

**Lab BL02**  
**Integrating DB2 UDB for**  
**iSeries and XML**



**ITSO iSeries Technical Forum 2003**



---

## Contents

<b>Lab 1. Verifying the environment to work</b> .....	1
Task 1: Configuring a PC connection to the iSeries server .....	2
Task 2: Verifying the Coded Character Set ID .....	4
Task 3: Mapping the ITSO folder on the IFS .....	6
Task 4: Creating a directory to store your files .....	10
Task 5: Getting started with WebSphere Studio Application Developer .....	11
<b>Lab 2. Working with XML Columns</b> .....	17
Task 1: Importing source code to be used in the PC side .....	19
Task 2: Creating your schema and database tables .....	23
Task 3: Reviewing the information in an XML document .....	27
Task 4: Exporting XML documents to the IFS .....	31
Task 5: Enabling an XML Column .....	34
Task 6: Storing your document into the XML Column .....	37
Task 7: Running queries against XML Columns and a side table .....	39
<b>Lab 3. Decomposing XML data using an XML Collection</b> .....	43
Task 1: Preparing the WebSphere Studio Application Developer Workbench ..	44
Task 2: Reviewing the information in the XML file .....	45
Task 3: Reviewing the associated DTD file .....	49
Task 4: Reviewing the associated DAD file .....	51
Task 5: Working with the three related files .....	53
Task 6: Decomposing the XML file .....	54
Task 7: Running queries against XML Columns and side table (optional task) ..	56
<b>Lab 4. Composing XML documents from existing data</b> .....	57
Task 1: Preparing the WebSphere Studio Application Developer Workbench ..	58
Task 2: Reviewing the information in DTD file .....	58
Task 3: Reviewing the DAD file .....	59
Task 4: Transferring the files before composition .....	62
Task 5: Composing XML documents using the SQL Mapping DAD file .....	63
Task 6: Viewing the XML file .....	65
<b>Lab 5. Additional exercises to work with XML documents</b> .....	67
Task 1: Updating elements or data attributes in an XML Column .....	67
Task 2: Using SQL overrides during XML Composition .....	71
Task 3: Validating composed XML documents using a DTD .....	73
Task 4: Using the RDB node DAD file for document composition .....	78
Task 5: Cleaning the server of objects and files created during lab .....	79



---

## Lab 1. Verifying the environment to work

eXtensible Markup Language (XML) is the evolution of HTML like a mechanism to share information between computers in a simple and documented form. It is the standard for data interchange in e-business. When you manage information in XML format you know exactly with what type of information is working.

The most important part of XML is that you can publish information on the Web very easily. You can:

- Locate the information in the tables that your applications already are using
- Design a Web page for your data
- Transform the data using the DB2 attributes and send them to the Web

### Objectives

You will work with five different labs that are related to the requirements of your business – deploying information to the Web. When you complete these five labs, you will understand how easy is to deploy information to the Web when XML is used. You will also understand how easy it is to put the information that you are receiving into your database.

There are many different ways (and tools) available to communicate your information via the Web. XML is a product completely integrated with your database tables. Your information in XML format will be tied with the data that your applications are using now.

You will work with the two different forms to store information. And, you will transform data in your tables to XML format. Among the activities that you will perform are:

- Store an XML document into your database table
- Decompose an XML document to update your database tables
- Compose an XML document to publish information from your tables

### Lab prerequisites

Before you begin this lab, be sure the following prerequisites are available:

- An IBM @server iSeries or AS/400e server with OS/400 V5R1 (or higher) with:
  - 5722-SS1 Option 12: Host Servers
  - 5722-SS1 Option 13: OS/400 - System Openness Includes
  - 5722-SS1 Option 30: OS/400 - QShell Interpreter
  - 5722-SS1 Option 34: OS/400 - Digital Certificate Manager
  - 5722-SS1 Option 39: OS/400 - International Components for Unicode
  - 5722-DE1 DB2 UDB Extenders
  - 5722-DE1 Option 1: DB2 UDB Text Extender
  - 5722-DE1 Option 2: DB2 UDB XML Extender
  - 5722-TC1 TCP/IP Connectivity Utilities
  - 5722-WDS WebSphere Studio Application Developer
- A PC with Client Access Express V5R1M0 with the latest Service Pack applied
- A Windows IP address HOSTS file containing an I400WS entry that point to your iSeries server

**Note:** You need to install the latest PTFs for the products on your iSeries server.

## Time required

The time required to efficiently complete this lab is 20 minutes.

## Required information for working with the lab

In many steps of these labs, you need to provide information about your environment of work. You are asked for a group number. Every time you see a double character (XX), you need to replace it with your group number. For example, DBXMLXX is the user profile for group "xx", your group number.

In this document, we use the term "schema" as a qualified or unqualified name that provides a logical grouping for SQL objects. Starting with V5R1, the term "schema" is used in the same sense as the term "collection". This is an OS/400 library created with automatic DB journaling enabled and local DB catalog views.

A user profile, DBXMLXX, has been created for you, with the password DB2UDB.

An IFS structure is already created, but you need to add a directory to store the XML documents created during the labs.

---

## Task 1: Configuring a PC connection to the iSeries server

Before you begin, you must configure a connection between your PC and the iSeries. This connection will be used to transfer files and execute code in the Server.

- \_\_\_ 1. Double-click the **iSeries Navigator** icon on your Windows Desktop.  
If this is your first time running iSeries Navigator on your PC, a message window appears indicating that there are no connections to the server yet. Click **Yes**.
- \_\_\_ 2. The Add Connection - Welcome window (Figure 1) appears. Complete the following steps:
  - a. In the Server input field, type I400WS.
  - b. Move your cursor to the Description field and type iSeries Version 5 or any description you prefer.
  - c. Click **Next**.

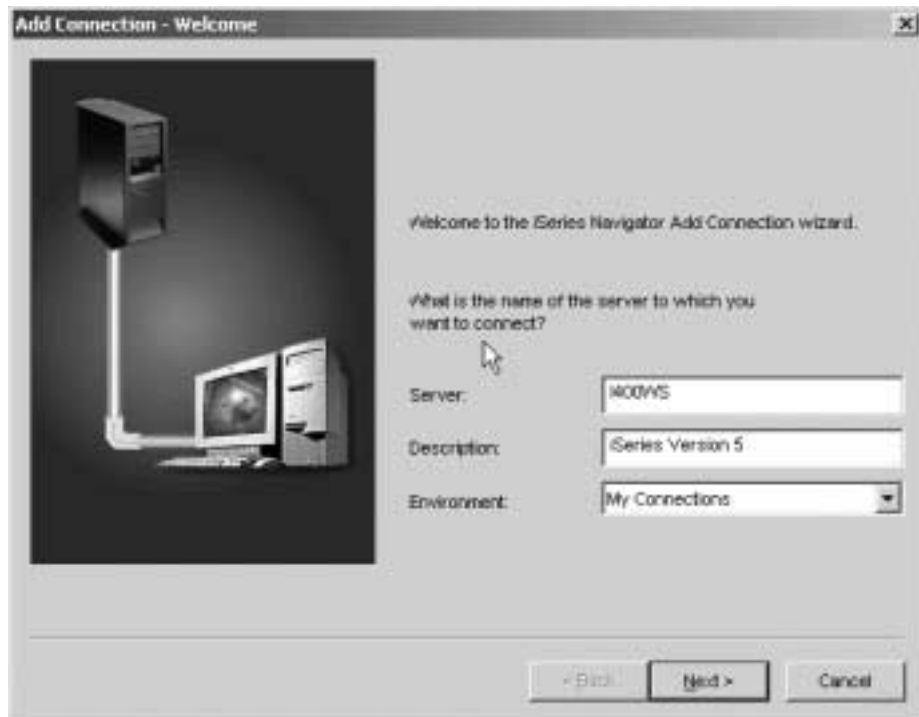


Figure 1. Add connection - Welcome window

- \_\_\_ 3. The Add Connection - Signon Information window appears. Select the **Use default User ID, prompt as needed** radio button and type your team profile (DBXMLXX) in the input field shown in (XX is your team number).

**Note**

For the purposes of this lab, the server name I400WS is used for the primary connection. If you decide to use different server names, you must make appropriate changes throughout this and subsequent lab documents.

- \_\_\_ 4. Click **Next**. The Verify Connection window appears. Click **Verify Connection** to verify that the Host Servers functions on the server are up and running. If you receive an error message in this step, notify your lab supervisor or your system administrator (if you are in your office). When the verification is successful, click **OK** and then click **Finish**.
- \_\_\_ 5. From the iSeries Navigator window, click the plus sign (+) in front of the **I400WS** connection. A new window appears prompting you for your user name and password (see Figure 2).

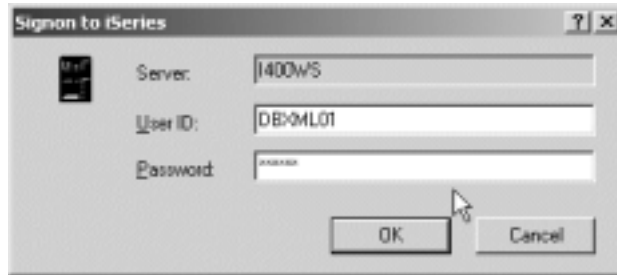


Figure 2. Signon to the iSeries server

Because this is your first time signing on to the iSeries server, the client checks the server for the available functions to be displayed in the left panel.

## Task 2: Verifying the Coded Character Set ID

When you need to transfer files from the PC to the iSeries, be sure that you are using the correct conversion. For this reason, you need to verify your user profile and make changes if necessary:

- \_\_\_ 1. Click the plus sign (+) in front of the **Users and Groups** icon to expand it. See Figure 3.

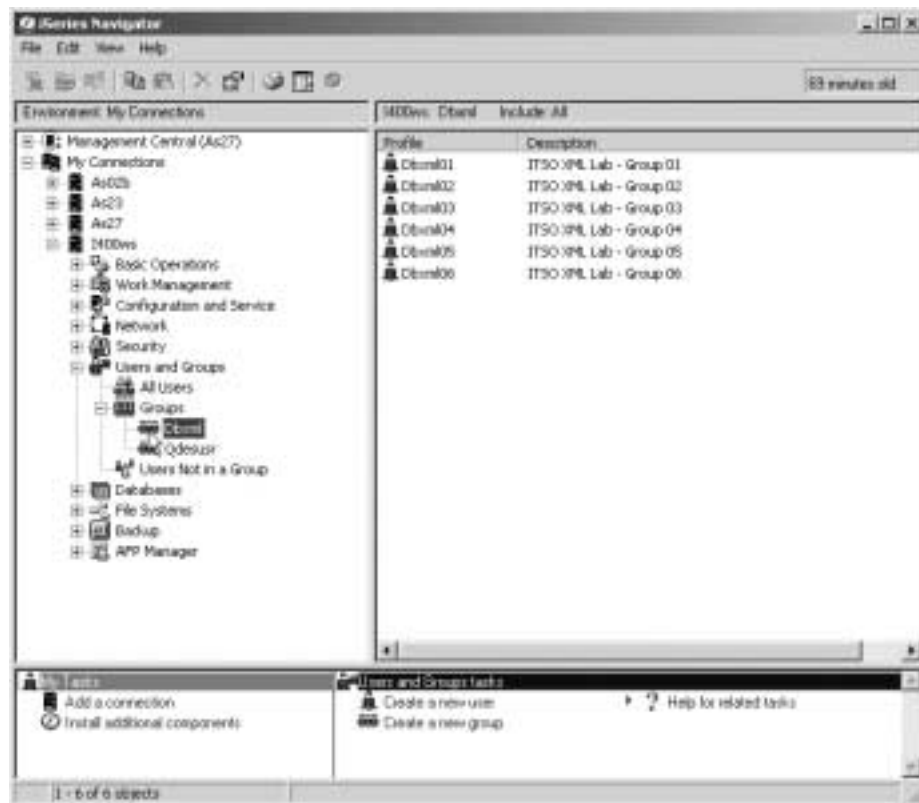


Figure 3. Main Operations Navigator window



- \_\_\_ 2. Click the plus sign (+) in front of the **Groups** icon. A list of existing group profiles displays. Locate the group **DBXML** (note that this is the main user profile; there is no group number) and click it.
- \_\_\_ 3. The right pane updates with the list of users in this group. Double-click your user profile, and a Properties window (Figure 4) appears.



Figure 4. Properties of your user profile

- \_\_\_ 4. Click the **Jobs** button. The Jobs window opens. In the Jobs window, select the **International** tab. Compare the CCSID values of your user profile with the values listed in the Figure 5.

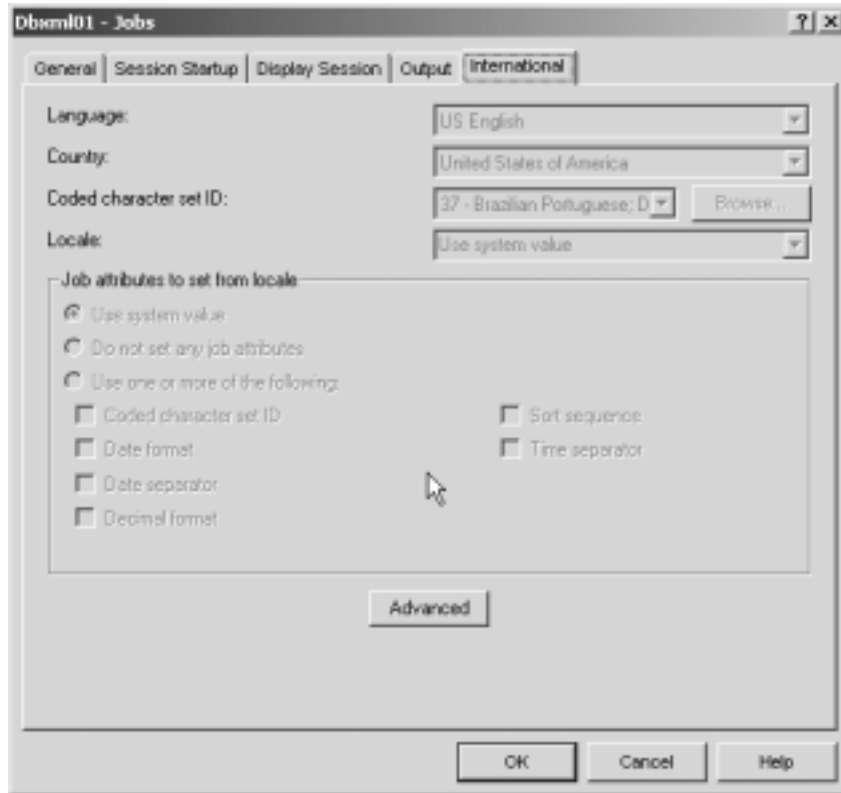


Figure 5. CCSID values for a user profile

- \_\_\_ 5. If you see a different value, ask to your instructor or system administration for changes. The value for Coded character set ID must be 37.

You have now completed this task.

---

### Task 3: Mapping the ITSO folder on the IFS

When you work with XML, you need PC tools to help you with development tasks. In this lab, you create XML code in your PC and transfer the files to the iSeries.

This task explains how to map an existing directory in the IFS as a PC drive. We created a place to organize the work for all groups. Mapping this directory will help you to send files to the iSeries.

Share the IFS folder with NetServer:

- \_\_\_ 1. Go to the iSeries Navigator Main Window. Click to expand **File Systems-> Integrated File System**. This expands to show the main file systems supported by iSeries. Then expand **Root**, which expands to show the main folders in the IFS. See Figure 6.

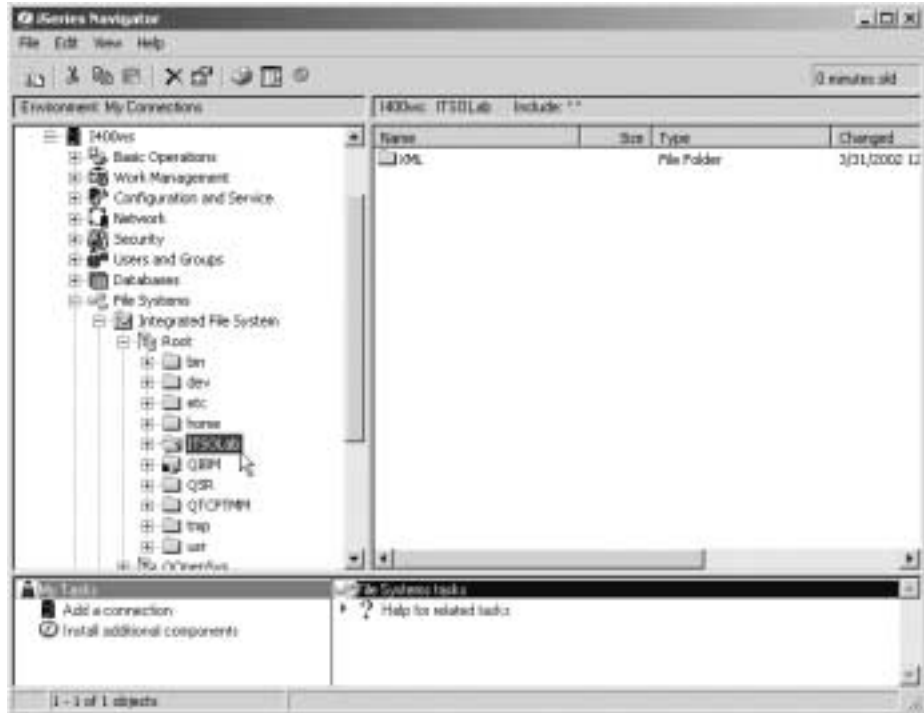


Figure 6. IFS directories

2. Right-click the **ITSOLab** icon. This directory is used to manage all the directories for all groups. If this folder appears with a hand on the base of the icon, someone may have already created the Share. In this case, skip to 5 on page 8.

From the pop-up menu, select **Sharing-> New share** as shown in Figure 7.

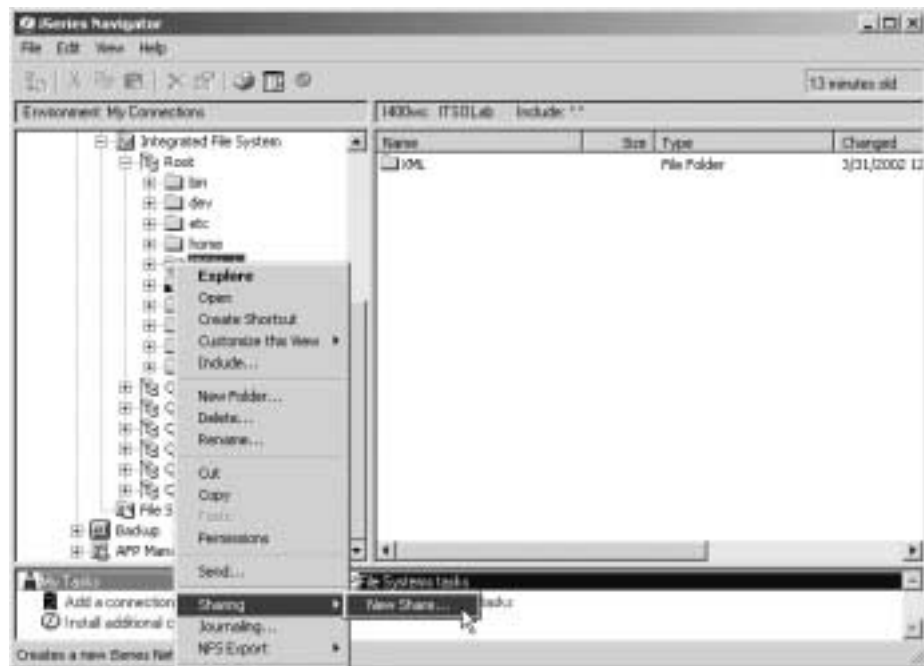


Figure 7. Sharing the ITSOLab directory

- \_\_\_ 3. The iSeries NetServer File Share window (Figure 8) opens. In the Description field, write a meaningful identifier. For Access, select **Read/Write** because you need to place your files here.

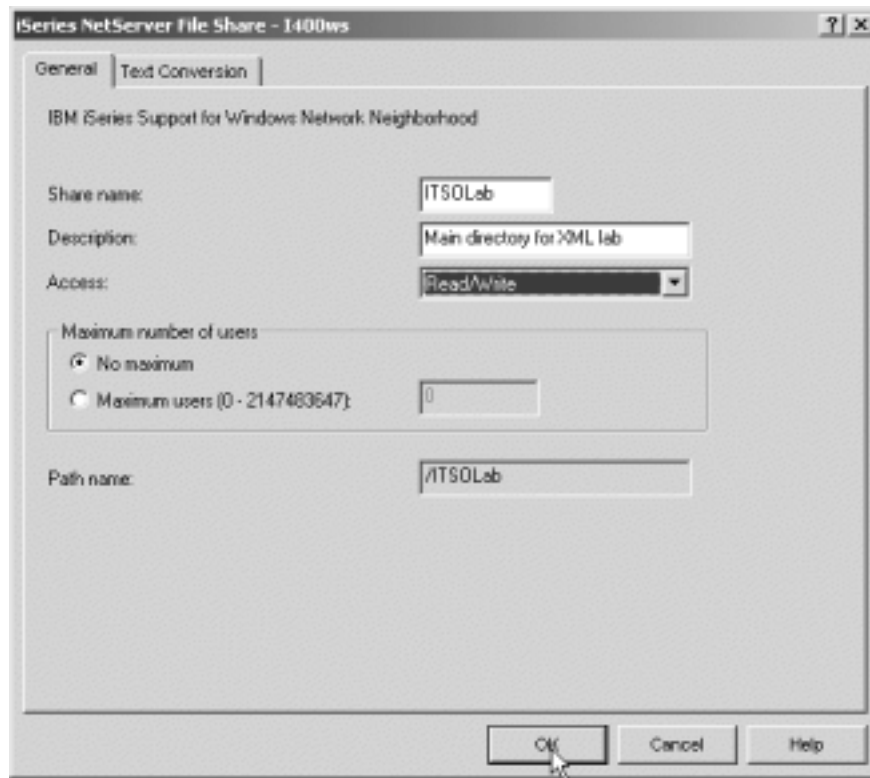


Figure 8. Adding a description to your Share

- \_\_\_ 4. Click the **OK** button to confirm your task. After this operation, the ITSOLab folder is shared for others to access. The folder appears with a hand on the base of the icon (in the main window of iSeries Navigator).
- \_\_\_ 5. Assign a disk drive letter to this folder. You can go to iSeries Navigator main window and click the **File Shares** icon to see a list of all file shares (Figure 9).



Figure 9. iSeries Navigator file shares

\_\_\_ 6. Right-click the entry for **Itsolab** and select **Map Network Drive** as shown in Figure 10.

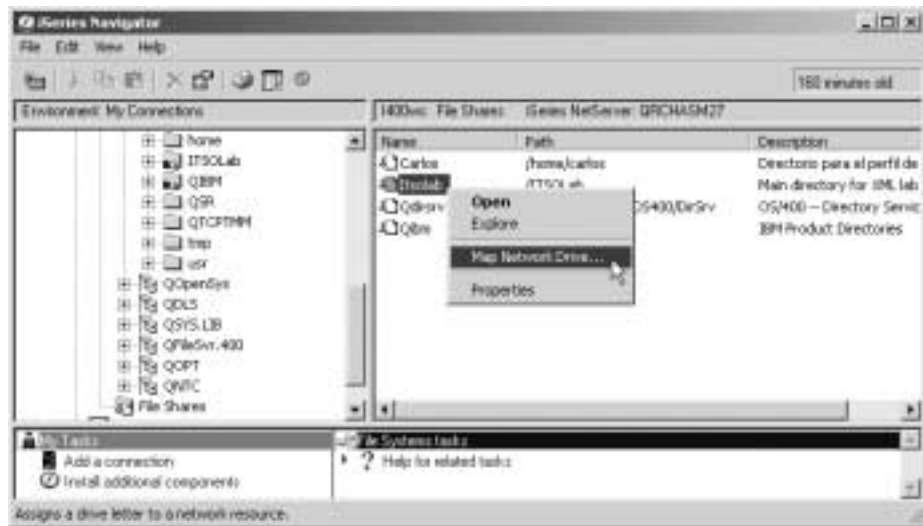


Figure 10. Menu option to map a drive

\_\_\_ 7. On the Map Network Drive window (Figure 11), complete the fields as explained here:

- a. The Local drive field is already filled in. In this lab, we use letter "I" for the main folder in this lab. If drive I is not selected, click the list button and change it.

The drive letter assigned is the next letter in ascending order to the last drive assigned. If you are in your office working with this lab, more than one drive may be assigned to your PC.

- b. In the Connect as field, verify that your user ID is entered.
  - c. Enter your password.
  - d. Verify that the **Reconnect at logon** option selected.
- Click the **OK** button to confirm your options.

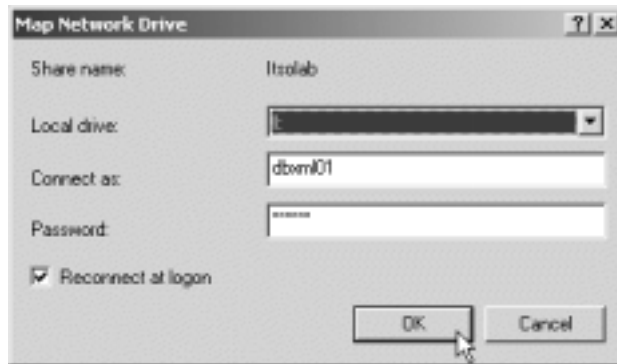


Figure 11. Information about your mapped disk drive

This task is now completed.

---

## Task 4: Creating a directory to store your files

It is important to organize the information in the IFS, especially when many developers are working with e-business solutions as XML and many directories are created. Find information can be a challenge if the data is not well organized.

For this lab, the folder ITSOLab is the main directory. In the previous task, you mapped it. Now, create your folder to store your XML files:

- \_\_\_ 1. Click the **ITSOLab** icon. When the list expands, you see the XML folder. If there more folders, ignore them. All labs will work inside XML folder.
- \_\_\_ 2. Right-click the **XML** folder and select **New Folder...** as shown in Figure 12.

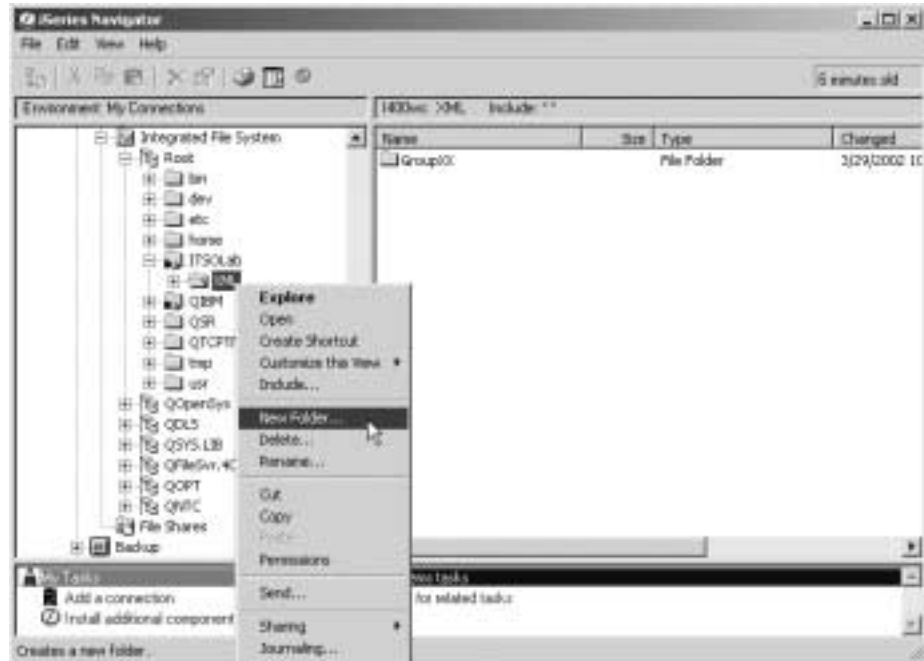


Figure 12. New folder

- \_\_\_ 3. A New Folder window (Figure 13) opens where you name your container. Write the name of your folder `GROUPXX`, where `xx` is your group number. Click the **OK** button.

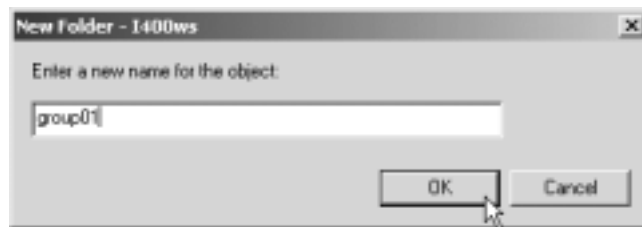


Figure 13. New folder information

The right pane updates with your folder as part of the list.

This task is now completed.

---

## Task 5: Getting started with WebSphere Studio Application Developer

WebSphere Studio Application Developer is IBM's latest suite of e-business application development tools. This tool is focused in productivity and integration. It provides a visual interface armed with powerful environments to create applications.

We use the editing facilities provided with WebSphere Application Developer to create and update XML files.

WebSphere Application Developer provides additional tools that can be used to browse relational database schemas, build SQL queries, and generate Document Access Definitions (DAD) files using simple drag-and-drop techniques to

associated XML content with relational database entities. These tools can make DAD file creation and maintenance easier.

