

Oracle XML DB 11g Release 1
X-Files demonstration: Installation notes
and User-guide.

An Oracle White Paper
October 2007

Introduction.....	2
Architecture	2
Installation.....	4
Server Pre-requisites.....	4
Oracle Software.....	4
Client Pre-requisites	4
Oracle Software.....	4
Non Oracle Software.....	4
Database Configuration Changes.....	5
Installing the Application	6
Overview	9
Using the X-Files application	11
Database Native Web Services	21
XML Searching.....	30
XML Index-driven search	30
XML DB Repository full-text search	39
XMLSchema-driven searching	40

Oracle XML DB 11g Release 1

X-Files demonstration: Installation notes and User-guide

Introduction

The X-Files application is designed to high-light some of the key features of the Oracle XML DB Repository. It provides a simple browser-based user interface, modeled after Oracle's Files-Online product to the Oracle XML DB repository. In particular the application demonstrates the following features of Oracle XML DB

- Developing simple Oracle XML DB applications using HTTP, Java and XSL stylesheets.
- Deploying Java servlets in the database under the Oracle XML DB HTTP server.
- Using the Oracle XML DB XSLT Virtual Machine to perform XSLT processing.
- The Content Management (CM) capabilities of the Oracle XML DB repository
- Repository based full text searching
- XML Index and XML Schema based intelligent search
- Developing AJAX based applications using Database Native Web Services

Architecture

The X-Files application consists of a collection of Java servlets and XSL Stylesheets. The servlets are accessed via the Oracle XML DB Native HTTP server and executed using Oracle Database's Java Virtual Machine. Each Servlet generates an XML Document which represents the result of the operation it performs. The XML is transformed into HTML using Oracle Database 11g's native XSLT virtual machine (XSLT-VM). The XSLT-VM allows Oracle XML DB to provide high-performance, scaleable XSLT processing.

The Java Servlets use a mixture of JNDI, JDBC and PL/SQL to access the contents of the XML DB repository. The JNDI services are provided by a simple JNDI implementation that uses JDBC to map standard JNDI methods directly into operations on RESOURCE_VIEW and PATH_VIEW or the functions and methods provided by the DBMS_XDB package. The JNDI specification provides a very natural abstraction for providing the basic functionality of a content management system.

Simple repository operations such as lookup, list directory, new folder, delete and save make use of the JNDI implementation. More advanced operations such as *version* and *publish* are performed using methods that use JDBC to directly invoke the appropriate PL/SQL packages.

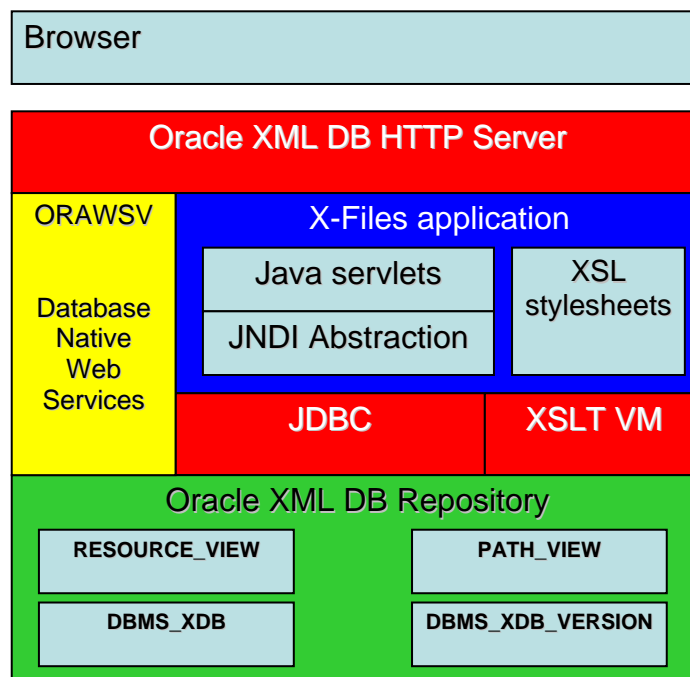
The X-Files application makes extensive use of the XSLT rendering capabilities of Oracle XML DB. The entire user-interface is generated using a set of simple XSLT style-sheets which transform the XML documents generated by each servlet into the HTML pages seen by the user.

This approach has the advantage that the definition and generation of the look-and- feel of the application is totally abstracted from the XML that defines which objects appear on each page. The appearance of the application can be altered by simply replacing the XSL style sheets. The “View XML” and “View XSL” buttons at the bottom of each page of the X-Files application allow direct access to the raw XML and associated XSL style sheet.

Certain parts of the X-Files application, such as the Source Code Viewer, and Search interface, demonstrate an alternative approach, based on Database Native Web Services (DNWS) and AJAX. This allows for a more interactive user-experience, where sections of the HTML page are regenerated dynamically. The AJAX approach migrates application logic from the database to the web browser.

In this model, the application logic is coded in JavaScript rather than Java. When the application requires data it makes SOAP requests to the ORAWSV servlet. This servlet can execute dynamic SQL or XQuery statements and invoke PL/SQL procedures and functions. The results are returned as XML to the client where browser based XSL transformation is applied to generate the user-interface.

The architecture of the X-Files application is summarized in the following diagram.



Installation

Server Pre-requisites

The following software is required to run the X-Files application

Oracle Software

- Oracle Database 11g release 11.1.0.6.0 or later, with the XML DB, Oracle Text and Oracle JVM features installed.

Client Pre-requisites

The installation process uses VB Scripting and the HTTP protocol to upload the source code into the Oracle XML DB repository, SQL*PLUS scripts are used to compile the Java servlets, register the virtual path for each servlet with the Oracle XML DB HTTP server and configure the repository.

It should be noted that since the application is a pure-Java application, it could also be deployed using a Java container such as Oracle's OC4J or the Oracle Internet Application Server.

The following software is required to install the X-Files application

Oracle Software

- Oracle Client (SQL*PLUS and Oracle Net Services) 11.1.0.6.0 (Production) or later. The application can be installed into a remote database, however both SQL*PLUS and Oracle Net Services must be installed on the client machine in order to perform a remote install. Currently remote installs are only supported on the Windows platform.

Non Oracle Software

- Microsoft Internet Explorer 7.0 with the latest service packs. The X-Files application has not been tested with any release of other browsers including Firefox, Mozilla, Netscape, Safari or Opera.
- Microsoft Windows Scripting Technologies version 5. Windows Scripting is used by the installation process to load the application source code into the Oracle XML DB repository. You can verify the version of Windows Scripting installed on your machine by opening a command prompt and typing the command cscript.
- Microsoft Core XML Services (MSXML) versions 4.0sp2 and 6.0 are required in-order to perform a remote install and run the application.

MSXML is used to process the configuration files when installing X-Files into a remote database. It is also required by the AJAX based components of the X-Files application.

At the time of writing the latest version of this software can be downloaded from <http://www.microsoft.com/downloads/Search.aspx?displaylang=en>

Database Configuration Changes

Installing the X-Files application will make the following changes to the configuration of the target database.

- **Oracle XML DB HTTP Server:** Installing the X-Files application will enable the database's native HTTP Server. Ensure you have read the XML DB documentation regarding the use of the XML DB HTTP Server before installing this application into a database that contains production data. This information can be found in the Oracle XML DB Developers guide.
- **Database Native Web Services:** Some components of the X-Files application use AJAX,. Consequently installing the X-Files application will enable the Database Native Web Services servlet. Ensure you have read the XML DB documentation regarding the use of the Database Native Web Services before installing this application into a database that contains production data. This information can be found in the Oracle XML DB Developers guide.
- **Database Schema XFILES:** The X-Files application requires that a database schema called XFILES. Installing the X-Files demonstration will automatically create this user. If the XFILES schema already exists it will be dropped as part of the installation process.
- **XML DB Repository:** The X-Files application is installed in the XML DB repository. The following folders will be deleted and recreated during the installation process.

/home/XFILES
/XFILES
/CABO

Any existing content in these folders be deleted as part of the installation process.

- **XML DB repository search:** Installing the X-Files application will enable text searching of the repository. The package DBMS_XDBT will be used to create a full-text index on the contents of the repository. Ensure you have read the XML DB documentation regarding the use of the DBMS_XDBT package before installing this application into a database that contains production data. This information can be found in the Oracle XML DB Developers guide.
- **Database Schema XDBPM:** The X-Files demonstration also requires the XDBPM packages be installed. These packages are installed into a locked database schema called XDBPM. Installing XFILES will drop and re-create the database schema XDBPM and re-install the XDBPM packages. XDBPM is a locked account with XDBADMIN role. It also has read access to some catalog tables owned by XDB and SYS. This account should remain locked.
- **Anonymous User:** Installing XFILES will unlock the anonymous user and configure XML DB so that the anonymous user can access all repository content secured with an ACL that grants read access to public. This prevents users from having to authenticate with the database before being able to access the XFILES system. Once the Anonymous account is unlocked all users will have read-only access to public content without have to authenticate with the database. Users will only

have access to content that is not readable by public once they have successfully authenticated with the database.

Ensure you have read the documentation related to unlocking the anonymous account and enabling anonymous access to the XML DB repository before installing this application. This information can be found in the Oracle XML DB Developers guide.

Installing the Application

To install the Oracle XML DB 11g Release 1 X-Files application unzip the contents of the file XMLDB_11gR1_XFiles.zip into a folder of your choice. Ensure that there are no spaces in any of the parent folder names. After unzipping this file the target folder should contain a subfolder called XFilesIII. Edit the file config.bat (Windows) or config.sh (Unix). The file declares the environment variables that define where the XFiles demonstration will be installed. On Windows the file looks like this:

```
set HOSTNAME=xfiles.oracle.com
set ORAHTTPPORT=80
set USERTBLSPACE=USERS
set TEMPTBLSPACE=TEMP
set ORASQLUSER=XFILES
set ORASQLPASSWORD=xfiles
set ORASYSPASSWORD=oracle
set TNSALIAS=XFILES
set REMOTE=FALSE
```

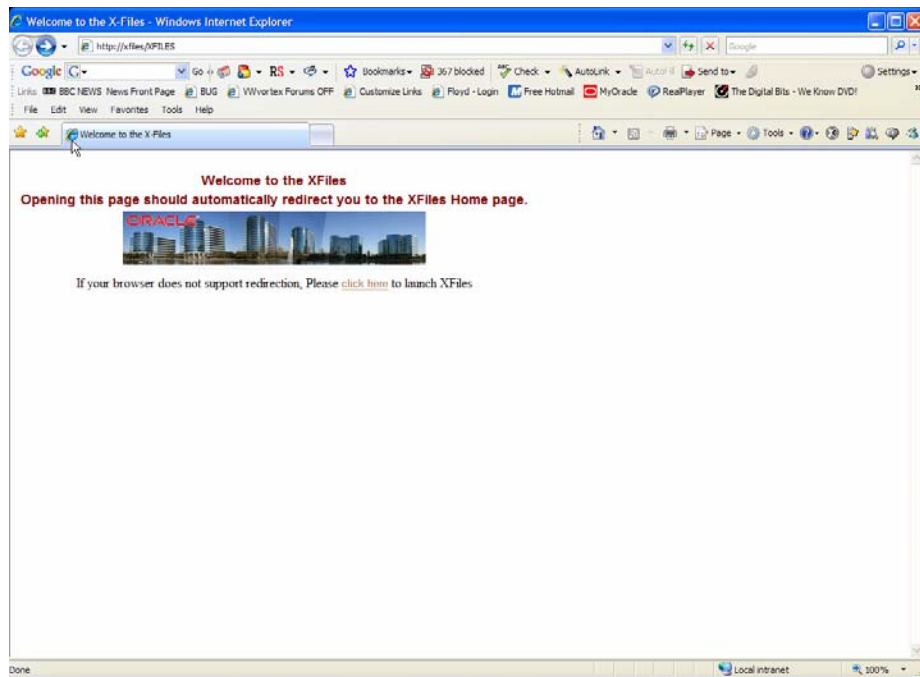
Edit the file to ensure that the values of these parameters are customized for your environment before starting the installation. The parameters are as follows

HOSTNAME	The name of the machine running the database.
ORAHTTPPORT	The Port number that will be used by the Oracle XML DB HTTP Server. Make sure this port number is not currently in use.
USERTBLSPACE	The tablespace that will be used by the XFILES account.
TEMTBLSPACE	The temporary tablespace that will be used by the XFILES account
ORASQLUSER	The database schema for the XFILES account. Do not change this value
ORASQLPASSWORD	The password for the XFILES database user. You may change this value
ORASYSPASSWORD	The sys password. The XFILES installation script needs to connect as SYSDBA in order to set up the XDBPM and XFILES accounts.
TNSALIAS	The name of the database instance where XFILES is to be installed
REMOTE	This should have the value 'TRUE' if the target instance is on a remote machine, FALSE otherwise. If the target instance is remote then HTTP will be used to upload the source documents

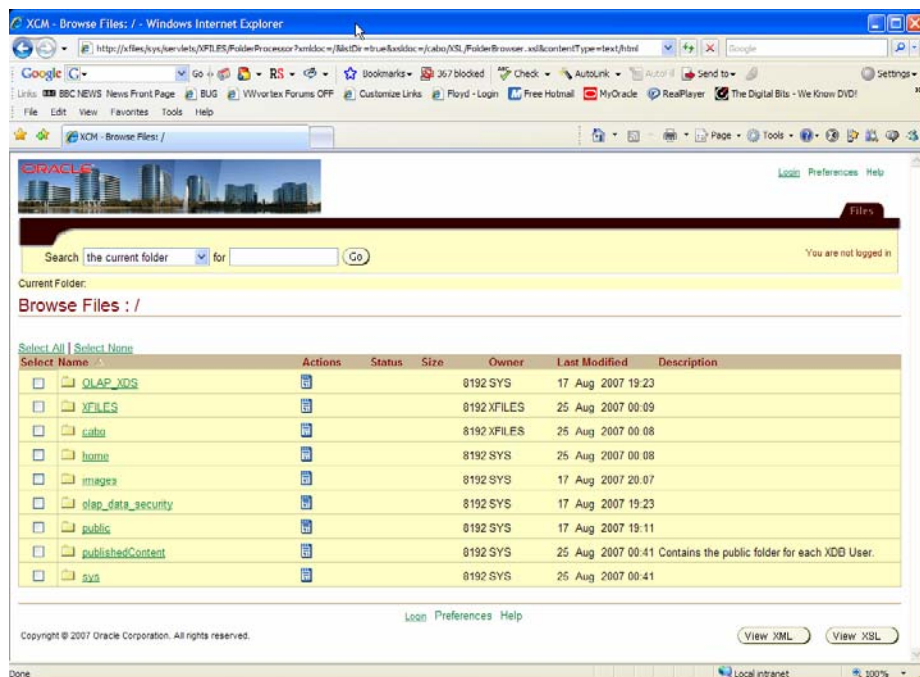
Once the values in config.bat or config.sh are correct install the application by running install.bat (Windows) or install.sh (Unix/Linux).

