

IBM[®] DB2[®] Universal Database[™]



IBM Video Central for e-business tutorial

Version 8.1

IBM[®] DB2[®] Universal Database[™]



IBM Video Central for e-business tutorial

Version 8.1

Before using this information and the product it supports, be sure to read the general information under Appendix C, "Notices" on page 83.

This document contains proprietary information of IBM. It is provided under a license agreement and is protected by copyright law. The information contained in this publication does not include any product warranties, and any statements provided in this manual should not be interpreted as such.

Order publications through your IBM representative or the IBM branch office serving your locality or by calling 1-800-879-2755 in the United States or 1-800-IBM-4YOU in Canada.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 2001, 2002. All rights reserved.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures	v	Overview	17
Chapter 1. About this tutorial	1	Prerequisites	17
Tutorial overview	1	Video Central tasks	17
Prerequisite skills	1	Task 1: Defining the WAS remote server instance	17
Tutorial duration	1	Task 2: Defining the Video Central Web project and Importing the source code	21
Before you begin	2	Lesson summary	26
Conventions used in this tutorial	2	Next step	26
Chapter 2. The Video Central sample application	3	Chapter 5. Review of the Video Central design	27
Duration	3	Duration	27
Before you begin	3	Overview	27
Overview	3	Video Central design	27
Discovering Web services	3	The Web Interface Layer (WIL).	27
What is a Web service?.	3	Developing the Business Logic Layer (BLL)	27
How to access a Web service	5	Data modelling	37
The Video Central scenario	5	Lesson summary	39
Business rationale	5	Next step	40
The Video Central architecture	7	Chapter 6. Deployment of Video Central	41
Lesson summary	9	Duration	41
Next step	10	Prerequisites	41
Chapter 3. Enabling the DB2 environment for Video Central	11	Overview	41
Duration	11	Video Central tasks	41
Before you begin	11	Task 1: Prepare the environment to deploy Video Central from WebSphere Studio Application Developer	41
System prerequisites	11	Task 2: Deploy the Web services to the WebSphere Application Server	43
Overview	11	Lesson summary	50
Video Central tasks	11	Next step	51
Task 1: Creating the 'ivc' user ID	11	Chapter 7. Creating and running a Web based sample client for Video Central	53
Task 2: Creating the sample database.	12	Duration	53
Task 3: Enabling the sample database to use DB2 XML Extender	12	Prerequisites	53
Task 4: Enabling the sample database to use DB2 Net Search Extender	13	Overview	53
Task 5: Creating and populating the Video Central tables	14	The design of Video Central sample client	53
Lesson summary	15	Class and sequence diagrams for Video Central sample client	54
Next step	15	Video Central tasks	56
Chapter 4. Getting started with WebSphere Studio Application Developer	17	Task 1: Preparing the client project	56
Duration	17		

Task 2: Creating the proxies using the WebSphere Studio Application Developer	64	What is the DB2 Net Search Extender?	76
Task 3: Deploying the sample client to the WebSphere Application Server	67	What is the IBM WebSphere Studio Application Developer?	77
Task 4: Running the sample client.	68	About the Web services environment.	77
Lesson summary	69	About the XML Development Environment	78
Chapter 8. Solving common problems	71	Appendix B. Glossary	81
Chapter 9. Additional information	73	Appendix C. Notices.	83
Appendix A. Reference information	75	Contacting IBM	85
What is the DB2 XML Extender?	75	Product information	85

Figures

1. Components of Web services	4	22. Selecting the Web service to be deployed	44
2. Video Central architecture	9	23. Java Bean to be deployed.	45
3. Creating and configuring a new server instance	18	24. Configuring the Java Bean as a Web service	46
4. Server instance created	19	25. Select the public methods to deploy	47
5. Configuring the data source	20	26. Web service binding proxy generation	48
6. Add external JAR files.	22	27. Content of WSDL file	49
7. Adding the Web project to the server	23	28. All Web services deployed	50
8. Adding the Web project to the server continued	23	29. Components of Web services	53
9. Before importing the JAR files	24	30. Class diagram for Video Central sample client, Video Client	55
10. Select resources to import	25	31. Sequence diagram for Video Central sample client, Video Client	56
11. After preparing Video Central project	26	32. Add external JAR files.	58
12. Video Central overall class diagram	28	33. Adding the client Web project to the server	59
13. Class diagram for Customer Infraction Web service	30	34. Highlight source.	60
14. Class diagram for Wish List Web service	32	35. Importing the client code from a JAR file into VideoClient project	61
15. Sequence diagram for Add Customer Infraction	34	36. Highlight webApplication	62
16. Sequence diagram for Query Customer Infraction	35	37. File system screen	63
17. Sequence diagram for Add Wish List	36	38. Successful import of Video Central client, VideoClient	64
18. Diagram of the data model — part 1	38	39. Generating the BusinessRegistration proxy	66
19. Diagram of the data model — part 2	39	40. Proxies created for all WSDL files	67
20. Check the HTTP server	42	41. Adding the client servlets	68
21. Check that the WAS Admin Server is working	43		

Chapter 1. About this tutorial

Tutorial overview

The IBM® Video Central tutorial that includes a step-by-step guide and a complete working application demonstrates DB2® as a Web service provider. Web services are based on emerging technologies including: Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description, Discover and Integration (UDDI) specification. SOAP is a standard message format used to invoke Web services. The application of the Video Central tutorial transports SOAP messages using the Internet standard protocol, Hyper Text Transfer Protocol (HTTP). Web service requesters can be developed using the Web service provider's WSDL document. For more information on Web services, refer to <http://www.ibm.com/software/webservices>.

Web services is a new development model for integrating applications. It provides a standard technique to describe, publish, discover, and invoke business logic in a distributed computing environment. This tutorial includes complete documentation required to successfully deploy six Web services to a WebSphere® Application Server (WAS). A client application is also provided to demonstrate these Web services in action. The DB2 XML Extender's dynamic XML mapping capabilities and the DB2 Net Search Extender's high speed full-text retrieval engine are demonstrated in this tutorial.

Prerequisite skills

This tutorial is intended for the professional Java™ programmer who has a good working knowledge of the following:

- WebSphere Application Server
- Web services concepts (for more information, refer to "Chapter 2. Enabling the DB2 environment for Video Central")
- DB2 Universal Database™
- Object-oriented programming concepts
- Java programming language

Tutorial duration

This tutorial should take approximately two hours to complete.

Before you begin

The following products are used in the tutorial and should be installed before you begin:

- WebSphere Application Server (WAS), Version 4.0 Advanced Edition Single Server
- DB2 Universal Developer's Edition (DB2 UDE), Version 8.1, including the DB2 XML Extender and the DB2 Net Search Extender (The XML and Net Search Extender are included in the box but are on a separate CD and require a separate installation)
- WebSphere Studio Application Developer (WSAD), Version 4.0

Conventions used in this tutorial

Throughout this tutorial:

- <DB2_install_dir> is used to refer to the installation directory of DB2
- <WAS_install_dir> is used to refer to the installation directory of WebSphere Application Server
- <WSAD_install_dir> is used to refer to the installation directory of WebSphere Studio Application Developer

Chapter 2. The Video Central sample application

Duration

This chapter should take approximately 20 minutes to complete.

Before you begin

You should be familiar with object-oriented programming concepts.

Overview

This chapter will teach you about Web services and the Simple Object Access Protocol (SOAP). It will also familiarize you with the Web services implemented in Video Central and the architecture of Video Central.

Discovering Web services

The following section provides an overview of Web services.

What is a Web service?

A Web service is a set of related application functions that can be invoked from another application. Unlike traditional Web applications, which use HTML for information presentation, Web services use XML for information interchange. Web services can be implemented and accessed using many different programming language and platforms. This tutorial will utilize the Java language to implement the Web service requester and provider on a Windows[®] platform.

Web services are modular applications that perform specific tasks and are accessible through open protocols like HTTP. Businesses can dynamically mix and match Web services to perform complex operations using a flexible programming model.

The concept of Web services depends on the definition of three component roles (see Figure 1):

Service provider

- Provides e-business services
- Publishes availability of these services through a registry (optional)

Service broker

- Provides support for publishing and locating services

Service requester

- Locates required services using the service broker
- Binds to services available from the service provider
- Invokes the service using the defined interface

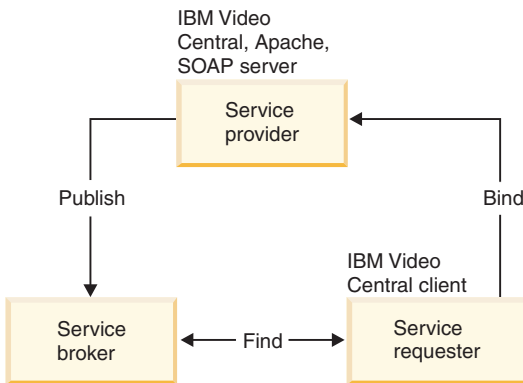


Figure 1. Components of Web services

The meeting place for Web services where businesses will form partnerships is the Universal Description, Discovery and Integration (UDDI) registry. The UDDI specification defines open, platform-independent standards that enable businesses to share information in a global Business Registry, discover each other's services, and define how they interact over the Internet. The registry is poised to become a vast library of functions that developers can use to create dynamic applications quickly. IBM is one of the key participants in defining the UDDI specification, which dozens of industry leaders have already endorsed.

Web services are published in the UDDI Business Registry by means of the Web Services Description Language (WSDL). They interact with each other by means of the Simple Open Access Protocol (SOAP). Web services can be grouped into three categories:

- *Business information* - where a business shares information with consumers or other businesses. In this case, the business is using Web services to expand its reach. Examples of business informational Web services are news streams, local weather reports, or stock quotations.
- *Business integration* - where a business provides transactional, "for fee" services to its customers. In this case, the business enables a global value net of suppliers that they can leverage as they conduct commerce. Examples of business integration Web services include bid and auction e-marketplaces, reservation systems, and credit checking.

- *Business process externalization* - where a business differentiates itself from its competition through the creation of a global value chain. In this case, the business uses Web services to dynamically integrate its processes.

For more information on UDDI, refer to <http://www.uddi.org>

How to access a Web service

A company may publish its business functions it offers on the Web. Potential consumers or clients discover the availability of these services on the Web through registries such as the UDDI Business Registry.

Web services are a natural extension to the distributed and client-server models. In Web services, methods can be invoked on remote objects using an XML standard format known as SOAP. The SOAP message can be exchanged using various transport protocols including MQSeries[®] and HTTP. The UDDI registry provides a directory of available services and service providers. The interface is defined using the language independent XML standard, WSDL.

IBM is enabling its key products with Web services capabilities. Tools to automatically generate, publish, find, test and invoke Web services are available. This tutorial will teach you how to use some of these tools. IBM also provides another way to submit SQL statements and, if you choose, control the format of the returned data. This support is known as Web services Object Runtime Framework (WORF) and Document Access Definition eXtension (DADX). For more information, refer to:

<http://www.ibm.com/software/data/pubs/papers/db2webservices/db2webservices.pdf>

The Video Central scenario

Video Central is a generic Web service provider for video rental applications (Business-to-Business applications). The purpose of Video Central is to provide a central data repository that can be accessed by registered Web-based applications. The Video Central application provides a suite of Web services to address the business needs of the client in two areas: **business services**, which serve the administrative needs of the client, and **customer services**, which enable the client both to serve and manage their customers.

Business rationale

Video rental business owners face the following limiting factors when conducting their day-to-day rental business operations:

- Manual and repetitive data entry of new rental title information.
- Limited title description (manual entry of such information as storyline, characters, and crew).

- Inability to suggest titles to their customers based on their renting history and preference.
- Some of the value-add services (wish list or previous rentals history) are limited to their local data.
- Restricted ability to evaluate credentials of new or existing customers to be entrusted with expensive rental items.
- Loss of revenue due to stolen titles by new customers (rent and disappear).

To summarize, rental business owners' data access is limited to the information hosted by their own databases. This current phase of Video Central provides the following services.

Video Central business services

We designed one business-related Web service in this tutorial: business registration.

Business registration: This Web service allows a client (for example, a video store) to register their business with Video Central. Once a business registers with Video Central, they are able to use the other Web services available.

Video Central customer services

We designed five specific customer-related Web services: customer registration, customer infractions, customer rented list, customer wish list, and movie search.