In this lab, you have to create DAD files using a basic XML editor to give you a better feel for the information contained in a DAD file and the role it plays within XML Extender.

- \_\_\_ 1. Using the Start menu navigator of Windows, locate the WebSphere Application Developer product. Click **Start-> Programs-> IBM WebSphere Studio Application Developer-> IBM WebSphere Studio Application Developer**. The Welcome to WebSphere Studio (Figure 14) opens.

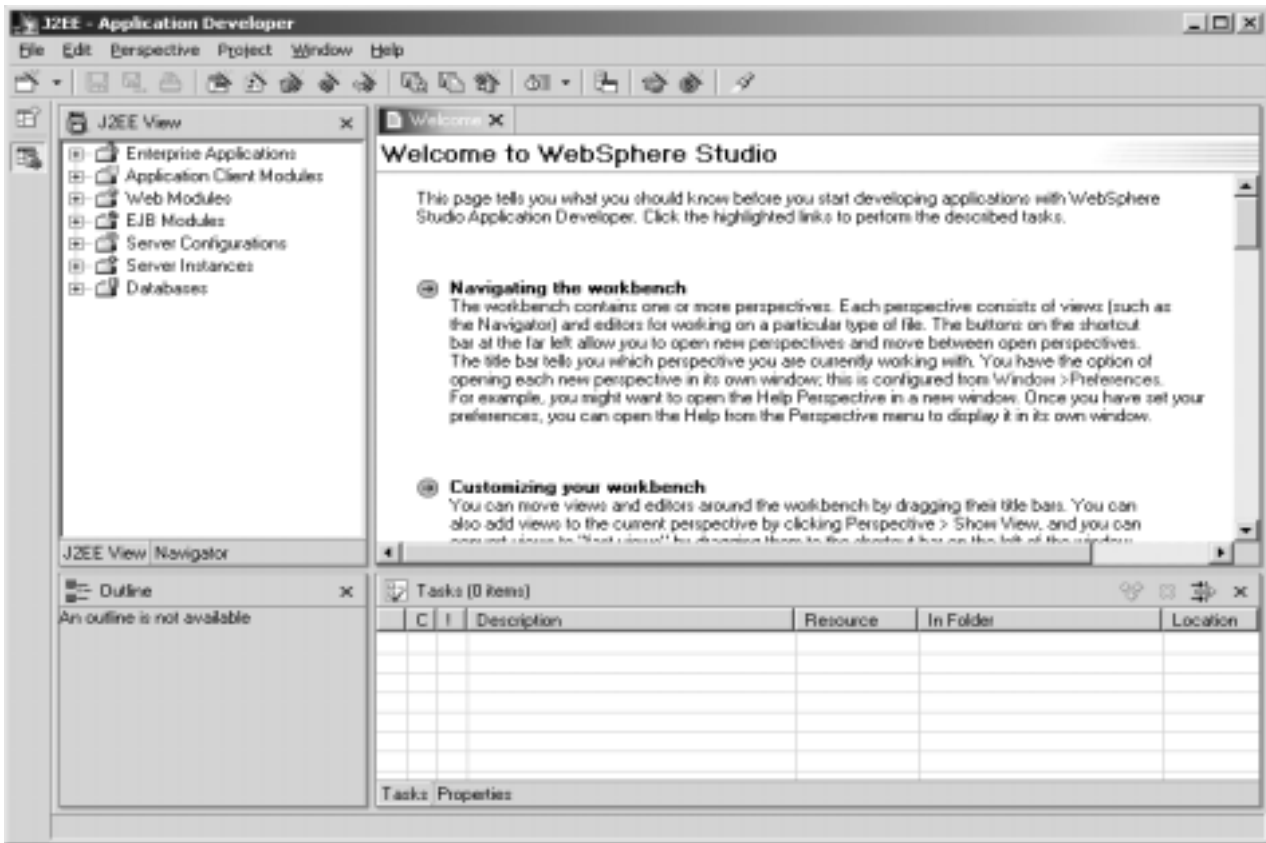


Figure 14. WebSphere Studio Application Developer welcome window

If the workstation that are you using, is already customized, the Welcome window does not open. Instead, a window reflecting the developer environment opens.

- \_\_\_ 2. Create a new project. The Open The New Wizard (left-most icon in the toolbar) calls a wizard and helps you organize your workbench. Click the **Open The New Wizard** icon.
- \_\_\_ 3. Tell the wizard what kind of project you are working on. As shown in Figure 15, select **Simple** for the resource. In this lab, we use the basic workbench. Note in the right pane that **Project** is selected. Click the **Next** button.



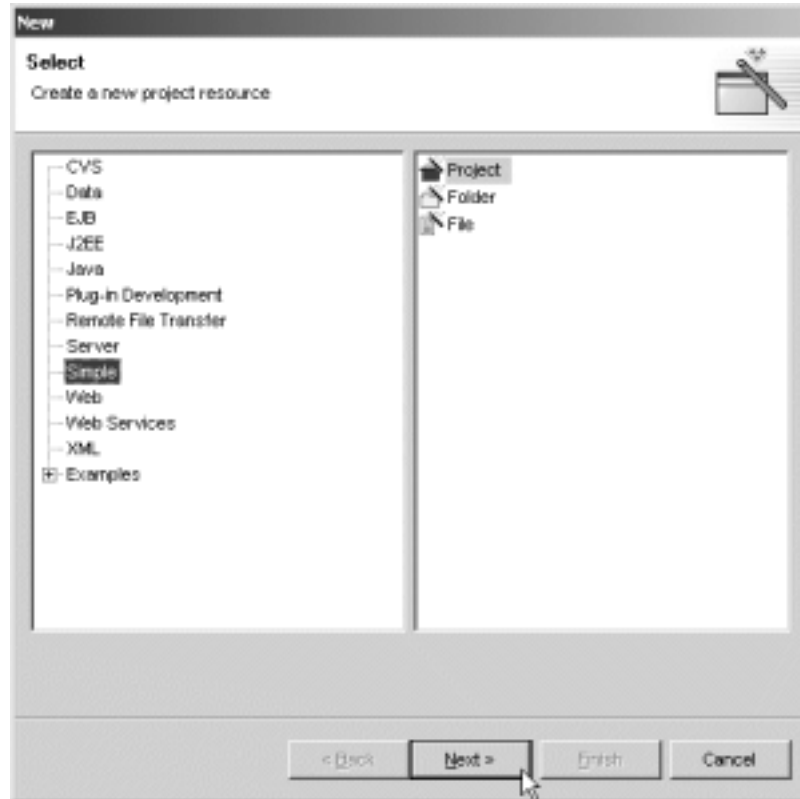


Figure 15. WebSphere Application Developer: Creating a new project

4. On the New Project display (Figure 16), enter XML Project for GroupXX as a description for this project. Click the **Finish** button.

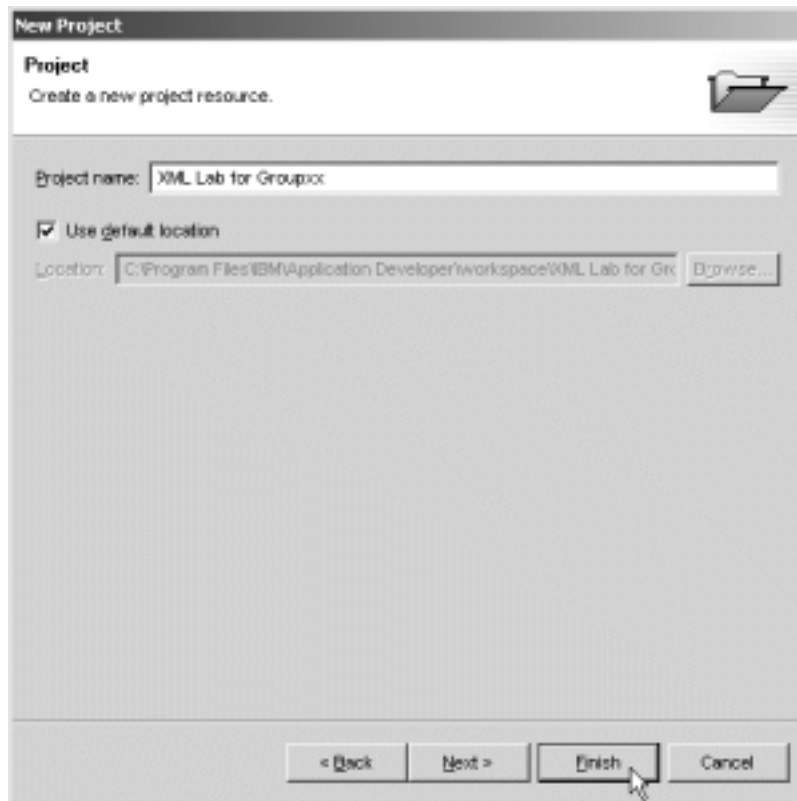


Figure 16. Naming the project

Your project is created. The window shown in Figure 17 contains an example only. In the upper left pane, you see the entry for the project created for you.

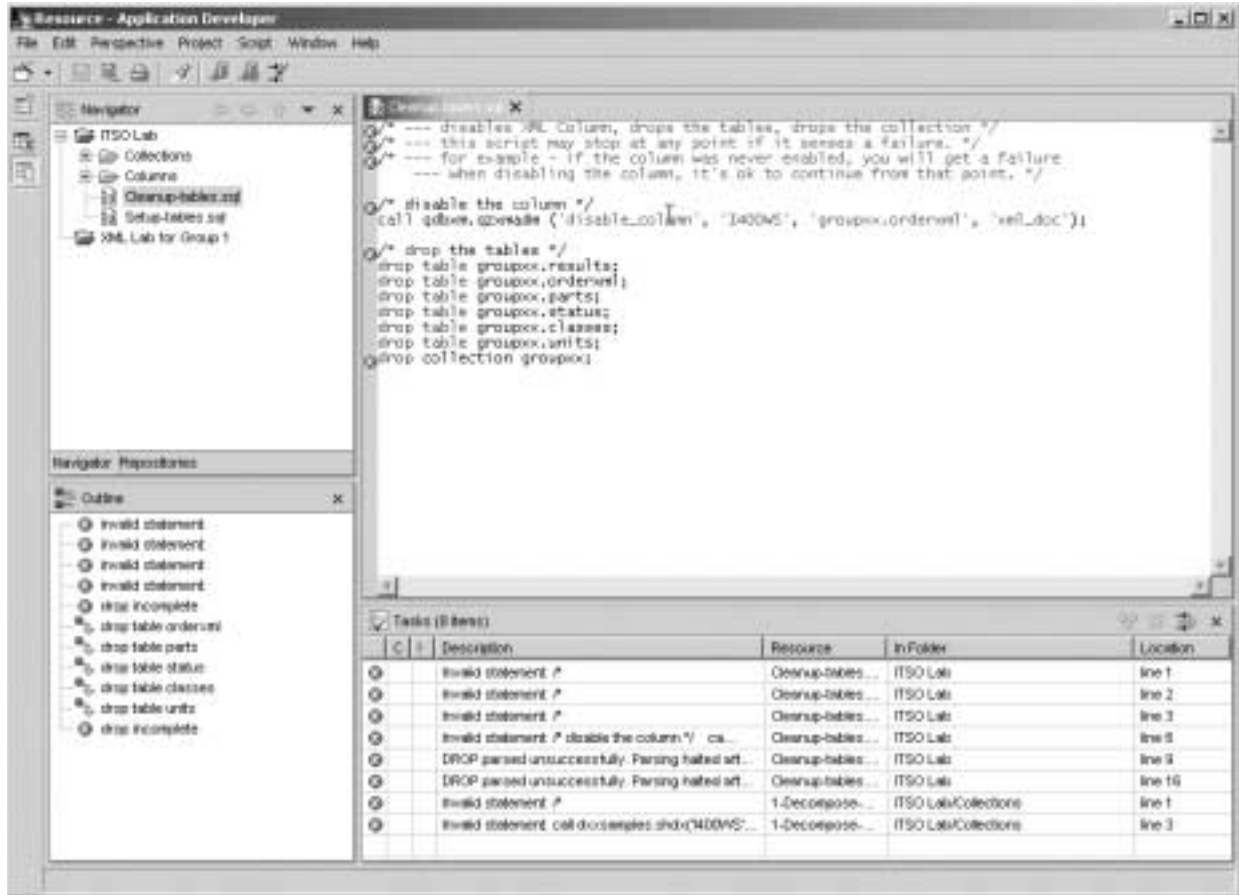


Figure 17. WebSphere Application Developer workbench

The main window is called the *workbench*. It contains four panes with information about your work:

- The *Navigator* pane displays projects and folders used to organize your work and the various files associated with each project and folder. You can select items in the navigator and perform various actions on them, depending on the type of the item (such as editing a DTD file).
- The *Editor* pane displays files that are currently being viewed or edited. If more than one file is currently being edited, it will appear as a tab along the top of the Editor pane.
- The *Outline* pane represents a view of the information being edited. When you edit an XML DTD, the Outline pane displays the various elements and attributes defined in the DTD. When you edit a DAD file, the Outline shows the elements and attributes for the file being edited.
- The *tasks* pane serves as a “to-do” list of tasks that need to be performed. Task pane use varies by application. The Tasks section is used by compile processes to show the compile-time errors encountered. Clicking a compile error entry in the Task pane brings up an edit session on the source generating the error.

5. Because this lab is an XML development project, we need to activate more facilities from WebSphere Application Developer. Create a Perspective, which in WebSphere Application Developer represents a canned set of

windows and actions, optimized for performing a given task like working with XML data. Click **Perspective-> Open-> Other** (Figure 18).

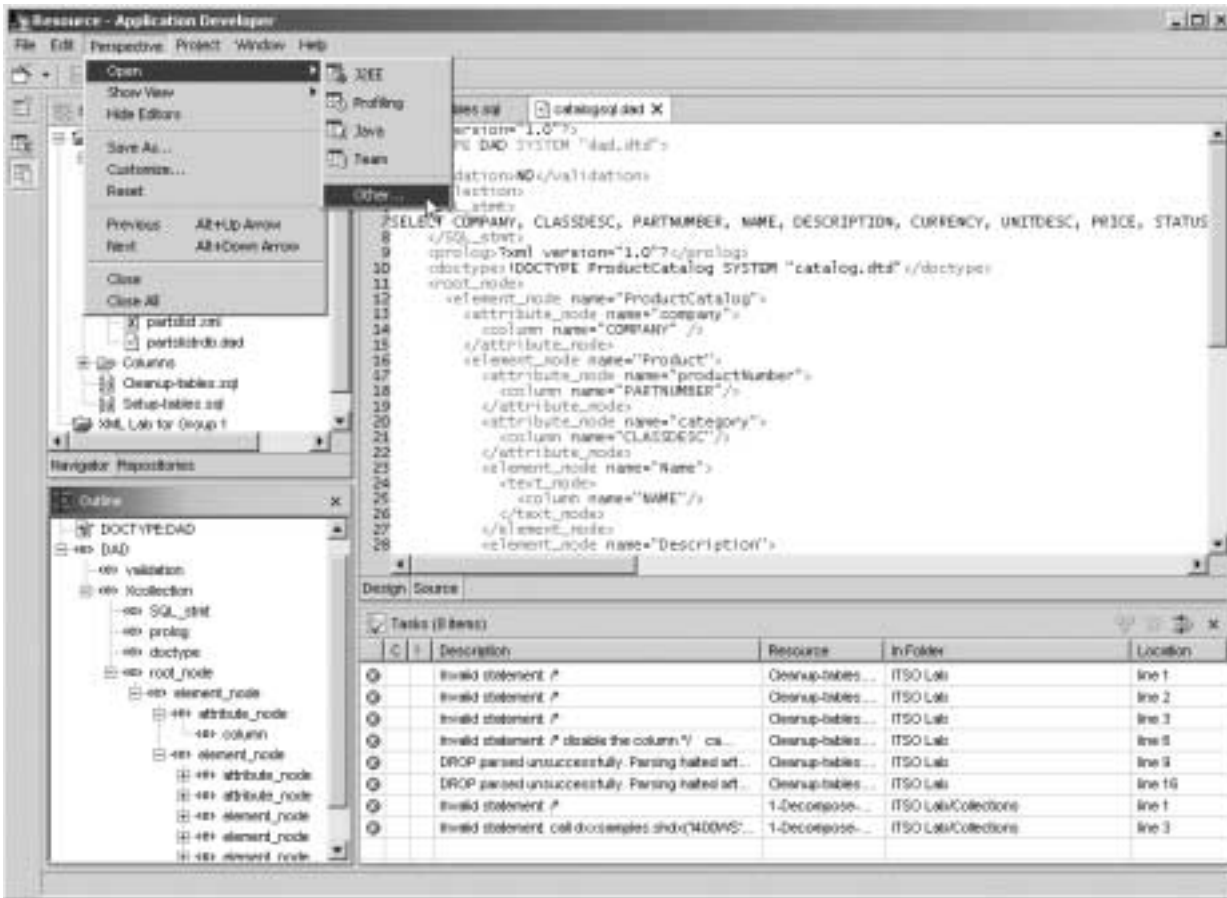


Figure 18. Activating a perspective for your workbench

- \_\_\_ 6. At the end of the list, select **XML**. A more productive workbench to work with XML now appears.
- \_\_\_ 7. In this lab, we do not use the Tasks pane (Figure 19) because all code will be used for the server. Click the **Close** button.

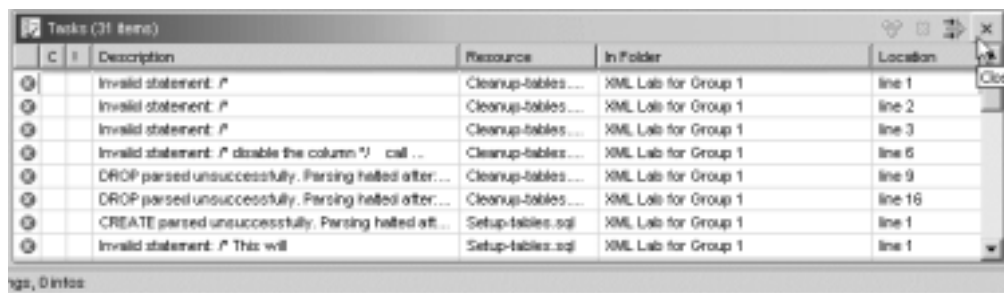


Figure 19. Closing the Tasks pane

This task is completed.

You have now created the environment and objects necessary to work with your XML lab. You have now completed this lab.

---

## Lab 2. Working with XML Columns

When you decide to use XML documents, there are two ways to store them:

- Using XML Columns
- Using XML Collection

In this lab, we work with XML Columns. An XML Column contains the entire XML document that can be updated, searched, and extracted; it is created as an XML user data type (such as XMLVARCHAR) in a single column of a database table. Other columns in the same table may be used for storing other non-XML data. In summary, an XML Column represents another type of data type that can be associated with a column in a database table.

In addition to simple data storage, XML Columns support the notion of side tables that can be used to store a subset of information from the XML document found in the associated XML Column. When you define an XML Column, you can specify

- The number of side tables to create
- The name and data type for the columns appearing in each side table
- The particular XML element or attribute values that will be the source of data stored in each column

Side tables are very useful when you want to hold key information about your XML documents. Side tables are managed by DB2 UDB XML Extender. You do not need to create them, and you do not need insert, update, or delete entries in a side tables. Side tables are created when an XML Column is enabled for a given database table. Database triggers are used by DB2 UDB XML Extender to insert, update, and delete side table entries as XML documents are inserted, updated, and deleted from the corresponding XML Column. Side tables can be queried by applications.

A key benefits of using side tables is for simplifying and optimizing queries performed against the key data items maintained in a side table. You can create an index over columns in a side table to further optimize query performance.

XML Columns are a good way to store complete XML documents for archival reasons. The side tables associated with XML Columns provide an easy and efficient mechanism to quickly locate XML documents based on queries on individual data items within the document.

### Objectives

In this lab, you learn the steps required to setup and use an XML Column associated with one of the tables in our database. The XML Column will be used to store purchase orders received by our example lawn and garden supply center.

You will configure DB2 UDB XML Extender to generate side tables to hold a subset of the information contained in the XML purchase order documents. You will also run a number of queries to see how can you extract and manipulate the data stored in XML Columns and their associated side tables.

To understand how DB2 UDB XML Extender works, you will:

- Review the XML documents that you will store in an XML Column
- Create a Document Access Definitions (DAD) file, describing a side table

- Use the DAD file to enable an XML Column to an existing database table
- Use functions provided with DB2 UDB XML Extender to store XML documents in the XML Column
- Use other XML functions to extract and update values associated with elements and attributes stored in the XML Column

## Lab prerequisites

Prior to completing this lab, you must have completed Lab 1, “Verifying the environment to work” on page 1. You must also verify the following items:

- Your user profile has the correct parameters
- You have created a folder for your group
- You have mapped your folder in the IFS; all code has a reference to the mapped drive
- Verify that the product WebSphere Studio Application Developer is installed

## Time required

The time required to efficiently complete this lab is 30 minutes.

## Required information for working with the lab

Be sure that the drive letter for your folder in the IFS is drive “I”. If you are working in your office and you do not assign this letter to the IFS, you have to change all references to it in the source code. We recommend that you work with the “I” drive for IFS.

Every time you need to run code in the iSeries, you are prompted for your server information as shown in Figure 20.

The Connect to Server dialog contains two parts. First, it asks you for the server name. Click the **OK** button. Then, the system validates your access information and returns the name of the *local relational data base name*. The local name is used for all labs.

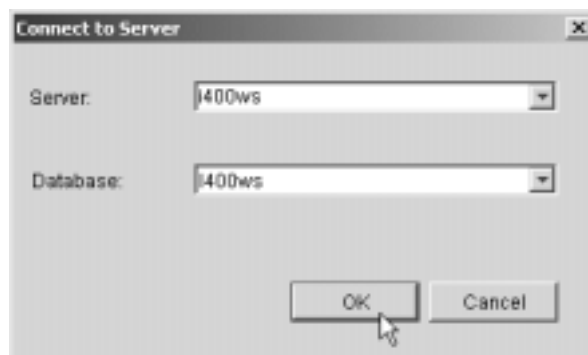


Figure 20. Connecting to the iSeries

If you do not have an active connection, you are prompted to supply your user ID and password to access the server.

## Task 1: Importing source code to be used in the PC side

### Before you begin

Ensure that you have observed the prerequisites specified for this lab and given consideration to the “I” drive issue.

All files that you need to complete this lab are stored in the IFS as a zip file. In this task, you need to import this code to the WebSphere Application Developer environment.

Here are the steps necessary to import ASCII files:

1. Start WebSphere Studio Application Developer from the Windows desktop. Click **Start-> Programs-> IBM WebSphere Studio Application Developer-> IBM WebSphere Studio Application Developer**.

When the product is loaded, the workbench is ready for you.

2. From the menu bar, click **File-> Import...** as shown in Figure 21.

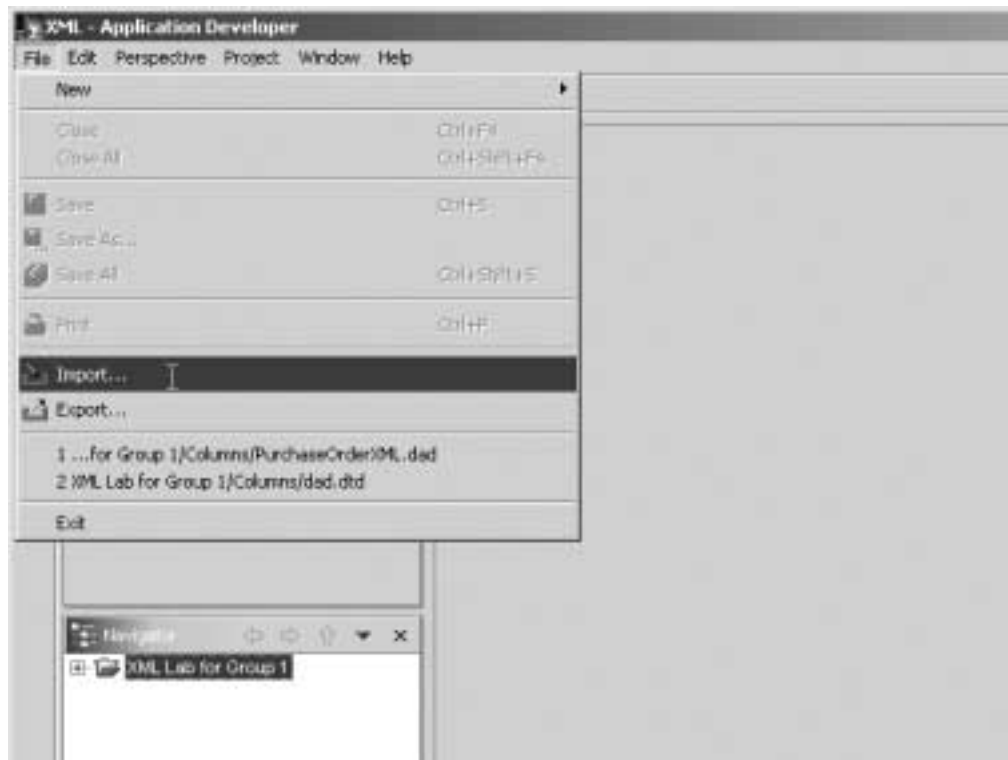


Figure 21. Importing a zip file

3. On the Import - Select dialog (Figure 22), click **Zip file** to select the type of resource.

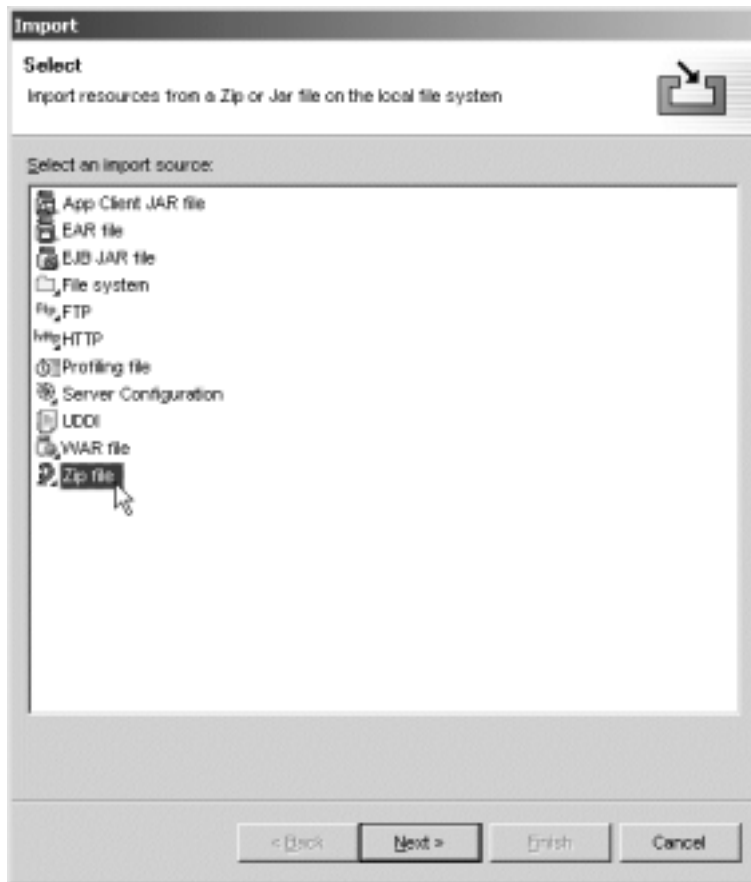


Figure 22. Define the type of resource to import

- \_\_\_ 4. Click the **Next** button to confirm your selection.
- \_\_\_ 5. On the Import Zip file dialog (Figure 23), click the **Browse** button to locate your zip file.



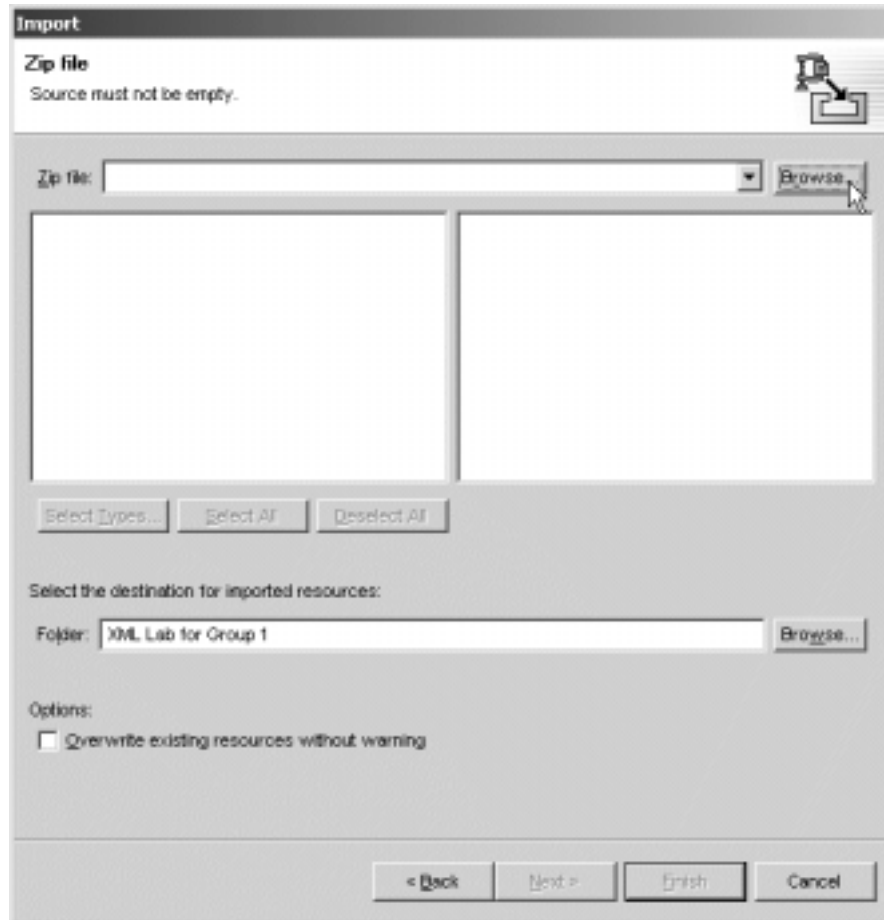


Figure 23. Locating the file to import

\_\_\_ 6. In the Open window (Figure 24), locate and select the "I" drive in the Look in drop-down list. In this lab, the drive is named **itsolab on 'i400ws' (I:)**.