Once the installation is complete you should be able to access the XFILES home page at the following URL <http://hostname:oraclehttp/XFILES>

This should display the XFILES welcome screen:

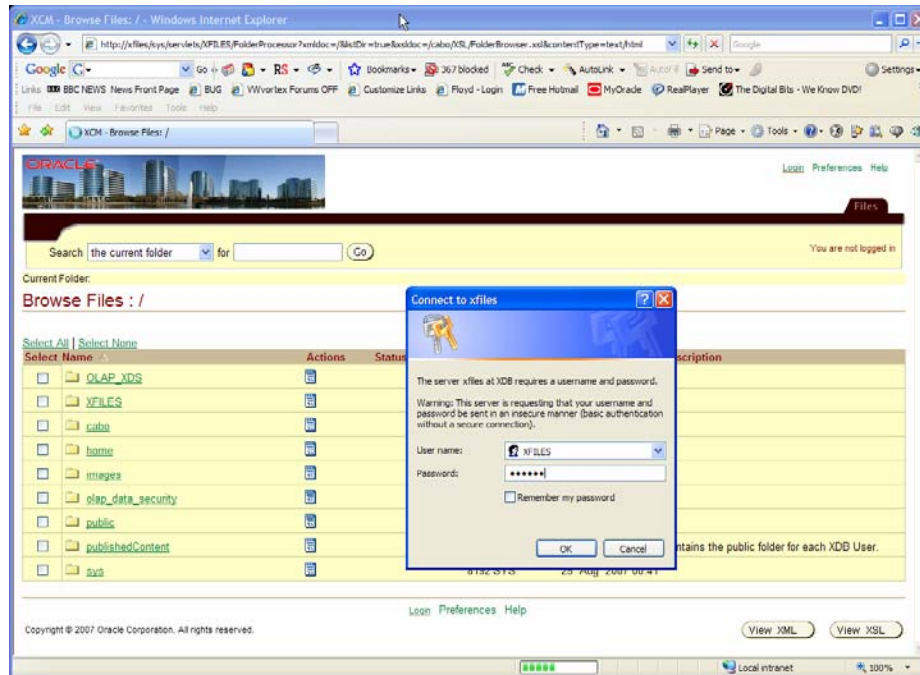


It should then automatically open the X-Files Folder Browser showing the root folder of the Oracle XML DB repository.



Overview

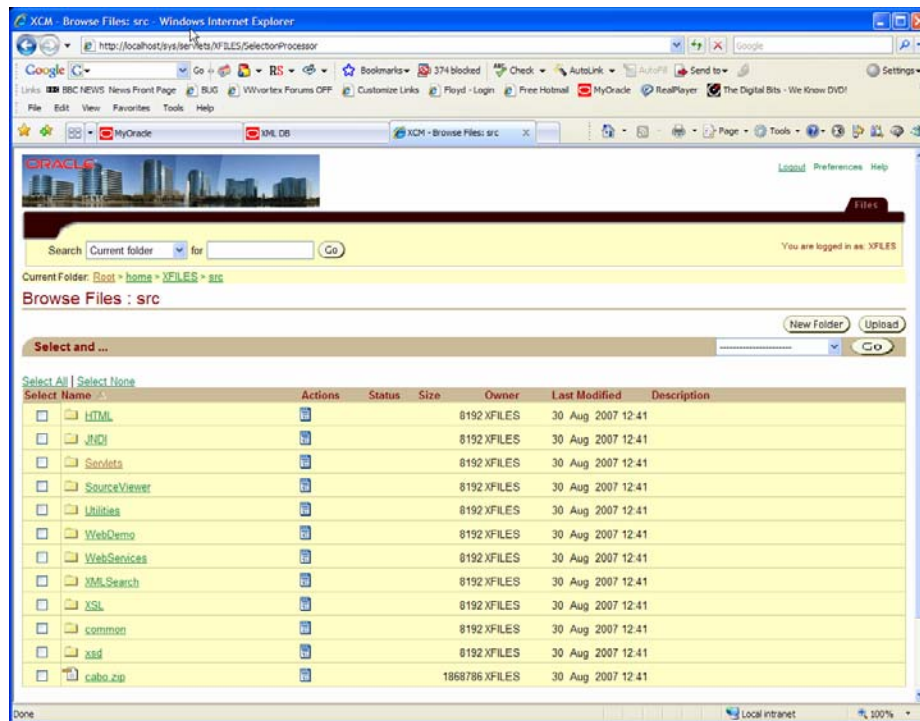
The X-Files application provides a simple intuitive HTML interface for browsing the Oracle XML DB repository as well as access to the basic content management and search capabilities of Oracle XML DB. In order to access the full functionality of the X-Files application it is first necessary to login to the database using HTTP authentication. Click the login link to authenticate with the database.



All of the source code for the X-Files application is loaded into the Oracle XML DB repository. The source code is stored in `/home/XFILES/src`. The key components of the application are as follows:

- The JNDI folder contains the source code the JNDI abstraction layer used by the X-Files application.
- The Servlets folder contains the source for the servlets and associated classes used by the X-Files application.
- The XSL folder contains the XSL stylesheets used to generate the X-Files user interface.
- The SourceViewer folder contains an AJAX-based applet for viewing source code.
- The WebDemo folder contains the source for a HTML based SQL*PLUS like applet, used to run some of the Oracle XML DB demonstrations
- The WebServices folder contains a small AJAX-based applet for demonstrating the database native web services capability of Oracle XML DB.

- The XMLSearch folder contains a set of small AJAX-based applet for text-based searching the Oracle XML DB repository, and intelligent XML-based searching of XML Schema content and XML Indexed content.



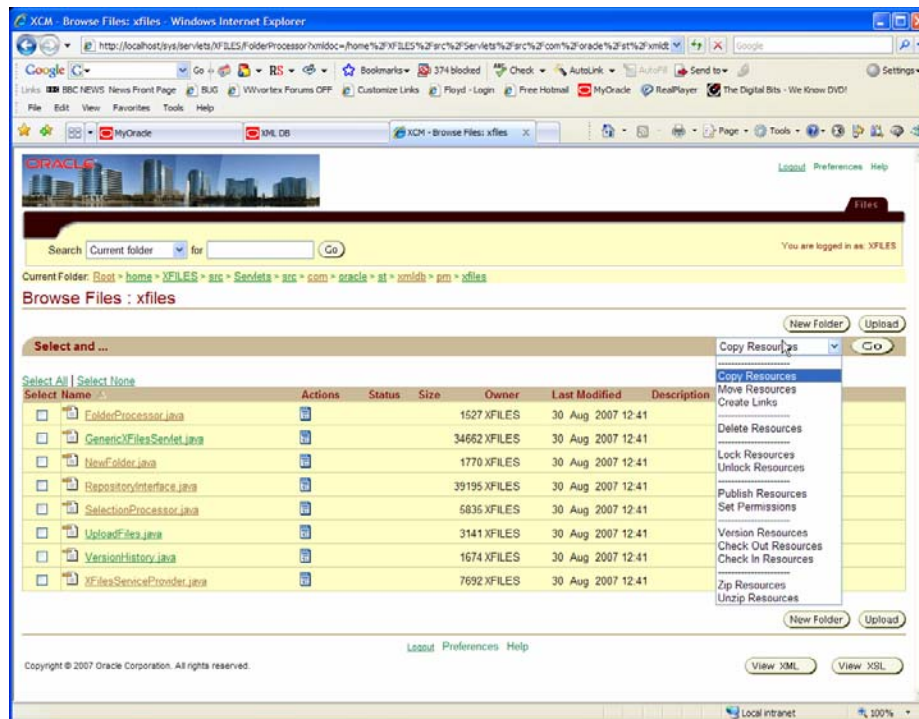
All of the servlets in the X-Files application extend from the GenericXFilesServlet. This servlet provides a framework that delivers basic services such as parameter processing, repository access and authentication, logging and error handling and XSL transformation. It uses the JNDI based RepositoryInterface and XFilesServiceProvider classes to access the XML DB repository.

The other servlets are as follows:

The FolderProcessor servlet is the foundation of the X-Files application. Its primary function is to generate the Folder Browser that is used to navigate up and down the folder tree and to select one or more of the resources in particular folder and perform a given operation on them. The servlet also provides functionality that allows the user to confirm the selected operation and to select a target folder for move, copy or link operations. The servlet enables copy, move, link, lock, unlock and publish operations on resources, changing the access control list on a resource, versioning, check-out and check-in operations as well as zipping and unzipping operations.

The SelectionProcessor servlet takes the list of resources generated using the FolderProcessor servlet and performs the specified operation on them.


The NewFolder servlet provides functionality to create a new Folder.





The UploadFiles servlet provides functionality to create new resources by uploading one or more files from the local file system. The servlet makes use of a multi-part mime processor to process the input stream generated by the browser. The servlet also supports automatic versioning of documents during upload processing.

The VersionHistory servlet provides functionality to access and display the version history associated with a versioned resource.

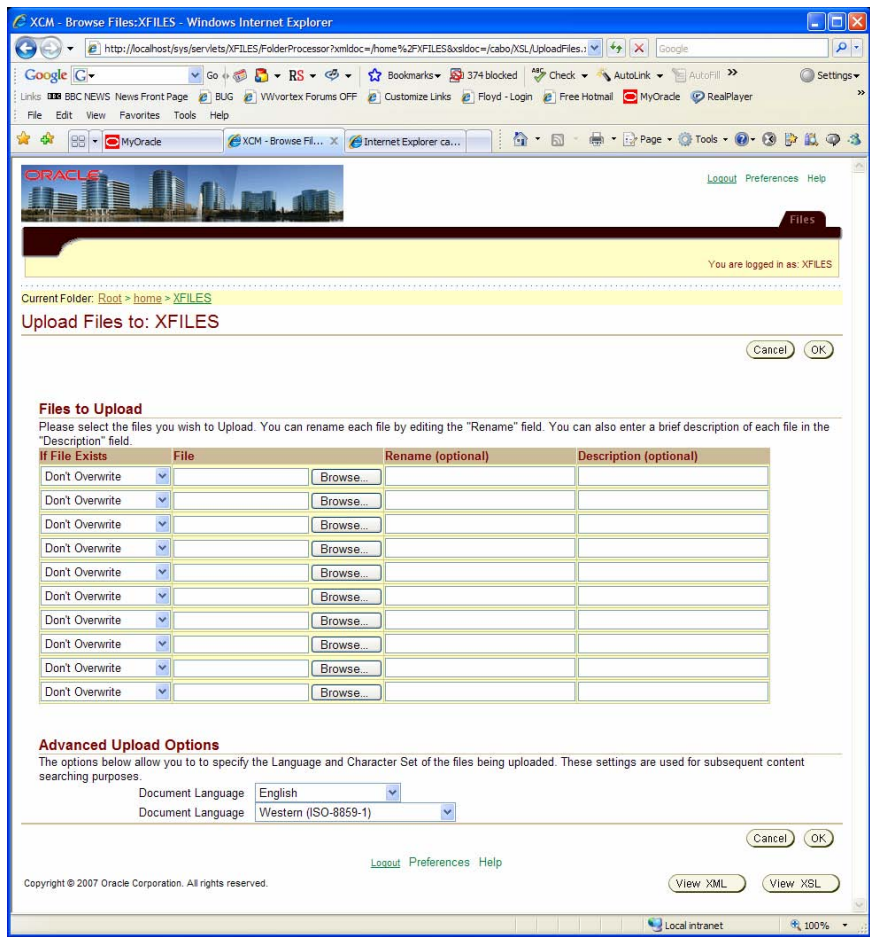
Using the X-Files application

Navigating the folder hierarchy using the X-Files application is very straightforward using the Folder Brower. In the Folder Browser each child folder of the current folder is represented by the  icon. The folder's name is a link. Clicking on the link will open the selected folder in the Folder Browser. The Folder Browser also shows the path to the current folder as a series of links. Each link represents one of the current folder's parents. Clicking on the link will open the selected folder in the Folder Browser

The list of resources display by the Folder Browser can be sorted by Name, Size, Owner and Last Modified Data. The currently sort column and order is indicated by the presence of a  (ascending) or  (descending) icon next to the column name in the column headings. The sort column is changed by clicking on the name of the new sort column. Clicking on the currently selected sort column reverses the direction of the sort.

New resources are created directly from the Folder Brower. If the current folder is protracted by an ACL that permits the current user to create a new resource in this location the New Folder and Upload buttons are displayed.