Customer registration: This Web service enables the registered business (that is, the video store) to register their customers with Video Central. Once a customer is registered, useful information (for example, rental history, wish list, infractions) about the customer can be stored, accessed, and analyzed, by using the other services available.

Customer infractions: This Web service enables the registered business to add infractions committed by a customer to the Video Central repository. This allows the business to keep track of the customer infractions for their customers without having to actually store the data in their local database. This Web service also enables the registered business to query infractions for a potential or existing customer at any time. Since Video Central is the central repository for many video stores, the business will be able to discover if a customer has an infraction from another store before renting a new item to them.

Customer rented list: This Web service enables the registered business to track a list of previously rented videos for each customer. This Web service also enables the registered business to query the rentals for a customer at any time.

Customer wish list: This Web service enables the registered business to add the titles of videos the customer has expressed an interest in watching. This allows the business to keep track of these titles for their customers and to query the wish list for a customer at any time. As well, the video store can use the lists to determine which videos are in demand by their customers, and alter their inventory.

Movie search: This Web service enables the registered business to search for movie titles and movie plot summaries. Instead of the business manually entering all the data into a local database, they can search online for current and constantly updated movie information. This Web service can be used by the staff or customers in the store to assist in video rental decision making.

The Video Central architecture

The Video Central application is based on a Web services architecture. More details on the Video Central design will be provided in "Chapter 5. Review of the Video Central design". As a part of this architecture, Video Central is essentially a dynamic and flexible provider of e-business services (over the Internet).

Service requesters may be developed to use the services provided by Video Central. When a service requester requires a service from Video Central, it will send its request to the service proxy object on the client machine. The proxy then packs the request and the related data into an XML format and sends the XML data within a SOAP encoded message. The SOAP runtime servlet is provided in WAS version 4.x. After the SOAP runtime servlet receives the request, it delegates the request to the appropriate Java Bean that implements the associated service to process the request.

The Video Central application is implemented as a three layer application:

1. Web Interface Layer (WIL)
2. Business Logic Layer (BLL)
3. Data Access Layer (DAL)

This three-tier design allows for a flexible component-based implementation of Video Central that will facilitate future enhancements through an extensible framework.

The Web Interface Layer can be implemented to support different Web interfaces such as Web services, HTTP Servlet or JavaServer Pages (JSP), and so on. This provides the maximum flexibility in end-user interface design. Video Central implemented Web services by utilizing the SOAP runtime for Web service invocation.

The Business Logic Layer is designed using Web services as the level of abstraction. Each Web service defines a set of business logic and the interface is described using WSDL (Web Services Description Language). In turn the WSDL interface can be published to a public or private UDDI (Universal Description, Discovery, and Integration) registry. A key component of the Video Central application involves the usage of the DB2 XML extender to manage the customer infraction repository for storage and retrieval. In addition, Video Central utilizes the DB2 Net Search Extender to facilitate its Movie Search Web service through its fast text search engine.

The Data Access Layer has been consolidated within the Data Access components using the IBM Data Access beans, wrapper for Java Database Connectivity (JDBC). The JavaBeans™ that implement the business logic use the Data Access components to store and retrieve data. DB2 UDB is used as the database server for the Video Central application.

Figure 2 shows the architecture of the Video Central application.

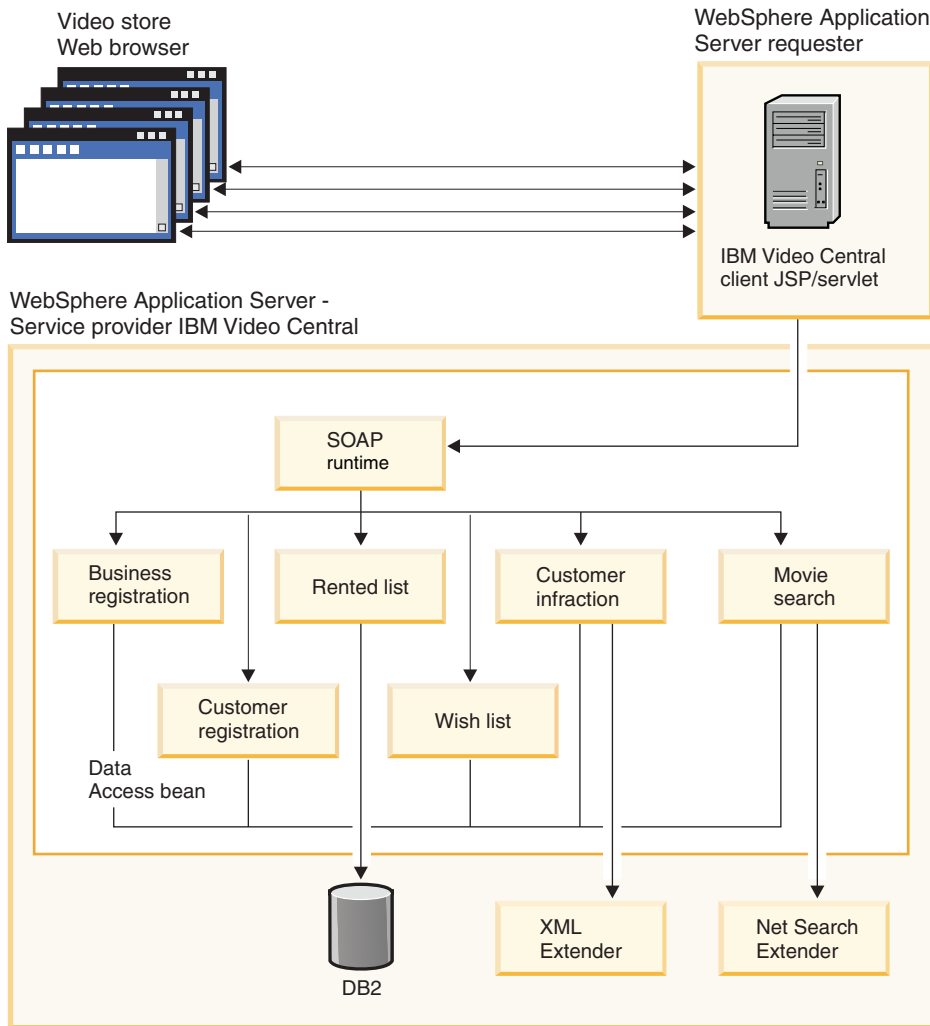


Figure 2. Video Central architecture

Lesson summary

In this chapter, you have:

- Learned about Web services
- Gained an understanding of the high-level architecture and design of the Video Central application

Next step

Next, we will learn about the required DB2 development environment and prepare it for deploying the Video Central application.

Chapter 3. Enabling the DB2 environment for Video Central

Duration

This chapter should take approximately 15 minutes to complete.

Before you begin

System prerequisites

- Microsoft® Windows NT®, Version 4.0, Service Pack 6a or higher, or Windows 2000
 - Java 2 SDK Standard Edition, Version 1.3
 - DB2 Universal Developer's Edition (DB2 UDE) Version 8.1
 - DB2 XML Extender (this comes with DB2 UDE Version 8.1, but is on a separate CD and requires separate installation)
 - DB2 Net Search Extender (this comes with DB2 UDE, v8.1 but is on a separate CD and requires separate installation)
-

Overview

This section will take you through setting up the DB2 environment for Video Central. This includes:

- Enabling the Video Central database for the DB2 XML Extender and the DB2 Net Search Extender

You will also:

- Create a new user with the name `ivc`
 - Create the tables required by Video Central
-

Video Central tasks

Task 1: Creating the 'ivc' user ID

If you have not already done so, create an `ivc` user ID on your operating system.

Windows 2000:

1. Click **Start** → **Settings** → **Control Panel**.
2. Select **Administrative Tools** → **Computer Management** → **Local Users and Groups**.
3. Right click **Users** → **New User** from the menu bar.

4. Create a new user named `ivc` with password `VideoCentral`. Deselect **User must change password at next logon**.

Windows NT:

1. Click **Start** → **Administrative Tools (common)** → **User Manager**.
2. Right click **Users** → **New User** from the menu bar.
3. Create a new user named `ivc` with password `VideoCentral`. Deselect **User must change password at next logon**.

Task 2: Creating the sample database

In this tutorial, we use the `sample` database as the Video Central database. To create the `sample` database, do one of the following:

- Run `db2samp1` from a command line.
- Click **Start** → **Programs** → **IBM DB2** → **First Steps**, click **Create Sample Databases** in the **First Steps** window.

Task 3: Enabling the sample database to use DB2 XML Extender

To store or retrieve XML documents from DB2 with XML Extender, you must first enable the database for XML.

When you enable a database for XML, the DB2 XML Extender:

- Creates all the user-defined types (UDTs) and user-defined functions (UDFs)
- Creates and populates control tables with the necessary metadata that the XML Extender requires
- Creates the `db2xml` schema and assigns the necessary privileges

The following are the steps to enable the `sample` database to use the DB2 XML Extender.

1. From a DB2 command window, run
`<DB2_install_dir>\tutorials\VideoCentral\db\dbEnableXML.bat`.

This script will:

- a. Connect to the `sample` database.
- b. Bind the XML Extender bind files to the `sample` database.
- c. Enable the XML Extender.

Note: The script

`<DB2_install_dir>\tutorials\VideoCentral\db\dbDisableXML.bat`
is provided in case you want to disable your database for XML.
There is no need to do this for the purpose of this tutorial.

2. Verify DB2 XML Extender is enabled by entering the following commands from a DB2 command windows:
 - a. `db2 connect to sample`.

- b. `db2 select procname from syscat.procedures where procschema='DB2XML'`.

You should see a list of the following stored procedure names:

- DXXDISABLECOLL
- DXXDISABLECOLUMN
- DXXENABLECOLL
- DXXENABLECOLUMN
- DXXGENXML
- DXXGENXMLCLOB
- DXXINSERTXML
- DXXRETRIEVEXML
- DXXRETRIEVEXMLCLOB
- DXXSHREXML

The Video Central application uses two of these stored procedures, namely DXXGENXML and DXXSHREXML, which are both used in the Customer Infraction Web service. DXXGENXML is used when querying customer infractions used to retrieve customer infractions from DB2 as XML documents. DXXSHREXML is used when adding a customer infraction which is received as an XML document into DB2.

For additional information on the XML Extender, refer to the *XML Extender Administration and Programming* manual available online at <http://www.ibm.com/software/data/db2/extenders/xml/ext/library.html>.

Task 4: Enabling the sample database to use DB2 Net Search Extender

1. From a DB2 Command Window, run
`<DB2_install_dir>\tutorials\VideoCentral\db\dbEnableNX.bat`.
This script will:
 - a. Start DB2.
 - b. Connect to sample database.
 - c. Start the DB2 Net Search Extender Daemon.
 - d. Enable the database for Net Search Extender.
 - e. Enable the text columns (For Video Central, the Net Search is indexed on the following columns: `ivc.titleinfo.name` and `ivc.titleplot.plotsummary`).

Note: the script
`<DB2_install_dir>\tutorials\VideoCentral\db\dbDisableNX.bat` is

provided in case you want to disable your database from using the DB2 Net Search Extender. There is no need to do this for the purpose of this tutorial.

2. Verify DB2 Net Search Extender is enabled by opening the `<DB2_install_dir>\tutorials\VideoCentral\db\movieSearch.log` and checking for any errors.

Note: If there are errors, see "Chapter 8. Solving common problems".

For additional information on the Net Search Extender, refer to the Net Search Extender Administration and Programming manual available online at <http://www.ibm.com/software/data/db2/extenders/netsearch> and "Appendix A— Reference information".

Task 5: Creating and populating the Video Central tables

1. Open a DB2 command window to run the batch file.
2. From the DB2 command window, run `<DB2_install_path>\tutorials\VideoCentral\db\tblCreate.bat`.
This script will:
 - a. Create the tables required for Video Central in the sample database.
 - b. Create the necessary indexes and constraints.
 - c. Populate the necessary tables (code tables, DAD table, Movie Search tables).

Note: The script `<DB2_install_dir>\tutorials\VideoCentral\db\tblClean.bat` is provided in the event you no longer want to run Video Central and want to drop the Video Central tables. There is no need to do so for the purpose of this tutorial.

3. From a DB2 command window, type the following commands:
 - a. DB2 connect to sample.
 - b. DB2 list tables for schema ivc.