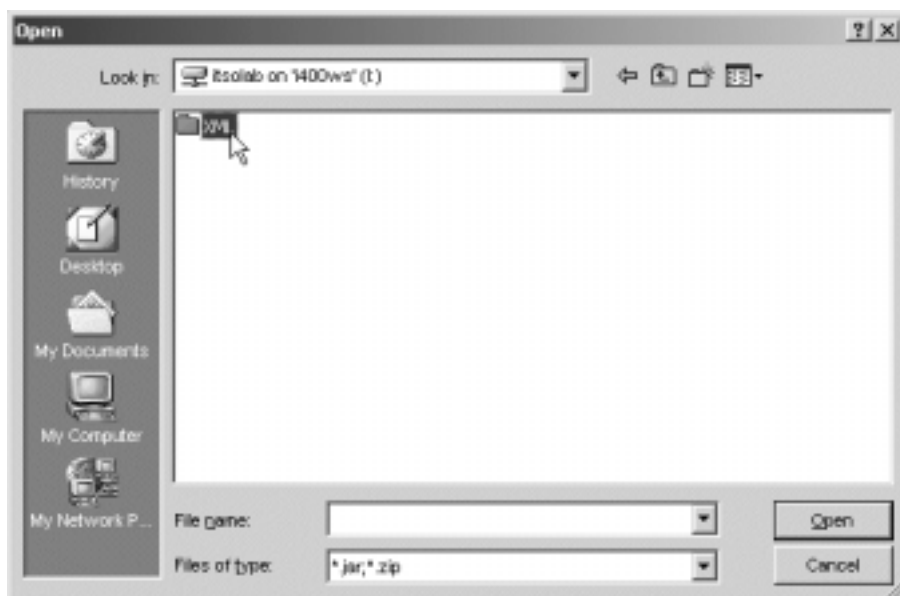


Figure 24. XML folder in the ITSOLab (IFS)

- \_\_\_ 7. Double-click the **XML** folder. This folder contains a list of files. If you are the first (or only) group accessing this folder, one file only exists – GroupXX.ZIP.
- \_\_\_ 8. Click **GroupXX.ZIP** to open it. The File name entry is already filled for you. Click **Open** to accept it. For the folder under Select the destination for imported resources, click the **Browse** button and select your **XML Lab for Group XX** subdirectory as shown in Figure 25.

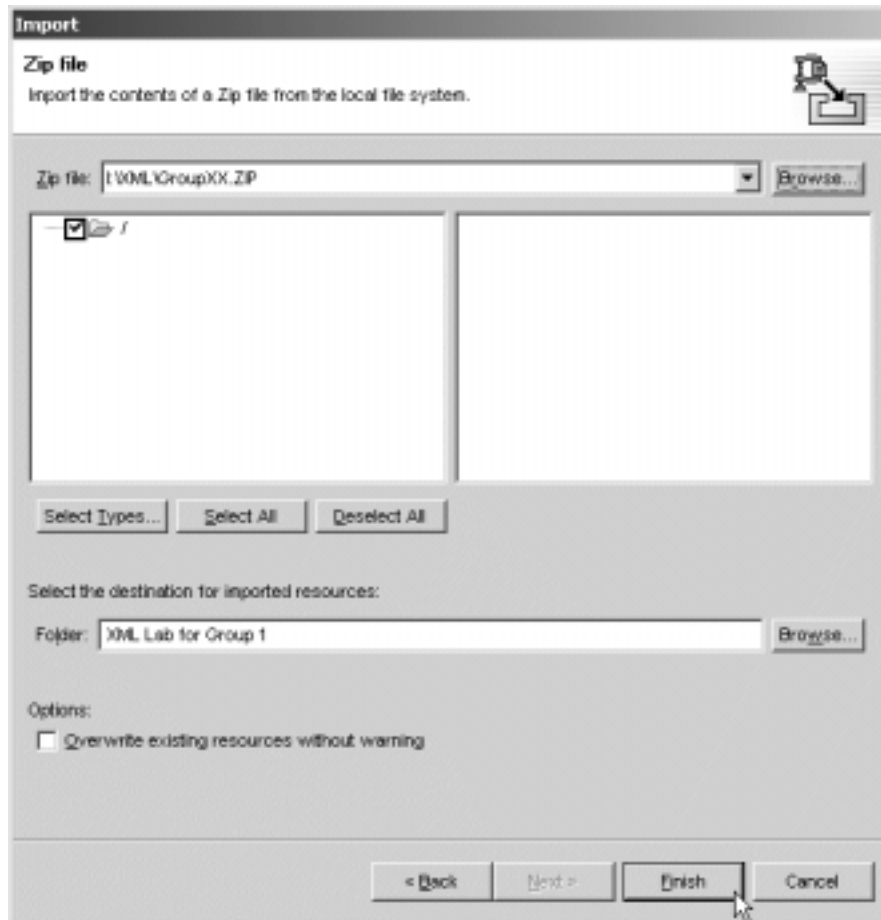


Figure 25. Last step before importing the zip file

- \_\_\_ 9. Click **Finish**. Your files are now transferring to your workbench. A progression bar appears. When the transfer is finished, your workbench is updated.  
You have transferred two folders (Collections and Columns) and two files in the root directory (Cleanup-tables.sql and Setup-tables.sql).
- \_\_\_ 10. In the Navigator pane (Figure 26), verify the file names that you need to complete this lab.

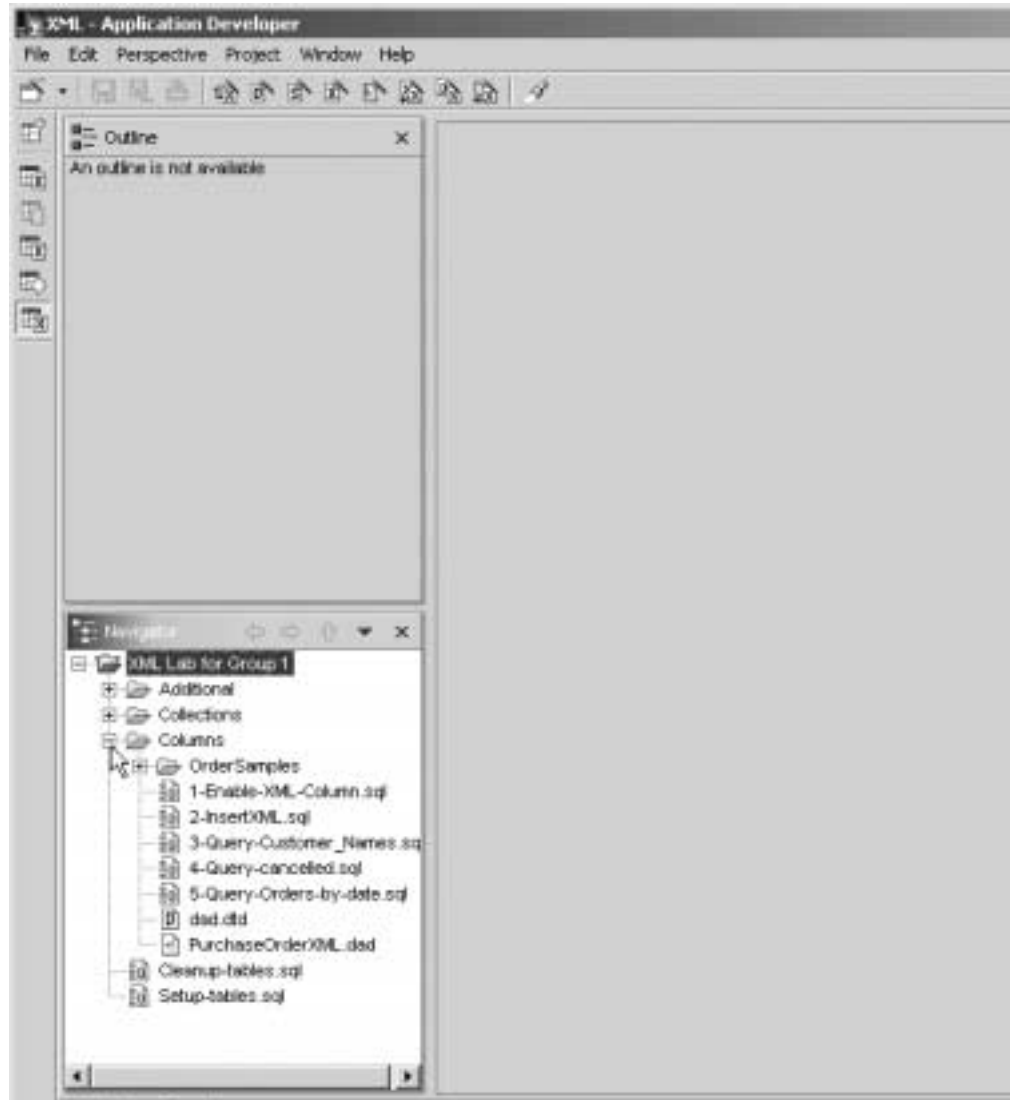


Figure 26. Verifying the content of your project

\_\_\_ 11. Open the **Columns** folder to see what files are inside.

This task is now completed.

---

## Task 2: Creating your schema and database tables

In this task, you create the schema and database tables necessary to work with your lab. You also populate the database tables by executing an SQL script.

In some steps, you are asked to open and close the files with *WebSphere Application Developer Editor*. You are asked again to open them with the *System Editor*. The purpose of this is to compare how the source code is presented using both tools. You need to know how to achieve the best productivity when building e-business solutions.

The main difference between using the Application Developer Editor and the System Editor is the standards associated with your source statements. We recommend you use both options when doing the labs.

- \_\_\_ 1. Locate the SQL Script named **Setup-tables.sql** in the main folder of your project. This script creates the schema and database tables used by your group throughout this lab.
- \_\_\_ 2. To open a file, there are several options depending what are you trying to do. In this step, open it with the default editor and change the references for your group number. Right-click the **Setup-tables.sql** file and click **Open** as shown in Figure 27.

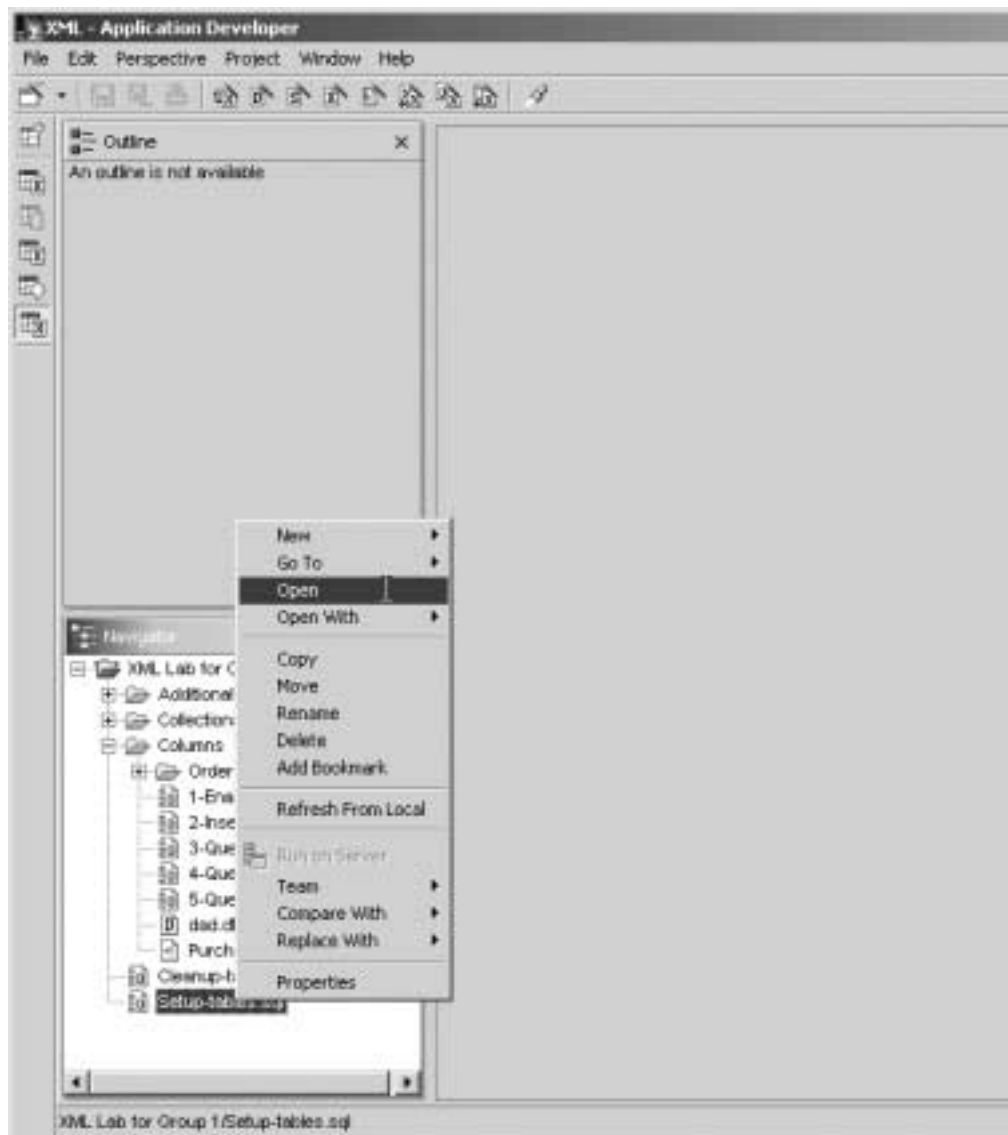


Figure 27. Opening the SQL Script file

- \_\_\_ 3. After you open the file, the components of your workbench are updated. The right pane contains the code of the SQL Script. The statements have tokens with colors to facilitate the reading of code.

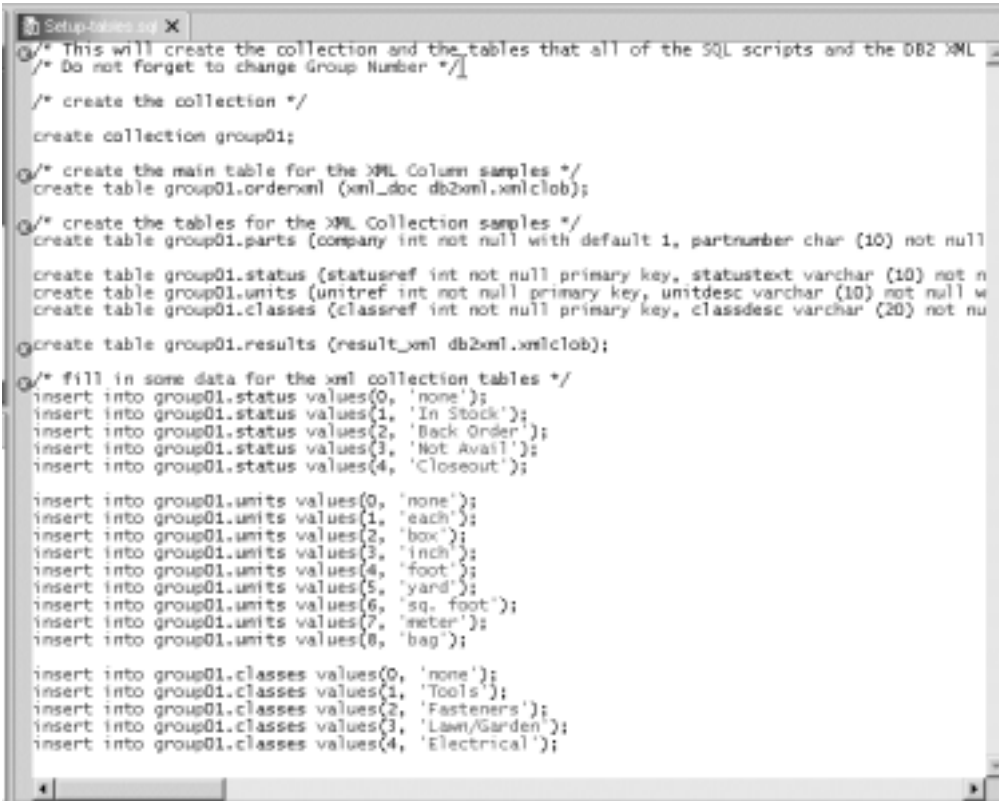
The *Outline* pane contains messages about your source code. These messages are not sent by a compiler, but because syntax checking is active in your editor.

Change all references for **GroupXX** to your group number now. You must also change the name of your schema, the place where all tables are created and, the place where the tables exists to insert rows.

For example, we Group01 as a valid group name. Figure 28 shows the source code with statements already replaced for Group01.

#### Hint

You can use the search and replace function of your editor for more efficiency.



```
Setup-tables.sql x
/* This will create the collection and the tables that all of the SQL scripts and the DB2 XML
/* Do not forget to change Group Number */

/* create the collection */
create collection group01;

/* create the main table for the XML Column samples */
create table group01.orderxml (xml_doc db2xml.xmlclob);

/* create the tables for the XML Collection samples */
create table group01.parts (company int not null with default 1, partnumber char (10) not null

create table group01.status (statusref int not null primary key, statustext varchar (10) not n
create table group01.units (unitref int not null primary key, unitdesc varchar (10) not null w
create table group01.classes (classref int not null primary key, classdesc varchar (20) not nu

create table group01.results (result_xml db2xml.xmlclob);

/* fill in some data for the xml collection tables */
insert into group01.status values(0, 'none');
insert into group01.status values(1, 'In Stock');
insert into group01.status values(2, 'Back Order');
insert into group01.status values(3, 'Not Avail');
insert into group01.status values(4, 'Closeout');

insert into group01.units values(0, 'none');
insert into group01.units values(1, 'each');
insert into group01.units values(2, 'box');
insert into group01.units values(3, 'inch');
insert into group01.units values(4, 'foot');
insert into group01.units values(5, 'yard');
insert into group01.units values(6, 'sq. foot');
insert into group01.units values(7, 'meter');
insert into group01.units values(8, 'bag');

insert into group01.classes values(0, 'none');
insert into group01.classes values(1, 'Tools');
insert into group01.classes values(2, 'Fasteners');
insert into group01.classes values(3, 'Lawn/Garden');
insert into group01.classes values(4, 'Electrical');
```

Figure 28. Setup-tables.sql modified

- \_\_\_ 4. The Editor continues showing warnings about errors in your code. These errors are detected because the source was coded to be executed in the server. The pair of */\** and *\*/* are not SQL statements; they are only comments for the server and used as documentation.
- \_\_\_ 5. From the menu bar, click **File-> Save Setup-tables.sql**.
- \_\_\_ 6. Exit the Editor. From the menu bar, click **File-> Close**.
- \_\_\_ 7. The SQL Script is ready to be executed in the host. Right-click the file and select **Open With-> System Editor**.

You see a status bar that shows the progression of loading the SQL Script file. When completed, a new window opens for you to work with the script file. Figure 30 shows the same interface used by Run SQL Scripts....

- \_\_\_ 8. Verify that you are using the SQL naming convention in your session. From the menu bar, select **Connection-> JDBC Setup....** On the JDBC Setup window (Figure 29), click the **Format** tab and verify that **SQL (\*SQL)** is entered in the Naming convention field.

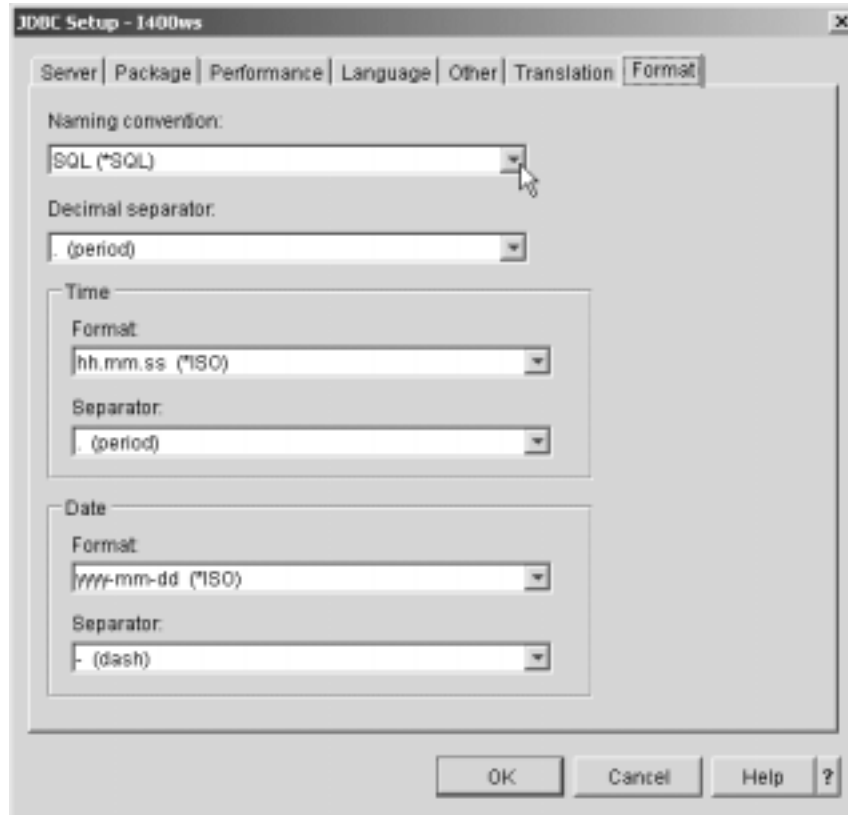


Figure 29. Setting the SQL naming convention

- \_\_\_ 9. Figure 30 shows the window with the SQL Script already open. You can execute all statements from the menu bar by clicking **Run-> All**. Or, you can click the **Run All** icon on the toolbar. If you prefer to follow the execution of this script, you can execute the statements one by one.

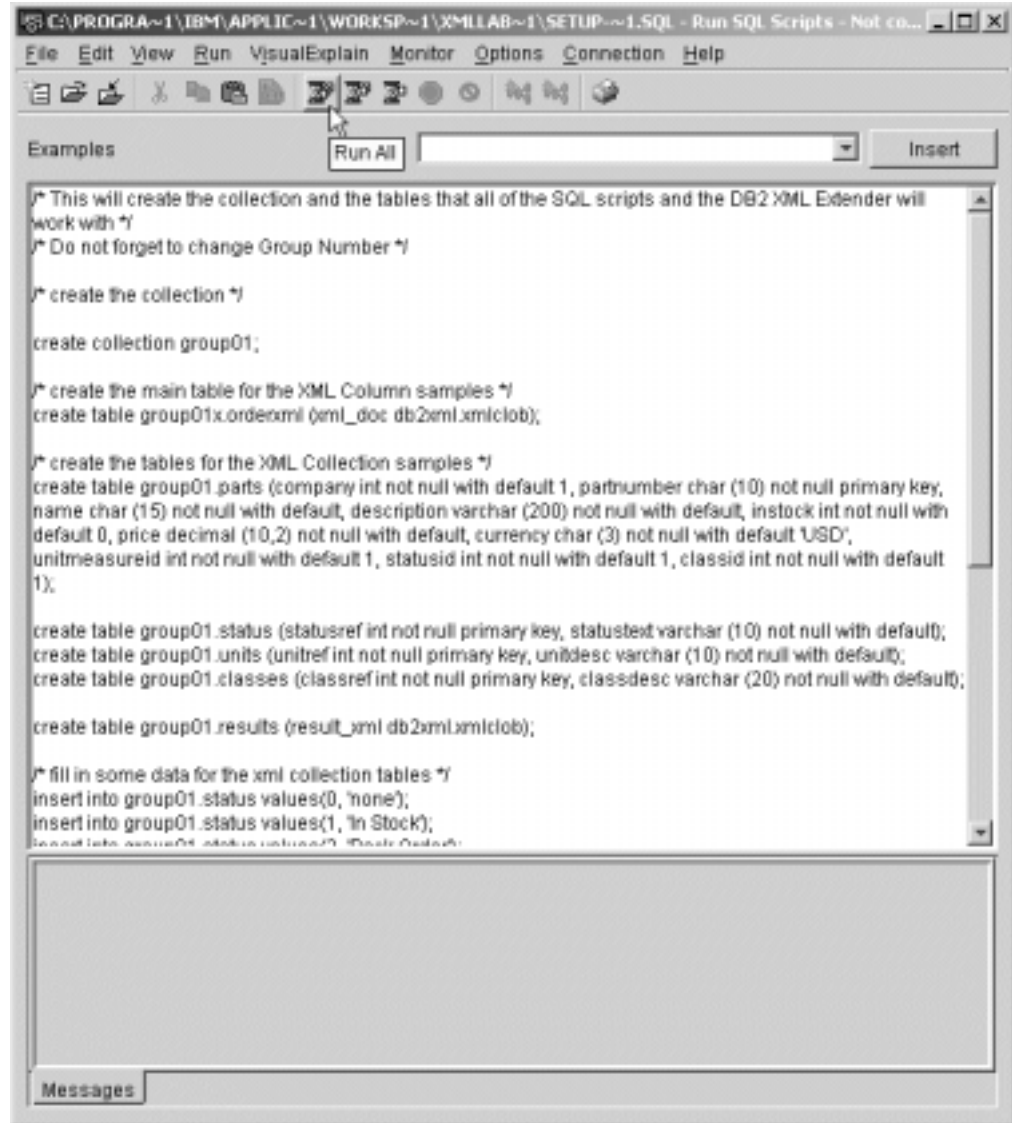


Figure 30. Running SQL Script to create schema and tables

- \_\_\_ 10. You are prompted for your server information, which we explained at the beginning of this lab. Click **OK** to run the SQL Script.
- \_\_\_ 11. After all SQL Script statements are executed, click **File-> Exit**.

This task is now completed. Now we have a schema with populated tables on the iSeries server.

### Task 3: Reviewing the information in an XML document

An XML document is very simple in that it only contains tags and data. It is something like the data portion of an HTML document. An XML document does not contain presentation tags.

The tag names used in an XML document represent the data. This is the main reason why these types of documents are used to share information in the e-business world.

In this task, you review a typical XML file:

- \_\_\_ 1. Go to the WebSphere Application Developer main window.
- \_\_\_ 2. In the Navigator pane (Figure 31), expand the folders **Columns->OrderSamples**. You should see six XML documents files listed inside the OrderSamples folder.

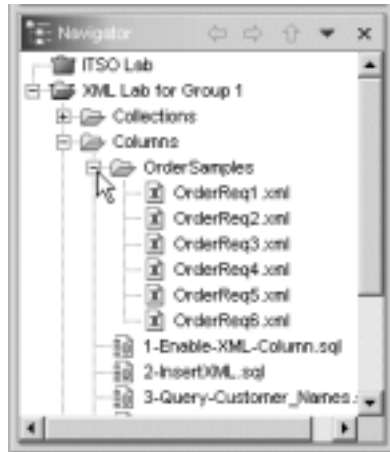


Figure 31. OrderSamples folder

- \_\_\_ 3. Open one of these files and examine the information inside. For example, using Web Browser in WebSphere Application Developer, right-click the **OrderReq1.xml** file and select **Open With-> Web Browser**.
- \_\_\_ 4. The opened XML file appears as shown in Figure 32. XML is built of tags and data. The red color is used for a tag. The black color is where the data is.

Review how the information is grouped for the Purchase Order. The Header tag is defined for the Customer and Order sections. Customer has an ID and a name. Order has an ID, date, type, etc.



```
<?xml version="1.0" ?>
- <PurchaseOrderXML version="0.1">
  - <Header>
    - <Customer id="CN000001">
      <Name>Rons Hardware</Name>
    </Customer>
    - <OrderInfo orderID="ON74334458" date="2001-10-05" type="new">
      <Shipping />
      <TotalMoney currency="USD">260.00</TotalMoney>
      <Notes>None</Notes>
    </OrderInfo>
  </Header>
  - <ItemList>
    - <Item partnum="PN00000015">
      <Description>Screw Driver</Description>
      <Quantity>10</Quantity>
      <Price>1.00</Price>
      <Notes />
    </Item>
    - <Item partnum="PN00000003">
      <Description>Sheet Metal Scr</Description>
      <Quantity>50</Quantity>
      <Price>5.00</Price>
    </Item>
  </ItemList>
</PurchaseOrderXML>
```

Figure 32. Opening an XML file with Web Browser

5. To continue reviewing this XML file, open the same file using the XML Editor. Right-click the **OrderReq1.xml** file and click **Open With-> XML Editor**.

