vary depending on your version of the LDAP server, which corresponds to the OS/400 release. IBM recommends enhancing the existing schema by adding only new attributes and object classes and not modifying the existing definitions.

For OS/400 V4R3 and V4R4, the LDAP Version 2-compatible schema modifications are applied to text files stored in the iSeries IFS. Attribute changes should be appended to the end of /qibm/userdata/os400/dirsrv/UsrAt.txt. Object class modifications should be appended to the end of /qibm/userdata/os400/dirsrv/UsrOC.txt.

The following code shows the changes added to UsrAt.txt to define one additional attribute called animalInterests, with a syntax of case ignore string (cis), a maximum length of 256 bytes, and normal access control permission:

```

attribute animalInterests cis
animalInterests 256 normal

```

The changes to be added to UsrOC.txt to define the AFVisitor object class that allows the optional attribute animalInterests are as follows:

```

objectclass AFVisitor allows
animalInterests

```

For OS/400 V4R5, the LDAP Version 3-compatible schema modifications are applied by one of the following methods:

- Use the Directory Management Tool (DMT), a workstation-based Java application. DMT is useful for making a small number of dynamic changes to an LDAP schema. Refer to the Download and Installation section for information on installing DMT. Once installed, DMT can be launched as follows: Start/Program/IBM SecureWay Directory/Directory Management Tool.
- Modify LDAP server's schema text files as mentioned earlier. Note that this requires restarting the server after changes are made.
- Use OS/400 Operations Navigator to import the schema modifications while the LDAP server is down. The schema modifications must be defined in a text file in LDAP Data Interchange Format (LDIF).
- Use the ldapmodify utility from a workstation command line, OS/400 command line, or OS/400 Qshell command line. The schema modifications must be defined in a text file in LDIF. The command from OS/400 Qshell would be as follows:

```

ldapmodify -d <admin_name> -w
<admin_password> -f <ldif_pathname>

```

Note that when using ldapmodify from the OS/400 command line, the text file must be located in the AS/400 IFS and the -h (hostname) and -p (port number) options are not required. Figure 2 shows the LDIF file to modify the schema. LDAP Version 3 requires an object identifier (OID) to be included in the schema. For this example, a text string is used to represent the OID of the new attribute and the new object class.

### User Registration Servlet


The UserRegistry servlet is invoked by an

action of the visitor registration form. The servlet uses XMLTransform's methods to convert the form's data into an XML document, and then it transforms the XML document into a DSML document. The DSML document is then passed to the UserRegistry Bean's add method to update the user registry with the new visitor's information. UserRegistry dynamically creates and returns an HTML page with the results of the registration to the Web browser.

### **UserRegistry Bean**

UserRegistry is a nonvisual JavaBean specific to the user registry. It provides properties that allow JSPs and servlets to customize a user registry. For example, setUserRegistryParentDN() allows a servlet to override the user registry's default location in the directory hierarchy. The UserRegistry servlet uses the add method of UserRegistry to add new users. After setting several properties, UserRegistry Bean invokes LDAPBean's add method to add a user. UserRegistry's event listener receives events from LDAPBean, translates the events into messages, and sets a status property.