You should see a list of 10 tables as follows:

- BUSINESSINFO
- DAD
- QUERY_RESULT_TAB
- TITLEINFO
- TITLEPLOT
- USERINFO
- USERINFRACT
- USERINFRACT_TMP

- USERRENTALS
- USERWISHES

Lesson summary

In this chapter, you have:

- Created a new user with the name `ivc`
- Created the sample database
- Enabled the Video Central database for the DB2 XML Extender and the DB2 Net Search Extender
- Created and populated the tables required by Video Central

Next step

Next, we will learn about the required development environment and prepare it for running Video Central.

Chapter 4. Getting started with WebSphere Studio Application Developer

Duration

This chapter should take approximately 20 minutes to complete.

Overview

This chapter is a quick introduction to IBM WebSphere Studio Application Developer. It will explain how to set up the Java development environment for Web services.

Prerequisites

- WebSphere Application Server (WAS) Version 4.0, Advanced Edition Single Server
 - WebSphere Studio Application Developer (WSAD) Version 4.0
-

Video Central tasks

Task 1: Defining the WAS remote server instance

This task is to create a Video Central server project and create the definition for WAS remote server within WebSphere Studio Application Developer.

Start the WebSphere Studio Application Developer

Start WebSphere Studio Application Developer by clicking on **Start** → **Programs** → **IBM WebSphere Studio Application Developer** → **IBM WebSphere Studio Application Developer**.

Create and configure the Video Central server

1. Create the Video Central server:
 - a. In the Server perspective, create a Server project by clicking on **File** → **New** → **Server Project**.
 - b. Name the project VideoCentralServer. Click **Finish**.
2. Create a new server instance and configure it:
 - a. In the Server perspective, right click on **Server Configurations** in the Server Configuration navigation panel and select **New** → **Server Instance and Configuration**.

- b. Name the Server VideoCentralServer and in the **Server instance type** select **WebSphere Servers** → **WebSphere v4.0 Remote Server**. You should now see the following (see Figure 3).

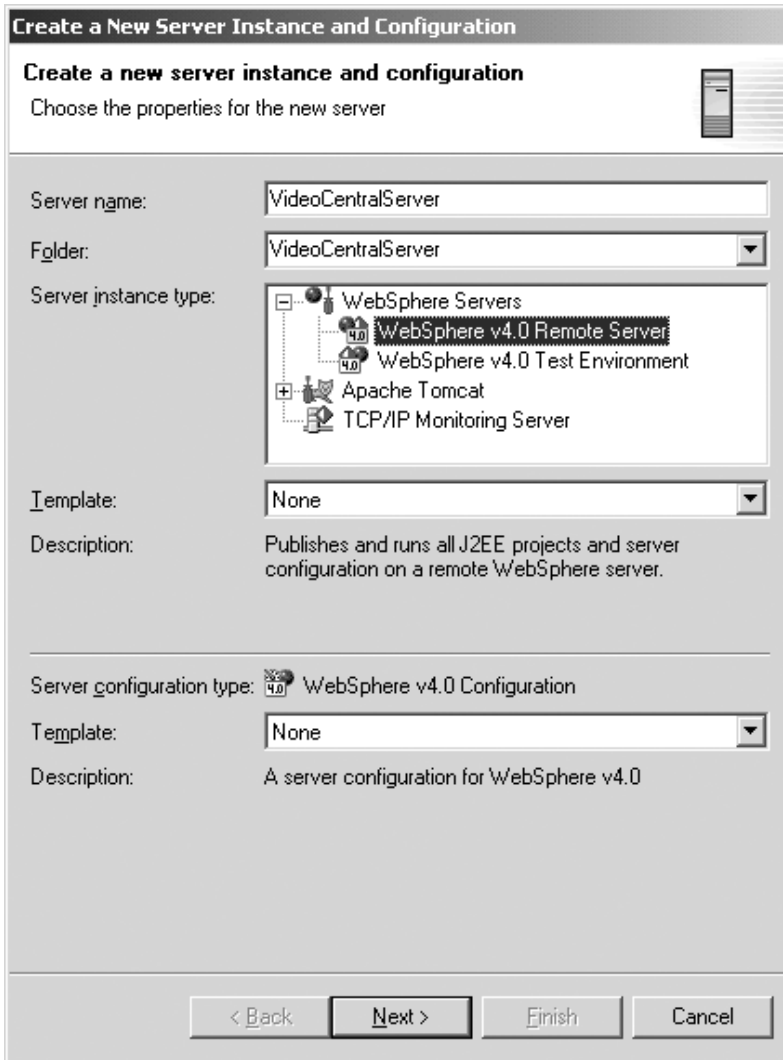


Figure 3. Creating and configuring a new server instance

- c. Click **Next**.
- d. In the **Host Address** field, replace 127.0.0.1 with localhost.
- e. In the **WebSphere installation directory** field, type in the location where you have installed WebSphere Application Server 4.0 Advanced Edition Single Server (that is, <WAS_install_dir>). Click **Next**.

- f. Select **Create a new remote file transfer instance** —> **Copy file transfer mechanism**. Click **Next**.
- g. In the **Remote target directory** field, type in the location where you installed WebSphere Application Server 4.0 Advanced Edition Single Server (that is, <WAS_install_dir>). Click **Next**.
- h. Set the HTTP port number to 9080 instead of 8080. Click **Finish**. You should now see the following in your Navigator panel (see Figure 4).

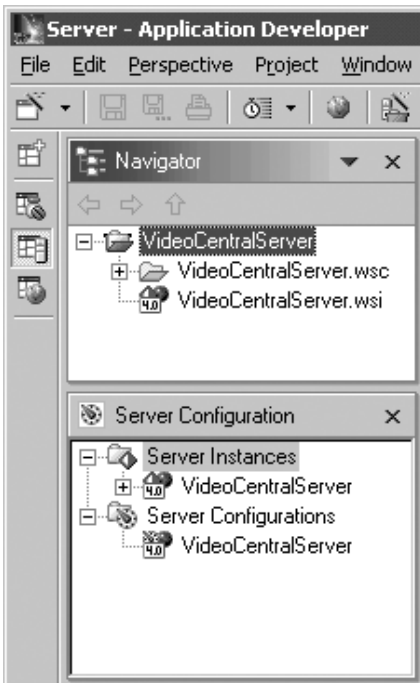


Figure 4. Server instance created

3. In the Server Configuration panel, under the **Server Configurations** folder right click on **VideoCentralServer** —> **Open**.
 - a. In the General tab, select **Enable Administration Client**.
 - b. In the EJB tab, deselect **Enable EJB test client**.
 - c. Click **File** —> **Save VideoCentralServer**.
 - d. Close **VideoCentralServer** window.
4. Configure the data source:
 - a. In the same window, select the Data source tab and in the JDBC driver list, highlight **Db2JdbcDriver**.
 - b. Add the sample database:

- 1) Click the **Add** button beside the **Data source defined in the JDBC driver selected above** box.
- 2) In the **Name** field, type `sample`.
- 3) In the **JNDI name** field, type `jdbc/Sample` (Note: the JNDI name field is case sensitive so `Sample` must be capitalized here)
- 4) In the **Database name** field, type `sample`
- 5) In the **Default User ID** field, type `ivc`
- 6) In the **Default User password** field, type `VideoCentral`
- 7) Click **OK**.

Note: The Video Central application establishes connections to DB2 by using the user ID and password stored in the `VideoCentral.properties`.

You should see the following (see Figure 5).

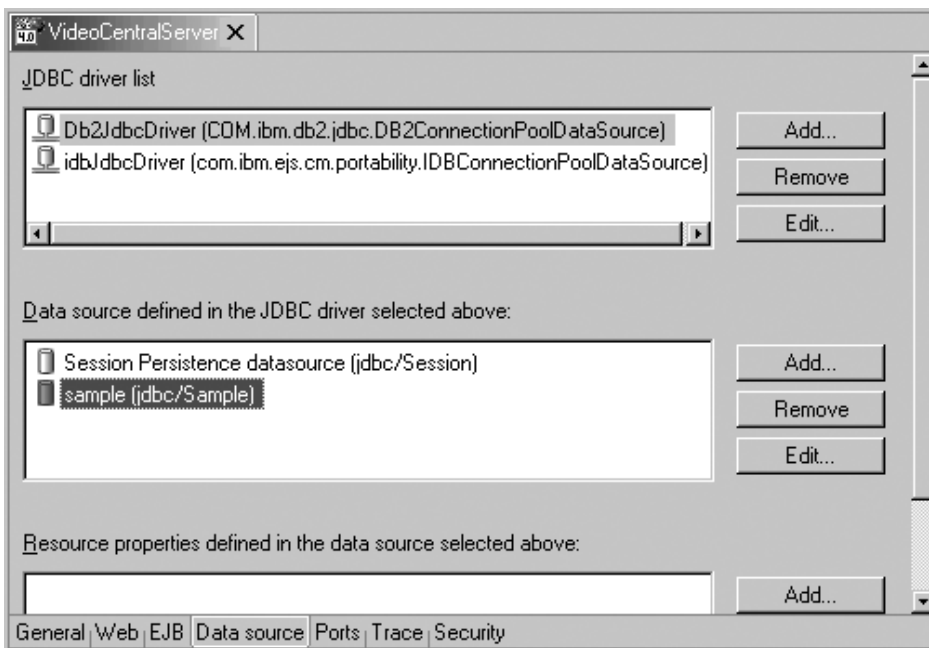


Figure 5. Configuring the data source

- c. Close the VideoCentralServer by clicking X and choosing **Yes** when the windows pops up to ask if you want to save changes.
5. In the Server Configuration panel, under the **Server Instances** folder right click on **VideoCentralServer** → **Open**.

- a. Select the Paths tab and click on **Add External JARs** and add the following:
 - <DB2_install_dir>\java\db2java.zip
 - <DB2_install_dir>\tools\databasean.jar
 - <WAS_install_dir>\lib\j2ee.jar

Note: The db2java.zip file is a set of classes that provide JDBC 2.1 support for accessing DB2 databases. These classes will communicate with the DB2 server using a JDBC type 2 driver architecture.
- b. Close the VideoCentralServer by clicking on **X** and choosing **Yes** when the windows pops up to ask if you want to save changes.

Task 2: Defining the Video Central Web project and Importing the source code

Create the VideoCentral Web project

1. Create the VideoCentral Web project:
 - a. In the Web perspective, click **File** —> **New** —> **Web project** to create a new Web project. Name the project VideoCentral and the Enterprise Application Project name VideoCentralEAR. Click **Next**.
 - b. You will see the **Module Dependencies** screen. Keep all the defaults. Click **Next**.
 - c. You should now be at the **Define Java Build Settings** screen. Select the Libraries tab, click on **Add External JARs** and add the following external jars in the **JAR Selection** pop-up window:
 - <WSAD_install_dir>\plugins\com.ibm.etools.webservices\runtime\soap.jar
 - <DB2_install_dir>\tools\databasean.jar

Note: The soap.jar is required by the Video Central to perform the SOAP message parsing required by Web services. The databasean.jar is a set of classes provided by IBM to simplify access to DB2 from Java.

You should now see the following (see Figure 6).