Figure 33 shows the source code for your XML file. Here, you see colors for both tags and fields. You also see that several ItemList sections make up an order and Item has a part number, description, quantity, etc.

```

1 <?xml version="1.0"?>
2
3 <PurchaseOrder XML version="0.1">
4   <Header>
5     <Customer id="CN000001">
6       <Name>Rons Hardware</Name>
7     </Customer>
8     <OrderInfo orderID="ON74334458"
9       date="2001-10-05"
10      type="new">
11       <Shipping></Shipping>
12       <TotalMoney currency="USD">260.00</TotalMoney>
13       <Notes>None</Notes>
14     </OrderInfo>
15   </Header>
16   <ItemList>
17     <Item partnum="PN00000015">
18       <Description>Screw Driver</Description>
19       <Quantity>10</Quantity>
20       <Price>1.00</Price>
21       <Notes />
22     </Item>
23     <Item partnum="PN00000003">
24       <Description>Sheet Metal Scr</Description>
25       <Quantity>50</Quantity>
26       <Price>5.00</Price>
27     </Item>
28   </ItemList>
29 </PurchaseOrder></XML>

```

Figure 33. Opening an XML file with XML Editor

- \_\_\_ 6. XML Editor offers you more options to facilitate your work. Click the **Design** tab located at the bottom of the *Editor* pane. This view of the code is more structured and easier to read. It is similar to the information shown in the *Outline* pane, but has more functions.
- \_\_\_ 7. In the Editor pane, using the XML Editor, find the date of this order. Click the plus sign in front of **PurchaseOrderXML** tag. The list is expanded and you see the main structures for this document: Header and ItemList.
- \_\_\_ 8. The date information is in the Header section. Click the plus sign in front of **Header** tag. The list expands and you see the Customer and OrderInfo tags.
- \_\_\_ 9. Expand the **OrderInfo** tag. Now you see that the entry for date is visible with the value 2001-10-05. Figure 34 shows a view of the file structure.

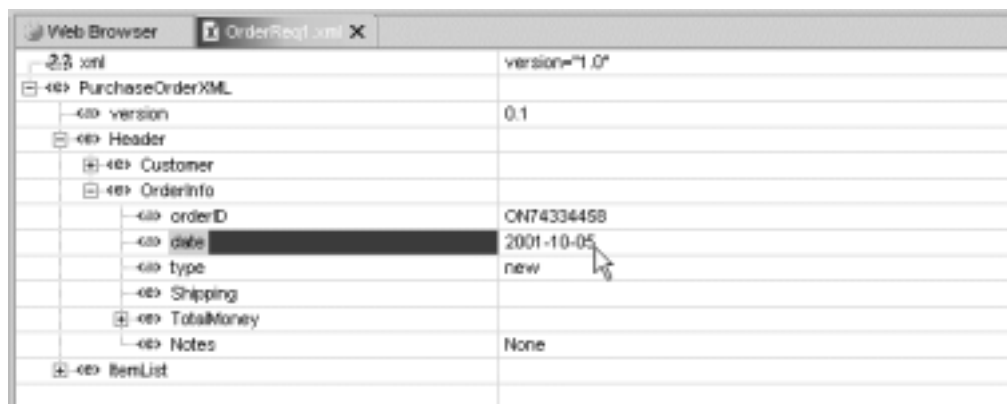


Figure 34. Date value for the tag in the XML file

- \_\_\_ 10. Close the two opened views of the file. This time, use the close button (marked with an "X") of the tab for the view. Figure 35 shows a portion of the Editor pane.

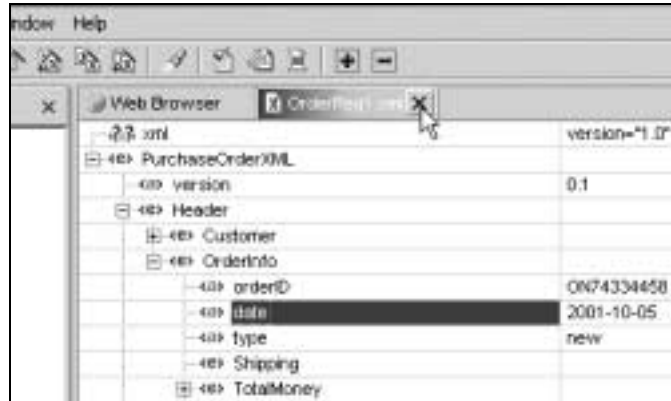


Figure 35. Closing the view with the Close button

You have now completed this task.

---

## Task 4: Exporting XML documents to the IFS

In this task, you transfer all the XML documents to the IFS. DB2 UDB for iSeries XML Extenders needs these files in the IFS. Depending of your experience, you can use the tool of your choice and skip these instructions.

In this lab, you are asked to use the Export function of WebSphere Application Developer. It is already integrated and easy to use.

Follow these steps:

- \_\_\_ 1. Locate the **OrderSamples** in the Navigator pane.
- \_\_\_ 2. Click the **OrderReq1.xml** file to select it. Then hold down the shift key and click the **OrderReq6.xml** file to select the group of entries as shown in Figure 36.

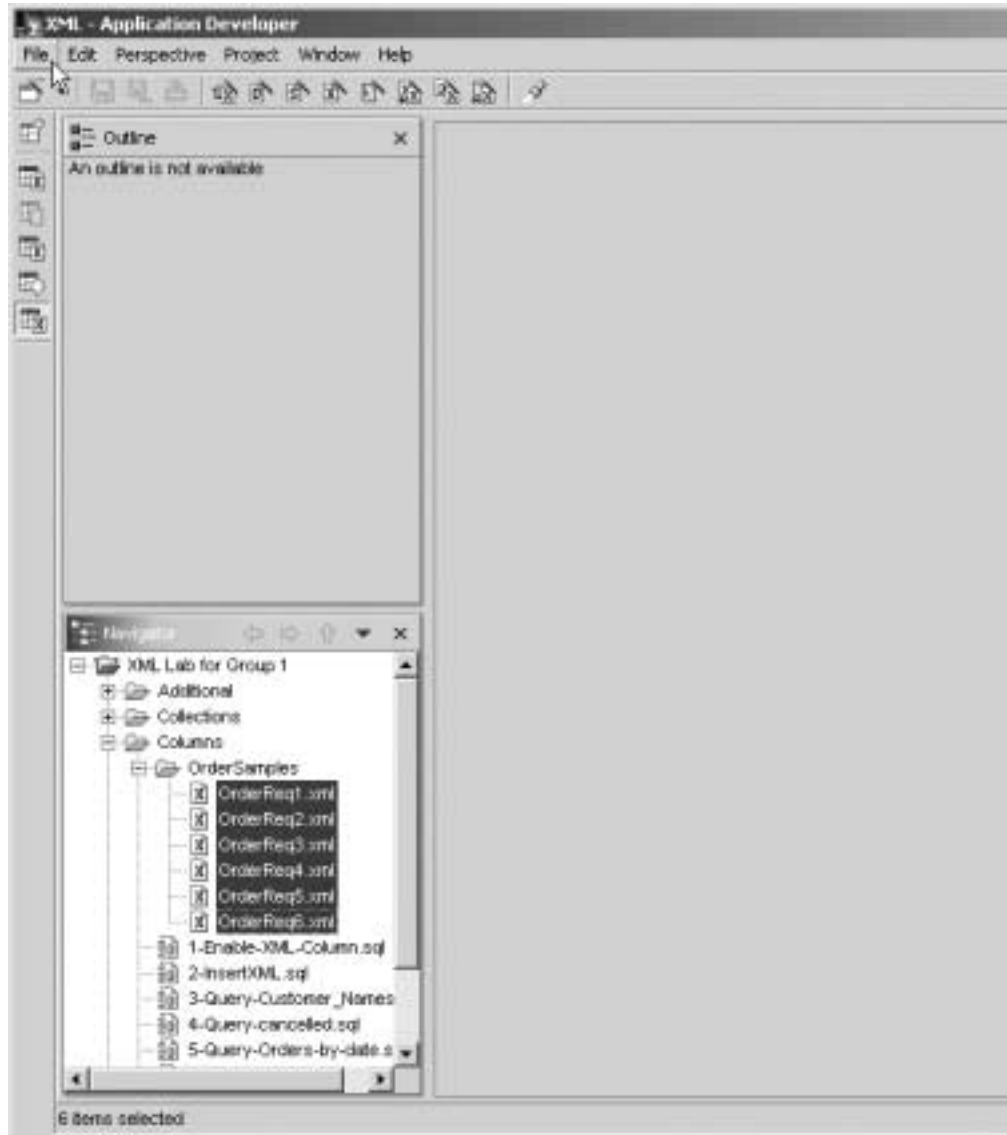


Figure 36. Selecting a group of files to import

3. Click **File-> Export....** The Export - Select dialog (Figure 37) opens and shows the resources available.

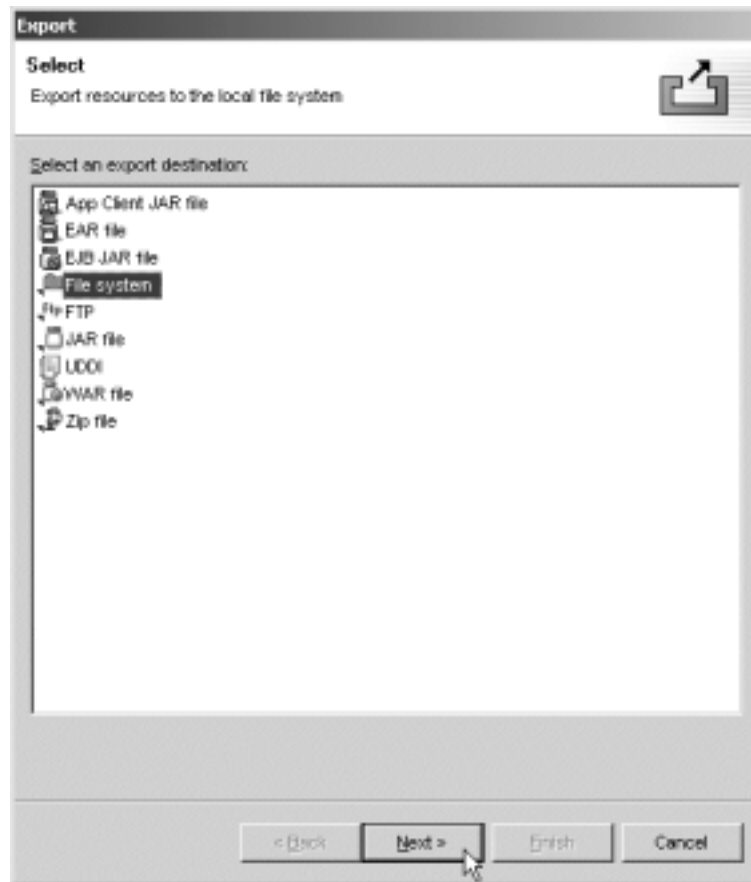


Figure 37. Resource available for export function

- \_\_\_ 4. Select **File system**. Click the **Next** button.
- \_\_\_ 5. The Export - File System window (Figure 38) opens. Specify the target directory, which is **I:\XML\GroupXX** in this lab.

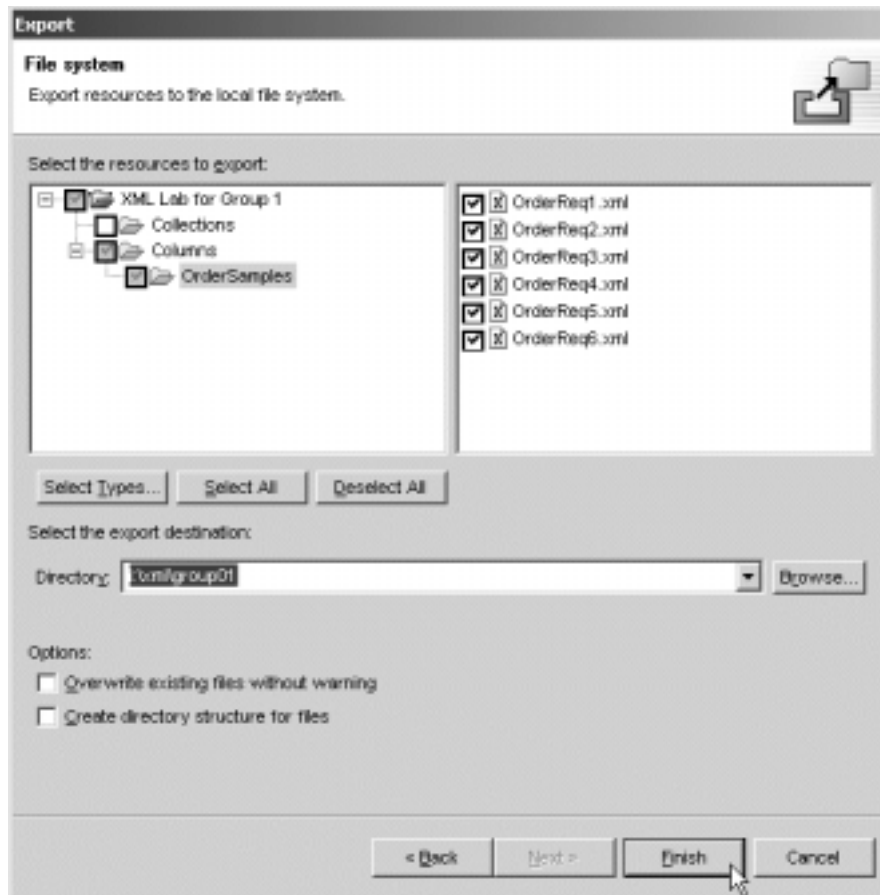


Figure 38. Selecting the export directory for your files

\_\_\_ 6. Notice that the list has been expanded in the Select resources to export box. Click **Finish** to transfer your files. A progression bar activates.

You have now completed this task.

---

## Task 5: Enabling an XML Column

Before you use an XML Column for the first time, you must enable the column in the database table. This process associates the Document Access Definitions (DAD) file with the column and creates any side tables defined within the DAD file.

When you created the tables for this lab in Task 2, "Creating your schema and database tables" on page 23, a table named ORDERXML was created with a single XML Column of type XMLCLOB. This is a UDT provided by DB2 UDB XML Extender that is capable of storing and XML document.

Before you can enable an XML Column, the DAD file associated with the XML Column must be available in an IFS directory.

\_\_\_ 1. In this exercise, you use a side table. One of the benefits of using a side table is performance. The entire document is in the XML Column and key information in the columns can be shared with your traditional applications that are already running.

Using the DAD Editor, open the PurchaseOrderXML.dad file. To do this, right-click **Purchase OrderXML.dad**, and click **Open With-> DAD Editor**.

- \_\_\_ 2. Locate the table name that is created for this XML Column. To find this information, go to the **Design** tab on the Editor pane. Under XColumn, you see a table entry and beneath the table entry, you see the *name* entry.
- \_\_\_ 3. Replace the Schema name for your table. Figure 39 shows a DAD file changed to Group01 (using the Source view).

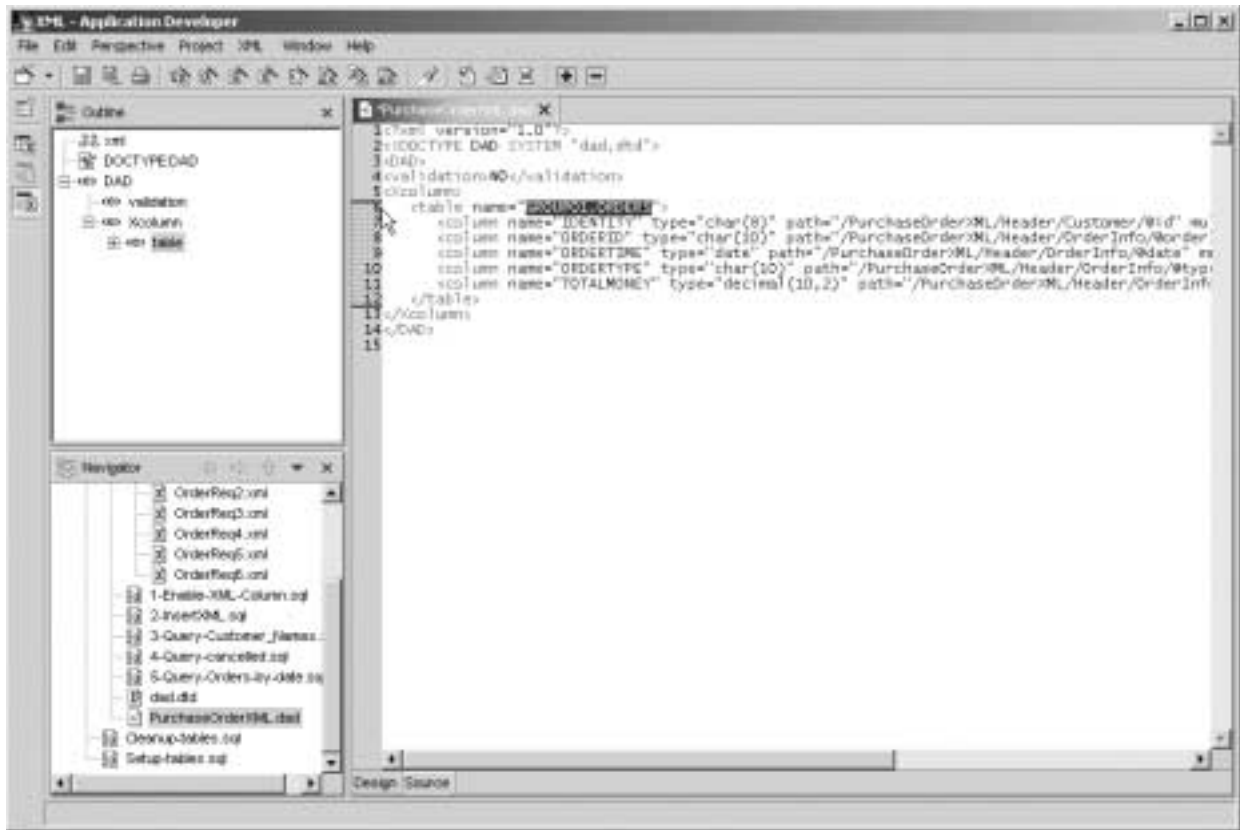


Figure 39. PurchaseOrderXML.dad file changed for use the correct side table

- \_\_\_ 4. Save the file. Click **File-> Save PurchaseOrderXML.dad**.
- \_\_\_ 5. Exit from the Editor. Click **File-> Close**.
- \_\_\_ 6. Using the export function of Application Developer, transfer the DAD file. Click the **PurchaseOrderXML.dad** file and use the Export function as explained in Task 4, "Exporting XML documents to the IFS" on page 31. Use the same target directory.
- \_\_\_ 7. You are ready to enable the XML Column using the DAD file. Expand the Columns folder. Then locate and right-click the **1-Enable-XML-Column.sql** file. Select **Open With-> System Editor**.

This action launches the Run SQL Script utility in iSeries Navigator. Figure 40 shows two SQL sentences that are executed.

**Note**

Be sure you replace xx with your group number.

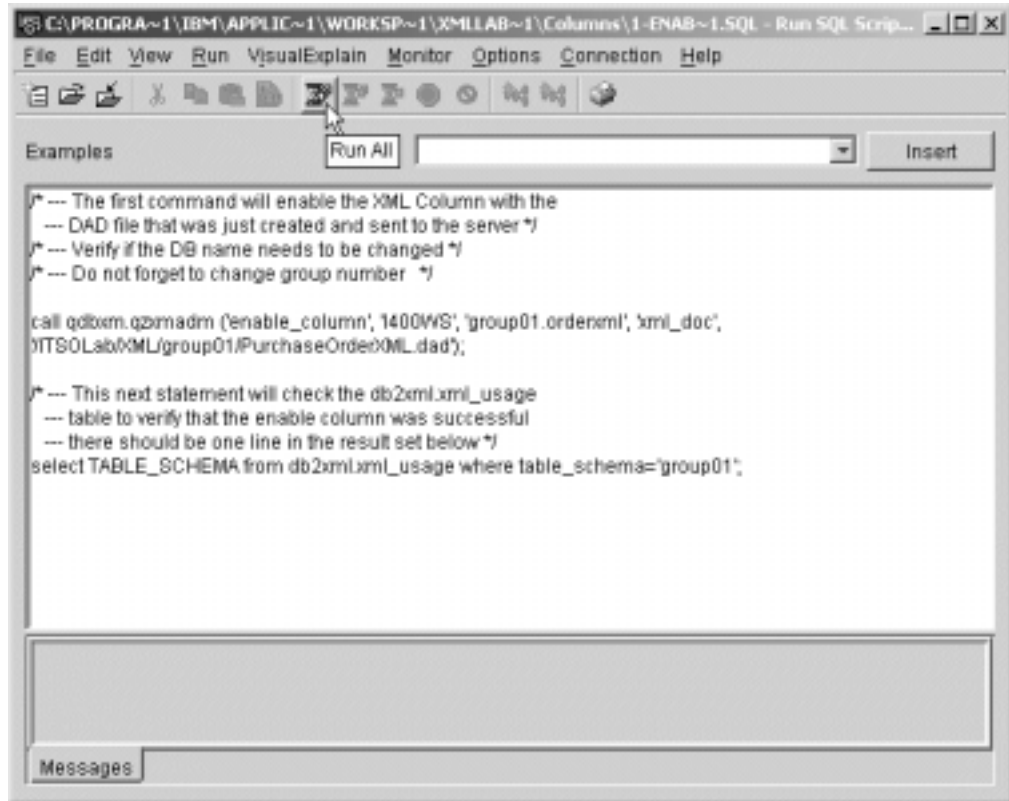


Figure 40. Enabling an XML Column

The first sentence is a *Call* statement to enable the XML Column. The program called is *qzxadm* and is part of DB2 UDB XML Extender. The parameters used by this program are:

- **enable\_column**: The action to perform
- **I400WS**: The name of the relational database name; use capital letters
- **GROUPXX.orderxml**: The name of the table within schema
- **xml.doc**: The name of the column to be enabled
- **ITSOLab/XML/GROUPXX/PurchaseOrderXML.dad**: Your DAD file

**Note**

If you experience problems running this command, try using a 5250 session. Open a 5250 session and invoke the Qshell interface. On the command line, type *Qsh* and type the following command:

```
dxxadm enable_column I400WS GROUPxx.orderxml xml_doc
/ITSOLab/XML/Group10/PurchaseOrderXML.dad)
```

The second sentence is a *Select* statement where only one column is retrieved for you. The purpose of this statement is to check whether DB2 UDB XML Extender has accepted your request for enabling your column.



DB2XML is the product library, and XML\_USAGE is a directory used by the product.

- \_\_\_ 8. Run your SQL Script to enable the XML Column and to verify if it has been accepted. Use the **Run All** icon to execute it. Provide your server ID and then the database name in the dialog window. Enabling a table column takes some time. Wait for the completion of the process.

**Do not continue if...**

...no record was selected. Tell your instructor or system administrator that DB2 UDB XML Extender has not accepted your request.

A common problem is when you do not replace your group number correctly, or you specify the wrong local relational database name for your system

- \_\_\_ 9. Exit from iSeries Navigator. Click **File-> Exit**.
- \_\_\_ 10. Because you made changes to the script file, you are asked if you want to keep the changes in your file. Respond by clicking **Yes** to this message.
- \_\_\_ 11. If you have time, review what was created automatically in your schema. Four triggers and a database table were created by DB2 UDB XML Extender. The database table **ORDERS** is a side table to help you in the search of elements from the XML documents.

This task is now completed.

---

## Task 6: Storing your document into the XML Column

Your database table is ready to store your XML documents. You have already enabled the column in your table and it is waiting for your documents. An SQL Script file is already created for you.

Follow these steps:

- \_\_\_ 1. In the Navigator pane, locate and right-click the **2-InsertXML.sql** file. Then click **Open With-> System Editor**. This launches Run SQL Script.
- \_\_\_ 2. Make changes appropriate to work with your group number. Remember that the Replace function is available. It is more accurate and faster since you need to change the schema name and the IFS directory where your data is.
- \_\_\_ 3. Figure 41 shows you the SQL Script with all references replaced for Group01.

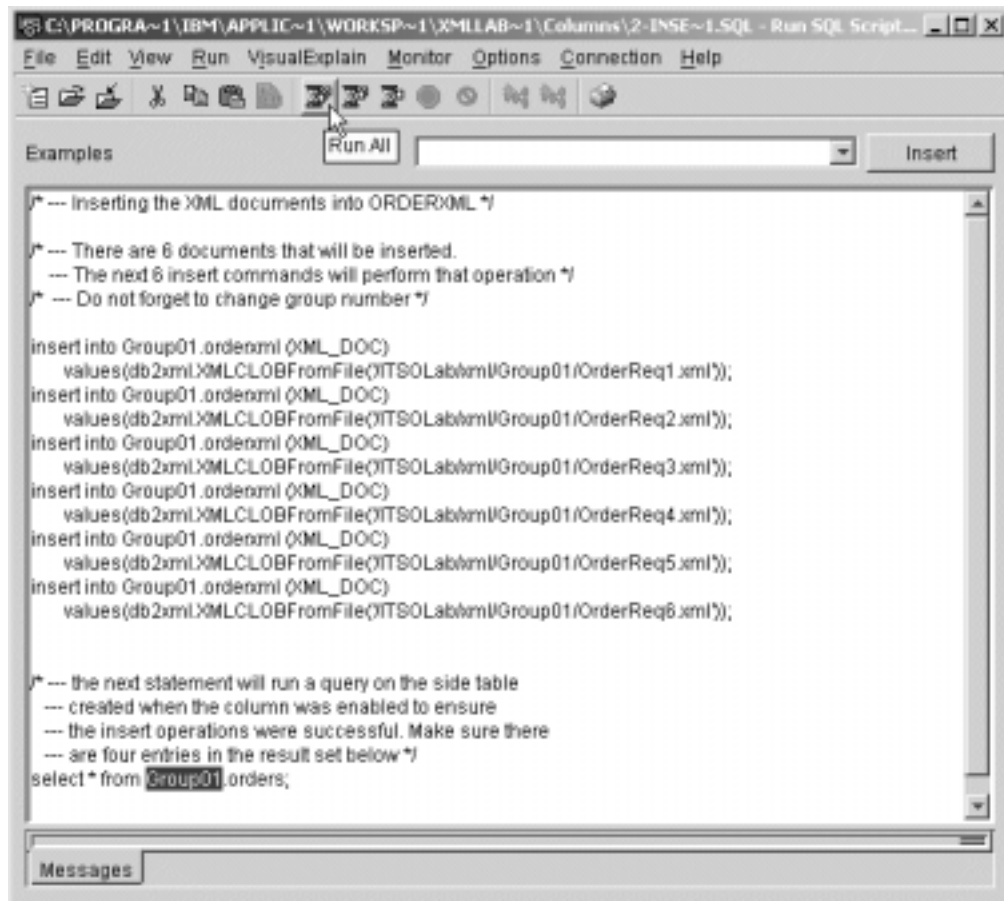


Figure 41. SQL Script to insert XML documents in the database table

There are two types of SQL statements in this script:

- The *Insert* statements to insert a row in the database table: Note that there is no reference to the side table. This is done by DB2 UDB XML Extender to preserve the integrity between XML documents and database rows.
- The *Select* statement at the end of the script used to verify that the six XML documents were accepted by DB2 UDB XML Extender. *All* columns of the table are selected in this statement.

Run the SQL Script using the **All** icon.

When the Select statement is executed, six are displayed as shown in Figure 42.

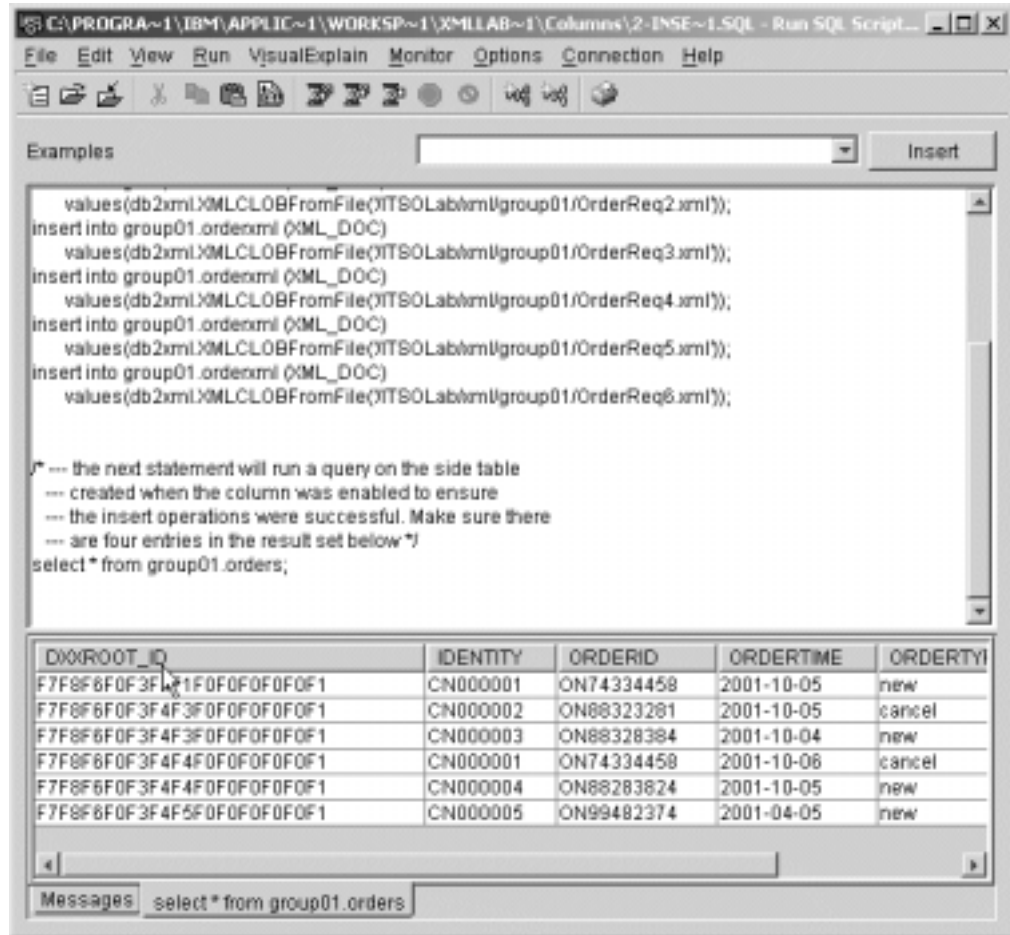


Figure 42. XML documents accepted by DB2 UDB XML Extender

- \_\_\_ 4. The Select statement displays all columns in the table. The DXXROOT\_ID column was created by DB2 UDB XML Extender and is used to correlate side tables entries with corresponding XML documents (used to generate side table content).