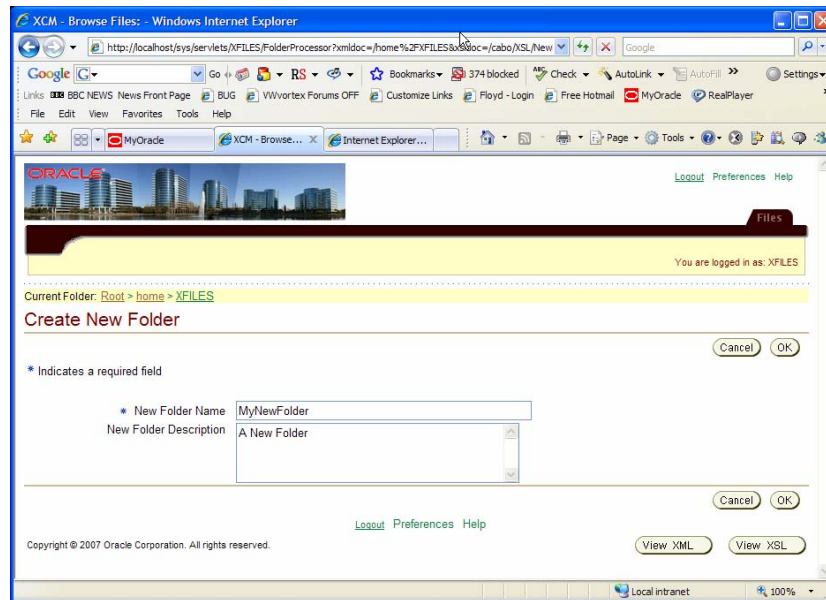
Clicking the Upload button opens the Upload Files dialog.



New resources are created by using the Browse button to select up to ten files from the local file system. The dialog allows an alternative name to and short description to be supplied for each file. It also allows the user to specify what action to take when the name of the file being uploaded matches the name of an existing resource. The available choices are Don't Overwrite, Overwrite or Create Version.

Once the files to be uploaded have been selected, clicking the OK button uploads the chosen files into the current Folder.

Clicking the New Folder button opens the New Folder dialog.



A new folder is created by entering a name and a short description for the new folder and clicking on the OK button.

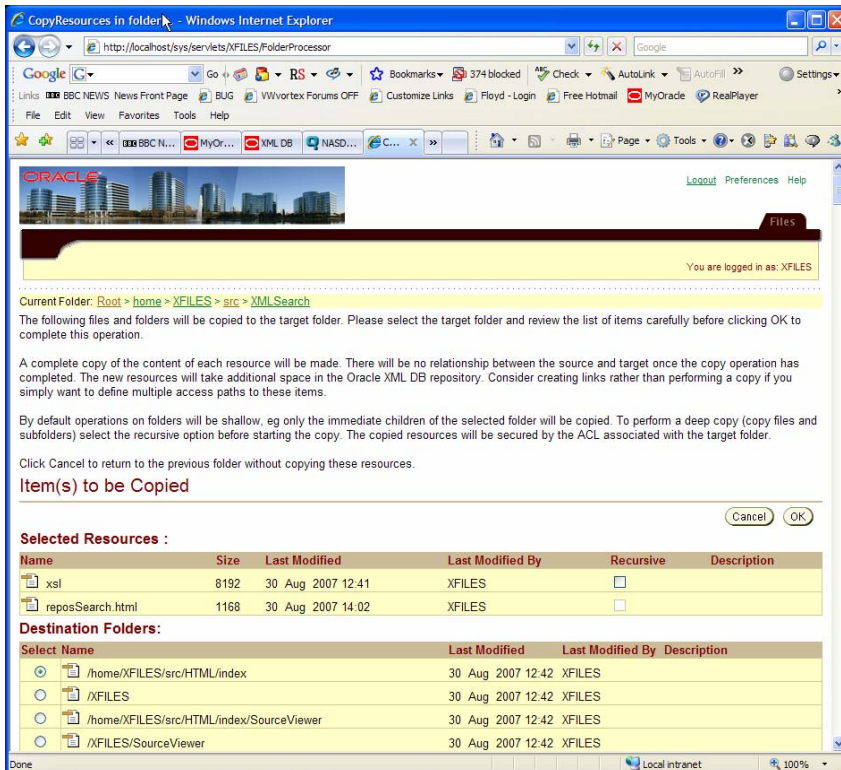
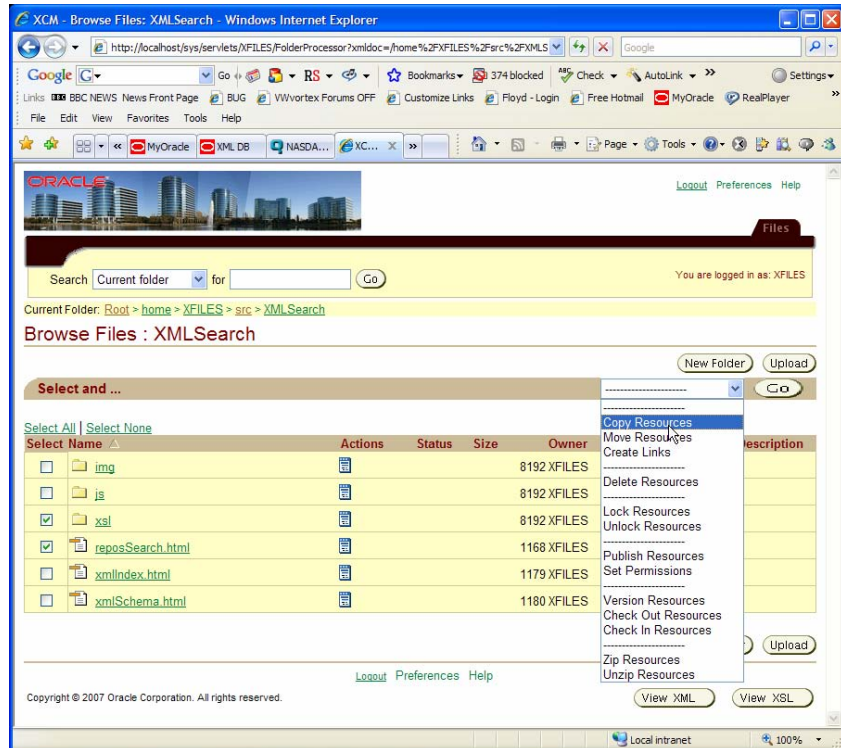
The Folder Browser is also used to perform content management operations on the resources in the current folder. To perform a given operation on one or more of the resources, choose the required operation from the drop down menu of supported operations. Select the set of resources to be operated on by placing a tick in the check box associated with each of the resources in question. Finally click on the GO button to initiate the operation.


Clicking the GO button will open the Operation Confirmation dialog. The dialog will display the set of selected resources and request confirmation that the chosen operation should be performed on these resources. At this point it may also be necessary to provide additional information in order for the operation to proceed. Examples of additional information include the target folder for an operation or indication of whether operations on folders should be recursive.

The Folder Browser supports the following operations

- **Copy Resource:** Copy the chosen resources to the destination folder. The destination folder and whether folder operations are recursive is specified using the Operation Confirmation dialog. Recursive operations copy all the files and folders that are descendants of the source folders. Non-recursive operations are restricted to copying files that are a direct child of the source folders.

A copy operation results in a completely new resource being created for each resource processed. There is no relationship between the source and target of the copy once the operation is complete. The new resources will take up additional space in the Oracle XML DB repository. The new resources will be secured by the ACL of the target folder.





- **Move Resource:** Move the chosen resources to the destination folder. The destination folder is specified using the Operation Confirmation dialog. A move operation has no effect on the ACL of the target resources.
- **Link Resource:** Create links to the chosen resources in the target folder. The target folder is specified on the Operation Confirmation dialog . Creating a new link to a folder allows the folder's contents to be accessed via the link, but does not create new links to the contents of the folder. A link operation has no effect on the ACL of the target resources.
- **Delete Resource:** Remove the chosen resources from the folder. Whether folder operations are recursive is specified using the Operation Confirmation dialog. Recursive operations will delete all files and folders that are descendants of the source folders. Non-Recursive operations will fail on a non-empty folder.
- **Lock Resource:** Take a persistent DAV Lock on the chosen resources. The target folder and whether folder operations are recursive is specified using the Operation Confirmation dialog . Recursive operations will lock all files that are descendants of the source folders. Non-Recursive folder operations will only lock files that are a direct child of the selected folder. Locked resources are indicated by the presence of the locked icon  in the status column of the Folder Brower.
- **Unlock Resource:** Release a persistent DAV Lock on the chosen resources. The target folder and whether folder operations are recursive is specified using the Operation Confirmation dialog. Unlock operations can only be performed on locked resources. Recursive operations will lock all files that are descendants of the source folders. Non-Recursive folder operations will only lock files that are a direct child of the selected folder.
- **Publish Resource:** Sets the ACL on the chosen resources to one that grants read-only access to public and links the resources into the user's publishedContent folder. The publishedContent folder tree is protected by an ACL that grants read-only access to public. The net effect of these operations is to make the chosen resources available to all repository users.


Publishing a folder creates a new folder and then creates links from the new folder to each of the resources in the source folder. This prevents subsequent recursive delete operations on the public folder from deleting the contents of the original folder. Whether folder operations are recursive is specified using the Operation Confirmation dialog. Recursive operations will process the source folder and all the files and folders that are descendants of the source folder. Non-recursive operations will only create links to files that are a direct child of the source folder.

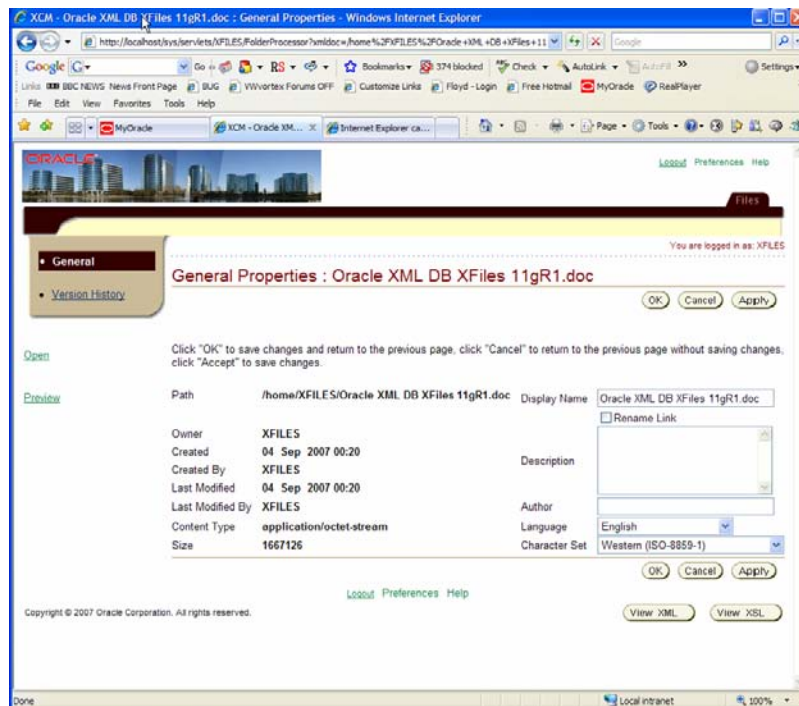
- **Set Permissions:** . Changes the ACL on the chosen resources. The ACL that is to be applied and whether folder operations are recursive is specified using the Operation Confirmation dialog. Recursive operations will change the ACL on all files and folders that are descendants of the source folders. Non-Recursive folder operations will only lock files that are a direct child of the selected folder.

- **Version Resource:** Initiates Oracle XML DB versioning on the chosen resources. Whether folder operations are recursive is specified using the Operation Confirmation dialog. Recursive operations will initiate versioning on all files and folders that are descendants of the source folders. Non-Recursive folder operations will only initiate versioning on files that are a direct child of the selected folder.

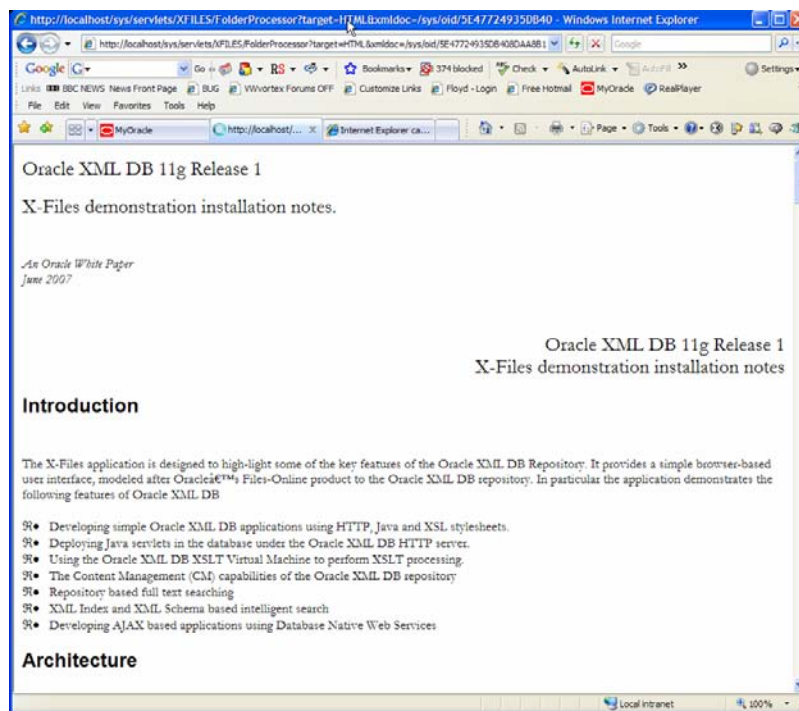
Once a resource has been versioned it can only be modified or deleted if it checked out. A versioned resource is indicated by the  icon in the status column of the Folder Browser.