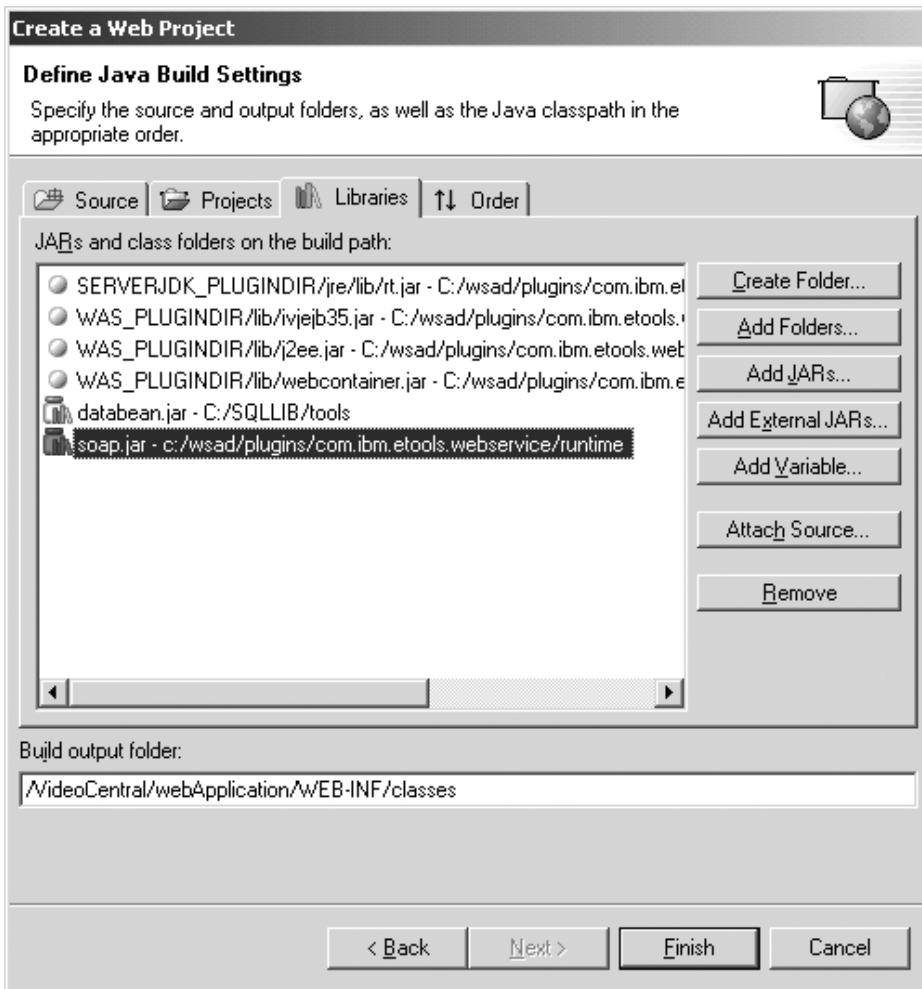


Figure 6. Add external JAR files

- d. Click **Finish**.
2. Add the VideoCentral Web project to the server:
 - a. In the Server Configuration panel, expand **Server Configurations** and right click on **VideoCentralServer**. Select **Add Project** → **VideoCentralEAR**.
You should now see the following in your Server Configuration panel window (see Figure 7).

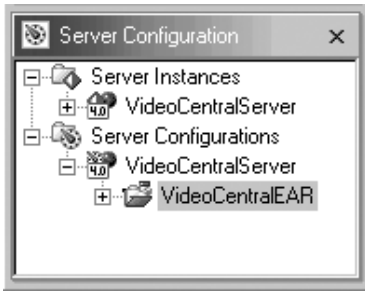


Figure 7. Adding the Web project to the server

- b. In the Navigator panel, right click on the VideoCentral Web project and select **Properties** → **Server Preferences**. Highlight the **VideoCentralServer** in the box and click **Apply** and then click **OK** (see Figure 8).

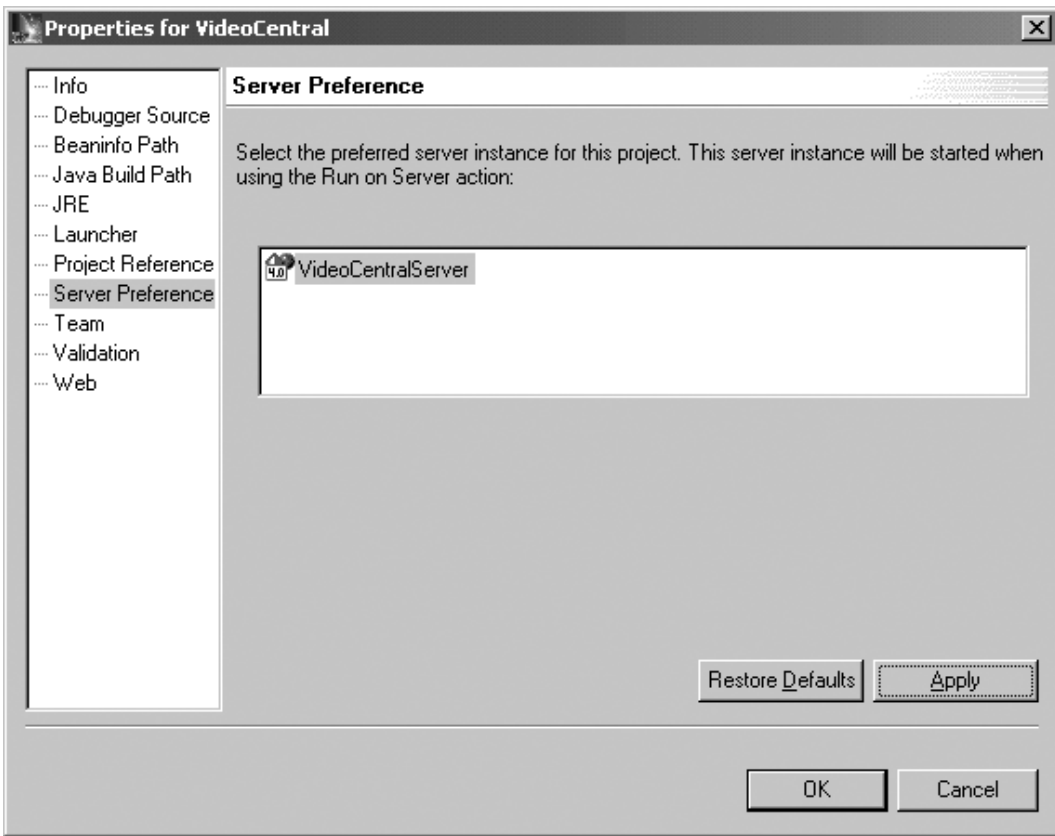


Figure 8. Adding the Web project to the server continued

Import VideoCentral.jar into WebSphere Studio Application Developer

1. Expand your newly created **VideoCentral** Web project and highlight **source** (see Figure 9).

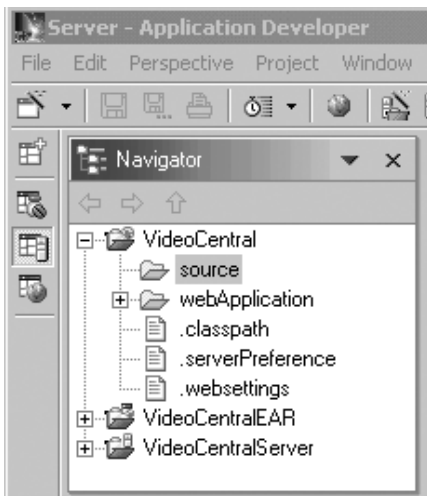


Figure 9. Before importing the JAR files

2. Select **File** —> **Import**.
3. For the import source, select **ZIP file**. Click **Next**.
4. Browse for
<DB2_install_dir>\tutorials\VideoCentral\lib\VideoCentral.jar
5. Click the **Select All** Button (see Figure 10).

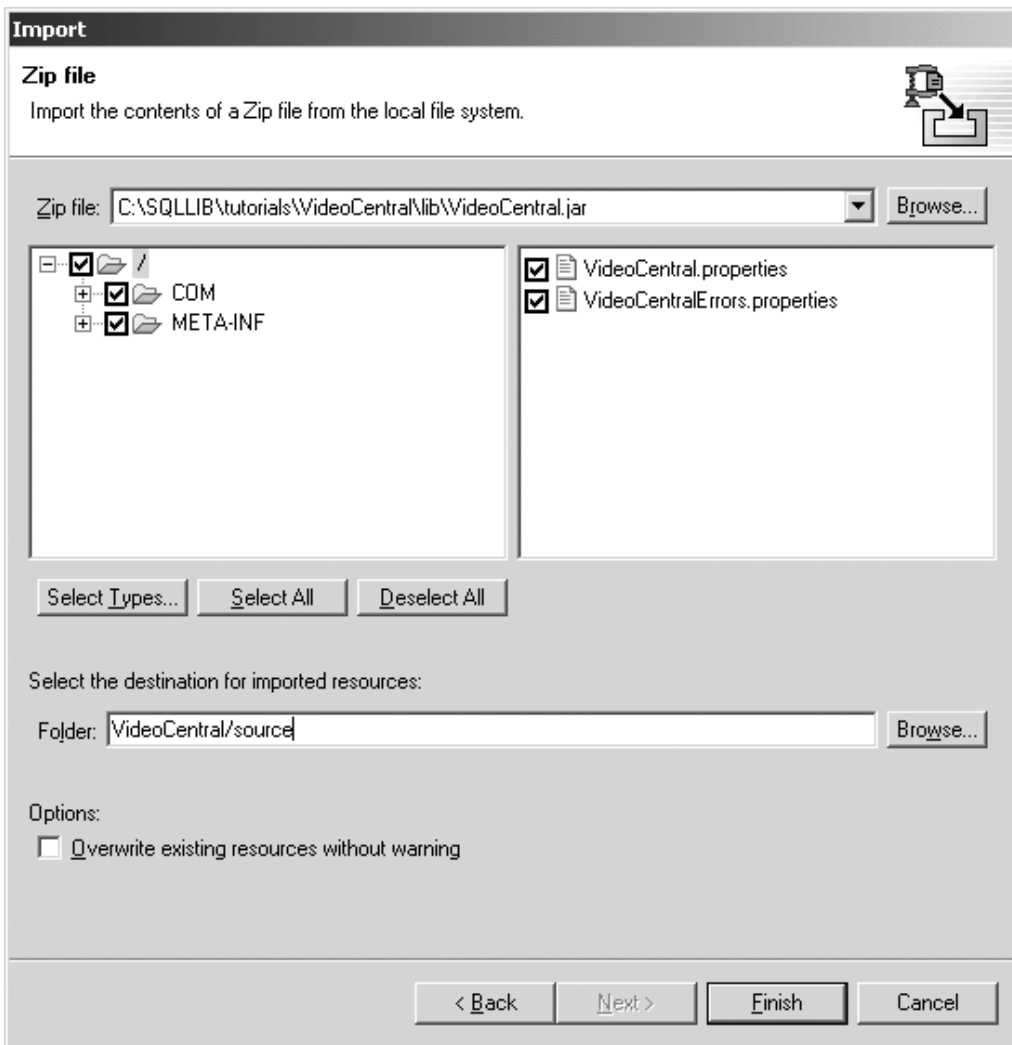


Figure 10. Select resources to import

6. Click **Finish**.

The files will now be imported into WebSphere Studio Application Developer and you will see on the action bar that the files are being compiled.

7. Once the files are imported into the VideoCentral Web project, expand the source folder and you should have a structure like the following (see Figure 11).

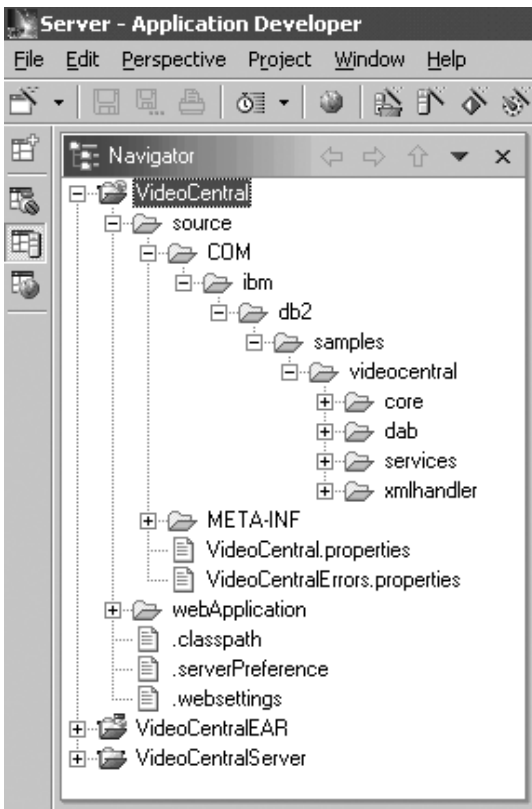


Figure 11. After preparing Video Central project

Lesson summary

In this chapter you have:

- Learned about WebSphere Studio Application Developer
- Created a WAS server reference within WebSphere Studio Application Developer to deploy and test applications
- Imported the Video Central code into the WebSphere Studio Application Developer

Next step

Next, we will explain in detail the design of the Video Central application.

Chapter 5. Review of the Video Central design

Duration

This lesson should take approximately 20 minutes to complete.