As XML documents are stored in the XML Column, DB2 UDB XML Extender generates a unique ID. The ID is stored in the DXXROOT\_ID column for use in queries that involve join operations between the XML Column in the table and its associated side tables.

Exit from the System Editor by clicking **File-> Exit**.

- \_\_\_ 5. Save the file when you are prompted, or use the save option before you exit from this system utility.

This task is now completed.

## Task 7: Running queries against XML Columns and a side table

DB2 UDB XML Extender provides a number of functions for extracting or updating individual elements and attribute values in an XML Column, even if the values are not stored in a side table.

In this task, you run a number of queries to demonstrate some of the accessing functions provided by DB2 UDB XML Extender for working with data in XML Columns:

- \_\_\_ 1. In the Navigator pane, expand the **Columns** folder and locate the **3-Query-Customer\_Names.sql** file.
- \_\_\_ 2. Open this file with the System Editor. Change references to your group number. Figure 43 shows the SQL Script changed for Group01.

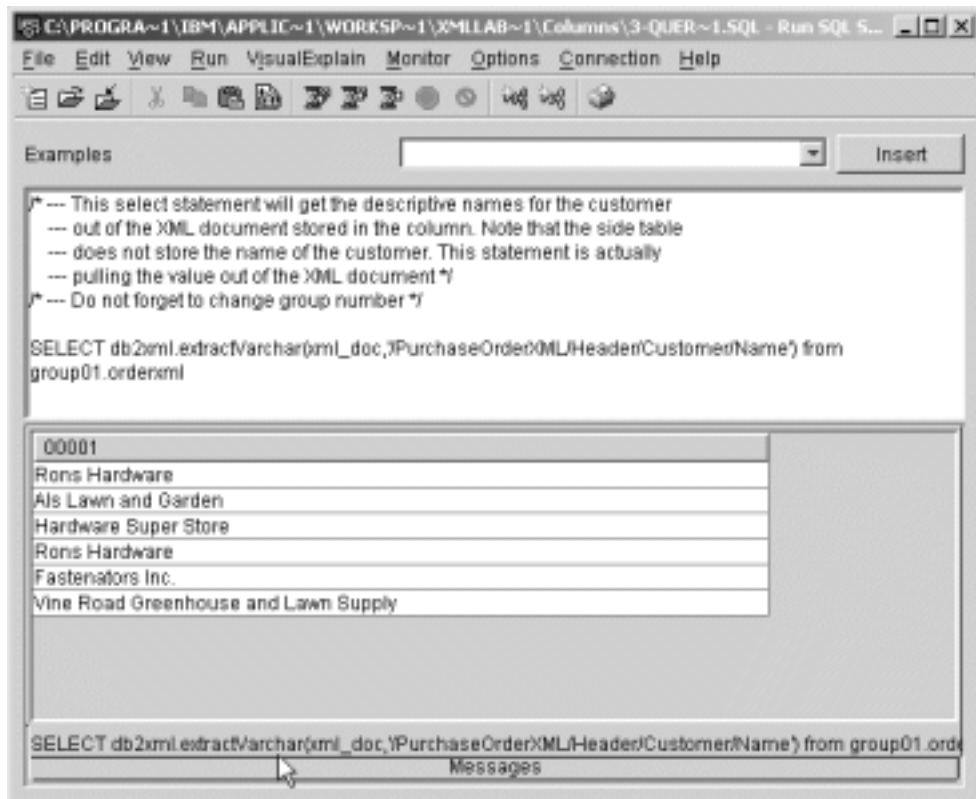


Figure 43. Querying the Customer Name in the orders

- \_\_\_ 3. Notice that this query uses the DB2 UDB XML Extender `extractVarchar` function. This function is used to extract individual elements or attribute values from data in an XML Column based on a location path that identifies the element or attribute of interest.

In this script, you extract the customer names from purchase orders stored in our XML Column. Also, note that the side table generated from our XML Column is not involved in this query.

Save your file changes and close the Run SQL Script window.

- \_\_\_ 4. Execute another script. Locate the **4-Query-cancelled.sql** file and open it using System Editor. Change the group number. An example of this query is shown in Figure 44.

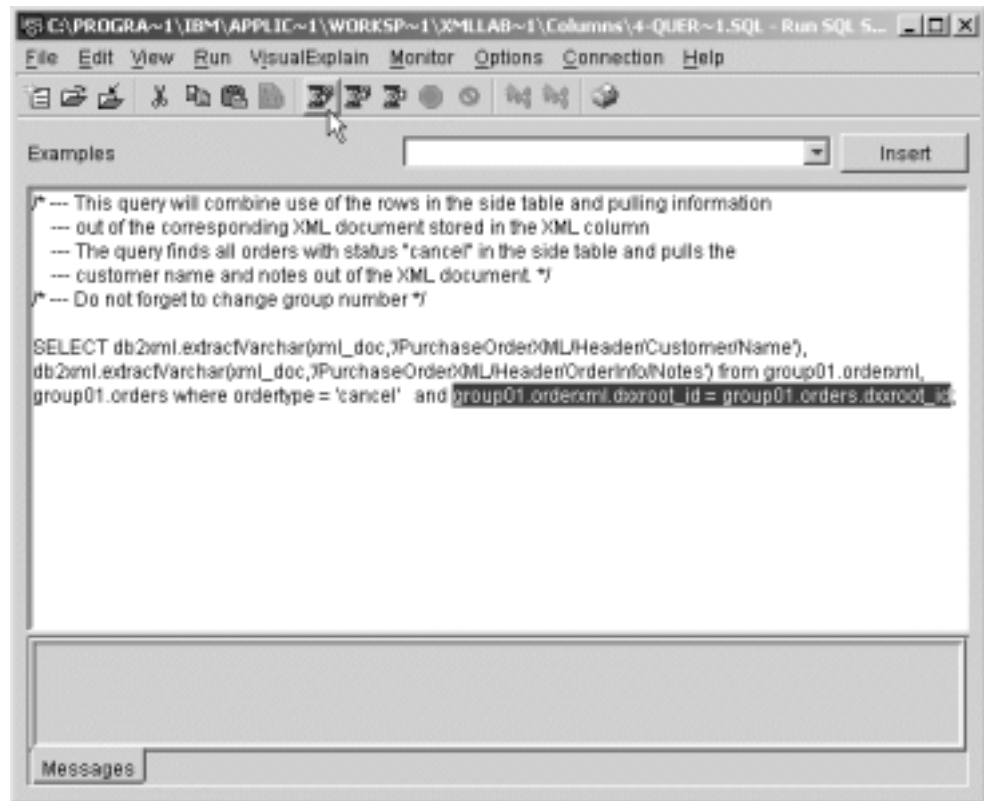


Figure 44. Query cancelled orders

- \_\_\_ 5. This query is designed to return the list of customer names and order notes for those orders that have been cancelled. The function `extractVarchar` is used to retrieve customer names and order notes. In this case though, the side table is used to select only those orders that have been canceled.

This query also demonstrates that DB2 UDB XML Extender generates a unique ID column for XML Columns that can be used to perform SQL join functions between the XML Column and its associated side table(s). This is done by the join condition:

```
group10.ordersxml.dxxroot_id = group10.orders.dxxroot_id
```

Save your file and close the Run SQL Script.

- \_\_\_ 6. Your last query for this task is to query all orders placed on a particular date. Locate the **5-Query-Orders-by-date.sql** file and open it with System Editor. An example of the code with the group number replaced is shown in Figure 45.

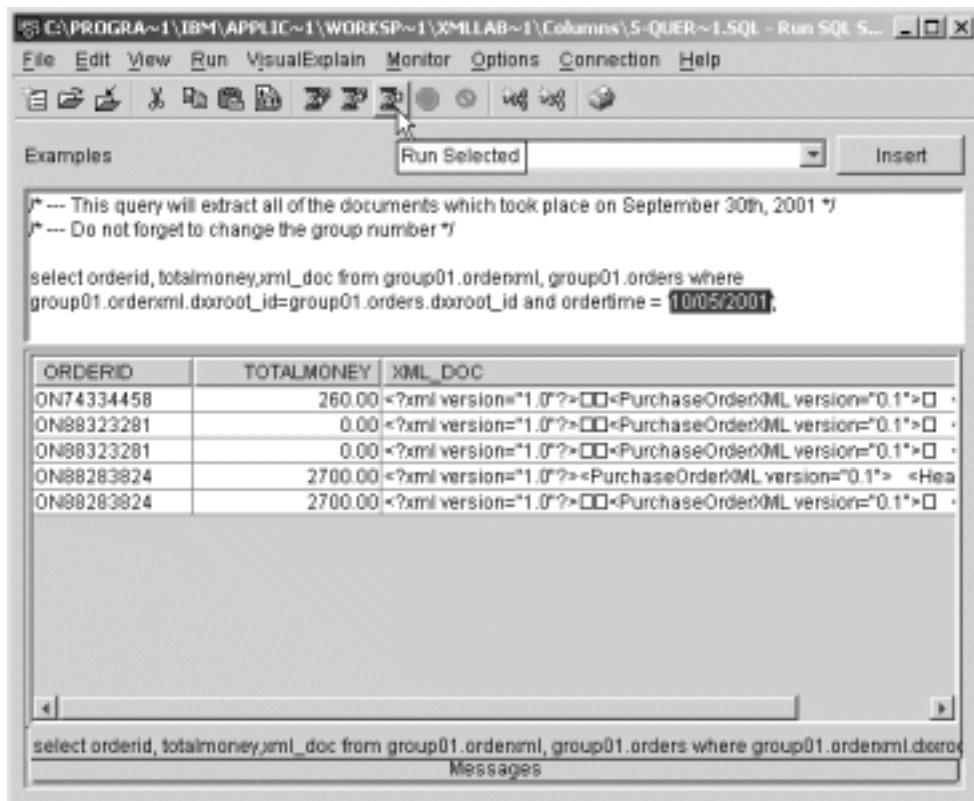


Figure 45. Query orders for a particular date

\_\_\_ 7. The *ordertime* column in the side table is used to select orders for a given date. Similar to the previous query, a join is performed between the side table and the XML Column table to select the purchase order documents matching the desired order date.

In this case, our result set includes the complete purchase order documents rather than selected fields from the order. Note the XML\_DOC column that contains the XML document.

When you finish reviewing this output, save your file and close the Run SQL Script.

\_\_\_ 8. Close the **Columns** folder.

You have now completed this task and Lab 2.

---

## Lab 3. Decomposing XML data using an XML Collection

Lab 2 explains how to store an XML document in a database column. XML Columns are a good option when you want to keep all the content of your documents. For example, you receive XML documents about products from different sources, but you are not interested in changing your current providers and want to keep all the information as it is.

When storing the documents in an XML Column, you feel comfortable because you know the information is in a safe place and available when you need it. Unlike XML Columns, *XML Collections* store XML data in database files as individual fields rather than as complete XML documents. Elements and attributes values from the XML document are mapped to columns in one or more database tables based on information specified in a Document Access Definitions (DAD) file.

XML Collections work with existing database tables. You map document information to your existing columns in tables. Be sure to use caution when mapping to the correct fields.

There are two forms of mapping XML data to database tables and columns that are supported:

- **Relational database (RDB) mapping:** Where the DAD file defines explicit relationships between an XML element or attribute to a database table or column
- **SQL mapping:** Where an SQL Query is used to extract information from one or more database tables, and the columns in the resulting table are mapped to an XML document

The RDB mapping technique can be used for both:

- **Decomposition:** XML document *to* the database
- **Composition:** XML document *from* the database

The SQL mapping technique can be only used for XML data composition from existing database information.

### Objectives

In this lab, you populate an existing database with information contained in an XML document. Our example lawn and garden center inventory database is updated from XML documents defining the set of new items received.

This lab demonstrates the support provided by XML Collections for decomposing XML data and storing it in one or more database tables.

By the end of this lab, you will be able to:

- Create a DAD file to map data from an XML part list document to an existing part information database.
- Use XML document decomposition facilities provided with XML Collections to extract part data from an XML document and store it in an existing database table, based on a specified DAD file.
- Identify with more accuracy when it is better to use XML Collections or XML Columns, as shown in the Lab 2.

## Lab prerequisites

You must have completed Lab 2, “Working with XML Columns” on page 17, before you start this lab. If you did not have time to complete the, you can read the material to learn the concepts introduced there. The most important part is in Task 2, “Creating your schema and database tables” on page 23. In this lab, the database tables used in the lab were created.

We recommend that you complete Lab 2, “Working with XML Columns” on page 17, before you begin this one.

## Time required

The time required to efficiently complete this lab is 30 minutes.

## Required information for working with the lab

The same requirements from the previous lab are valid here:

- You must assign drive “I” to the ITSOLab folder in the IFS.
- You must have the name of your server and be prepared to supply this information when prompted.

---

## Task 1: Preparing the WebSphere Studio Application Developer Workbench

### Before you begin

Ensure that you have observed the prerequisites specified for this lab and given consideration to the previous lab.

To do this lab, you do not need any of the files used in the previous lab. You can collapse the Columns folder in the Navigator pane and expand the folder that you need in this lab.

To prepare your workbench, follow these steps:

- \_\_\_ 1. Start WebSphere Studio Application Developer from the Windows desktop if it is not already open. Click **Start-> Programs-> IBM WebSphere Studio Application Developer-> IBM WebSphere Studio Application Developer**.
- \_\_\_ 2. WebSphere Studio Application Developer is recursive, if you exit from the product using the Close option, your configuration settings are preserved. Click the minus sign (-) to collapse the **Columns** folder as shown in Figure 46.



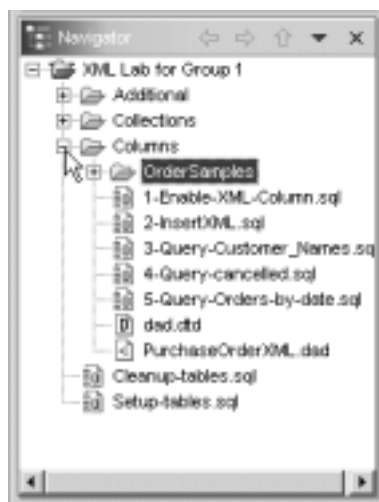


Figure 46. Collapsing the Columns folder

- \_\_\_ 3. Review the contents of the folder that you will use. Expand the **Collections** folder as shown in Figure 47.



Figure 47. Expanding the Collections folder

- \_\_\_ 4. From this list, you will work with all files associated with the part lists; the other file will be used in Lab 4, "Composing XML documents from existing data" on page 57.

This task is now completed.

---

## Task 2: Reviewing the information in the XML file

The XML document used for this lab has a more complex structure. In this task, you read the contents of the XML file:

- \_\_\_ 1. Locate the partslst.xml file and open it using the XML Editor. If this is the first time you open a file, XML Editor is the default. This means that you can select the **partslst.xml** file, right-click, and click **Open** as shown in Figure 48.

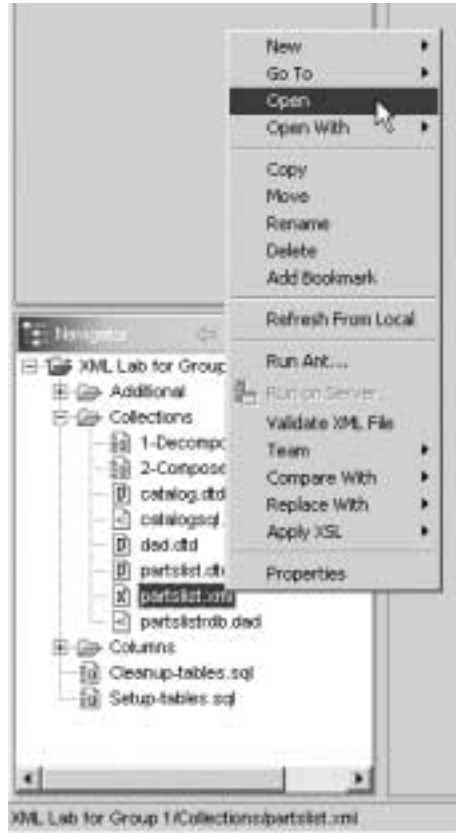


Figure 48. Opening a file using the default open option

2. When the file is loaded, the Editor and Outline panes are updated. First, you work with the Editor pane. Verify that the **Source** tab (at the bottom) is selected. You see the document content as shown in Figure 49, and tokens with colors for each component.

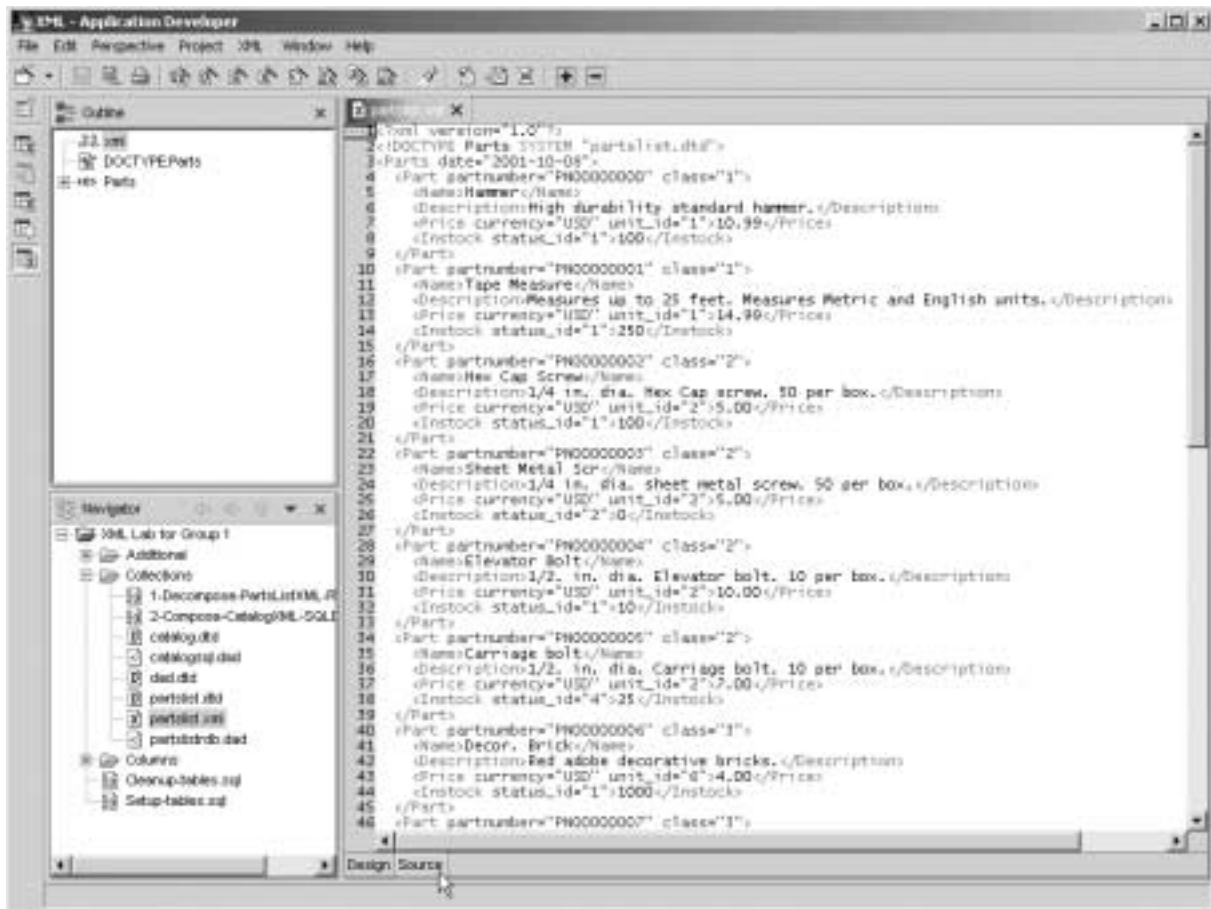


Figure 49. Source view of an XML document

- \_\_\_ 3. Skip the first two lines. Position your cursor on the third line (any column)  
 Notice that from line 3 to the end, the number of lines (left column) are shaded. This is to highlight the entire structure of the *Parts* document referred in the second line. Parts have many Part records. Parts also have tags that define each of the elements.  
  
 All the elements are in the color cyan on your screen. The attributes of each element are in magenta. And the data, is in black.  
  
 Position the cursor at **line 4**. Did you note the change? The line numbers are shaded from **line 4 to line 9**. This is done because the Part's boundaries (**<Part>** and **</Part>**).
- \_\_\_ 4. Do you want a better approach to read your data? Try the **Design** tab.  
 Figure 50 shows you the same document with all elements expanded for the first record of the document.



- \_\_\_ 7. Do not close the partslist.xml file. In line 2, you can read a reference to the partslist.dtd file. What is a DTD?

This task is now completed.

### Task 3: Reviewing the associated DTD file

A Document Type Declarations (DTD) file contains information about the elements and attributes that the XML file have.

When you decompose an XML document, you need to have a DTD to tell to DB2 UDB XML Extender how to format your data. In this task, you review the information in the DTD file for your XML document:

- \_\_\_ 1. From the Navigator pane, double-click the **partslist.dtd** file to open it.
- \_\_\_ 2. Verify that the Editor and Outline pane was updated.
- \_\_\_ 3. Change the view of your Editor. Click the **Source** tab (at the bottom). You see the structure of your document. Figure 52 illustrates a source view of a DTD file.

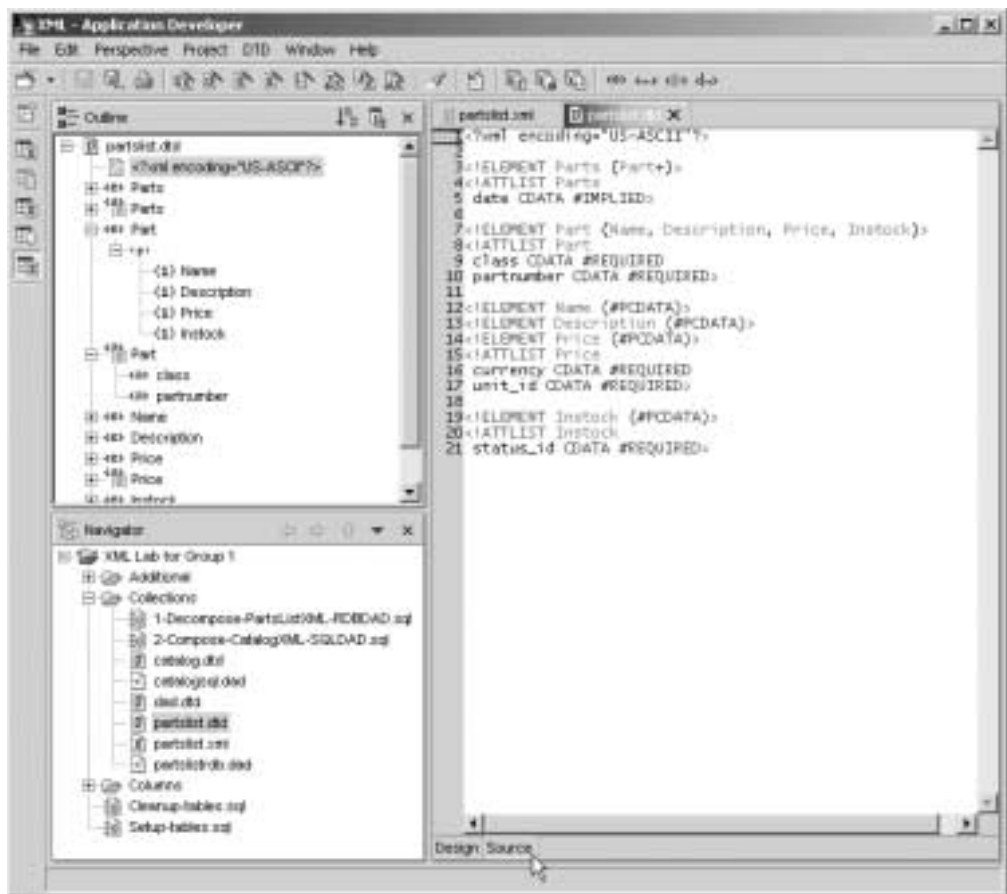


Figure 52. Source view of a DTD file

- \_\_\_ 4. WebSphere Studio Application Developer use the Outline pane to view the indentation of an XML file.

- 5. Go to the Outline pane and verify that the elements (letter “e” in blue color) and attributes (letter “a” in green color) are there. Figure 53 shows an example of how the Part element is created.

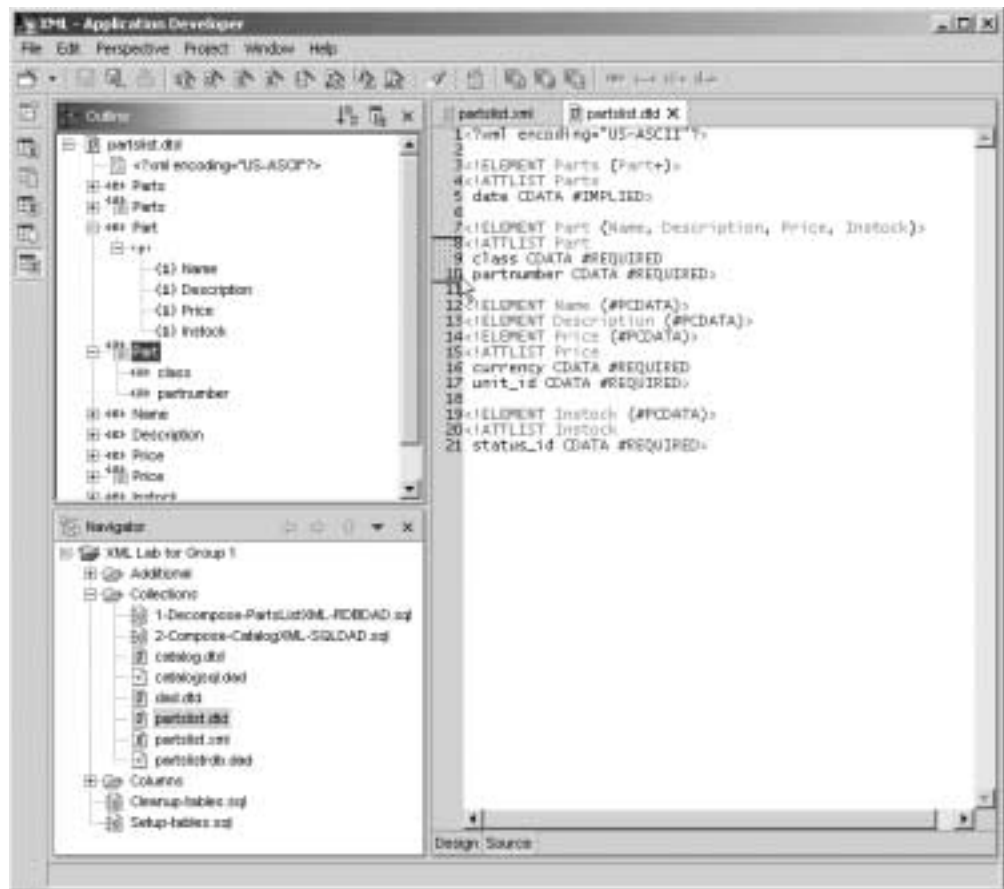


Figure 53. Outline pane for the DTD file

- 6. When you click an element or structure in the Outline pane, the corresponding line in the Editor is updated with the line numbers shaded. Click the date attribute and line 5 is shaded.

What do these keywords mean in this definition? If you change the Editor to the Design view, you have a great support. See Figure 54.