- **Check-Out Resource:** : Check-out a versioned resource so that it can be modified. Checking out a resource creates a new version of the document that can be modified. Whether folder operations are recursive is specified using the . Recursive operations will check-out all files and folders that are descendants of the source folders. Non-Recursive folder operations will only check-out files that are a direct child of the selected folder. A checked-out resource is indicated by the presence of the  icon in the status column of the Folder Browser.
- **Check-In Resource:** : Check-in a versioned resource so that it can be modified. Secure the current version of a checkout document so that no further modification can be made to this version. Whether folder operations are recursive is specified using the Operation Confirmation dialog . Recursive operations will check-in all files and folders that are descendants of the source folders. Non-Recursive folder operations will only check-in files that are a direct child of the selected folder.
- **Zip Resource:** Creates a zip file containing the selected resources. The zip file will be named *currentFolder.zip*. Whether folder operations are recursive is specified using the Operation Confirmation dialog. Recursive operations will zip all files and folders that are descendants of the source folders. Non-Recursive folder operations will only zip files that are a direct child of the selected folder.
- **Unzip Resource:** Unzips the selected zip files into the current folder.

The Folder Browser also provides access to the Resource Properties dialog. To access the Resource Properties dialog click the Resource Properties  icon. The Resource Properties dialog displays all the properties of the selected resource, and allows the file name, description and author to be edited.

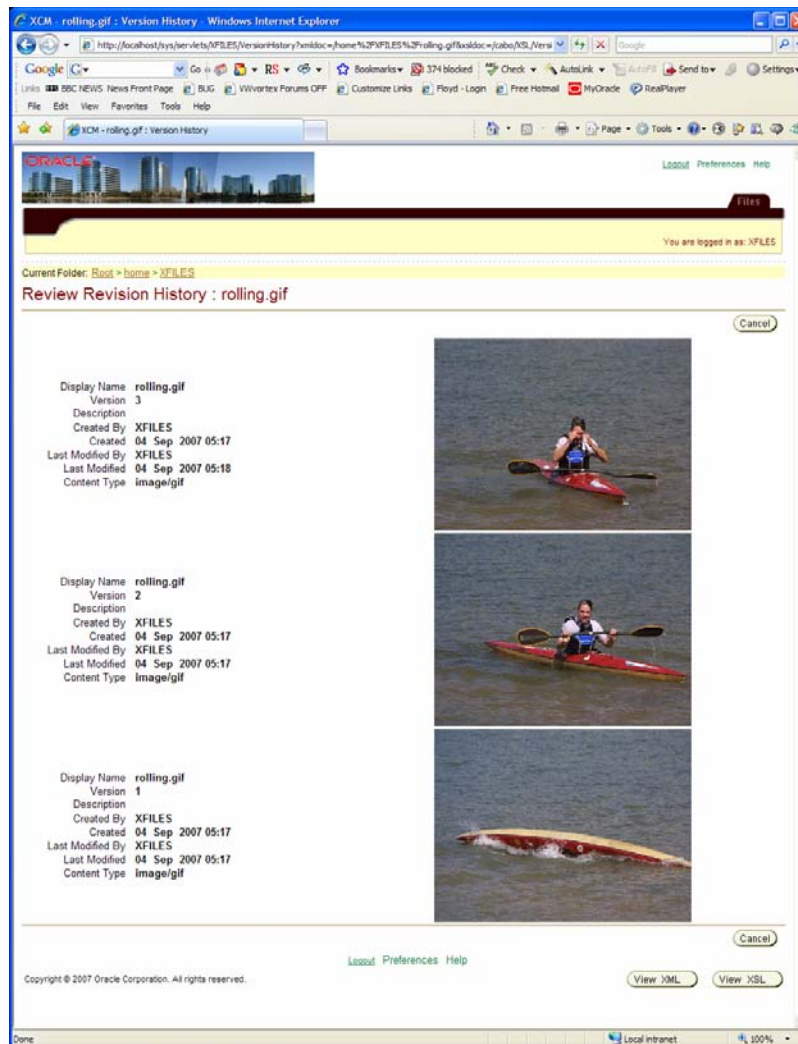


The Resource Properties dialog also makes it possible to open or preview the content of the resource.



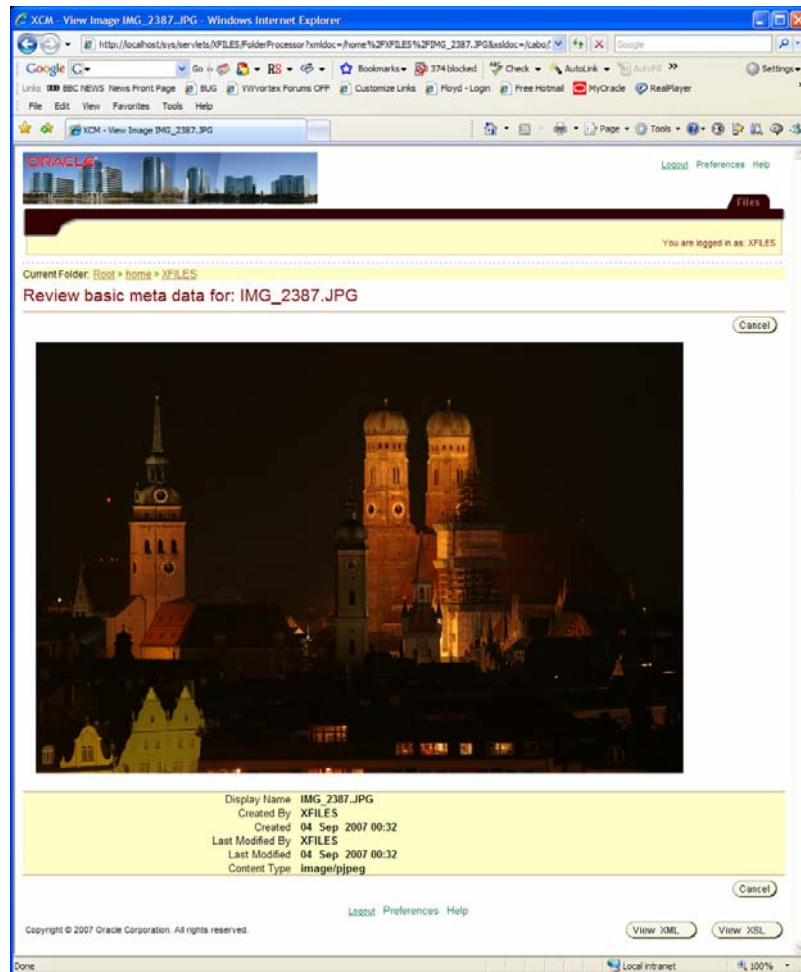
The preview is generated using the HTML rendering capabilities of Oracle Text and is only available in databases where the Oracle Text option has been installed.

The X-Files application is capable of displaying versioned resources. Clicking the versioned resource icon on the Folder Browser will open the version history dialog and show the metadata for each document in the version series. It will also show a thumbnail of the resource content for each version (In the current implementation thumb-nailed content is only available for images.)



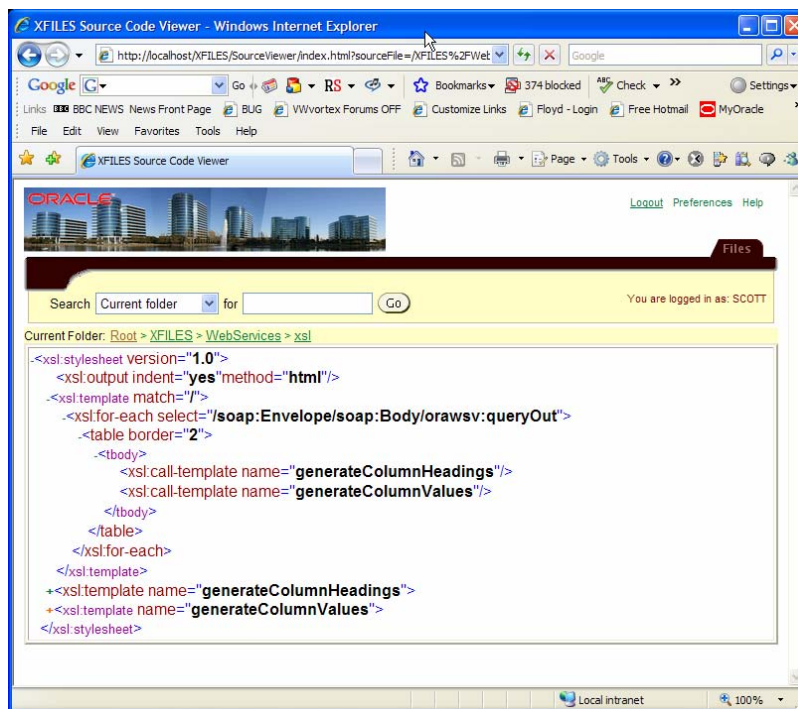
The X-Files application also includes a number of content-type specific viewers. Clicking on a specific type of resource will open the resource using the appropriate content viewer. Examples include:

- An Image Viewer, capable of displaying the extended metadata available with certain kinds of digital photograph where appropriate.



- A Purchase Order Viewer for use with the XML documents included with the Oracle XML DB basic features demonstration.
- A source code view that provides a convenient method of viewing the various different kinds of source code used by the X-Files application. This viewer is used to view Java, JavaScript and SQL files as well as various kinds of XML documents, including XML schemas and XSL stylesheets. The viewer is automatically invoked whenever a supported document is opened inside the X-Files application.

The viewer is a true AJAX application that makes use of Database Native Web Services. This means that viewer can only be accessed by users who have been granted the appropriate permissions.

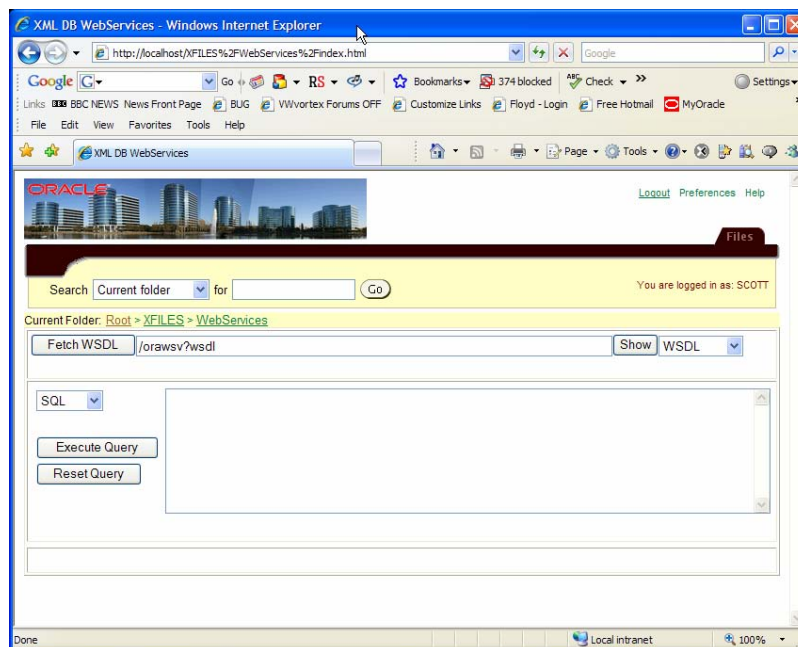


Database Native Web Services

The X-Files application acts as framework to highlight other features of Oracle XML DB. One of these is Database Native Web Services (DBNWS). Database Native Web Services simplifies the process of developing and deploying Web Services that provide access to data stored in an Oracle Database. They allow any PL/SQL stored procedure, function or package to be exposed as a SOAP service with zero-development and zero-deployment effort. Database Native Web Services also includes a SOAP service that supports execution of dynamic SQL queries and XQuery expressions. The database HTTP server, provided as part of Oracle XML DB Repository, allows these Web services to be accessed using HTTP and HTTPS, without any additional application server infrastructure.

Database Native Web Services are very efficient. They leverage the powerful XML capabilities of Oracle XML DB to automatically generate the WSDL, parse and process the SOAP request, and generate the SOAP response.

The DBNWS demonstration is located at the following URL: <http://server:port/XFILES/WebServices>. It can also be access by navigate to the folder /XFILES/WebServices using the Folder Browser and clicking on the index.html file. You must be connected as an authenticated database user in order to access the DNWS demonstration. The main page of the DNWS demonstration is shown below:



Database Native Web Services are implemented by the ORAWSV servlet, which is activated as part of the X-Files installation.

There are three additional permissions that are required in order to use the ORAWSV servlet and run the DBNWS demonstration. These are:

- **XDB_WEBSERVICES:** Enables access the ORAWSV servlet.
- **XDB_WEBSERVICES_WITH_PUBLIC:** Enables access to public stored packages, functions procedures and objects via the ORAWSV servlet.
- **XDB_WEBSERVICES_OVER_HTTP:** Enables access to the ORAWSV servlet via the HTTP protocol. Without this permission the ORAWSV servlet can only be accessed using the HTTPS protocol.