Overview

This chapter explains the design and implementation of the Video Central in the form of software component layers:

1. Web Interface Layer (WIL)
 2. Business Logic Layer (BLL)
 3. Data Access Layer (DAL)
-

Video Central design

This section will explain the design of Video Central. Some class diagrams and sequence diagrams are shown for your better understanding.

The Web Interface Layer (WIL)

The Web Interface Layer exposes the various Web services provided by Video Central to consumers. Web services may be exposed using a higher level abstraction layer. The design of the Video Central itself does not include this abstraction layer for its Web services, but utilizes SOAP runtime as this layer.

Developing the Business Logic Layer (BLL)

Class diagrams and sequence diagrams of the Java classes implemented in Video Central are provided here. The actual Java files are in the <DB2_install_dir>\tutorials\VideoCentral\lib\VideoCentral.jar and <DB2_install_dir>\tutorials\VideoCentral\lib\VideoClient.jar files included in this tutorial for your interest.

Review of the Video Central implementation

The following sections provide greater details regarding the Business Logic Layer of Video Central.

Class diagram for Video Central: Figure 12 shows the overall class diagram for Video Central.

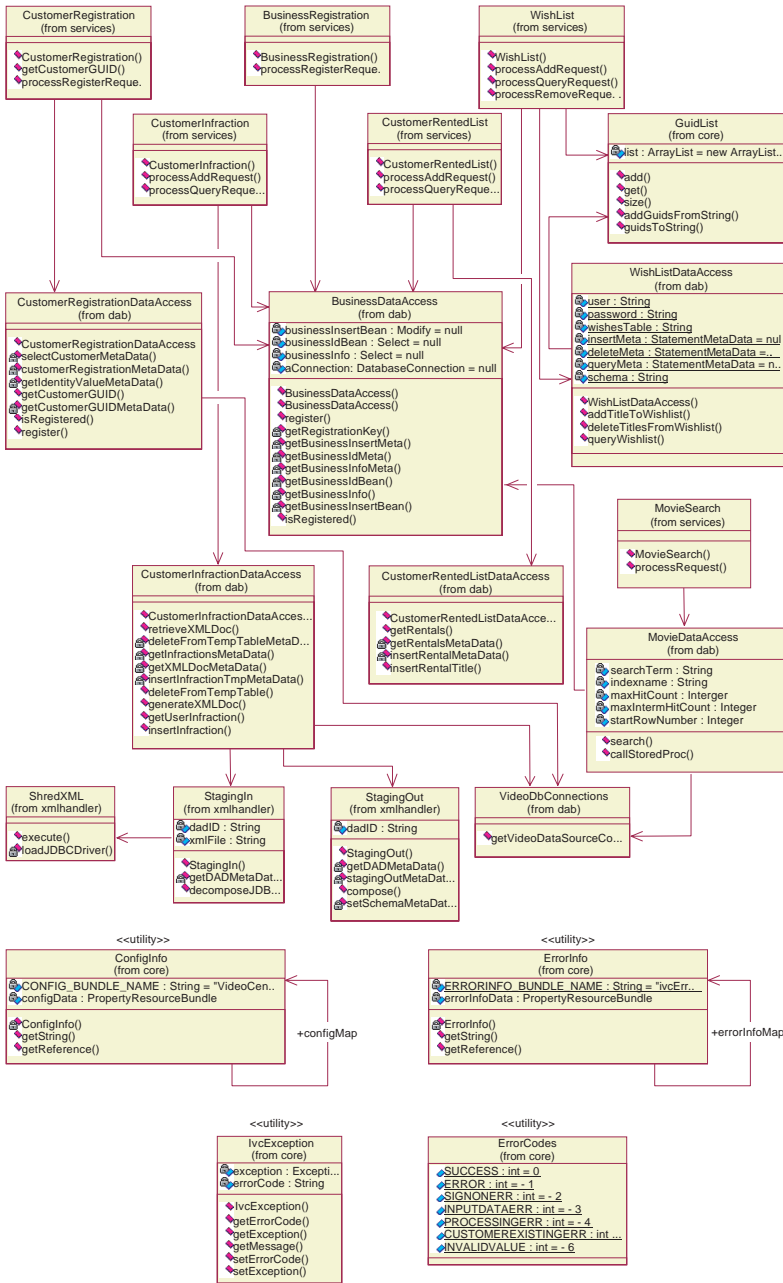


Figure 12. Video Central overall class diagram

Class diagram for Customer Infraction Web service: Each Web service is basically implemented in the same way. However, the Customer Infraction Web service is unique in that it demonstrates the use of the XML Extender.

Because of this, it deserves some special focus to show how the XML stored procedures are accessed and used. The following class diagram highlights the implementation of the Customer Infraction Web service (see Figure 13).

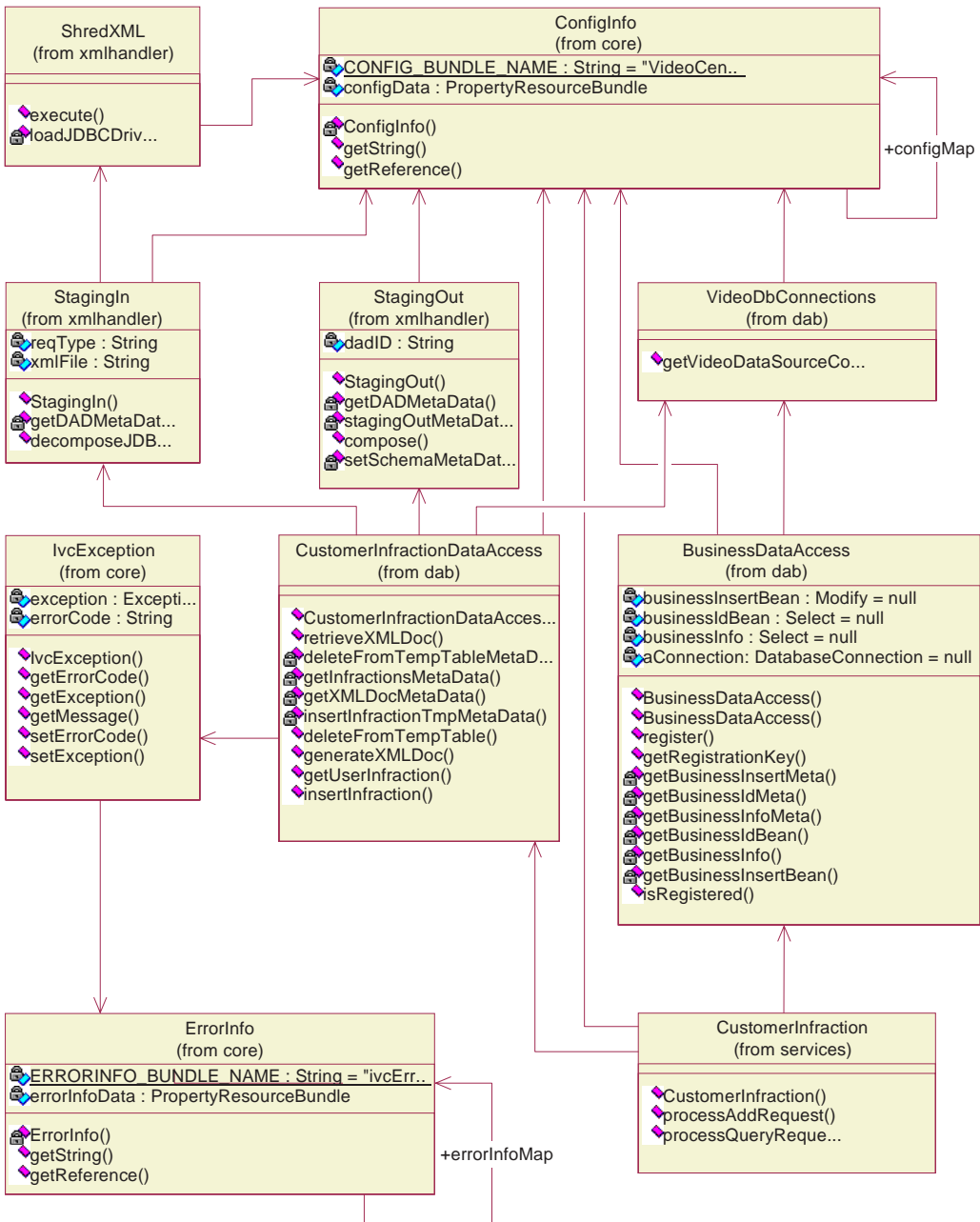


Figure 13. Class diagram for Customer Infraction Web service

In Figure 13, CustomerInfraction is the Business Logic Layer of the customer infraction Web service. While CustomerInfractionDataAccess, StagingIn, ShredXML and StagingOut compose the Data Access Layer, which involves

XML Extender to decompose and compose XML data. VideoDbConnections provides the functionality of obtaining database connections from a data source. BusinessDataAccess is involved in each Web service for business validation.

Class diagram for Wish List Web service: The implementation of the Wish List Web service uses Data Access JavaBeans (in the package `com.ibm.db.*`). All the other services were implemented with IBM Data Access beans (in the package `com.ibm.db.uibeans.*`). The `databean.jar` file provided with DB2 UDE contains the necessary classes required by the Data Access Layer of Video Central. The following class diagram highlights the implementation of the Wish List Web service (see Figure 14).

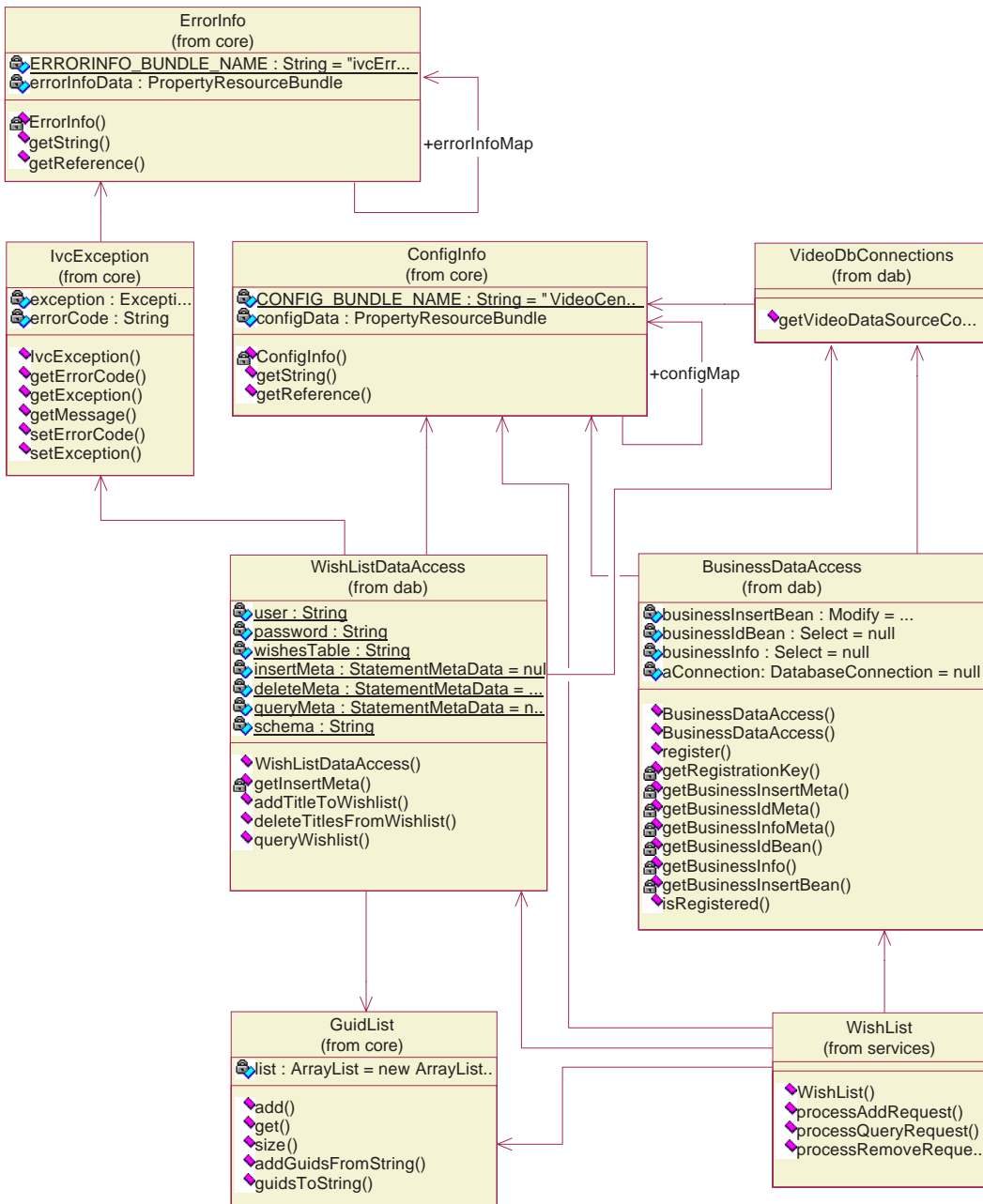


Figure 14. Class diagram for Wish List Web service

Sequence diagram for Add Customer Infraction: Since all the Web services are generally implemented the same way, we thought it unnecessary to provide sequence diagrams for all six services. The more interesting ones are

provided here. The sequence diagram for AddCustomerInfraction is worthy of special attention because it uses the XML Extender decomposition stored procedure DXXSHREDXML (see Figure 15). If you examine execute() method in ShredXML class, you will see the usage of this XML Extender decomposition stored procedure.