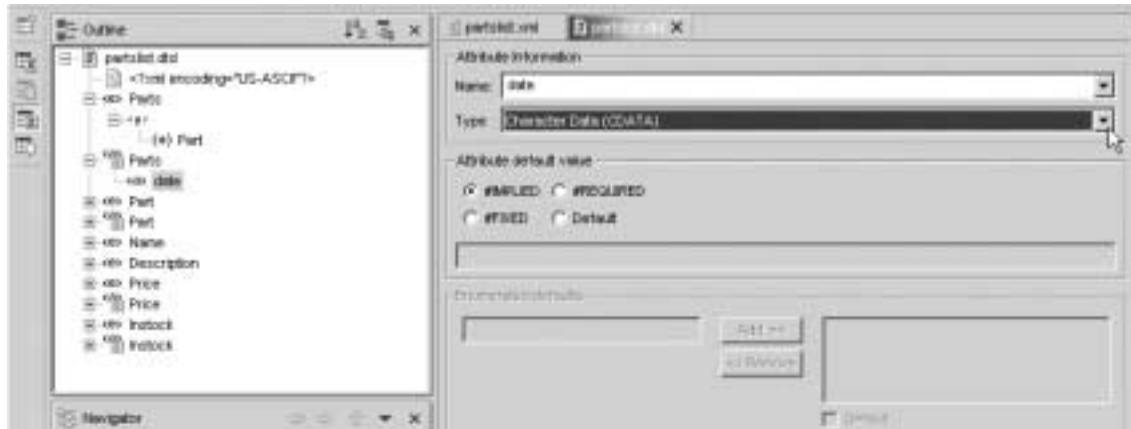


Figure 54. Prompt for definitions

\_\_ 7. Leave this file open.

This task is now completed.

---

#### Task 4: Reviewing the associated DAD file

You already know the information that comes in the XML file and the type of data in the document. Now you need to tell DB2 UDB XML Extender where to put this information.

When a document is decomposed, you can use as many tables as needed. In this lab, you decompose the XML document in one database table only. As we discussed earlier, you are responsible for the database table creation and the columns and their attributes. See Task 2, "Creating your schema and database tables" on page 23, in Lab 2, if you want to learn more about the structure of the *parts* database.

In this task, you work with the DAD file to provide information about how the information will be transformed to the database table:

- \_\_ 1. In the Navigator pane, locate and double-click the **partslistrdb.dad** file to open it.
- \_\_ 2. At the bottom of the Editor pane, click the **Source** tab to change to the source view of your code. To see the structure of the DAD file, you can see similar results using the Outline pane.
- \_\_ 3. Using the Outline pane, expand some level of the code. As soon as you expand a level, the corresponding details expand in the Editor pane.
- \_\_ 4. When you are creating a DAD file, the most important part is to define the attributes for each element. The Design view of the file allows you to do this. Click the **Design** tab (at the bottom of Figure 55) if necessary.

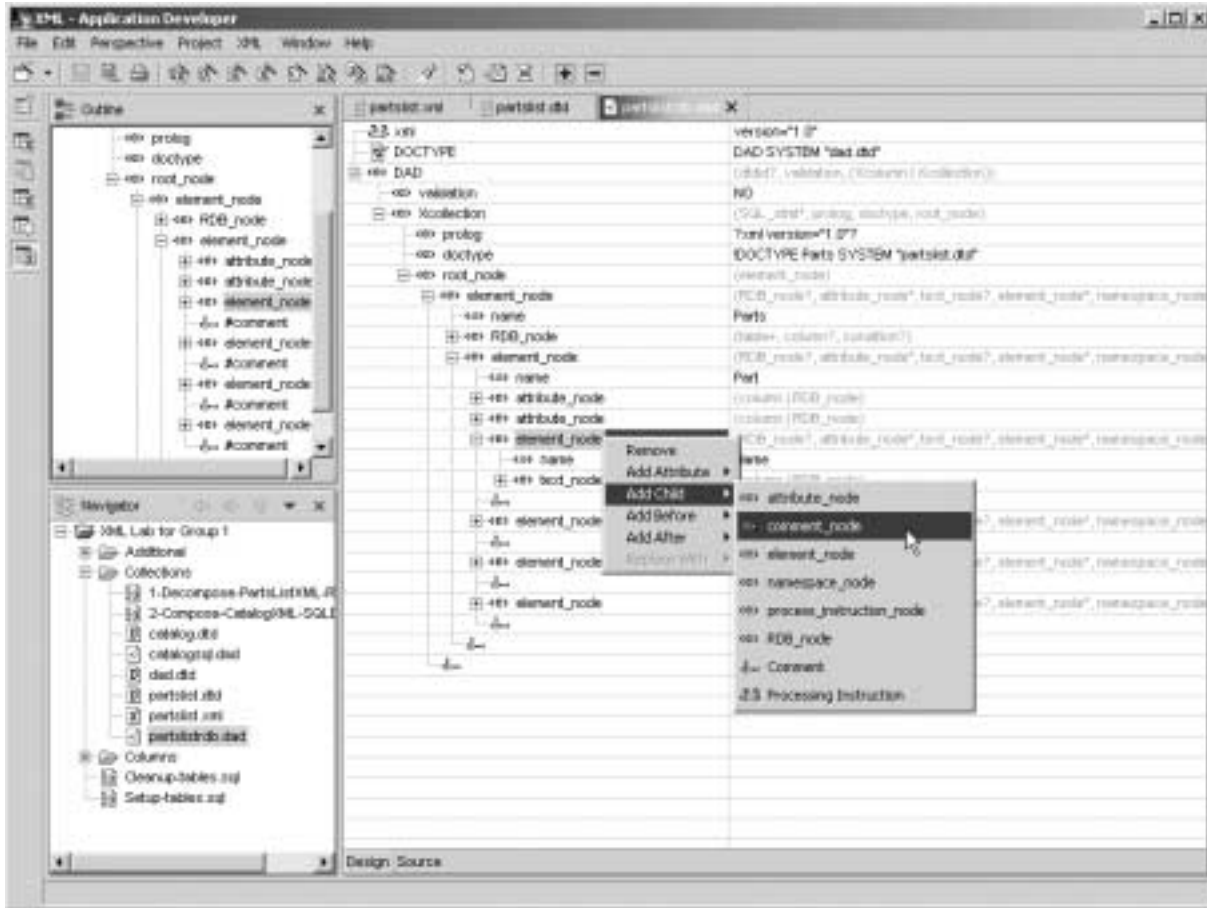


Figure 55. Adding attributes

- \_\_\_ 5. Make some changes to this DAD file. Specify where your columns and database table are located. Be sure to change the group number in your schema.
- \_\_\_ 6. Locate the references for the table name. Change the table name and the SQL statement that selects all rows. There are two lines in total.
- \_\_\_ 7. Change the references for the schema for the columns of the table. You need to change nine columns. You can use the Find & Change support for additional help. Press Ctrl+F in your keyword and a dialog box opens. See Figure 56 for details.





Figure 56. Searching and replacing the schema name

- \_\_\_ 8. Click the **Replace All** button.
- \_\_\_ 9. From the menu bar, click **File-> Save** to save the file.
- \_\_\_ 10. Close this window when you are finished but leave this file open.

This task is now completed.

## Task 5: Working with the three related files

The objective of this task is to summarize what you have learned. To decompose an XML file, you need to have a DTD file where elements and attributes are defined. And you need to have a DAD to be used by DB2 UDB XML Extender to transform the information from an XML file to database table.

A common problem when decomposing an XML document is when the DAD is not correctly coded, especially with numeric columns. Check all information related to the price of a part item and see how to define a numeric data. This data is transferred to the Price database column, which is defined as decimal.

After you verify that all references to the schema name have been replaced to use the schema for your group, send these files to the IFS:

- \_\_\_ 1. Close the three files currently open, using the **Close** button, as shown in Figure 57.



Figure 57. Closing files using the Close button

- \_\_\_ 2. Go to the Navigator pane (Figure 58). Click the **partslist.dtd** file. Then hold the shift key and click the **partslistrdb.dad** file to select the three files as a group.



Figure 58. Selecting the files to be transferred

- \_\_\_ 3. Use the Export function of WebSphere Studio Application Developer to transfer the files to the IFS.

This task is now completed.

---

## Task 6: Decomposing the XML file

To decompose an XML document, use a stored procedure shipped with DB2 UDB XML Extender. This program, calls the `dxxShredXML()` function to perform the decomposition and storage of the resulting data in the database table.

Input to the `dxxSHredXML()` function includes the DAD file defining the XML Collection and the XML document to be decomposed.

You need to run the decomposition process in the server:

- \_\_\_ 1. Go to the Navigator pane, locate the **1-Decompose-PartsListXML-RDBDAD.sql** file, and open it using the System Editor to invoke the Run SQL Script.
- \_\_\_ 2. Change the references to your group number.
- \_\_\_ 3. This script file has two SQL sentences:
  - **Function `dxxShredXML()`:** A *call* to program `shdx` that uses the `dxxShredXML()` function. This program needs the name of the relational database as a first parameter. The second parameter is the name of the DAD file. And the third parameter is the name of the XML file.
  - **Select statement:** The second statement is a *Select* statement to verify that DB2 UDB XML Extender has accepted your XML file.

Figure 59 shows you a copy of the SQL Script file.

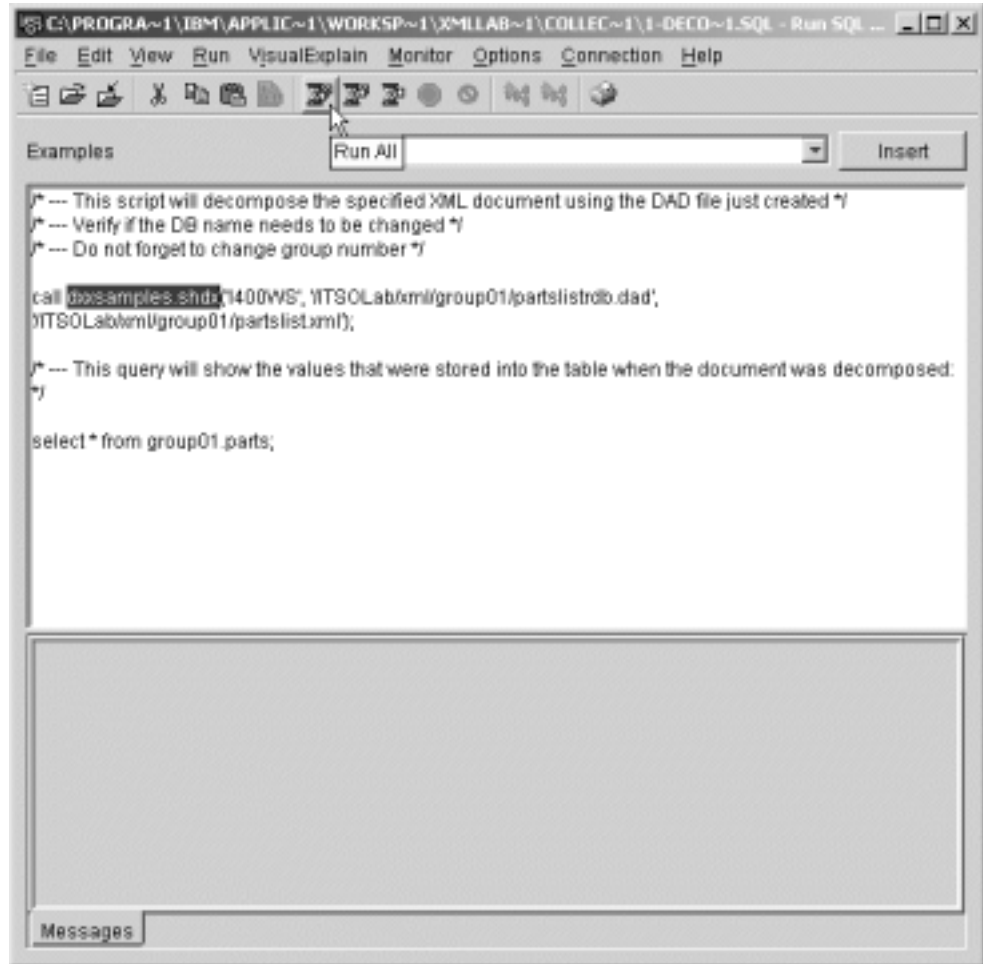


Figure 59. SQL Script to decompose a document

- \_\_\_ 4. Save the file. Click **File-> Save** or press Ctrl+S.
- \_\_\_ 5. Run your SQL Script. Figure 60 shows the first rows in your part file.

COMPANY	PARTNUMBER	NAME	DESCRIPTION
1	PN00000000	Hammer	High durability standard hammer.
1	PN00000001	Tape Measure	Measures up to 25 feet. Measures Me
1	PN00000002	Hex Cap Screw	1/4 in. dia. Hex Cap screw. 50 per box
1	PN00000003	Sheet Metal Scr	1/4 in. dia. sheet metal screw. 50 per
1	PN00000004	Elevator Bolt	1/2. in. dia. Elevator bolt. 10 per box.
1	PN00000005	Carriage bolt	1/2. in. dia. Carriage bolt. 10 per box.
1	PN00000006	Decor. Brick	Red adobe decorative bricks.
1	PN00000007	Sod-midgrade	Mid-grade sod 1 yard wide.
1	PN00000008	Sod-highgrade	High-grade sod 1 yard wide.
1	PN00000009	Garden Hose	Rubber garden hose. Ends sold sepa
1	PN00000010	Grass Seed	25lb grass seed
1	PN00000011	1 in. Conduit	1 in. dia. electrical conduit
1	PN00000012	Copper Wire	House-wiring grade copper wire. Insu
1	PN00000013	Elec. Box	Standard Size 2-outlet electrical wall k
1	PN00000014	Peatling Soil	6lb. cleaned peatling soil

Messages select \* from group01.parts

Figure 60. First records selected from the database table

- \_\_\_ 6. If you do not see any rows after the select statement runs, advise your instructor or system administrator. These records must be selected.
- \_\_\_ 7. Close the SQL Script windows to return to WebSphere Studio Application Developer.

This task is now completed.

---

### **Task 7: Running queries against XML Columns and side table (optional task)**

The DB2 UDB XML Extender has transferred all the information to databases tables. This means that you can execute any SQL statement you want. Using the Run SQL Script from iSeries Navigator is a good interface to the server. Look at the content of PARTS table in your GROUPxx collection. It should contain the elements of the XML document.

Take some time to work with this database, but do not forget you need to do another lab.

You have now completed this lab!

---

## Lab 4. Composing XML documents from existing data

As we explain in Lab 3, “Decomposing XML data using an XML Collection” on page 43, XML Collections decompose and compose XML documents into and from DB2 UDB for iSeries. Each element and attribute value in the XML document is mapped to columns in one or more database tables or vice versa.

*Decomposition* refers to the act of storing XML data across one or more database tables. *Composition* refers to generation of an XML document from existing database tables.

RDB mapping and SQL mapping represent two techniques for composing XML documents. Both describe the mapping process via a Document Access Definitions (DAD) file used by DB2 UDB XML Extender at runtime to perform the composition function. For SQL Mapping, the DAD file contains an SQL query that is used to extract information from one or more database tables.

The result set of this query is then mapped to individual XML document elements and attributes via additional instructions in the DAD file. Any number of DAD files may be used with a given database, enabling multiple types of XML documents to be generated from the same database schema.

### Objectives

The XML Collection feature of DB2 UDB XML Extender is used for this lab. In this case, you compose a new XML document from database tables.

To achieve these objectives, you will:

- Create a DAD file to specify how to generate an XML document from existing database tables using the SQL Mapping technique of XML document composition supported by DB2 UDB XML Extender
- Use the DAD file and XML document composition facilities provided with XML Collections to generate an XML product catalog from a part inventory database

### Lab prerequisites

This lab builds on the results of Lab 3, using information from the parts database to generate an XML-based catalog of the items available for sale from our example lawn and garden center.

You must complete Lab 3 before start this lab.

### Time required

The time required to efficiently complete this lab is 25 minutes.

### Required information for working with the lab

The same requirements from the previous lab are valid here:

- You must assign drive “I” to the ITSOLab folder in the IFS.
- You must have the name of your server and be prepared to supply this information when prompted.

---

## Task 1: Preparing the WebSphere Studio Application Developer Workbench

### Before you begin

Ensure that you have observed the prerequisites specified for this lab and given consideration to the previous lab.

In this lab, you continue to use the Collections folder. Your work is to create a catalog of parts available in your database table. This folder has the necessary files that you need.

To prepare your workbench, follow these steps:

- \_\_\_ 1. Start WebSphere Studio Application Developer from the Windows desktop if it is not already open. Click **Start-> Programs-> IBM WebSphere Studio Application Developer-> IBM WebSphere Studio Application Developer**.
- \_\_\_ 2. Expand the **Collections** folder in the Navigator pane to open it.
- \_\_\_ 3. Verify that you have two files related to XML for your parts catalog (DTD and DAD) and an SQL Script to invoke DB2 UDB XML Extender functions as shown in Figure 61.



Figure 61. Contents of the Collections folder

This task is now completed.

---

## Task 2: Reviewing the information in DTD file

Unlike working with columns, when you are decomposing or composing a document, a DTD file is necessary. In this task, you review the DTD file that is necessary to compose a document:

- \_\_\_ 1. Locate the **catalog.dtd** file and open it using the XML Editor or DTD Editor.
- \_\_\_ 2. Click the Source tab at the bottom of the Editor pane to use the Source view as shown in Figure 62.

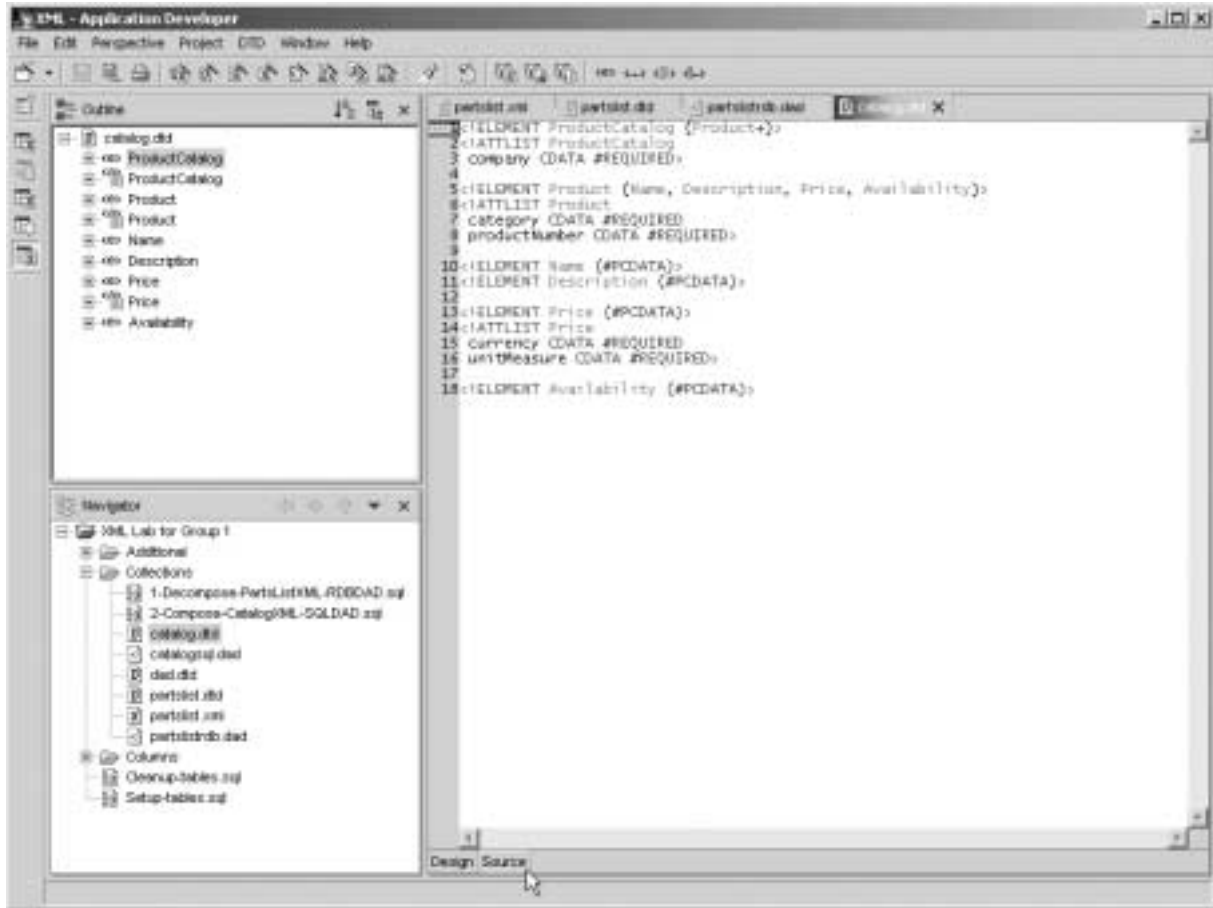


Figure 62. Source view of the DTD document

In examining this view, you can see that the DTD files are very similar. A DTD file shows the dependencies of the information. An element has attributes and sometimes it has a list of attributes.

\_\_\_ 3. Close this file.

This task is now completed.

### Task 3: Reviewing the DAD file

This DAD file is simple. It contains the information you want to have in the XML file. You can find relevant information about this file if you read the file's code:

\_\_\_ 1. From the Navigator pane, double-click the **catalogsql.dad** to open it.

The Editor and Outline panes are updated.

\_\_\_ 2. Go to the Editor pane and select the **Source** view for your source. You see a copy of the file as shown in Figure 63.

```

1 <?xml version="1.0"?>
2 <!DOCTYPE DAD SYSTEM "dad.dtd">
3 <DAD>
4   <validation>ND</validation>
5   <xcollection>
6     <SQL_stmt>SELECT COMPANY, CLASSDESC, PARTNUMBER, NAME, DESCRIPTION,
7     </prolog>?xml version="1.0"?</prolog>
8     <doctype>!DOCTYPE ProductCatalog SYSTEM "catalog.dtd"</doctype>
9     <root_node>
10      <element_node name="ProductCatalog">
11        <attribute_node name="company">
12          <column name="COMPANY" />
13        </attribute_node>
14        <element_node name="Product">
15          <attribute_node name="productNumber">
16            <column name="PARTNUMBER" />
17          </attribute_node>
18          <attribute_node name="category">
19            <column name="CLASSDESC" />
20          </attribute_node>
21          <element_node name="Name">
22            <text_node>
23              <column name="NAME" />
24            </text_node>
25          </element_node>
26          <element_node name="Description">
27            <text_node>
28              <column name="DESCRIPTION" />
29            </text_node>
30          </element_node>
31          <element_node name="Price">
32            <attribute_node name="currency">
33              <column name="CURRENCY" />
34            </attribute_node>
35            <attribute_node name="unitMeasure">
36              <column name="UNITDESC" />
37            </attribute_node>
38            <text_node>
39              <column name="PRICE" />
40            </text_node>
41          </element_node>
42          <element_node name="Availability">
43            <text_node>
44              <column name="STATUSTEXT" />
45            </text_node>

```

Figure 63. Source view of the DAD file

- \_\_\_ 3. Verify that you have a **<XCollection>** element. This element indicates that this is a DAD file for an XML Collection.

In line 6, there is an **<SQL\_stmt>** element. This element identifies the SQL query that will be executed to extract information from your database that is used to populate the XML document generated via this DAD file.

This SQL query has been provided for you. It extracts the list of parts from a parts table and does a join to resolve identifiers for unit of measure, part status, and part classification to human readable terms.

- \_\_\_ 4. Using the Editor's search/replace function or any method to advance on this line, change the references for your group number in the SQL query. Four tables are used: PARTS, UNITS, STATUS, and CLASSES. All of them reside in the same schema. Change to use the correct *schema* name for all of them.

- **<doctype>**: Identifies the DTD associated with the XML document.
- **<root\_node>**: Defines the root of the XML document that will be generated.



- **<element node>**: The first one defines the outermost element in the target XML document. It is named *ProductCatalog*.
- **<attribute\_node>**: The initial element defines an attribute for *ProductCatalog* named *company*. The **<column>** element associated with **<attribute\_node>** indicates that the value for this attribute comes from the *COMPANY* column of the result set generated by the SQL query statement.

The remaining **<element\_node>** and **<attribute\_node>** items define the remaining elements and attributes in our XML document.

5. There are many elements, attributes, and columns in this DAD file. Click the **Design** tab at the bottom of the pane to change to the Design view of the Editor (Figure 64).

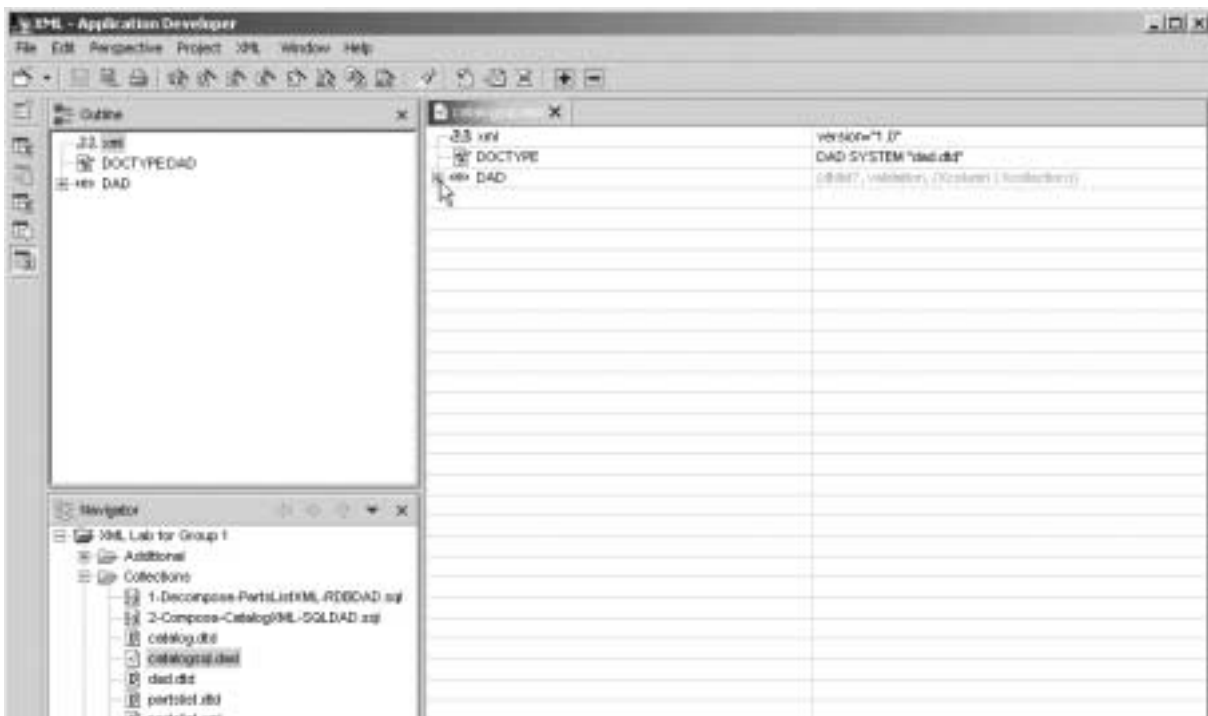


Figure 64. Design view (collapsed) of the DAD file

6. Click the **Expand All** button at the top of the Editor pane (Figure 65) to expand all elements and attributes of your file. This button is available only if you are working in the Editor pane.

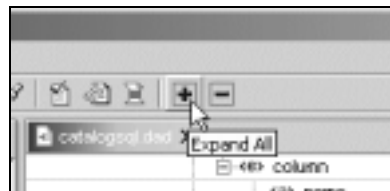


Figure 65. Using the Expand All button

Now you can see all statements as shown in Figure 66.



Figure 66. Design view (expanded) of the DAD file

\_\_\_ 7. To read the DAD file, locate the **<Attribute\_node>** that has an attribute of **name** equal to **productNumber**. This is the attribute of the *Product* element.

\_\_\_ 8. Review other elements and their attributes.

\_\_\_ 9. When you finish reviewing the components of this DAD file, close it.

This task is now completed.

## Task 4: Transferring the files before composition

### You must verify...

... that you have replaced the name of the schema.

You need to use the schema name for your group. Step 6 in Task 3 is where you replaced the name for your schema in the SQL statement to select the record to build the XML file.

Your iSeries server needs to have the DTD and DAD files in the IFS before you can do any composition from your database table. In this task, you transfer these files:

1. On the Navigator pane, expand the **Collections** folder. Click the **catalog.dtd** file. Then hold down the Shift key and click the **catalogsql.dad** file to select the files as a group (Figure 67).

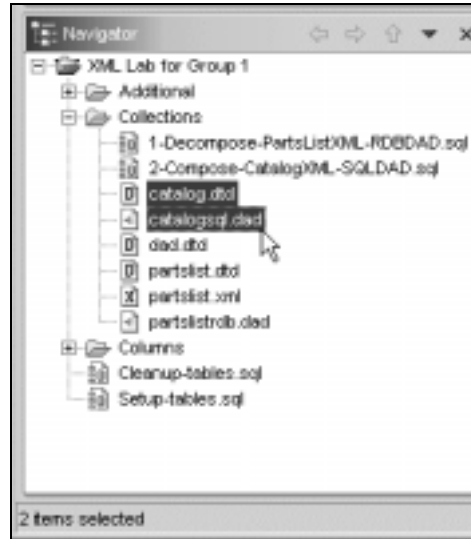


Figure 67. Selecting the files to export to the IFS

2. Use the **Export** function to send these files to the IFS.

This task is now completed.