These permissions can be granted to a specific user by connecting to the database as a DBA and executing the `enableWebServicesDemo()` procedure supplied by package `xdbpm.xdb_demo_helper`. The procedure expects a single argument which is the name of the database user. For example, to allow user SCOTT to use the DBNWS demo the following commands needs to be executed:

```
sqlplus system

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Sep 4 08:41:00 2007
Copyright (c) 1982, 2007, Oracle. All rights reserved.
Enter password:

Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - Beta
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> call xdbpm.xdb_demo_helper.enableWebServicesDemo('SCOTT');

Call completed.

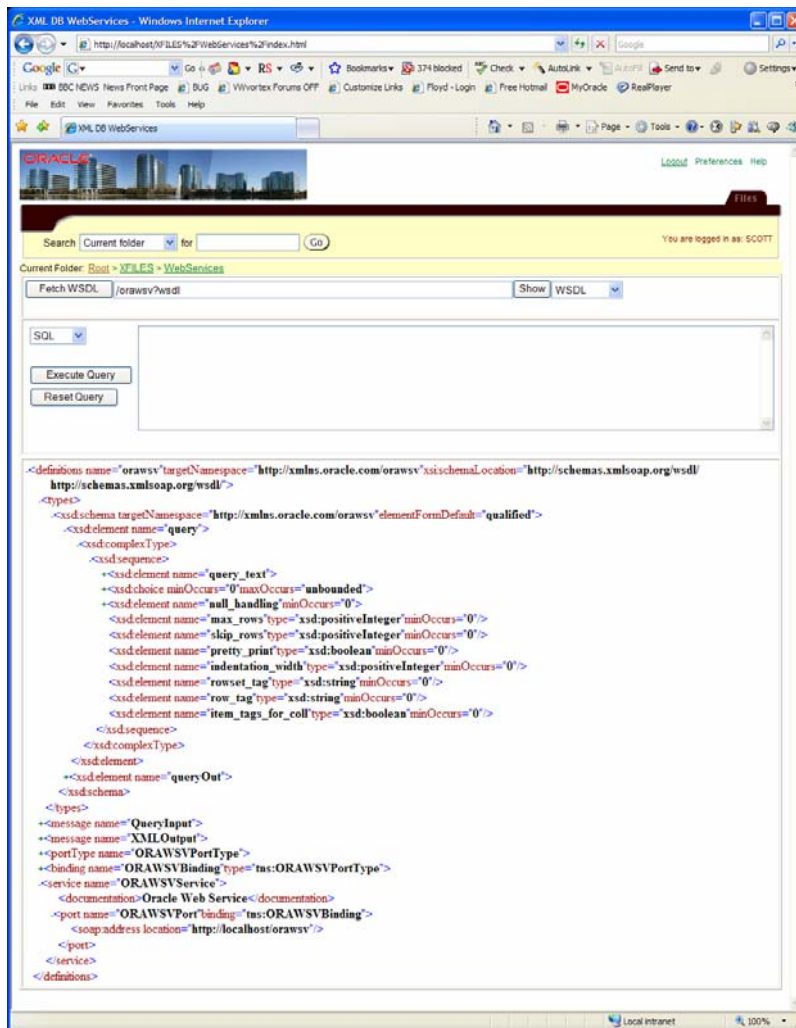
SQL> exit

Disconnected from Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - Beta
With the Partitioning, OLAP, Data Mining and Real Application Testing options
```

The Database Web Services demonstration provides the following functionality

- **View WSDL:** View the WSDL associated with the current service
- **View Request:** View the SOAP request
- **View Response:** View the SOAP response
- **View Data:** View the data in the SOAP response in tabular format
- **View Plan:** This feature is not available in the current implementation of DBNWS

The DBNWS demo can be used with the Dynamic SQL and XQuery service or with any PL/SQL package, procedure or function that the user has the ability to execute. When the demonstration is opened the Dynamic SQL service is selected. Clicking on the show button displays the WSDL document for the Dynamic SQL service.



To execute a SQL query enter the SQL statement into the query box and click the Execute Query button. The current implementation of the Dynamic SQL service only supports SQL Select statements.

The Execute query button invokes the following JavaScript function.

```
function executeQuery(language, query)
{
    xmlHTTP.open ("POST", "/orawsv", false);
    xmlHTTP.setRequestHeader ("SOAPAction", "/orawsv");
    xmlHTTP.setRequestHeader ("Content-Type", "text/xml; charset=utf-8");

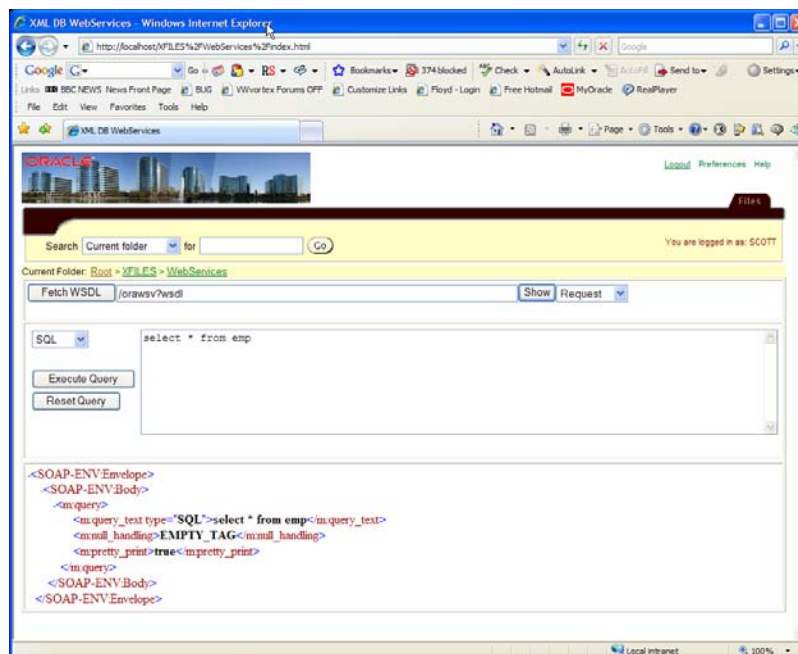
    requestText = "<SOAP-ENV:Envelope xmlns:SOAP-ENV=\"http://schemas.xmlsoap.org/soap/envelope/\"";
    requestText = requestText + "                        xmlns:SOAP-ENC=\"http://schemas.xmlsoap.org/soap/encoding/\"";
    requestText = requestText + "                        xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"";
    requestText = requestText + "                        xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\">";
    requestText = requestText + "    <SOAP-ENV:Body>";
    requestText = requestText + "        <m:query xmlns:m=\"http://xmlns.oracle.com/orawsv\">";
    requestText = requestText + "            <m:query_text type=\"" + language + "\"><![CDATA[" + query + "]]></m:query_text>";
    requestText = requestText + "            <m:null_handling>EMPTY_TAG</m:null_handling>";
    requestText = requestText + "            <m:pretty_print>true</m:pretty_print>";
    requestText = requestText + "        </m:query>";
    requestText = requestText + "    </SOAP-ENV:Body> ";
    requestText = requestText + "</SOAP-ENV:Envelope>";

    soapRequest = new ActiveXObject ("Msxml2.DOMDocument.4.0");

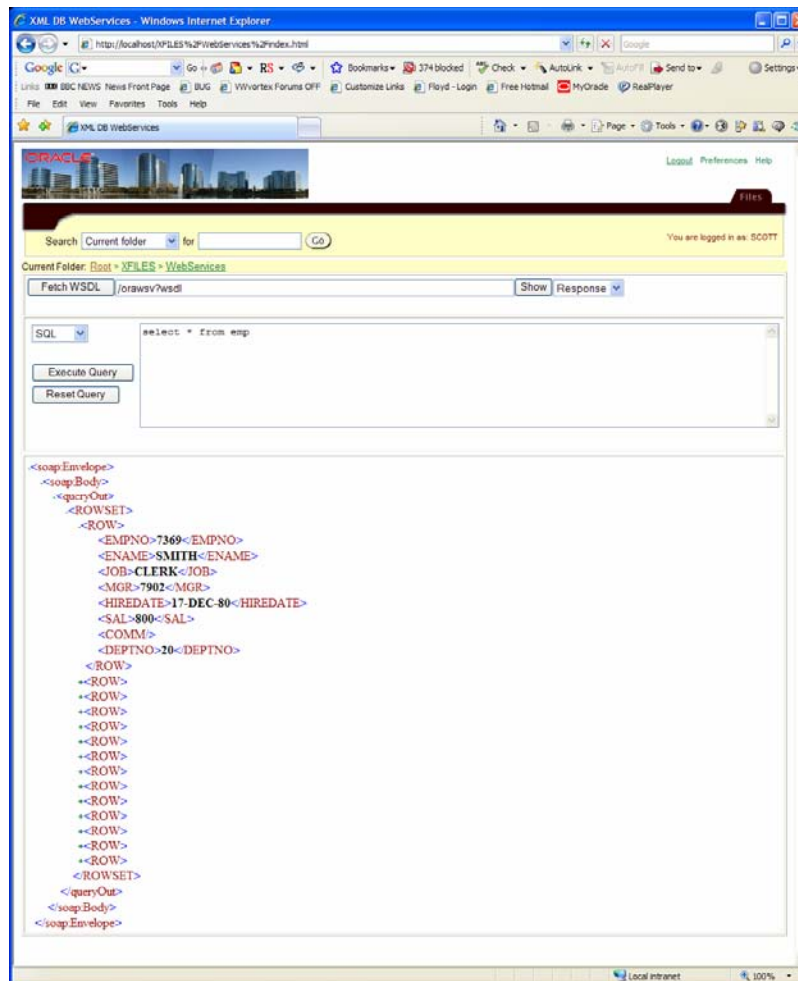
    soapRequest.async="false";
    soapRequest.loadXML(requestText);
    xmlHTTP.send(soapRequest)
    soapResponse = xmlHTTP.responseXML
    self.document.main.display.selectedIndex = 2;
    doTransform(soapResponse, PrettyPrintXSL, resultArea);
}
```

The function takes the current SQL or XQuery statement and embeds it into a SOAP request. The statement is wrapped with a CDATA section to avoid problems with special characters. The SOAP request is submitted to the ORAWSV servlet using the XMLHTTP control. The function then waits for the response from the ORAWXV servlet (note, since the response is not handled asynchronously this is technically not AJAX). When the response is received the function uses a client side XML transformation to render the SOAP response as color coded XML in result window.

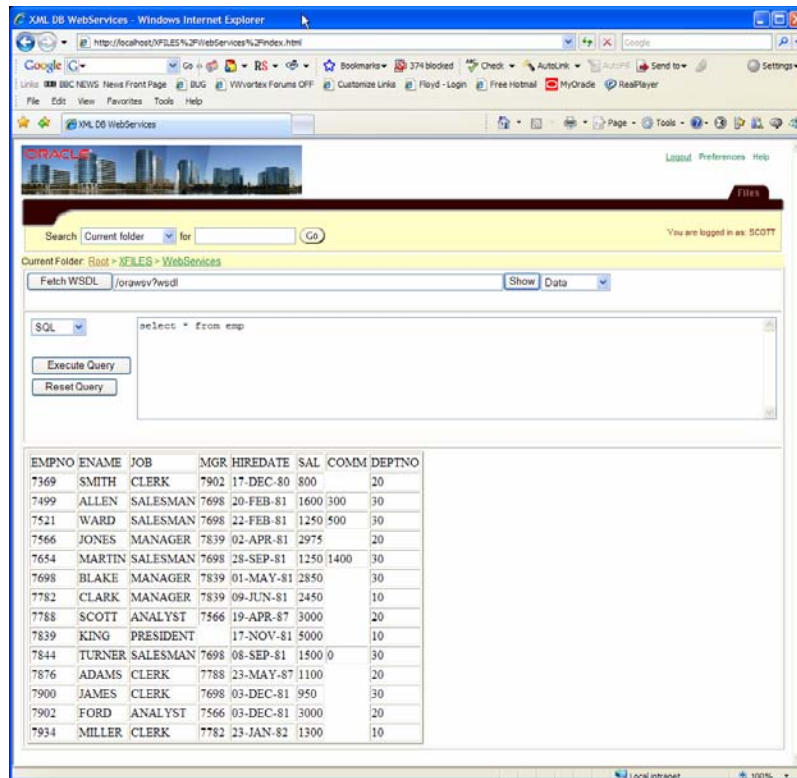
To view the SOAP Request, choose Request from the list of choices in the Display menu and click the Show button:



To view the SOAP Response, choose Response from the list of choices in the Display menu and click the Show button:



To view the data in the SOAP Response in tabular form, choose Data from the list of choices in the Display menu and click the Show button. This will use a generic XSL stylesheet to perform a client side XML transformation of the Response document.