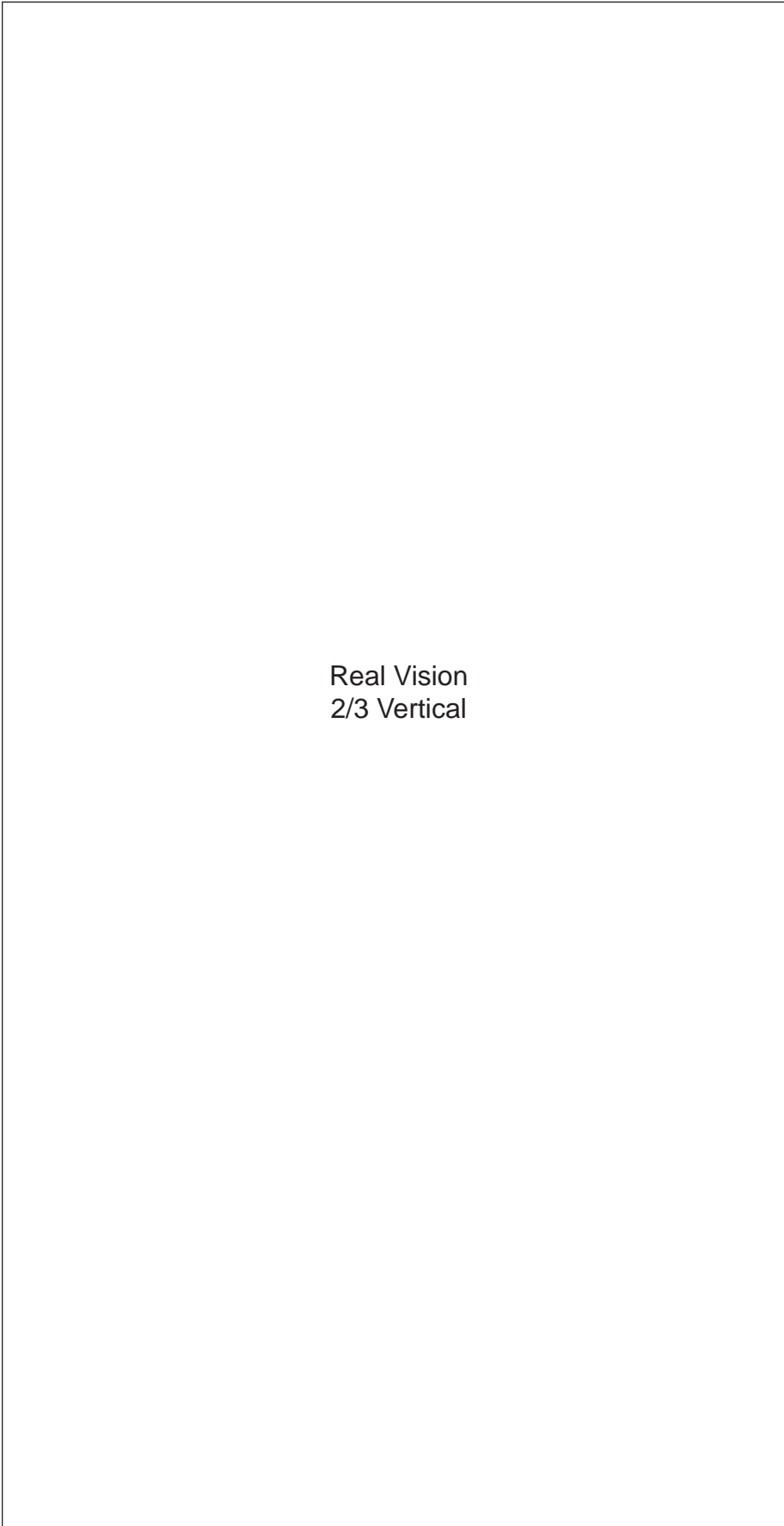
### **Working Together**

The examples in this article demonstrate many of the newest Internet technologies available from an iSeries that can work together to maintain a Web site's user registry in an LDAP directory. These examples can be expanded to provide additional user management facilities, such as updating existing visitor information. JSP and servlet developers will find it much easier to work with a user registry through the use of XML to structure data and JavaBeans as accessors. For additional implementation details and Installation and Configuration instructions, be sure to read the sidebar "Configuring LDAP as an E-business Registry: Additional Implementation Details" on the MC Web site at [www.midrangecomputing.com/mc](http://www.midrangecomputing.com/mc). 

**Pat Fleming** is a senior software engineer for the iSeries at IBM in Rochester, Minnesota. He is currently an IBM WebSphere Application Server for iSeries architect. Prior to that, he was an OS/400 Directory Services (LDAP) architect. He can be reached at [flemingp@us.ibm.com](mailto:flemingp@us.ibm.com).

### **REFERENCES AND RELATED MATERIALS**

- IBM HTTP Server page: [www.as400.ibm.com/products/http/httpindex.htm](http://www.as400.ibm.com/products/http/httpindex.htm)
- IBM WebSphere page: [www.as400.ibm.com/WebSphere](http://www.as400.ibm.com/WebSphere)
- iSeries 400 Directory Services (LDAP) Web page: [www.as400.ibm.com/ldap](http://www.as400.ibm.com/ldap)
- *XML Bible*. Elliotte Rusty Harold. New York City: Hungry Minds, 1999
- Zvon XSL Tutorial Web site: [www.zvon.org/HTMLOnly/XSLTutorial/Books/Book1/index.html](http://www.zvon.org/HTMLOnly/XSLTutorial/Books/Book1/index.html)



Real Vision  
2/3 Vertical