---

## Task 5: Composing XML documents using the SQL Mapping DAD file

The Composition process is run under the control of DB2 UDB XML Extender. It uses the `dxxGenXML()` function to compose an XML document. The XML document is created inside the QSYS.LIB.

This XML file needs to be transferred to a PC-like file system to be used by a Web browser or PC-like software with attributes to process XML files.

Follow these instructions to compose an XML document from an existing database table:

1. On the Navigator pane, open the **2-Compose-CatalogXML-SQLDAD.sql** file using the System Editor. Remember that this option invokes the Run SQL Script function of iSeries Navigator.
2. Replace all references for the schema name and the IFS folder to your group number. Take the time to verify that you replace the correct values. If you prefer, use the Replace function provided in the Run SQL Script support. A copy of the script using group01 is shown in Figure 68.

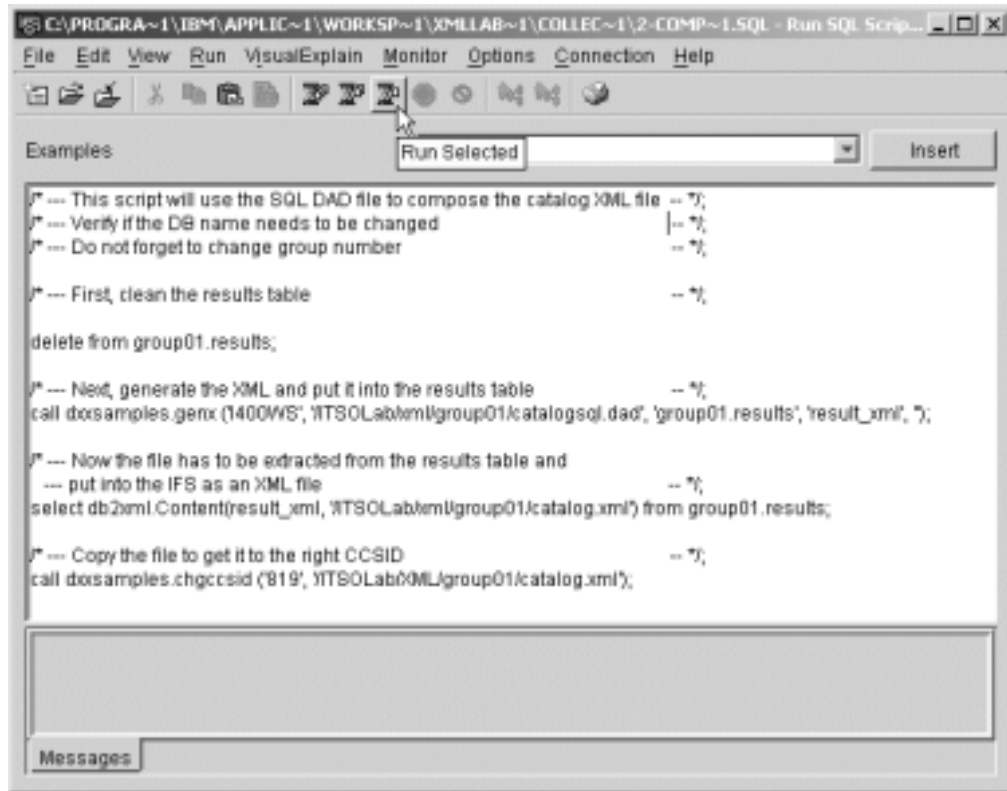


Figure 68. Script to Compose an XML document

- \_\_\_ 3. This script is prepared in a different way. Take note because you need to run it statement by statement.
- \_\_\_ 4. The first executable statement is a *delete* statement. Regardless of whether you are running this lab for the first time, make sure you perform this step. Position the cursor in the line where the delete statement is (any position) and click the **Run Selected** icon as shown in Figure 69.

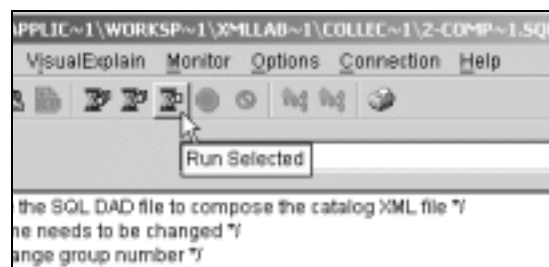


Figure 69. Running selected statements

- \_\_\_ 5. The next statement is a *call* statement. A call to a program is already supplied for you – *dxxsamples.genx*. This program uses the *dxxGenXML()* function to compose the XML document. The output table name is RESULTS. Click the **Run Selected** icon to run this statement to generate the RESULTS table in the OS/400 file system.

Your file is ready to be transferred to your IFS folder using a program supplied by UDB DB2 XML Extender.

- \_\_\_ 6. The next statement is a call to the program *Content*. The purpose of this program is to extract the information from the database table and write this information as an XML file in the IFS. Click the **Run Selected** icon to run this statement to generate XML file.
- \_\_\_ 7. You have stored the file in the IFS. You must make this file readable for PC software before it can be viewed. Click the **Run Selected** icon to execute the last statement to change the CCSID to the XML file.
- \_\_\_ 8. Close the Run SQL Script.

This task is now completed.

---

## Task 6: Viewing the XML file

Your XML document is now ready. Using Microsoft Internet Explorer Web browser, you can see the file directly from the IFS because a drive is mapped:

- \_\_\_ 1. From the Windows desktop, click **Start-> Run....**
- \_\_\_ 2. Type the name of the Explorer executable file in the Open field, which is `iexplore.exe` in our example (Figure 70). The extension is not necessary but, your PC may have a *bat* file. A *bat* file is selected first. When you write the extension `exe`, you are executing exactly the Explorer Web browser. Note the name of the `.exe` file.

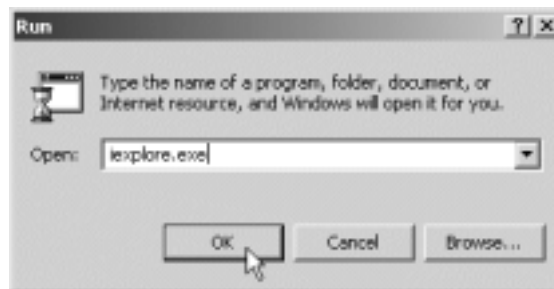


Figure 70. Running Internet Explorer

- \_\_\_ 3. Click the **OK** button.
- \_\_\_ 4. When the browser is loaded, open your XML file from the IFS. To do this, write the address for the file: `i:\xml\group01\catalog.xml` in the Address field of your browser (Figure 71).

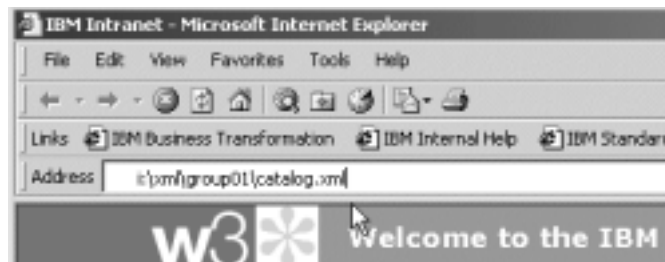


Figure 71. Opening the XML file from the IFS

- \_\_\_ 5. Your file is loaded and the browser uses its own viewer to format the output (Figure 72).

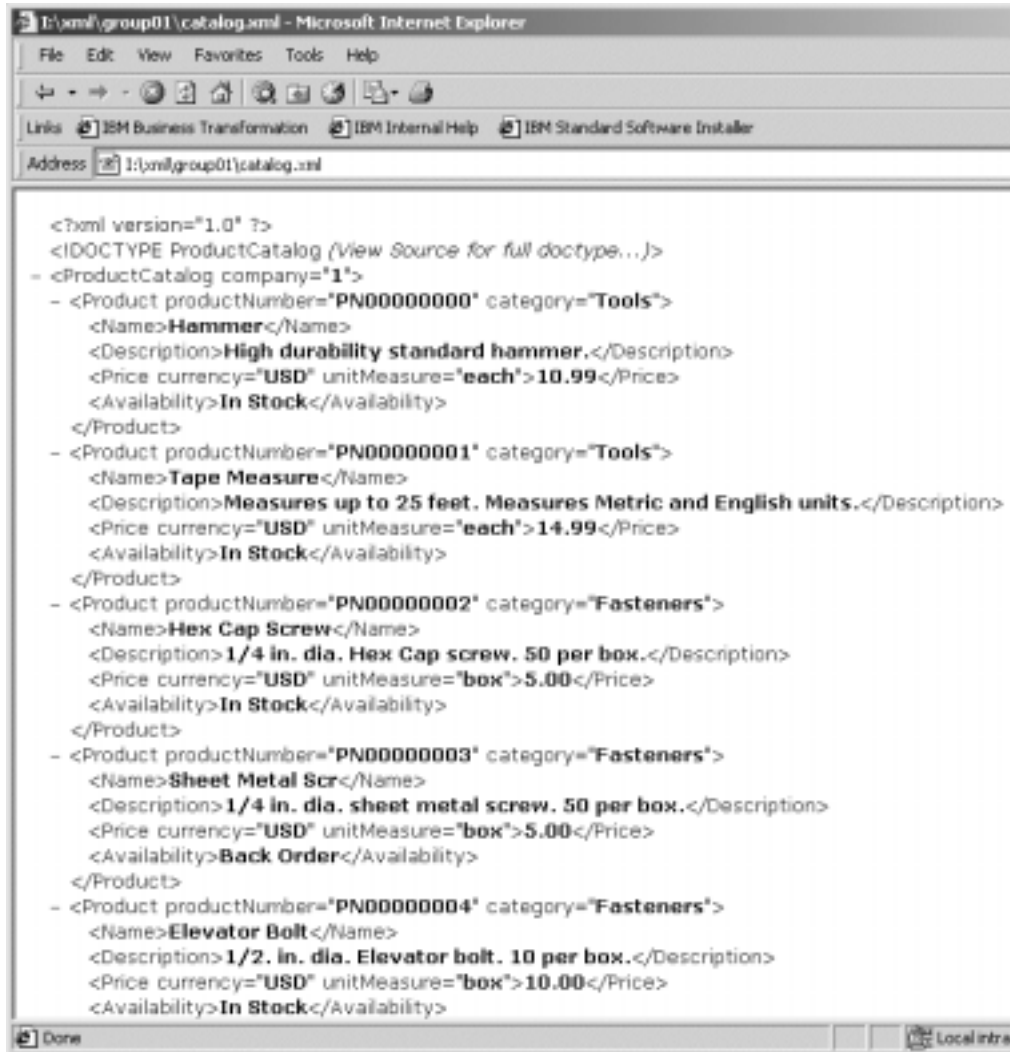


Figure 72. Web browser display

You have now completed this lab.

---

## Lab 5. Additional exercises to work with XML documents

This lab contains additional exercises to show you some of the more advanced features of the DB2 UDB XML Extender product. Each task is independent of the other. You may choose to work with the ones that apply more to your business.

### Objectives

This lab uses some of the functions of the DB2 UDB XML Extender product. They have been used to create some programs that will use some of the SQL Scripts provided for you.

To achieve these objectives, you will:

- Learn to update individual elements or attribute values in an XML Column
- Use SQL in a more efficient way to deal with the data to compose an XML document
- Use the DTD reference table maintained by DB2 UDB XML Extender
- Learn how to use a DAD file to compose and decompose an XML document

### Lab prerequisites

This lab builds on the results of Lab 4, “Composing XML documents from existing data” on page 57. You must complete Lab 4 before you start this one.

### Time required

The time required to efficiently complete this lab is 40 minutes.

### Required information for working with the lab

The same requirements from the previous lab are valid here:

- You must assign drive “I” to the ITSOLab folder in the IFS.
- You must have the name of your server and be prepared to supply this information when prompted.
- You are working with SQL scripts and some of them have many statements. Examine each statement and execute them one-by-one using the **Run Selected** icon in the Run SQL Script support of iSeries Navigator.

---

## Task 1: Updating elements or data attributes in an XML Column

In this task, you learn how to use the DB2 UDB XML Extender function to update individual elements or attribute values associated with data in an XML Column. Instead of replacing the entire XML document to update a particular element value, DB2 UDB XML Extender allows updating of individual items in the document.

In addition, the updates to the XML Column items that are mapped to a side table will cause the corresponding side table rows to also update.

To prepare your workbench, follow these steps:

- \_\_\_ 1. Start WebSphere Studio Application Developer from the Windows desktop if it is not already open. Click **Start-> Programs-> IBM WebSphere Studio Application Developer-> IBM WebSphere Studio Application Developer**.
- \_\_\_ 2. Expand the **Additional** folder.
- \_\_\_ 3. Locate the **Update-XML-Column.sql** file and open it using the System Editor. The Run SQL Script window opens.
- \_\_\_ 4. This script shows you some records before and after the update. You have to change the properties for this SQL session to open a new window each time a select statement is executed. Click **Options-> Display Results in Separate Window** as shown in Figure 73.

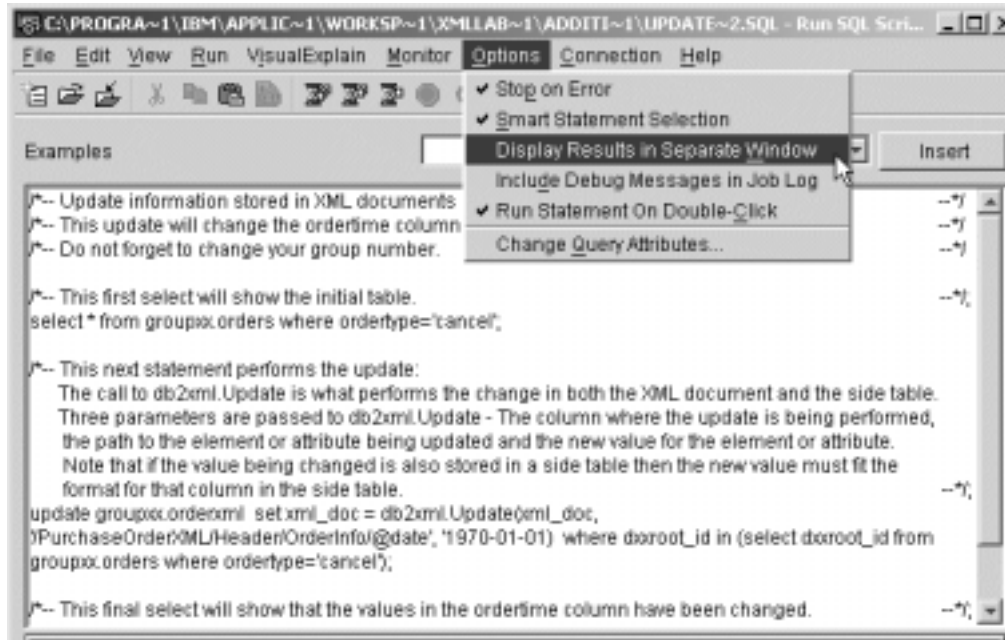


Figure 73. Changing the options for this session

- \_\_\_ 5. In the script file, change the references for your group number before any statement is executed.
- \_\_\_ 6. You can run this script with the **Run All** icon, but this time click the **Run Selected** icon to execute statement by statement to make it easier to see what is happening.
- \_\_\_ 7. The first statement is a *select* statement. See Figure 73. After you run this statement, you can see the orders. Move the horizontal bar to read the values of column ORDERTIME as shown in Figure 74.

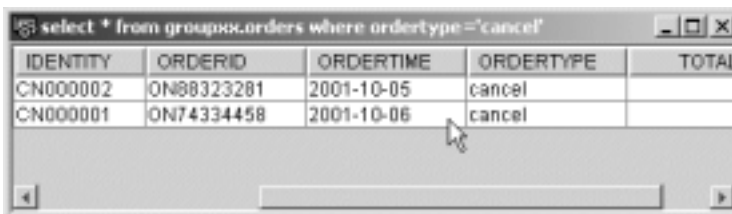
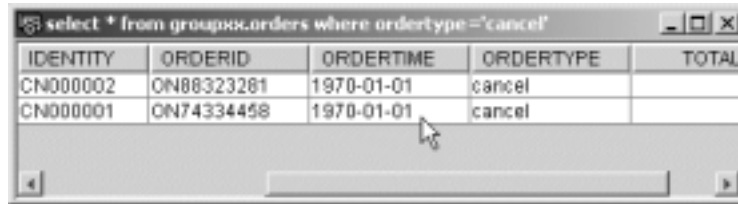


Figure 74. Selecting records before the update



- \_\_\_ 8. Run the next statement – *update*. It uses the DB2 UDB XML Extender function *db2xml.Update* to change the data attribute of an order to a new value. This statement sets the ordertime column for all cancelled orders to *1970-01-01*.
- \_\_\_ 9. Continue and run next statement. This *select* statement displays the same rows that are in the window for the first select statement. But this time, ordertime is updated with a new date (Figure 75).



The screenshot shows a window titled "select \* from groupxxx.orders where ordertime='cancel'". The window contains a table with the following data:

IDENTITY	ORDERID	ORDERTIME	ORDERTYPE	TOTAL
CN000002	ON88323281	1970-01-01	cancel	
CN000001	ON74334458	1970-01-01	cancel	

Figure 75. Selecting records after the update

- \_\_\_ 10. Note that these queries are performed against the side table and demonstrate that side tables values are updated as corresponding XML Column updates are made. The updates are done by SQL triggers provided by the product. Switch between these two windows with the records selected to see the updated column.
- \_\_\_ 11. Save your SQL script and close the session.

Now, you are going to try a more complex update to your XML document stored in the database table. You need to change two attributes:

- Quantity ordered for an item
- The total amount for the order

Use the **Run Selected** icon in the Run SQL Script to run statement-by-statement in this script:

- \_\_\_ 1. Locate the **Update-XML-Column-2.sql** file and open it using the System Editor (Figure 76).

```

C:\PROGRA~1\IBM\APPLIC~1\WORKSP~1\XMLLAB~1\ADDITI~1\UPDATE~1.SQL - Run SQL Scri...
File Edit View Run VisualExplain Monitor Options Connection Help
Examples
/-- Update information stored in XML documents that have already been inserted into an XML column --*;
/-- Do not forget to change your group number --*;

/-- This update will change the order from customer CN000003, order number ON88328384.
The quantity of tape measures ordered will be changed from 35 to 50 and the totalmoney value in
the result set and in the xml file will be changed as well. --*;
update groupxx.ordersxml
set xml_doc = db2xml.Update(xml_doc,
)PurchaseOrder/XML/ItemListItem(@partnum='PN00000001')Quantity, '50')
where droot_id in (select droot_id from groupxx.orders where orderid='ON88328384');

update groupxx.ordersxml
set xml_doc = db2xml.Update(xml_doc,
)PurchaseOrder/XML/HeadenOrderInfo/TotalMoney, '859.50')
where droot_id in (select droot_id from groupxx.orders where orderid='ON88328384');

/-- Write the file out to \ITSOLab\XML\groupxx\UpdatedOrder.xml --*;
select db2xml.Content(xml_doc, \ITSOLab\XML\groupxx\UpdatedOrder.xml)

```

Figure 76. Copy of the Update-XML-Column-2.sql file

- \_\_\_ 2. Examine the first two SQL statements, which are updates to your XML Column.
- \_\_\_ 3. Run the first update to change order ON88328384. For part number PN00000001, the quantity is changed to 50.
- \_\_\_ 4. Run the next update. The second update changes the amount for the same order. The TotalMoney column is changed to 859.50.
- \_\_\_ 5. You already changed the XML document. Run the next statement that is using the Content() function to extract the document from the DB2 UDB XML Extender database and placing it in the IFS. Examine the *where* clause. The statement selects the order ON88328384 and the corresponding information from the side table.
- \_\_\_ 6. After the XML is placed in the IFS, the next statement changes the character code of the file to 819, using the program chgccsid. Run this statement. You need to convert this file before you can view it on the PC.
- \_\_\_ 7. To demonstrate that the update operation was done in the side table, the next select statement displays the order information. Run this statement (Figure 77).

	IDENTITY	ORDERID	ORDERTIME	ORDERTYPE	TOTALMONEY
F0F0F1	CN000003	ON88328384	2001-10-04	new	859.50

Messages: select \* from groupxx.orders where orderid='ON88328384'

Figure 77. Order selected from the side table

- \_\_\_ 8. Do you want to see the order with a browser? Open your browser. Be sure you have a version that supports XML. Open the **UpdatedOrder.xml** file as shown in Figure 78.

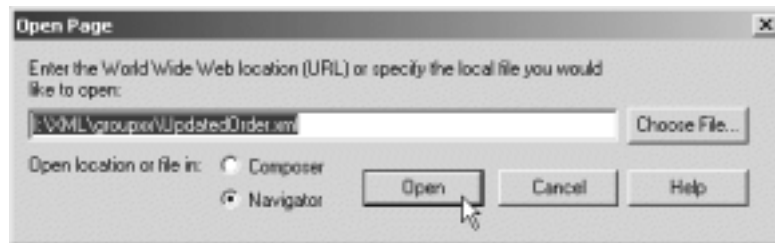


Figure 78. Opening the UpdatedOrder.xml file with a browser

- \_\_\_ 9. Make sure that the changes to the document were made. When you finish reviewing this file, save your SQL script and close your session.

This task is now completed.

---

## Task 2: Using SQL overrides during XML Composition

SQL overrides provide a more dynamic mechanism to control the set of data used to compose an XML document from an XML Collection. A DAD file is based on SQL Mapping and contains a fixed SQL statement that is used to extract data for XML document composition. An SQL override allows the SQL statement to be customized without having to update the DAD file for each query variation.

In this task, you use SQL overrides to create a number of separate products catalogs based on different class of product items offered by our lawn and garden center. First, you generate a catalog for tools parts:

- \_\_\_ 1. Locate the **Override-SQL.sql** file and open it using the System Editor. Click the **Run Selected** icon to run statement-by-statement when you are instructed (Figure 79).
- \_\_\_ 2. Run the first executable statement. This is a *delete* statement. If you already deleted the RESULTS table, ignore the error displayed at the bottom pane.
- \_\_\_ 3. To generate the XML document, you call program *genx*. This program invokes the `dxxGenXML()` store procedure.

In this program, the fifth parameter instructs to DB2 UDB XML Extender to use the override capability. The SQL to execute is coded in the following parameter. This provides the query to override the SQL query found in the DAD file.

You need to run this statement.

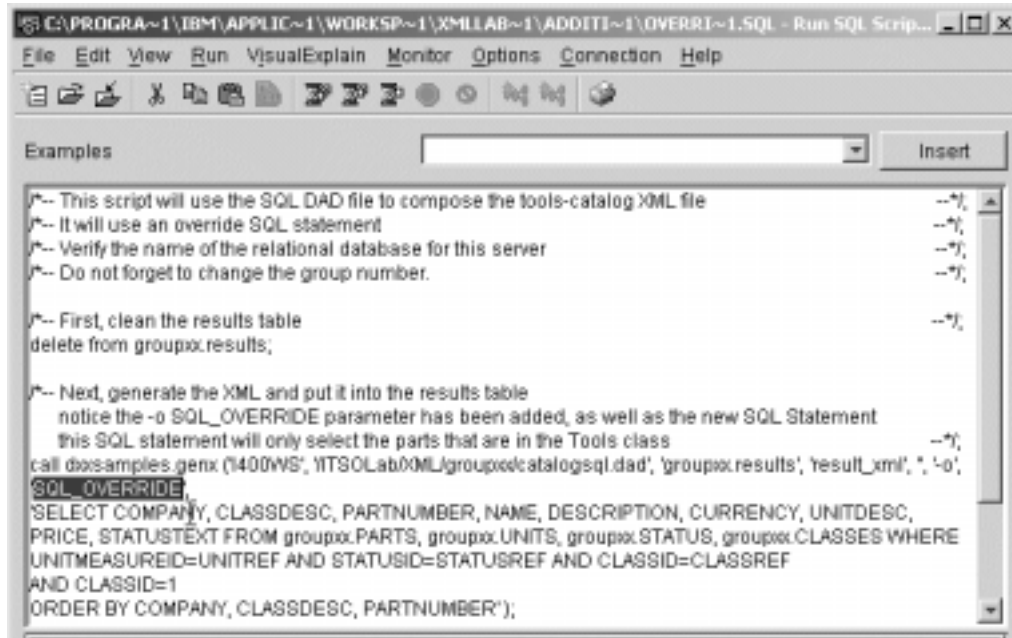


Figure 79. Overriding the SQL statement in the DAD file to generate XML file

- \_\_\_ 4. The XML file has been generated. Import this file from the DB2 UDB XML Extender executing the next *select* statement. Here, you use the Content() function to place the XML file in the IFS.
- \_\_\_ 5. The final statement changes the CCSID code to 819. This is mandatory before you can view the file with your browser.
- \_\_\_ 6. With your browser, open the **tools-catalog.xml** file.
- \_\_\_ 7. Save your script file.

#### Optional

You already generated the catalog for tools parts. If you want to try to generate other catalogs for fasteners, lawn and garden, and electrical items, follow these instructions. If you do not want to run these steps, skip to Task 3, "Validating composed XML documents using a DTD" on page 73.

- \_\_\_ 1. To generate a catalog for fasteners parts, select only the records with a class ID equal to **2**. Change the SQL to execute parameter to select only fasteners as shown in Figure 80.

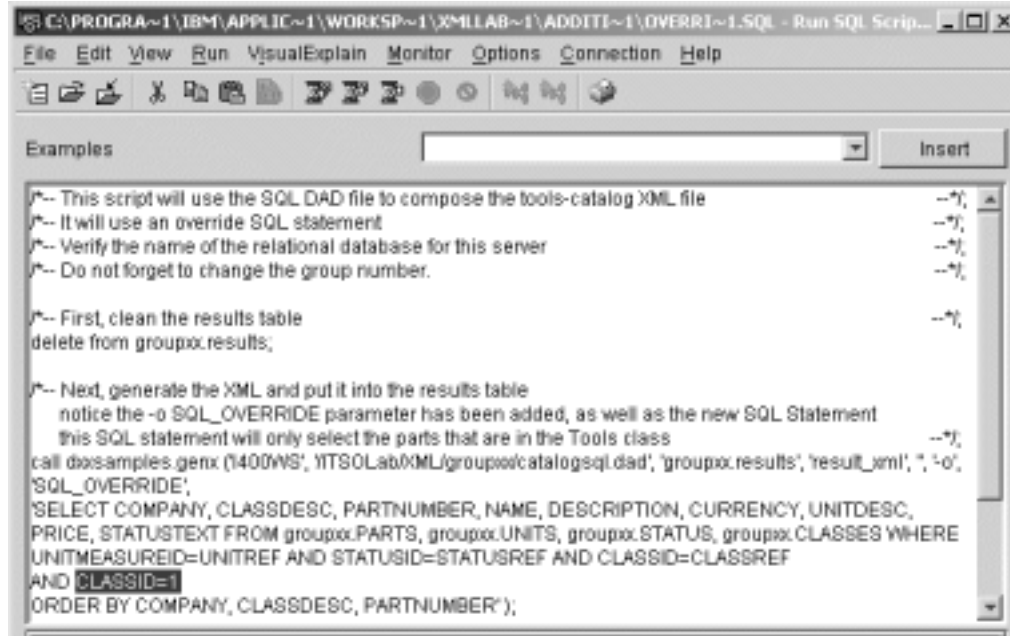


Figure 80. CLASSID column to select part type

- \_\_\_ 2. Locate the two references to **tools-catalog.xml** and change to **fastener-catalog.xml** so that a separate XML document is generated to contain the fasteners.
- \_\_\_ 3. Execute the SQL script to generate the **fastener-catalog.xml** file.
- \_\_\_ 4. Open the generated **fastener-catalog.xml** file with your browser to verify the work you do.
- \_\_\_ 5. Save this SQL script file with a meaningful name, if you want.
- \_\_\_ 6. Change CLASSID to a value of 3 for lawn and garden parts. Repeat steps 2 through 5 and use a different XML file name.
- \_\_\_ 7. After you verify that you created the new catalog, change CLASSID to a value of 4 for electrical items and again. Repeat steps 2 through 5.
- \_\_\_ 8. Close the SQL session.

This task is now completed.

### Task 3: Validating composed XML documents using a DTD

DB2 UDB XML Extender also supports registration of DTDs that may be used to validate XML data during composition, decomposition, and insertion into an XML Column. In this task, you store a DTD in the DTD reference table maintained by DB2 UDB XML Extender. You use this DTD during composition to generate a valid XML document using the SQL mapping technique provided by the DAD file.

Follow these instructions:

- \_\_\_ 1. Locate the **Insert-DTD.sql** file and open it using the System Editor. The file is shown in Figure 81.