To invoke a PL/SQL procedure package or function simply enter the appropriate URL in the Fetch WSDL field and click the Fetch WSDL button. The naming convention the URL to access PL/SQL packages, functions and procedures can be one of :

/orawsv/SCHEMA_NAME/PACKAGE_NAME | FUNTION_NAME | PROCEDURE_NAME?wsdl

or

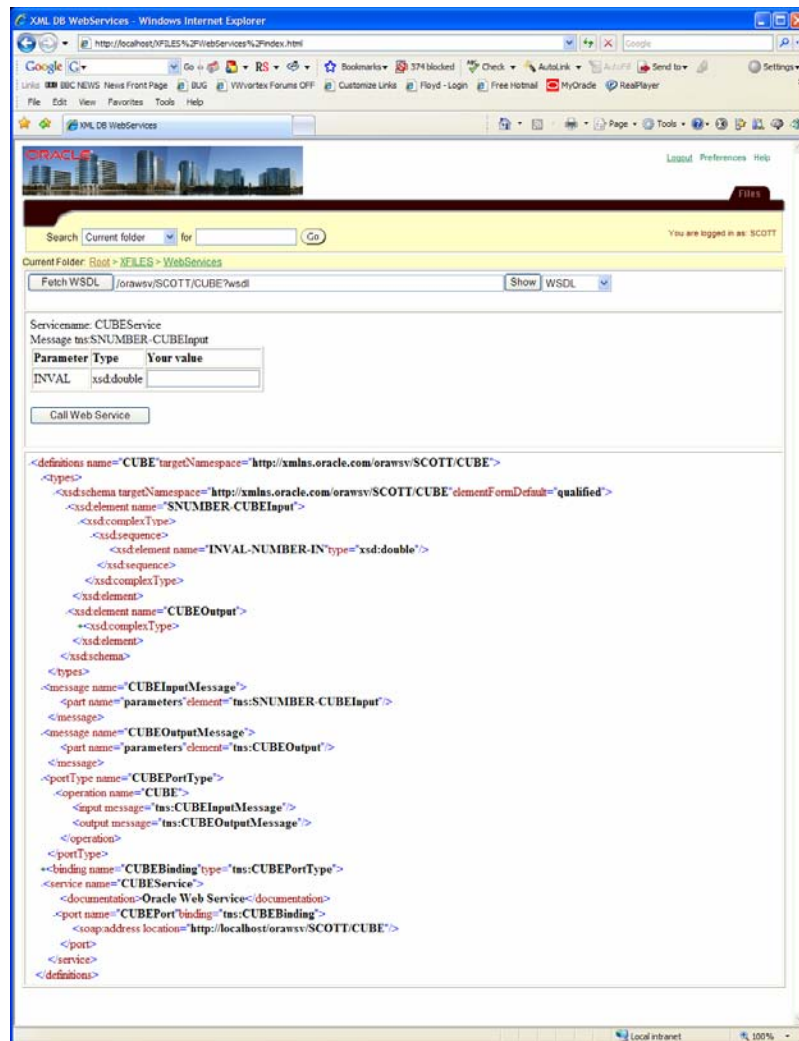
/orawsv/SCHEMA_NAME/PACKAGE_NAME/PROCEDURE_NAME or FUNCTION_NAME?wsdl

For instance, assuming user SCOTT owns a function called CUBE with the following definition

```
SQL> create or replace function cube ( inval number )
2   return number
3   as
4   begin
5       return inval * inval * inval;
6   end;
7   /
```

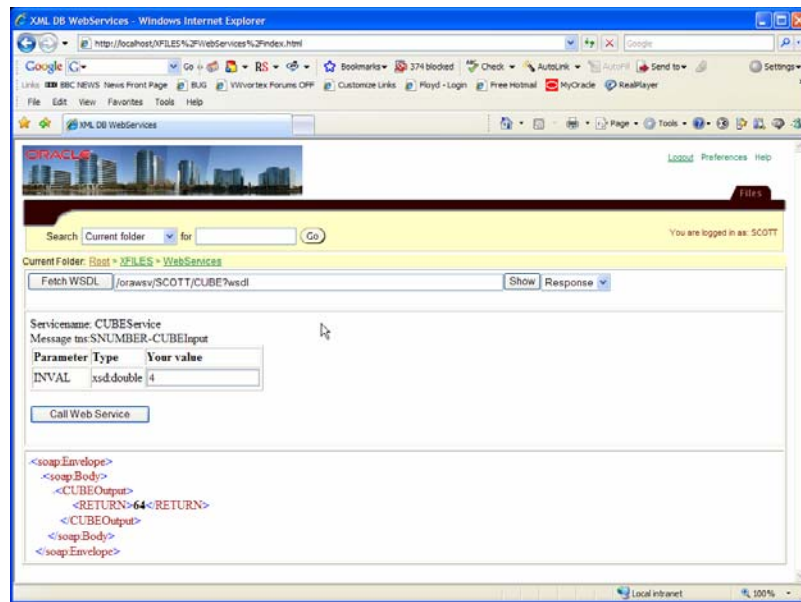
The URL to obtain the WSDL for this function would be **/ORASV/SCOTT/CUBE?wsdl**

To view the WSDL for the function enter the URL and click the Fetch WSDL button, then click the SHOW button to display the WSDL for the function.

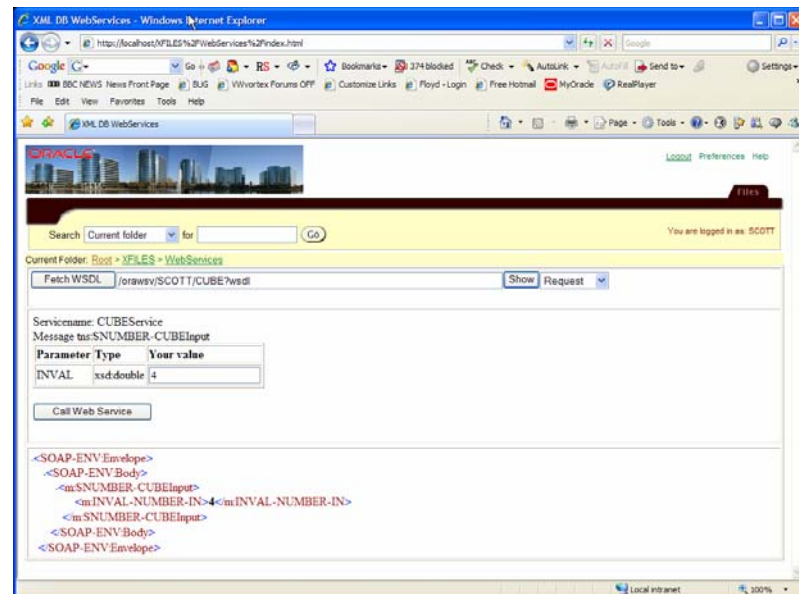


As can be seen the ORAWSV servlet has automatically generated a WSDL for the PL/SQL function CUBE. The DBNWS demonstration automatically generates a simple HTML input form for the SOAP service by applying an XSL stylesheet to the WSDL document. The service is invoked by entering a value into the INVAL input box and clicking the Call Web Service button.

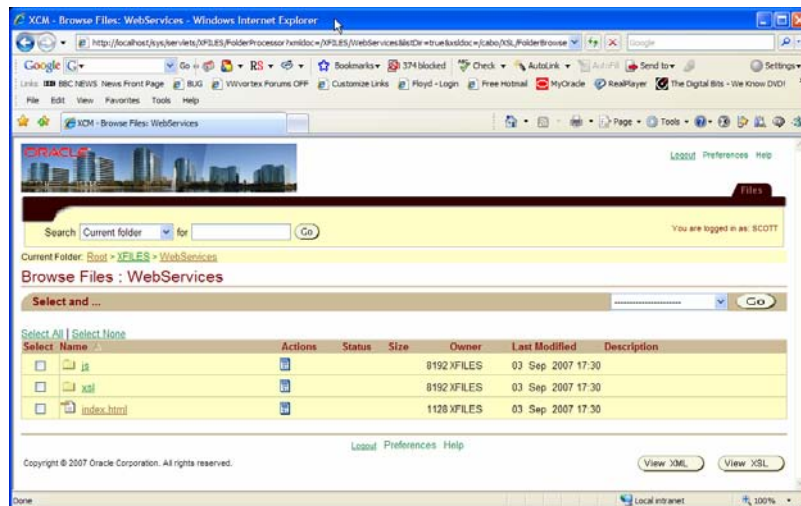
The DBNWS demonstration calls the Web Service using a simple JavaScript function that generates a SOAP request using the information in the WSDL, and then submitting the SOAP request to the ORAWSV servlet. Once the request has been processed it displays the SOAP response.



To view the SOAP Request, choose Request from the list of choices in the Display menu and click the Show button:



All of the source code for the DBNWS demonstration can be found by using the Folder Browser to navigate to the /XFILES/WebServices folder.



File index.html is a place holder used to launch the DBNWS demonstration. The actual body of the DBNWS demonstration is generated dynamically using the webServices.xsl stylesheet once the HTML page has been loaded.

The js folder contains the source code for the JavaScript functions used by the DBNWS demonstration:

- **webServices.js** is a JavaScript library that contains the functions used to create SOAP Requests, process WSDL documents, and process SOAP Responses.

The xsl folder contains the XSL stylesheets that are used to dynamically render the DBNWS demonstration.

- **webServices.xsl** generates the HTML framework for DBNWS demonstration.
- **ORAWSVInput.xsl** generates the custom HTML input form used with the Dynamic SQL service.
- **WSDL2Input.xsl** transforms a WSDL document into a generic HTML input form.
- **Response2Table.xsl** generates a tabular view of a SOAP Response document.

The DBNWS demonstration also uses the **XMLPrettyPrint.xsl** stylesheet to render the WSDL, SOAP Request and SOAP Response documents as color-coded XML. This stylesheet and its related CSS can be found under the folder /XFILES/common.

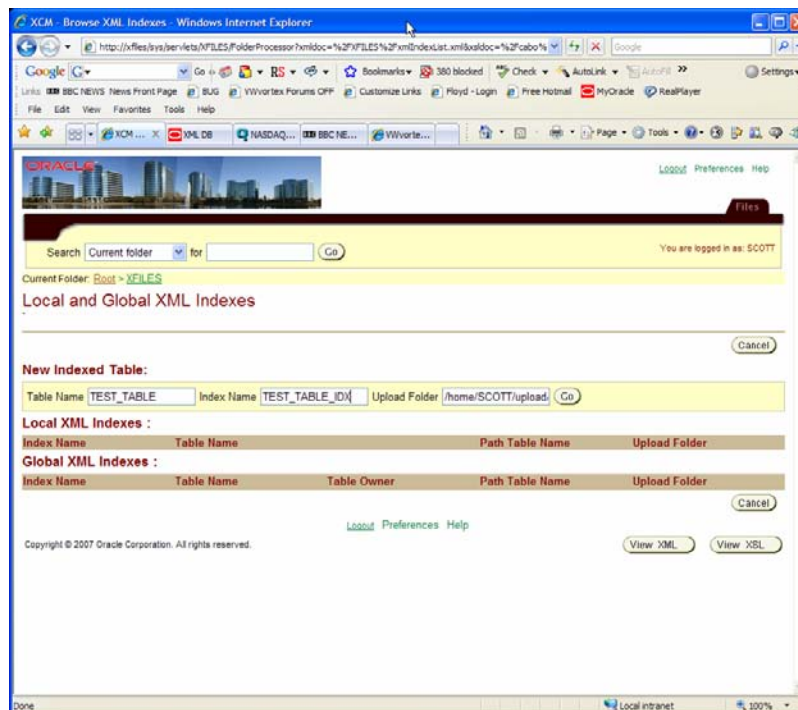
XML Searching

The X-Files application demonstrates 3 kinds of XML DB related search capabilities. These are XML DB Repository driven searching, XML Index driven searching, and XML Schema driven search. All of these search capabilities are accessed from the search dialog that is included in all of the X-Files dialogs. All of the search capabilities provided by the X-Files application are based on AJAX. Users must have the ability to use Database Native Web Services in order to use the search capabilities of the X-Files application.

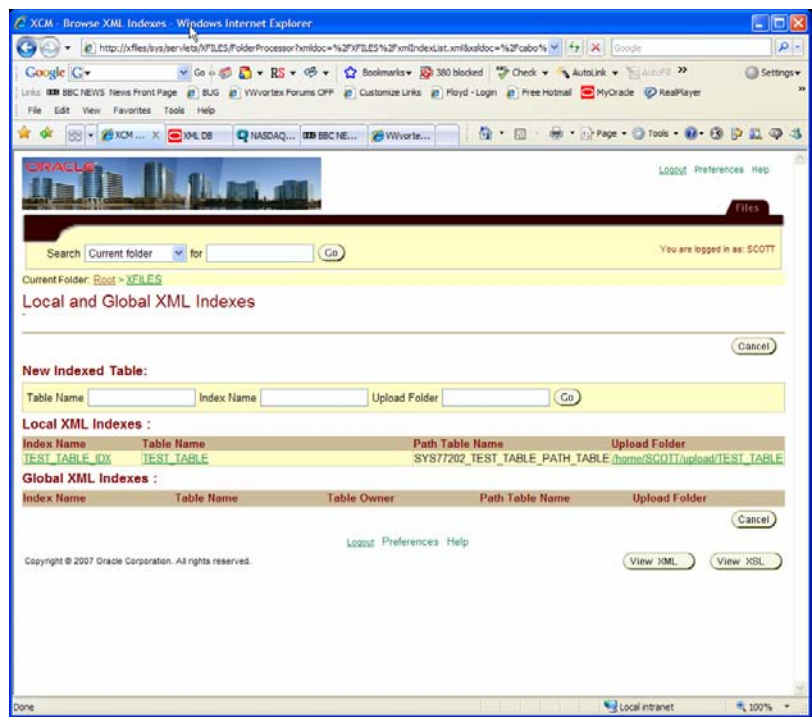
XML Index-driven search

The X-Files application also includes an intelligent search capability for XML content which is indexed using The Oracle XML DB XML Index. XML Index is a new feature of Oracle XML DB in Oracle Database 11g. XML Index driven searching provides a data-guide that enables intelligent searching of XML content when there is no advanced knowledge about the structure of the documents being processed. The data-guide is generated dynamically from meta-data maintained by the XML Index.

To access the XML Index Data-Guide select the XML Indexes option in the Search dialog and click GO. The application will respond with a list of the XML Indexes that are accessible by the current user. The XML Index Data-Guide is an AJAX based application; in order to use the Data-Guide a user must have been granted the ability to access the ORAWSV service.