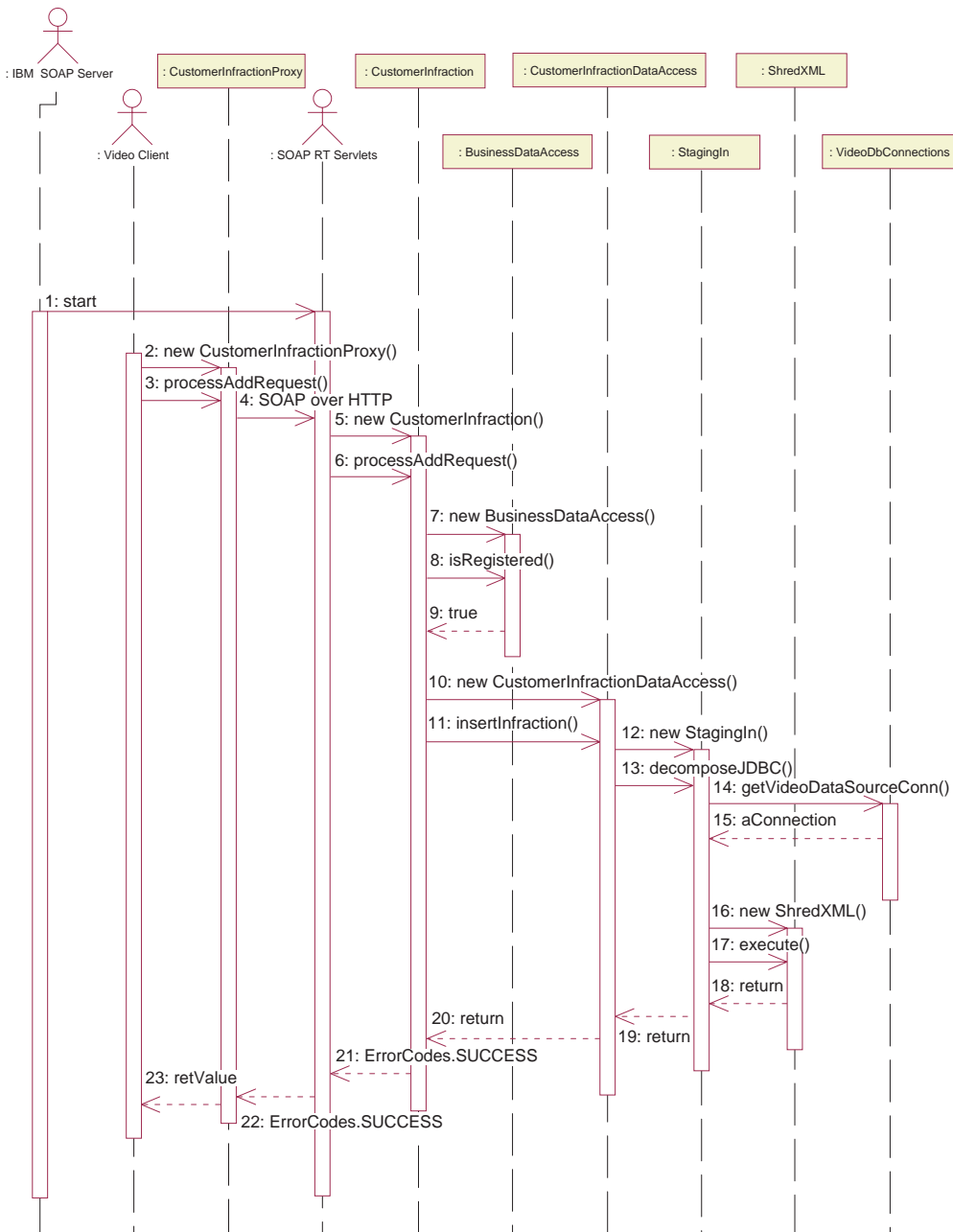


Figure 15. Sequence diagram for Add Customer Infraction

Sequence diagram for Query Customer Infraction: The sequence diagram for QueryCustomerInfraction is also worthy of special attention since it uses the XML Extender composition stored procedure DXXGENXML (see Figure

16). If you examine compose() method in StagingOut class, you will see the usage of this XML Extender composition stored procedure.

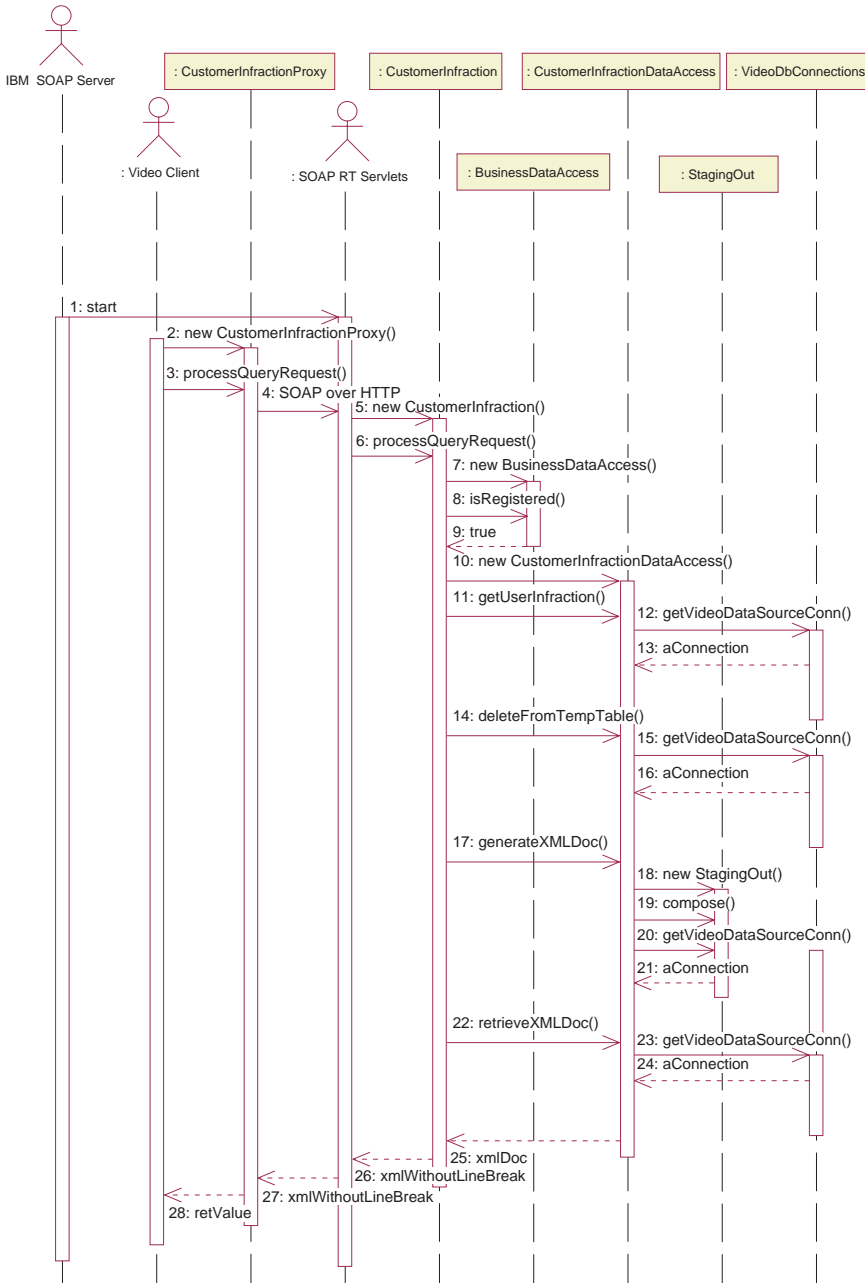


Figure 16. Sequence diagram for Query Customer Infraction

Sequence diagram for Add Wish List: The sequence diagram for adding Wish List is provided here to show the typical flow of a Web service. The Wish List service was implemented using a lower level of Data Access classes than the Data Access beans in the package `com.ibm.db.*`, but the sequence flow for the service is similar. See Figure 17. Steps 10 to 14 are to insert a title to a customer's wish list through the Data Access layer.

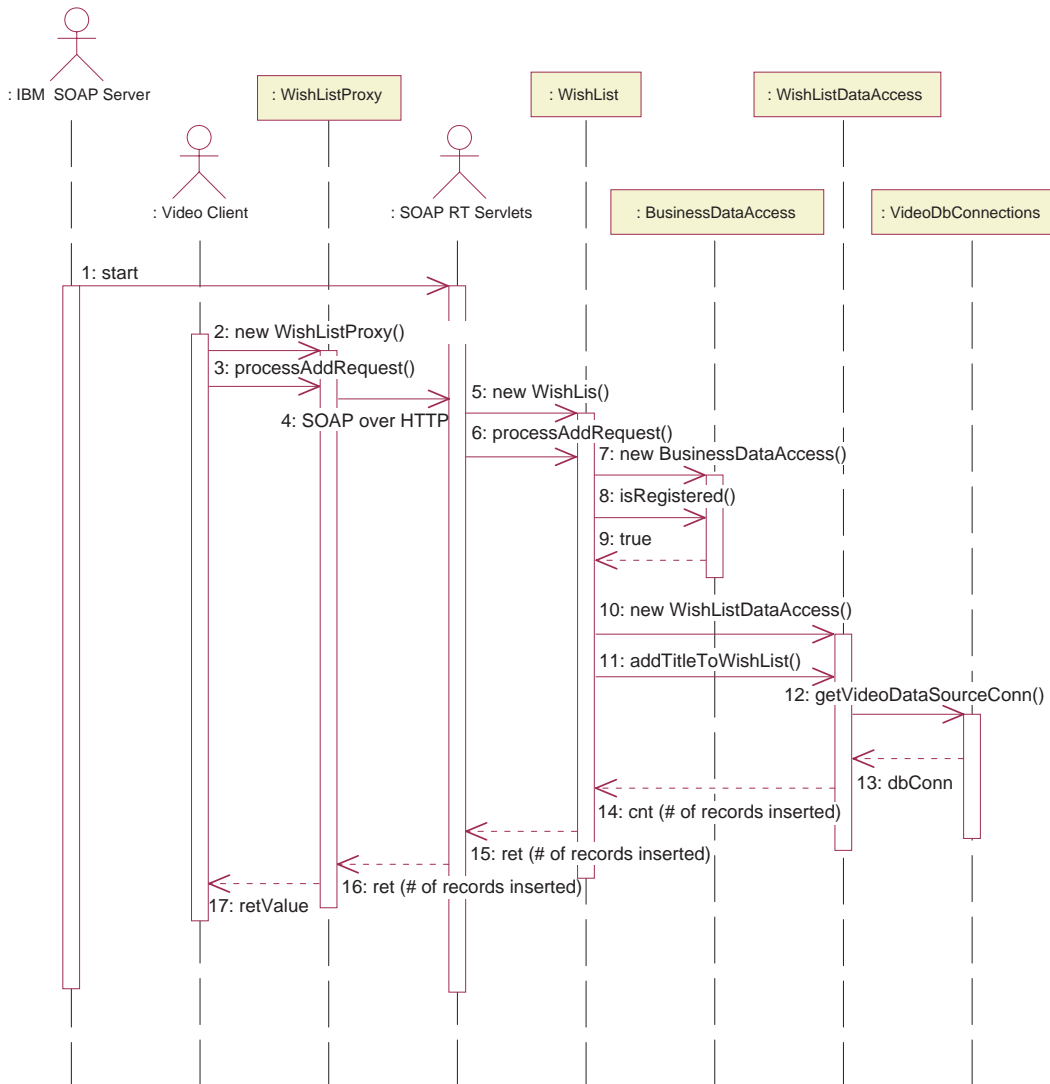


Figure 17. Sequence diagram for Add Wish List

Data modelling

The data model of the tables required by Video Central are described here. For more details, refer to the script files provided in <DB2_install_dir>\tutorials\VideoCentral\db.