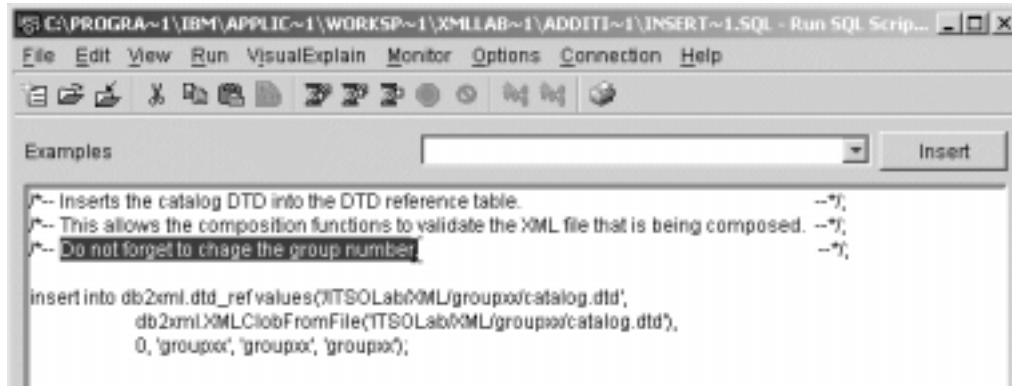


Figure 81. View of the Insert-DTD.sql file

- \_\_\_ 2. Examine the SQL script. This script inserts our catalog DTD into the DTD reference table. A table is used by DB2 UDB XML Extender for registration of DTDs that are referenced during XML document processing.
- \_\_\_ 3. Change the references to your group number. Remember that other students will be adding their DTD reference row to the same dtd\_ref table.
- \_\_\_ 4. Run the SQL script. After this operation, the DB2 UDB XML Extender is updated with an entry for your DTD.

You are ready to change the DAD to reference the DTD that is already stored. The DAD you use was used before in Lab 4, “Composing XML documents from existing data” on page 57. You are prompted to open this file and save it with a different name.

- \_\_\_ 1. Expand the **Collections** folder and open the **catalogsql.dad** file using the DAD Editor. Remember, you will save it later to another folder under another name.
- \_\_\_ 2. When the file is opened, click the **Design** tab at the bottom of the Editor pane.
- \_\_\_ 3. You need to turn on the validation. Locate the **<validation>** element and set this value to **YES** to indicate that XML documents composed using this DAD file should be validated against a given DTD.
- \_\_\_ 4. You are adding a **<dtdid>** element as a child of the **<DAD>** element. To do this, locate and right-click the **<DAD>** element and select **Add Child-> dtdid** as shown in Figure 82.

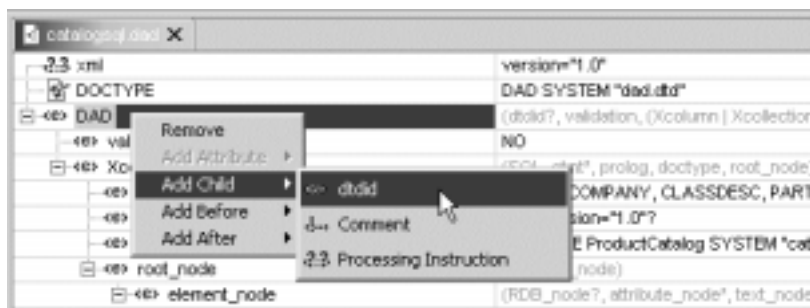


Figure 82. Adding the dtdid element

- \_\_\_ 5. Specify the value of this new element as /ITSOLab/XML/groupxx/catalog.dtd. Do not forget to change your group number. See Figure 83.

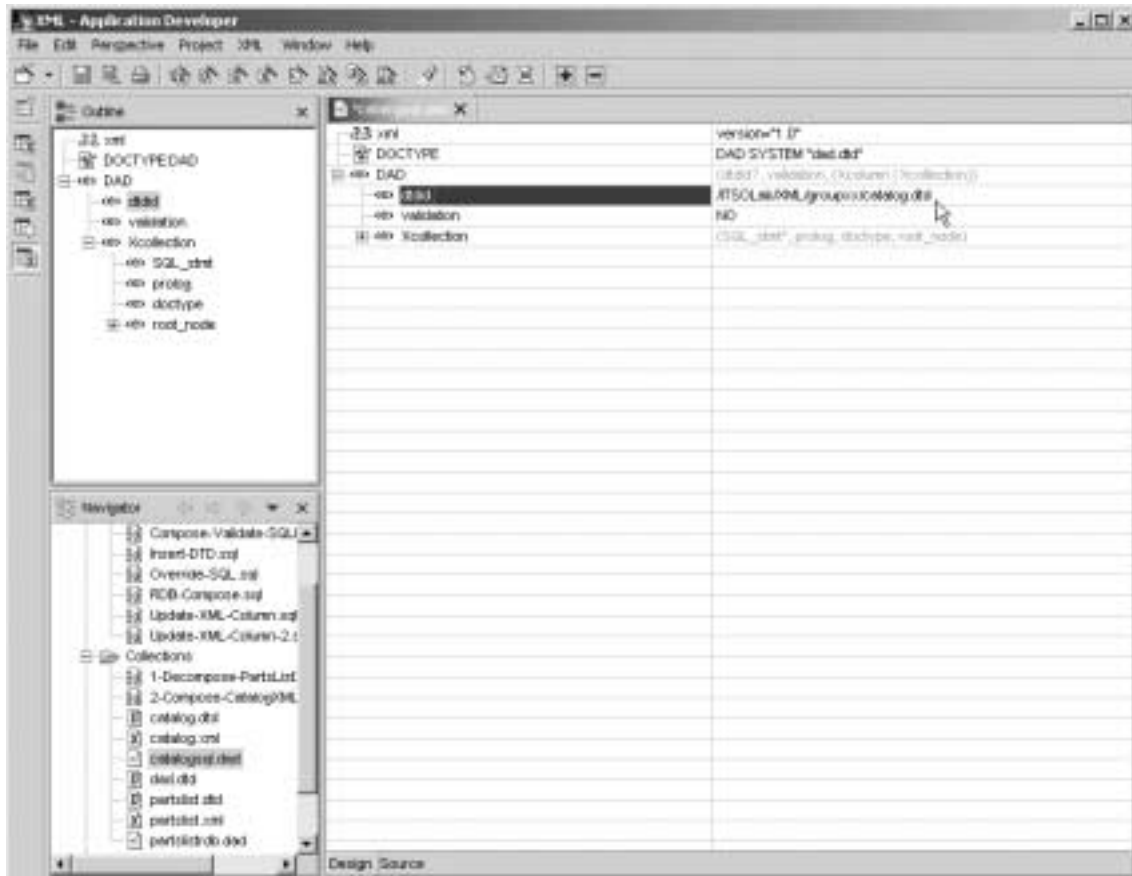


Figure 83. Reference to the DTD file

This entry identifies the DAD file that will be used to validate documents during composition.

- \_\_\_ 6. Click **File-> Save catalogsql.dad As...** to save this file with a different name in a different folder.
- \_\_\_ 7. In the Save As dialog (Figure 84), change the file name to catalogsqlvalid.dad in the File name field. Do not press Enter!
- \_\_\_ 8. Click the **Additional** folder icon and note that the folder name changes too.
- \_\_\_ 9. Verify your changes with the display shown in Figure 84. You may find a difference only with the name of your main folder. It is very important to save the file in the Additional folder because you need to transfer the new file to the IFS.

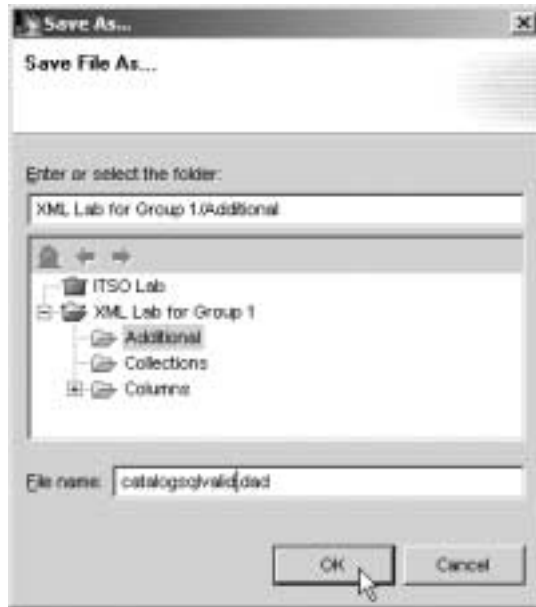


Figure 84. Saving the new file in a different folder with a different name

- \_\_\_ 10. Click **OK**.
- \_\_\_ 11. Close the **Editor** session for this file.
- \_\_\_ 12. Close the **Collections** folder.
- \_\_\_ 13. Click the **catalogsqlvalid.dad** file (in the Additional folder) and export it to your group's folder in the IFS.

Now you must generate the catalog. This time you are using a different name, so do not replace the name from the previous lab.

- \_\_\_ 1. Open the **Additional** folder and locate the **Compose-Validate-SQLDAD.sql** file. Open this file with the System Editor. The file is shown in Figure 85.

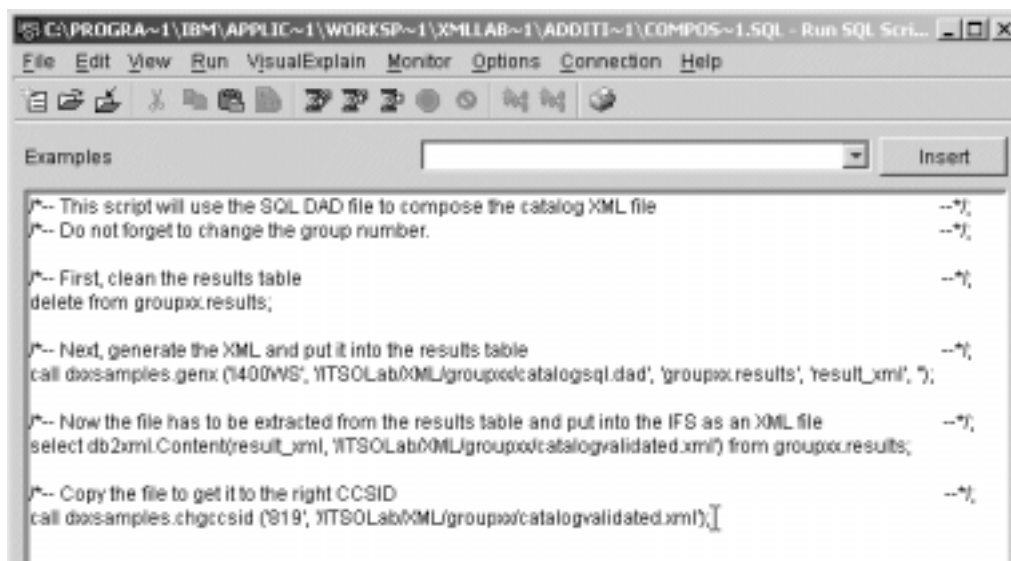


Figure 85. View of the Compose-Validate-SQLDAD.sql file



- \_\_\_ 2. Change the references to your group number. You need to change the name of your collection and the folder for your group, where you export the new DAD file.
- \_\_\_ 3. Save the file.
- \_\_\_ 4. As you can see in the SQL script, this script creates an XML file named catalogvalidated.xml from your database. Click the **Run All** icon to run the script. At the end, there is a CCSID conversion, which gives you the ability to view it with a browser.
- \_\_\_ 5. Close this SQL session.
- \_\_\_ 6. Using Microsoft Internet Explorer, open the **catalogvalidated.xml** file from the IFS. The file should look like the example in Figure 86.

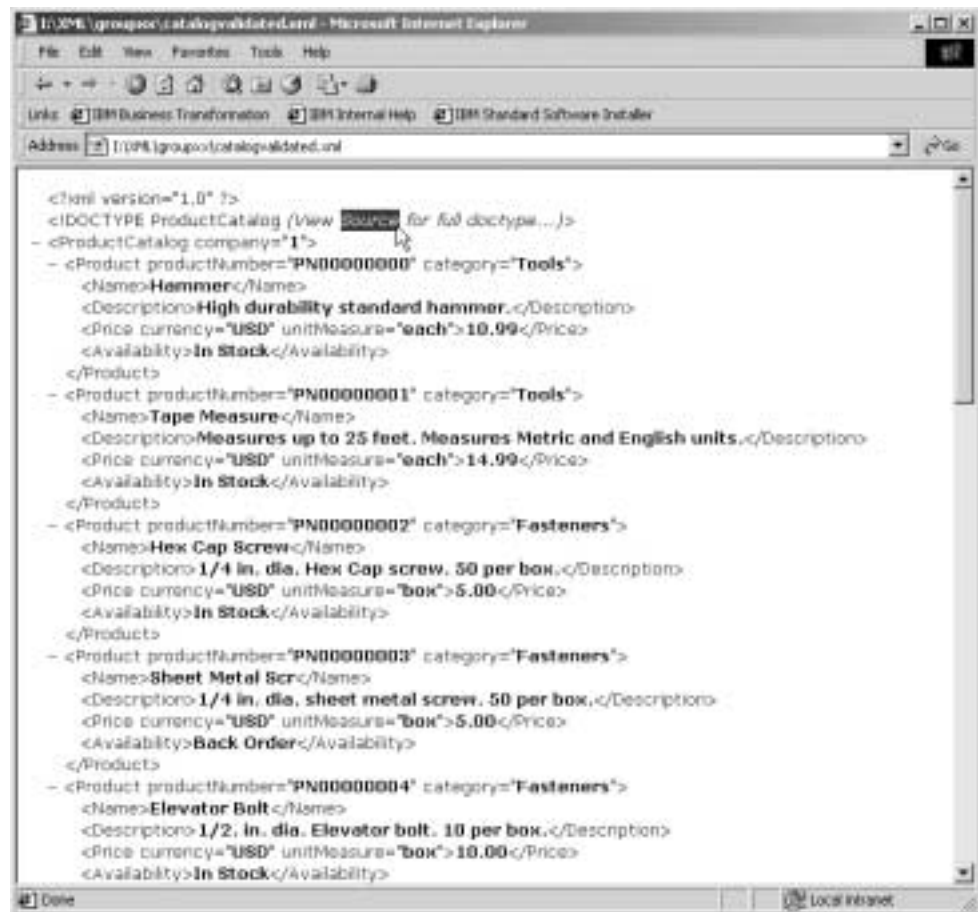


Figure 86. View of the catalogvalidated.xml file

- \_\_\_ 7. Notice that it contains a **DOCTYPE** reference. Using the menu bar of Microsoft Internet Explorer, select the **View-> Source**. The first part of the view is shown in Figure 87.

```

catalogvalidated.xml - Notepad
File Edit Format Help
<?xml version="1.0"?>
<!DOCTYPE ProductCatalog SYSTEM "catalog.dtd" [
<ProductCatalog company="1">
  <Product productNumber="PN00000000" category="Tools">
    <Name>Hammer </Name>
    <Description>High durability standard hammer.</Description>
    <Price currency="USD" unitMeasure="each"> 10.99</Price>
    <Availability>In Stock</Availability>
  </Product>
  <Product productNumber="PN00000001" category="Tools">

```

Figure 87. Part of the source code for the catalogvalidated.xml file

- \_\_\_ 8. Here, you can read that DOCTYPE makes a reference to the catalog.dtd file used during validation.
- \_\_\_ 9. Close the window with the view of the source code.
- \_\_\_ 10. Close Microsoft Internet Explorer session (optional).

This task is now completed.

## Task 4: Using the RDB node DAD file for document composition

In this task, you use the same RDB node DAD file created in the XML composition exercise (Lab 4, “Composing XML documents from existing data” on page 57) to compose an XML document based on the same set of elements and attributes used in the database schema mapping rules.

This task demonstrates the use of the RDB node mapping technique for bi-directional exchange of data between XML and relational data formats. Refer to Task 4, “Reviewing the associated DAD file” on page 51, in Lab 3, for information.

You use the same DAD file to compose an XML document. This demonstrates that the RDB mapping technique can be used for both XML document composition and decomposition.

- \_\_\_ 1. Open the **Additional** folder and locate the **RDB-Compose.sql** file. Open this file using the System Editor.
- \_\_\_ 2. Examine this script. It uses a DAD file with the RDB mapping technique to compose an XML document from our existing catalog data.
- \_\_\_ 3. Locate the string **FULL PATH TO THE DAD FILE** and replace it with the full path to the RDB mapping DAD file you created in Lab 3, “Decomposing XML data using an XML Collection” on page 43, **/ITSOLab/XML/groupxx/partslistrdb.dad**.
- \_\_\_ 4. Locate the string **OUTPUT FILE NAME** (two occurrences). Change each occurrence to the file to be generated during XML document composition. We recommend a name such as **/ITSOLab/XML/groupxx/rdbxml.xml**. If you decide to create the file with another name, write it down because you need to open this file later.
- \_\_\_ 5. Verify your changes as shown in the example in Figure 88.

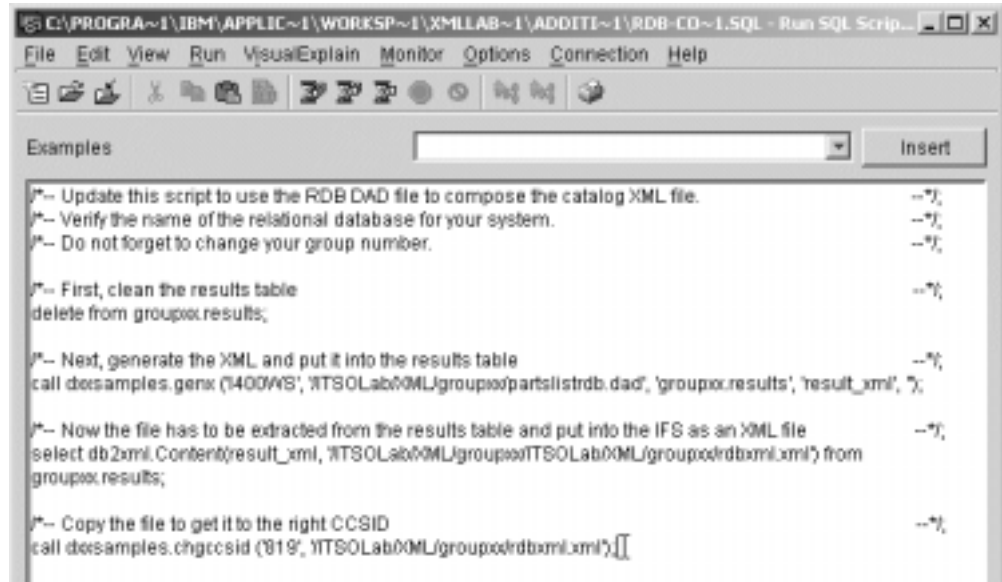


Figure 88. Composing an XML file using RDB technique

Before you run this script, review Figure 88 to see what it is happening. It removes the output from previous labs. It generates the XML document into the DB2 UDB XML Extender repository. It extracts the generated file and places it into the IFS. And it changes the file to use the right CCSID.

- \_\_\_ 6. Save your file.
- \_\_\_ 7. Click the **Run All** icon to run the script and generate your XML document.
- \_\_\_ 8. Close this SQL session.
- \_\_\_ 9. Verify your work, opening this file from the IFS, using a browser.

This task is now completed.

## Task 5: Cleaning the server of objects and files created during lab

### If you plan...

... to make a backup of the work you have done, do it now. Ask your instructor or system administrator about the resources you can use to make a backup of your library and the folder you created.

Some objects were created in the iSeries server and some were created in your PC. Your iSeries has many database objects related to your lab:

- Your schema
- Tables created using SQL scripts
- Triggers created by DB2 UDB XML Extender

Your iSeries server also holds many tables related to XML integration:

- DTDs
- DADs
- XML

In your PC, some directories and objects managed by WebSphere Application Developer are created.

In this task, you clean the server using an SQL script and iSeries Navigator. You clean the PC using WebSphere Application Developer tools.

First, clean the iSeries server of database objects:

- \_\_\_ 1. Go to the Navigator pane and open the **Cleanup-tables.sql** file using the System Editor.
- \_\_\_ 2. When the Run SQL Script window appears, change all references to your group number for the schema and verify that the relational database corresponds to the database of your system.
- \_\_\_ 3. You can read some notes about the process that this script runs. Note that if you receive an error in a sentence, you can continue using the Run Selected icon, sentence by sentence.
- \_\_\_ 4. The first SQL statement is a *call* to a function of DB2 UDB XML Extender. The program named in the call statement invokes this function and disables the column you enabled in Lab 5, "Enabling an XML Column" on page 34, in Lab 5.
- \_\_\_ 5. The second set of statements is a *drop* of database objects. At the beginning, it drops all tables created in Task 2, "Creating your schema and database tables" on page 23. Then, the schema is dropped from the system.
- \_\_\_ 6. Close the Run SQL Script window. Save the file (optional).

You have now cleaned the database objects from the server. Next you need to clean the IFS removing your folder:

- \_\_\_ 1. In iSeries Navigator, click **File Systems-> IFS-> Root-> ITSOLab** to open the main folder for labs.
- \_\_\_ 2. Click **XML-> group01**, right-click and select **Delete** as shown in Figure 89.

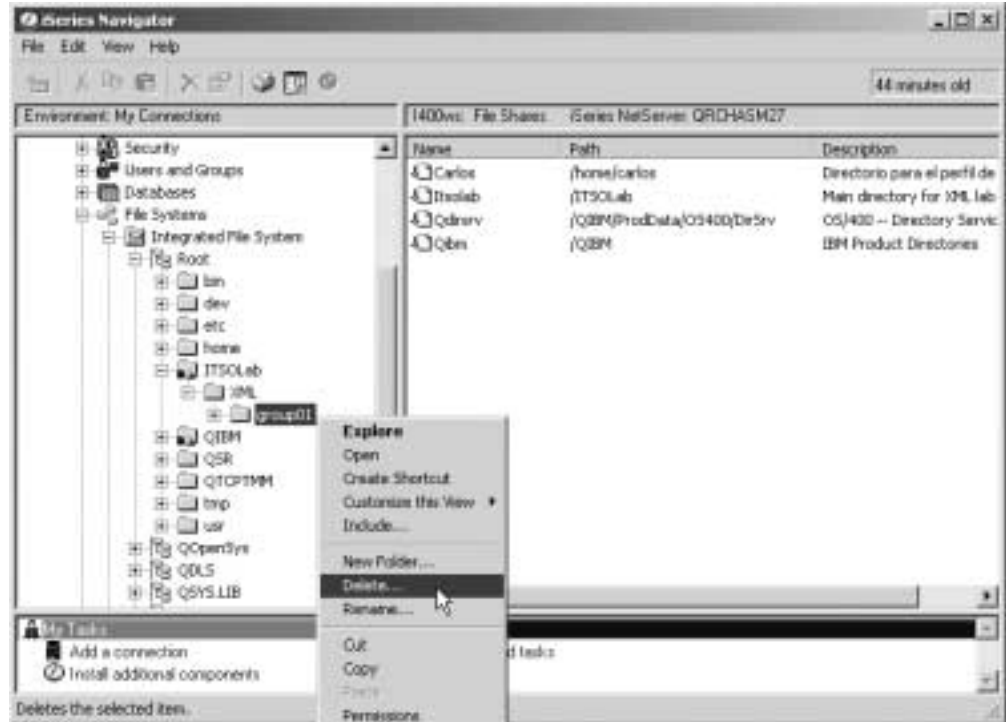


Figure 89. Deleting a folder

- \_\_\_ 3. A dialog (Figure 90) opens asking you to confirm whether you want to delete your folder. Click **Yes** to respond to this question.

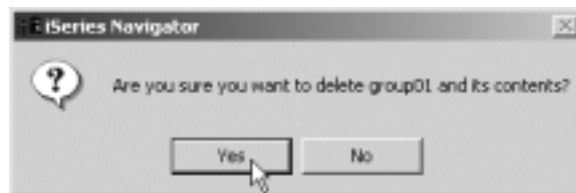


Figure 90. Dialog to confirm the folder deletion

- \_\_\_ 4. After the folder is deleted, close iSeries Navigator.

You have cleaned the IFS. And now, it is time to clean your PC:

- \_\_\_ 1. In the Navigator pane, select the project folder, right-click, and select **Close Project** as shown in Figure 91.

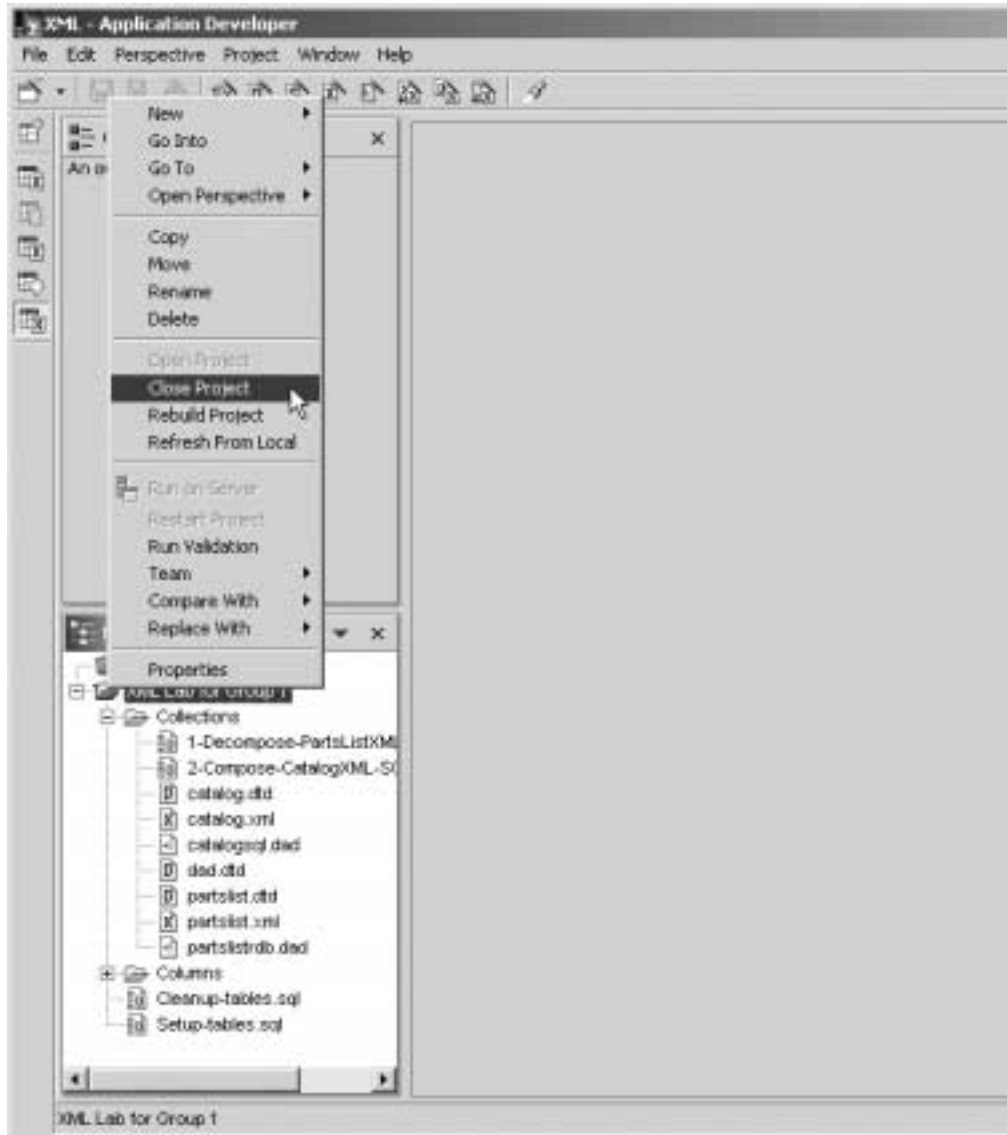


Figure 91. Closing the project

- \_\_\_ 2. Your project is closed. It is safe to copy this folder. Ask your instructor or your support department for resources to do a backup of your folder.
- \_\_\_ 3. Delete the project. This option removes your folder and objects from inside your PC file structure. Right-click your closed project and select **Delete** as shown in Figure 92.

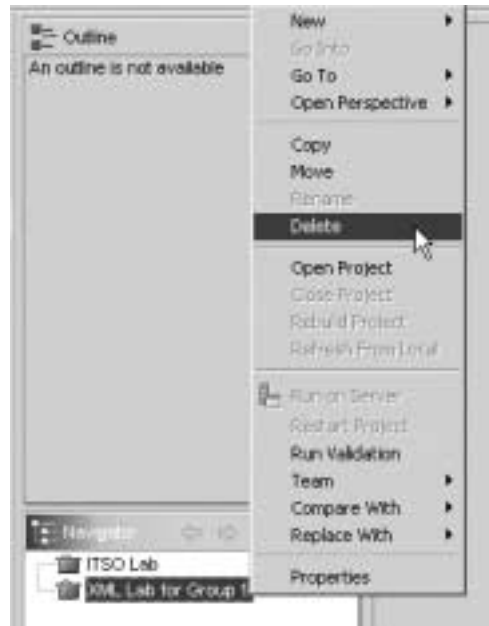


Figure 92. Deleting the project from your PC

4. A dialog asking if you want to delete your folder appears. Click **Yes** as shown in Figure 93.

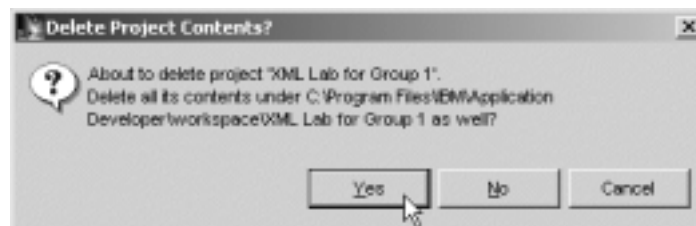


Figure 93. Confirming to delete the project

You have now completed this task and the entire lab.