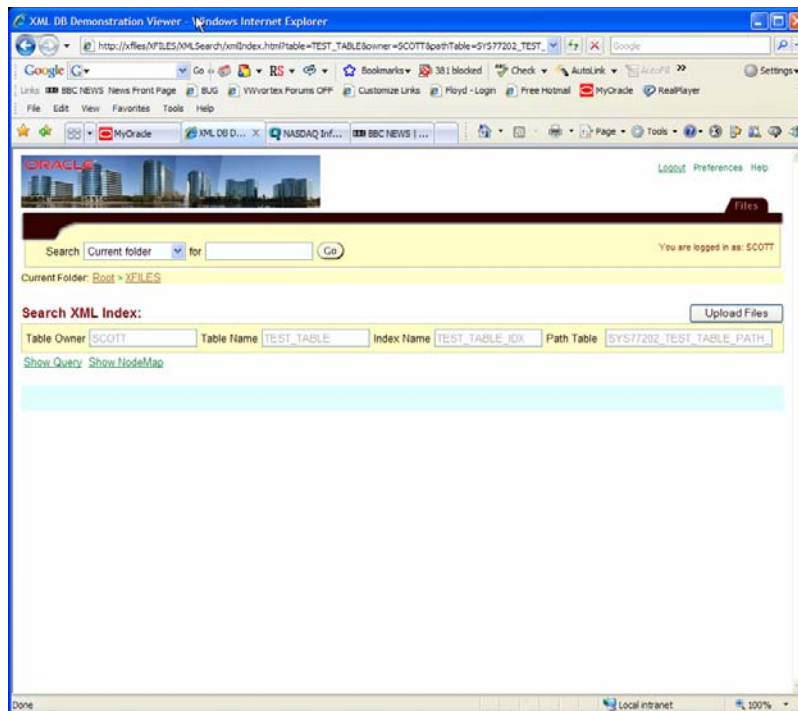


If the current user does not have access to any XML Indexes a new XML Indexed, binary XML Type table can be created by entering a Table Name and Index Name in the New Indexed Table section and clicking GO.







When an XMLType table is created from the XML Index Data-Guide, the X-Files application also creates a special folder that can be used to upload content into the table. When a document is stored in this folder the content is loaded directly into the table and the resource is discarded. This functionality is made possible by Oracle XML DB repository events. A resource configuration document is attached to the folder. The resource configuration document defines a postLinkIn event that call a procedure that captures the content of the newly created resource, inserts it into the target table, and then deletes the resource. This allows data can be inserted into the table by copying XML files into the folder.

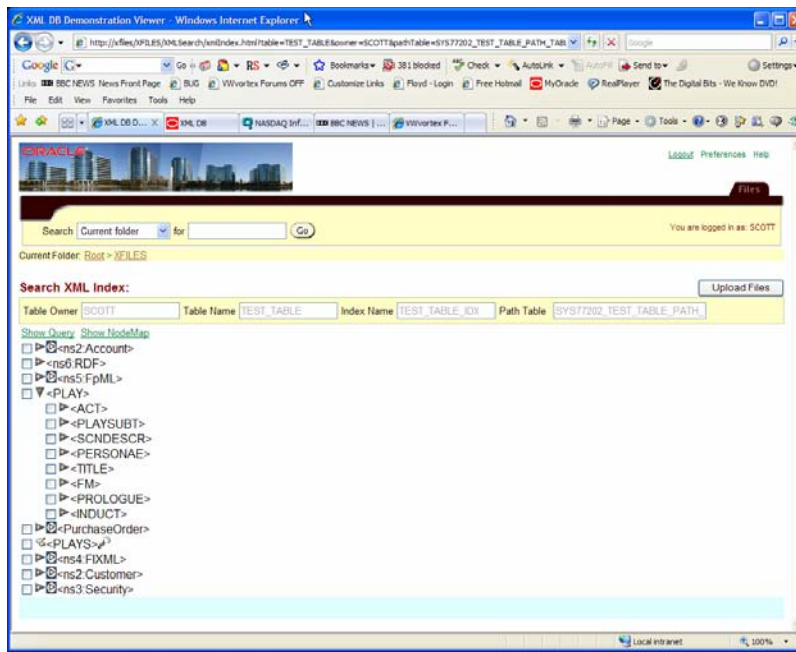
Clicking one of the available index or table names opens the XML Index Data-Guide.



If the table contains no data the XML Index Data-Guide will be empty when the form is opened. If the selected table is associated with an upload folder, data can be loaded into the table by clicking the Upload Files button. This will open a new window containing the Upload Files dialog, which can be used to load up to ten documents directly into the selected table. Note when the upload is complete the selected files will not appear in the Folder Browser since the postLinkIn event discards the resources.

After content has been loaded into the table the XML Index data-guide will show the possible root elements for the documents that have been loaded. It will also show the possible list of attributes and child elements for each root node. This list is obtained by querying the index metadata as distinct from querying the documents. This is a very efficient way of querying the structure of the XML documents.

The XML Index data-guide is presented in the form a simple tree control. When the data-guide is first opened all the known root elements are displayed along with their attributes and their immediate descendants. With-in the data-guide the list of descendants for a given element can be toggled by clicking the  (open) and  (close) icons. The list of attributes for a given element can be toggled by clicking the  (show) and  (hide) icons.

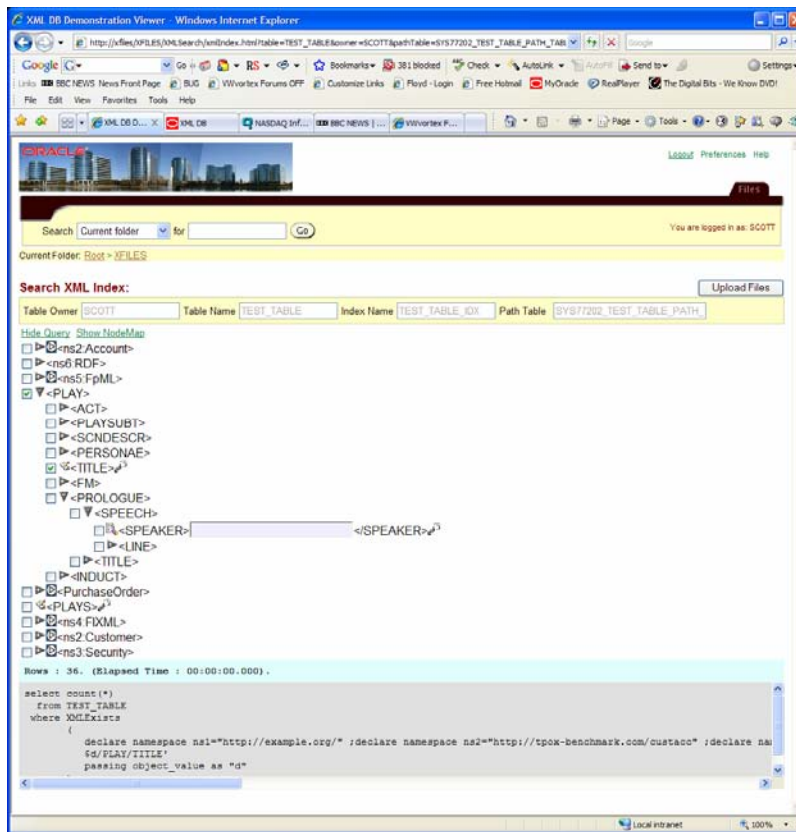


The Data-Guide works by creating a MAP of the possible nodes. The first time a branch is opened the PL/SQL procedure XDBPM.XMLINDEX_SEARCH_11100. getChildNodeMap() is invoked using Database Native Web Services. This procedure gets the list of possible Elements and Attributes for the selected node using meta-data obtained from the XML index.

The Data-Guide can be used to construct an XQuery statement on the fly. An XQuery expression is created by selected one or more the elements or attributes shown in the Data-Guide. To include an element of attribute in the XQuery expression simply place a tick in the check-box next to the required node. Once one more nodes have been selected the Data-Guide will use DBNWS to count the number of documents in the table that contain the selected nodes. To remove a node or node-tree from the query deselect the corresponding check box. The following example generates the XQuery expression

```
select count(*)
  from TEST TABLE
 where XMLExists
    (
      declare namespace ns1="http://example.org/" ;
      declare namespace ns2="http://tpox-benchmark.com/custacc" ;
      declare namespace ns3="http://tpox-benchmark.com/security" ;
      declare namespace ns4="http://www.fixprotocol.org/FIXML-4-4" ;
      declare namespace ns5="http://www.fpml.org/2005/FpML-4-2" ;
      declare namespace ns6="http://www.w3.org/1999/02/22-rdf-syntax-ns#" ;
      declare namespace ns7="http://www.w3.org/2001/XMLSchema-instance" ;
      $d/PLAY/TITLE'
      passing object value as "d"
```

The XQuery can be viewed by clicking on the Show Query link.

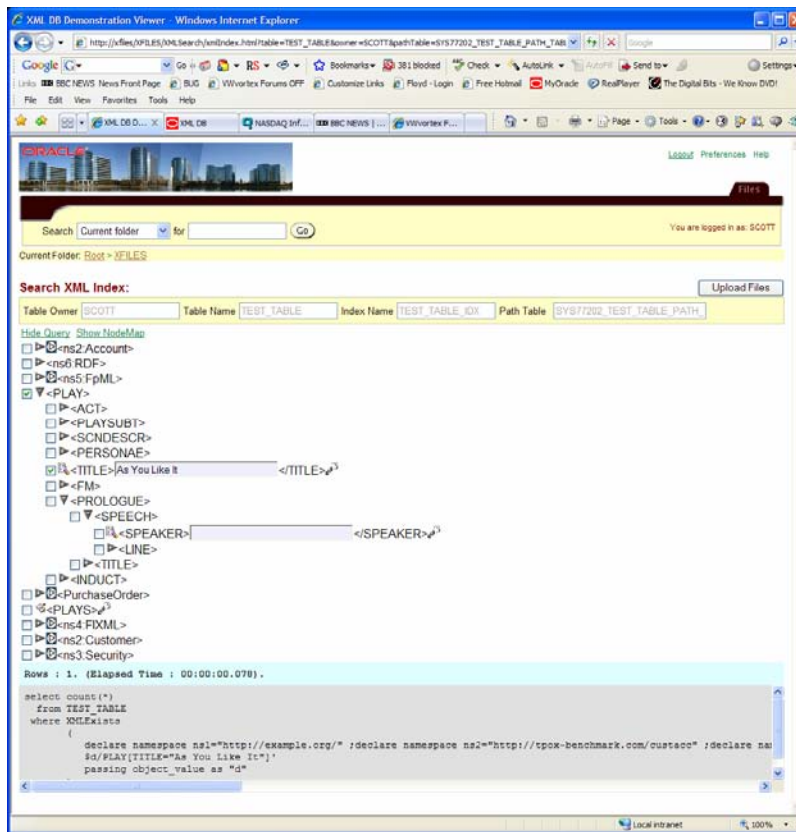


If the selected element is a leaf node then the open and close icons are replaced with the (Enter Text) icon the first time the open icon is clicked. Click the Enter Text icon to add an input box which can be used to enter a predicate value for the node. Click the (Erase Text) icon to clear the contents of the input box click and remove the predicate from the XQuery

In the above example the XQuery counts the number of documents that have a root element called PLAY with a child element called TITLE. As can be seen there are matching 36 documents. If the value “As You Like It” is entered a predicate for the TITLE element, then the new XQuery is:

```
select count(*)
from TEST TABLE
where XMLExists
(
  declare namespace ns1="http://example.org/" ;
  declare namespace ns2="http://tpox-benchmark.com/custacc" ;
  declare namespace ns3="http://tpox-benchmark.com/security" ;
  declare namespace ns4="http://www.fixprotocol.org/FIXML-4-4" ;
  declare namespace ns5="http://www.fpml.org/2005/FpML-4-2" ;
  declare namespace ns6="http://www.w3.org/1999/02/22-rdf-syntax-ns#" ;
  declare namespace ns7="http://www.w3.org/2001/XMLSchema-instance" ;
  $d/PLAY[TITLE="As You Like It"]'
  passing object value as "d"
```

and the number of matching documents is 1.





The XML Index Data-Guide is driven by a Node-Map. The Node-Map is an XML document that allows the browser to track the current state of the tree control. When a new branch is expanded for the first time, the Node-Map is updated with the information obtained from the XML Index. When a branch of the Data-Guide is opened or collapsed, the Node-Map is updated to reflect the desired state of the Tree-control. The updated Node-Map is then used to regenerate the Tree control via an XSLT transformation.

The Node-Map is also used to generate the XQuery that is used to count the number of matching rows. To view the Node-Map click the Show NodeMap link. The following is a fragment taken from the Node-Map associated with the previous query.