Overview of the data model

Figure 18 and Figure 19 show the tables that are used by the Video Central.

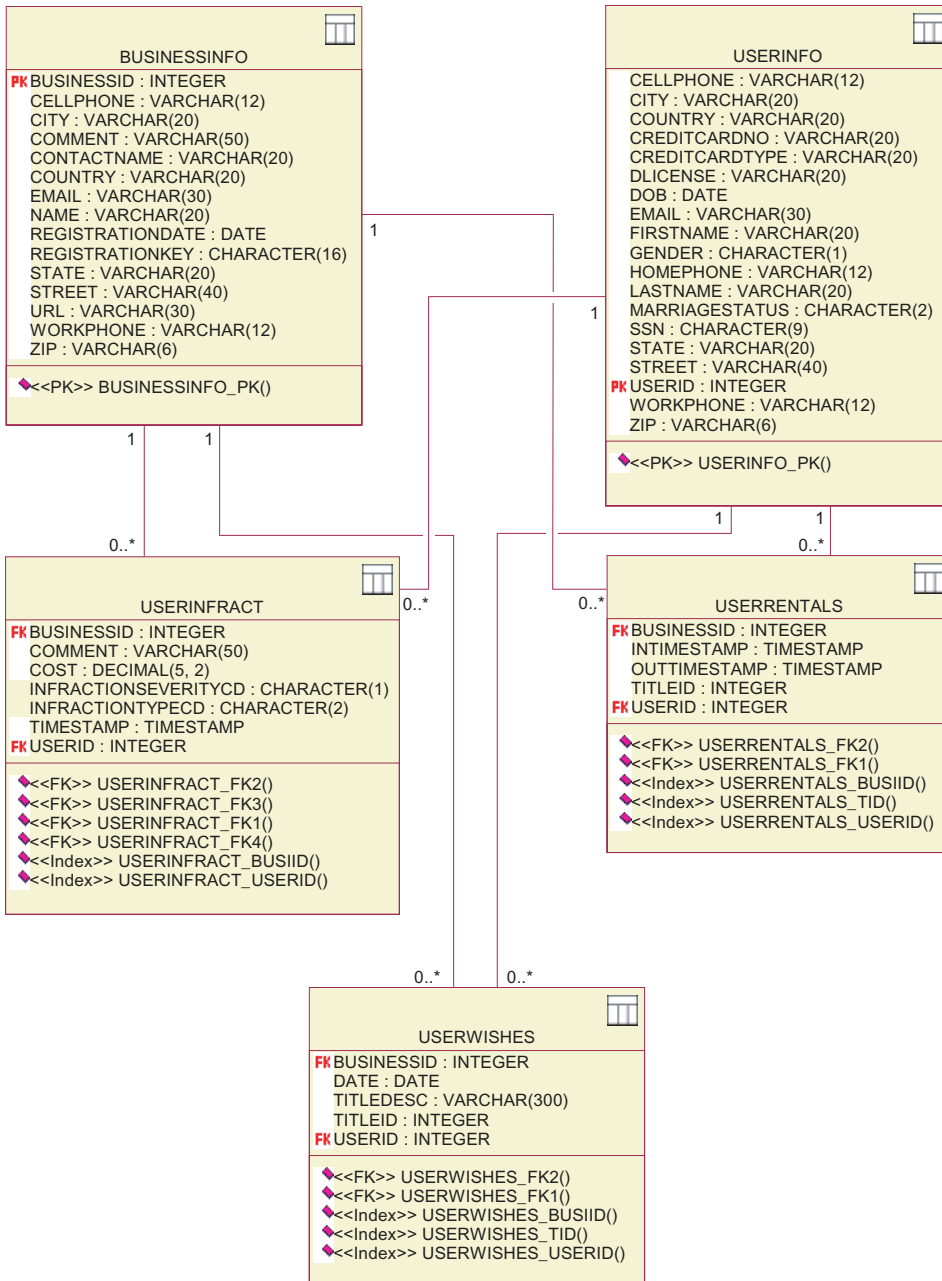


Figure 18. Diagram of the data model — part 1

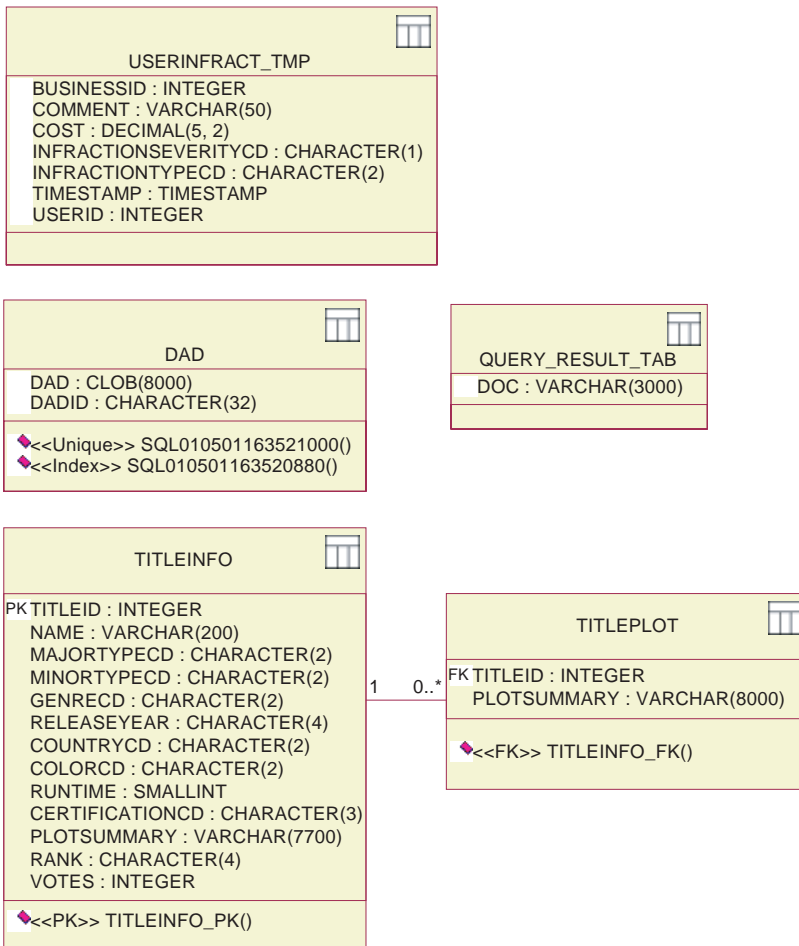


Figure 19. Diagram of the data model — part 2

The DAD table stores the Document Access Definition (DAD) files used for the composition and decomposition using XML Extender. USERINFRACT_TMP and QUERY_RESULT_TAB tables are used to store the customer infraction report for the customer infraction query. These two tables could be removed if the DxxGenXMLClob feature of XML Extender is used. TITLEINFO and TITLEPLOT tables store the information of video title and plot, and the indexes on columns TITLEINFO.NAME and TITLEPLOT.PLOTSUMMARY are used by Net Search Extender to provide the Movie Search Web service.

Lesson summary

In this chapter you have gained an understanding of the Video Central design and data model.

Next step

Next, we will deploy the Web services from WebSphere Studio Application Developer to the WebSphere Application Server.

Chapter 6. Deployment of Video Central

Duration

This lesson should take approximately 20 minutes to complete.

Prerequisites

- WebSphere Application Server 4.0, Advanced Edition Single Server
 - IBM HTTP server, Version 1.3.12 which can be obtained at <http://www.ibm.com/software/webservers/httpservers/download.html>
 - WebSphere Studio Application Developer and the VideoCentral.jar (Importing VideoCentral.jar into WebSphere Studio Application Developer is described in Chapter 4).
-

Overview

In this chapter you will deploy the Video Central code to the WebSphere Application Server (previously defined in Chapter 4) within WebSphere Studio Application Developer environment.

Video Central tasks

You will now learn how to deploy a Web service from WebSphere Studio Application Developer to the WebSphere Application Server.

Task 1: Prepare the environment to deploy Video Central from WebSphere Studio Application Developer

1. Make sure your DB2 and HTTP Server are started:
 - a. Go to your Windows Services and confirm that DB2 and IBM HTTP Server are started. If any are not started, start them now.
 - b. Open a Web browser and in the URL type localhost. You should see the following (see Figure 20).

The image shows a welcome screen for the IBM HTTP Server. At the top left is the IBM logo. The main heading reads "Welcome to the IBM HTTP Server" with "powered by Apache" in smaller text below it. A sub-heading says "Everything you need to start and use this server begins here...". Below this are three bullet points: "• Configure server", "• View documentation", and "• Visit our Web site". The background features a collage of images: a man's face, a woman's face, a man standing with arms crossed, and several interlocking gears. In the bottom right corner, there are some papers and a pair of glasses.

Welcome to the
IBM HTTP Server
powered by Apache

*Everything you need to start
and use this server begins here...*

- **Configure server**
- **View documentation**
- **Visit our Web site**

Figure 20. Check the HTTP server

2. Select the Servers panel in the Servers perspective from WebSphere Studio Application Developer. Right click on VideoCentralServer within this panel and select **Start**. Select the Console panel, and check that the message "Default server open for e-business" is present (see Figure 21).

```
Console
A DYNA0011E: Servlet cache file dynacache.xml not found; ca
A SRVE0169I: Loading Web Module: admin.
I SRVE0091I: [Servlet LOG]: JSP 1.1 Processor: init
I SRVE0091I: [Servlet LOG]: SimpleFileServlet: init
I SRVE0091I: [Servlet LOG]: action: init
A SRVE0169I: Loading Web Module: IBMVideoCentral.
I SRVE0091I: [Servlet LOG]: JSP 1.1 Processor: init
I SRVE0091I: [Servlet LOG]: SimpleFileServlet: init
I SRVE0091I: [Servlet LOG]: InvokerServlet: init
A SRVE0169I: Loading Web Module: IBMVideoCentralClient.
I SRVE0091I: [Servlet LOG]: JSP 1.1 Processor: init
I SRVE0091I: [Servlet LOG]: SimpleFileServlet: init
I SRVE0091I: [Servlet LOG]: InvokerServlet: init
A SRVE0171I: Transport http is listening on port 9,080.
A SRVE0171I: Transport http is listening on port 9,090.
U *** Server has started ***
A WSWR0023I: Server Default Server open for e-business
```

Figure 21. Check that the WAS Admin Server is working

3. Select the Servers panel in the Servers perspective from Websphere Studio Application Developer. Right click on VideoCentralServer within this panel and select **Stop**.

Note: VideoCentralServer is stopped because during Web services deployment the server will be automatically started, and you will get port in use error if the server is already started.

Task 2: Deploy the Web services to the WebSphere Application Server

Create and deploy the Web services (Java Bean based) using the WebSphere Studio Application Developer

1. Deploy the Web services from WebSphere Studio Application Developer to WebSphere Application Server:
 - a. Web Service Type Selection
In WebSphere Studio Application Developer, go to the Web perspective, expand the VideoCentral Web Project and select
VideoCentral/source/COM/ibm/db2/samples/videocentral/services/
BusinessRegistration.java
as the first Web service you will deploy (see Figure 22).