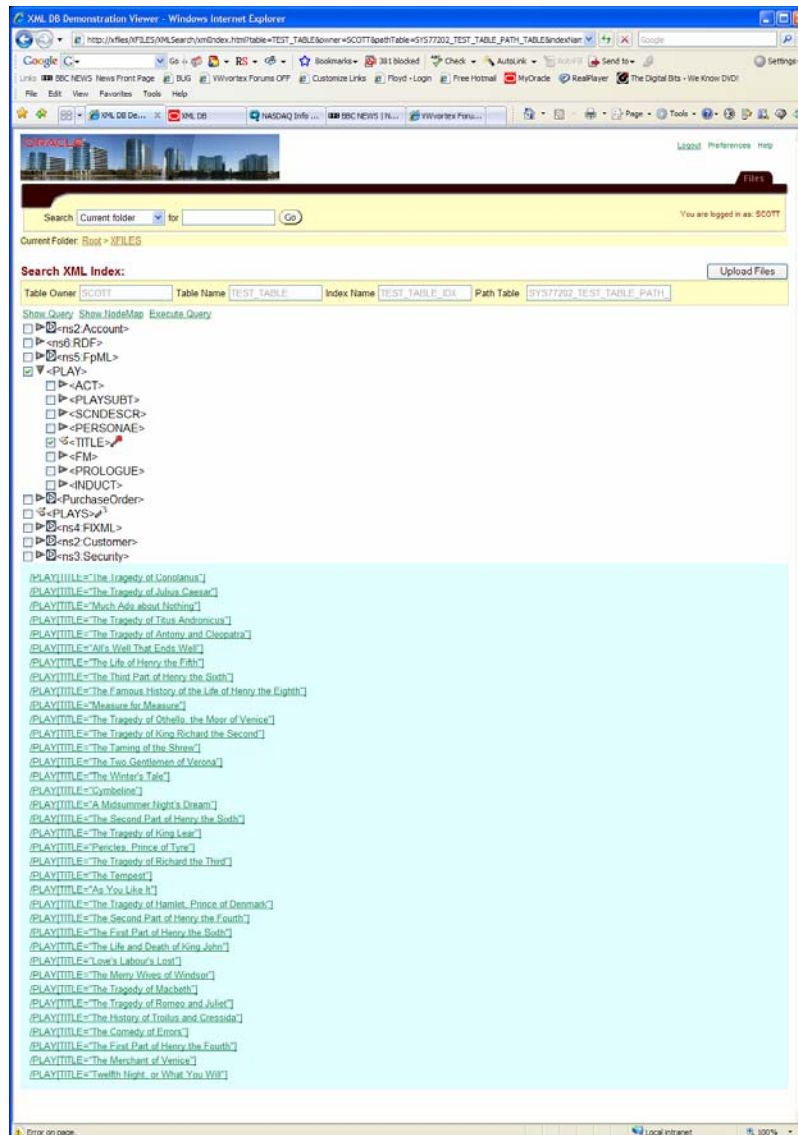
```

<Element ID="1878"isSelected="true"valueVisible="hidden"attributesVisible="block"childrenVisible="block"treeDepth="
  <Name>PLAY</Name>
  <Element>
    <Element ID="2322">ACT</Element>
    <Element ID="5FFF">PLAYSUBT</Element>
    <Element ID="41A4">SCNDESCR</Element>
    <Element ID="19A4">PERSONAE</Element>
    <Element ID="7049"isSelected="true"value="As You Like It"valueVisible="visible"attributesVisible="block"childrenVisible="
    <Name>TITLE</Name>
    </Element>
    <Element ID="6CA7">FM</Element>
    <Element ID="380F"isSelected="false"value=""valueVisible="hidden"attributesVisible="block"childrenVisible="block">
    <Name>PROLOGUE</Name>
    <Elements>
      <Element ID="7685"isSelected="false"value=""valueVisible="hidden"attributesVisible="block"childrenVisible="block">
      <Name>SPEECH</Name>
      <Elements>
        <Element ID="3F13"isSelected="false"value=""valueVisible="visible"attributesVisible="block"childrenVisible="blo
        <Name>SPEAKER</Name>
        </Element>
        <Element ID="440C">LINE</Element>
        </Elements>
      </Element>
      <Element ID="24B6">TITLE</Element>
      </Element>
    </Elements>
    <Element ID="3A5D">INDUCT</Element>
    </Elements>
  </Element>

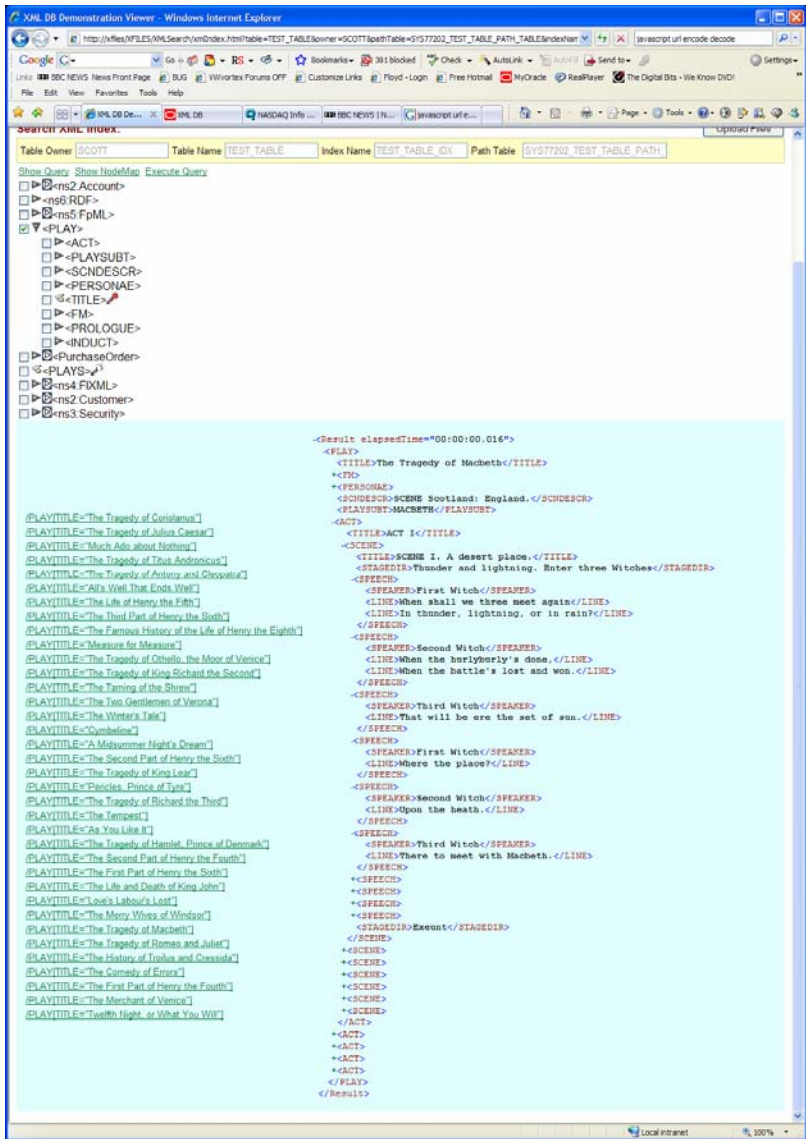
```

To list the documents which match the current query click the Execute Query link. The Execute Query link only becomes available once one of the elements or attributes in the Data-Guide has been marked as a unique identifier for the current set of documents. Only attributes or leaf-level elements can be marked as a unique identifier. To mark an element or attribute as a unique identifier click the  (possible identifier) icon. Once a node has been selected it will be identified by the  (unique identifier) icon. The current implementation does not support using combinations of elements or attributes for this purpose.

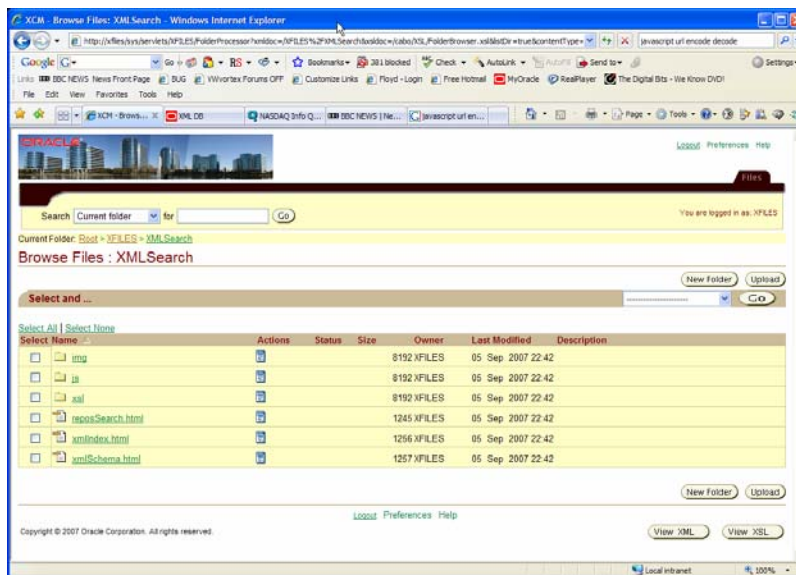
Once a unique identifier has been chosen, click the Execute Query link to execute the query. The matching documents are returned as a set of XQuery Path Expressions.



To view the complete content of one of the matching documents click on the associated Path Expressions



The source code for the XML Index Data-Guide is published under /XFILES/XMLSearch:



File `xmlIndex.html` is a place holder used to launch the XML Index Data-Guide. The actual body of the Data-Guide is generated dynamically using the `xmlIndex.xsl` stylesheet once the HTML page has been loaded.

The `js` folder contains the source code for the JavaScript libraries used by the Data-Guide:

- **xmlIndexSearch.js** is a JavaScript library that contains the functions that create the SOAP Requests to create and expand the node tree. These functions use Database Native Web Services to invoke the `getRootNodeMap()` and `getChildNodeMap()` methods of PL/SQL package `XMLINDEX_SEARCH_11100`, found in the XDBPM schema.
- **xmlSearch.js** is a JavaScript library that contains the functions that manage the Data-Guide tree-control, count and display the matching documents. These functions use Database Native Web Services to invoke the `executeRowCount()`, `executeQuery()` and `fetchDocument()` methods of PL/SQL package `XMLSEARCH_UTILITIES`, found in the XDBPM schema.

The `xsl` folder contains the XSL stylesheets that are used to dynamically render the XML Index data guide

- **xmlIndex.xsl** generates the HTML framework for Data-Guide
- **searchTreeView.xl** generates the tree-control

The Data-Guide also uses the **XMLPrettyPrint.xsl** stylesheet to render the WSDL, SOAP Request and SOAP Response documents as color-coded XML. This stylesheet and its related CSS can be found under the folder /XFILES/common.

XML DB Repository full-text search

Oracle XML DB includes support for creating a full-text index on the Oracle XML DB repository. This allows the power of full-text searching to be combined with metadata and path based search. The index is created using methods provided by package DBMS_XDBT.

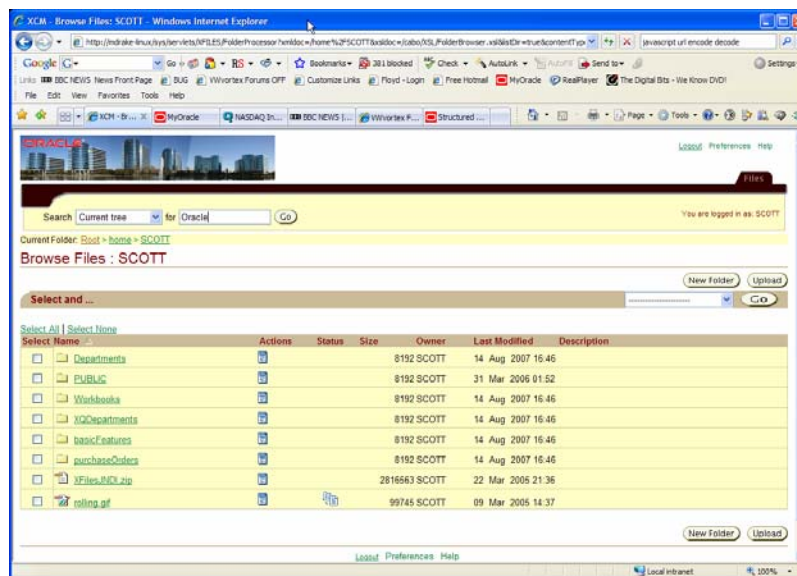
The current version of the X-Files application supports 3 types of full-text search. These are

1. **Current Folder:** Full-text search restricted to documents contained in the current folder.
2. **Current Tree:** Full-text search restricted to documents in the current folder tree.
3. **Repository:** Unrestricted search of the Oracle XML DB repository.

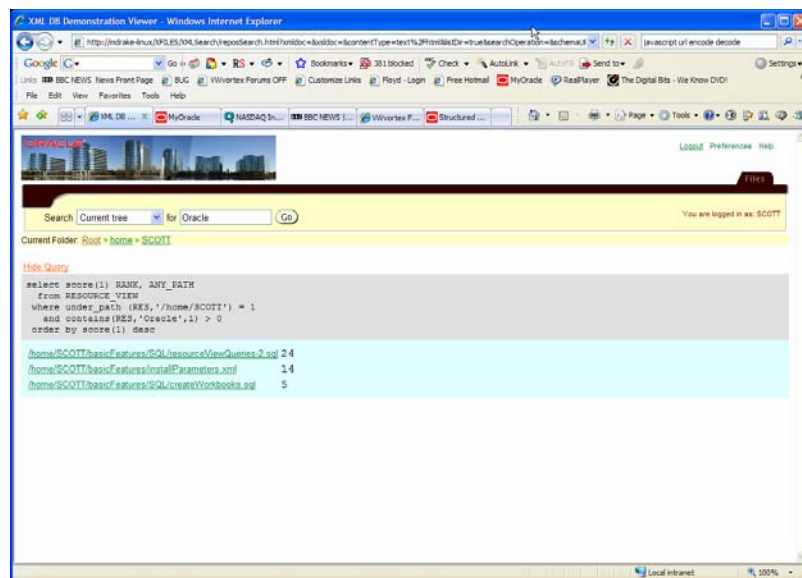
In order for a document to appear in the result set the document must be secured by an ACL that grants the user a minimum of read-only access to the content. Additionally the document will only appear in the result set of user has the permissions required to navigate to the folder containing the document.

To perform a full-text repository search select the type of search that is required and enter the search criteria into the search dialog, and click GO. The Full-Text search functionality makes use of Dynamic SQL web service, so a user must be granted the ability to use the ORAWSV servlet in order to perform a full text search of the repository.

The following example shows searching all of the folders under '/home/SCOTT' for the documents containing the word Oracle:



The search results show the path to the document that matched the search term and the score generated by the Oracle Text index.



To view any of the documents in the result set simply click on the appropriate link.

XMLSchema-driven searching

The X-Files application also incorporates a prototype of an XMLSchema-based Data-Guide. The functionality of this feature is similar to the XML Index Data-Guide except it uses the XMLSchema as a source of meta-data. However at this time the XMLSchema Data Guide is a work in progress and is not fully functional.



Oracle Database 11g Oracle XML DB

October 2007

Author: Mark D Drake

Contributing Authors: Oracle XML DB Development Team

Oracle Corporation

World Headquarters

500 Oracle Parkway

Redwood Shores, CA 94065

U.S.A.

Worldwide Inquiries:

Phone: +1.650.506.7000

Fax: +1.650.506.7200

oracle.com

Copyright © 2007, Oracle. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice.

This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission. Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.