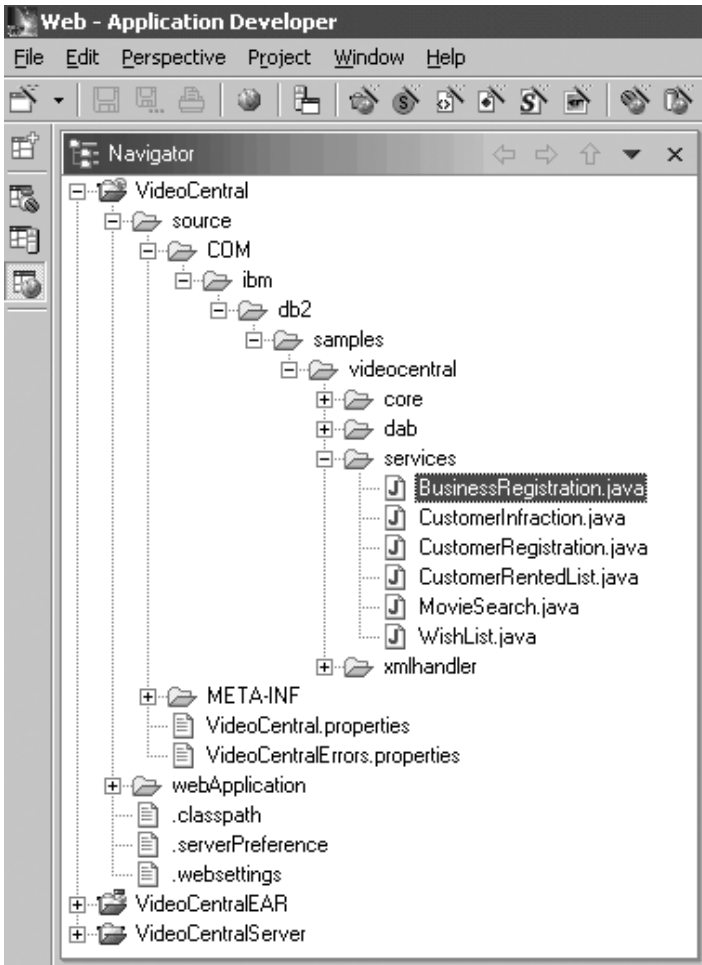


Figure 22. Selecting the Web service to be deployed

- b. Right click **BusinessRegistration.java** → **New** → **Other**, and click **Web service** in the **New** window coming up. Keep all defaults and click **Next**.
2. In the **Web Services Java Bean Selection** window, confirm that the bean is what you selected in **Step 1** (see Figure 23).

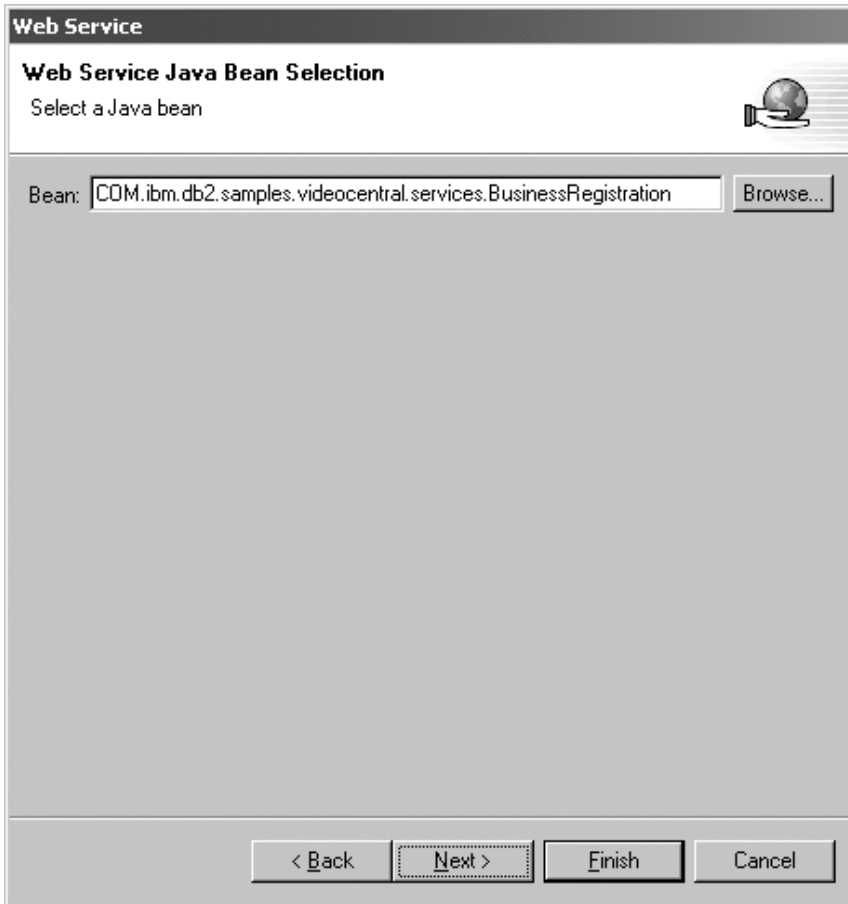
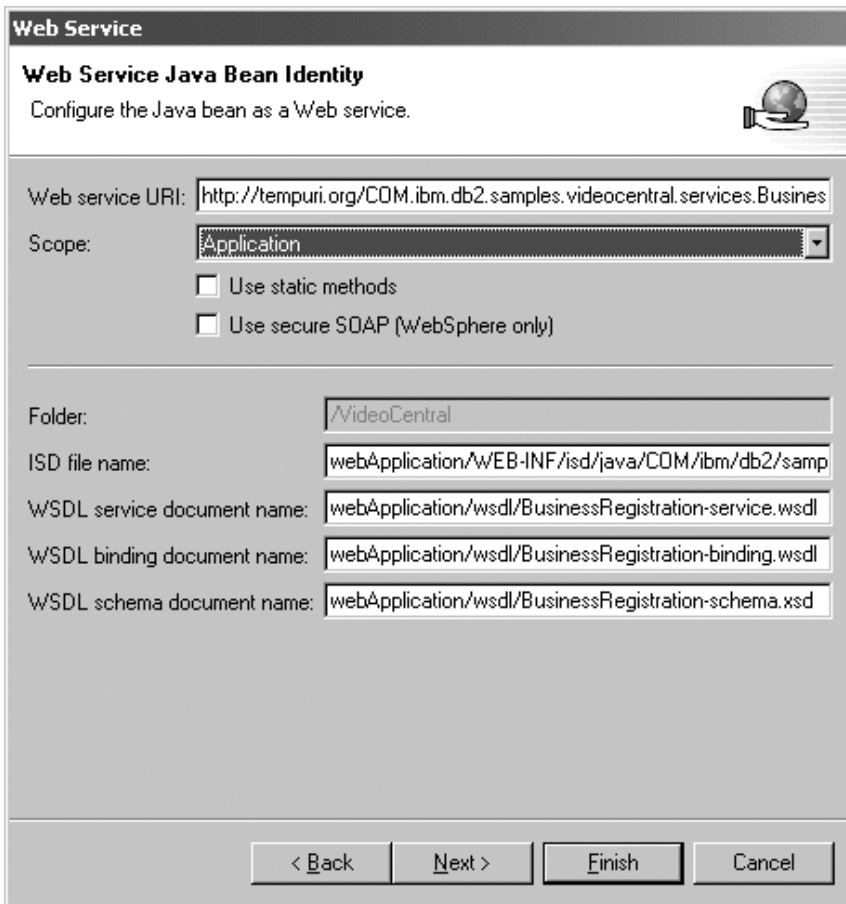


Figure 23. Java Bean to be deployed

Click **Next**.

3. Configure the Java Bean as a Web service. Leave the default settings and click **Next** (see Figure 24).



Web Service

Web Service Java Bean Identity

Configure the Java bean as a Web service.

Web service URI:

Scope:

Use static methods

Use secure SOAP (WebSphere only)

Folder:

ISD file name:

WSDL service document name:

WSDL binding document name:

WSDL schema document name:

< Back Next > Finish Cancel

Figure 24. Configuring the Java Bean as a Web service

4. Leave all methods selected (see Figure 25).

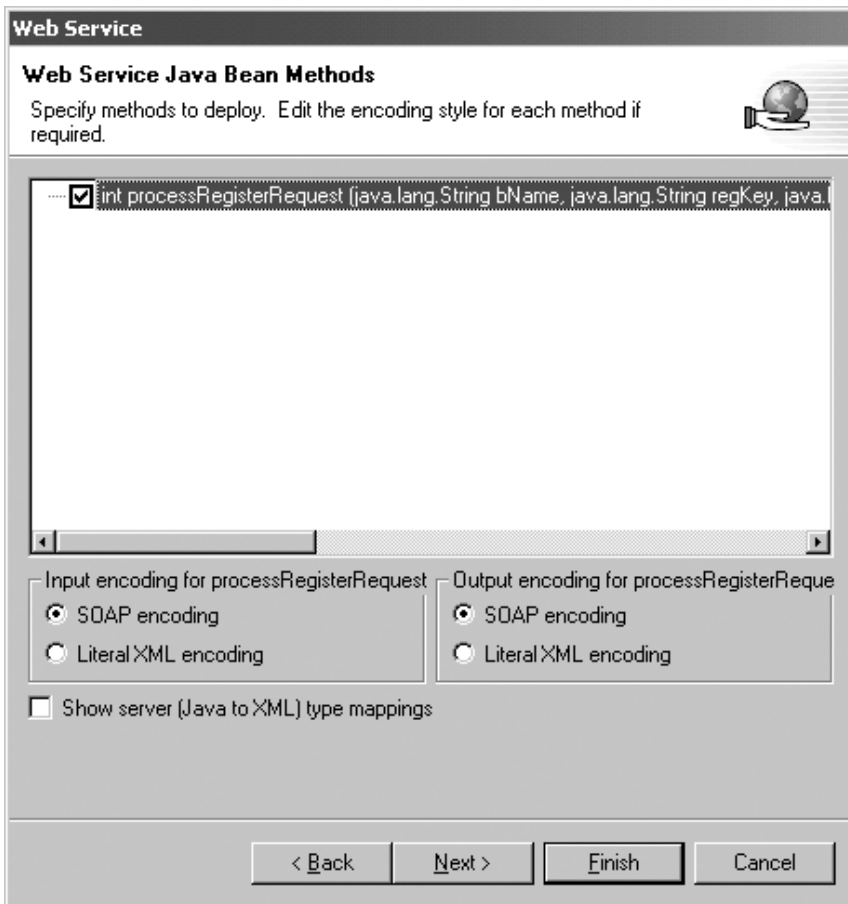


Figure 25. Select the public methods to deploy

Click **Next**.

- In the **Web Service Binding Proxy Generation** page, for the **Class** field, replace the class:

```
proxy.soap.COM.ibm.db2.sample.videocentral.services.  
BusinessRegistrationProxy
```

with

```
COM.ibm.db2.samples.videoclient.proxies.BusinessRegistrationProxy
```

See Figure 26. Thus, the proxy class can be created in the package `COM.ibm.db2.samples.videoclient.proxies.*` within the project `VideoCentral`



Figure 26. Web service binding proxy generation

6. Click **Finish** to deploy the service.
7. Select the Servers panel in the Servers perspective. Right click on VideoCentralServer within this panel and select **Stop**.

Note: If your deployment was not successful, refer to "Chapter 8. Solving common problems" for some probable causes.

At this point, the WSDL is created. You will see it in your VideoCentral Web project:

- VideoCentral > webApplication > wsdl > BusinessRegistration-binding.wsdl
- VideoCentral > webApplication > wsdl > BusinessRegistration-service.wsdl

If you would like to view the WSDL file using the default editor, double-click the WSDL file and select the **source** button (see the BusinessRegistration-service.wsdl in Figure 27). The binding document defines the protocol and format for operations and messages defined by a particular portType. The service document is a collection of related endpoints for bindings in the binding document.



```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="BusinessRegistrationService" targetNamespace="http://www.businessregistration.com/defin
<import namespace="http://www.businessregistration.com/defin
<service name="BusinessRegistrationService">
  <port name="BusinessRegistrationPort" binding="binding:Bus
    <soap:address location="http://localhost:9080/VideoCentr
  </port>
</service>
</definitions>
```

Figure 27. Content of WSDL file

Repeat Task 2, steps 1-7 for the remaining five Web services. The 6 services available for this phase of Video Central are:

- COM/ibm/db2/samples/videocentral/services/BusinessRegistration.java
- COM/ibm/db2/samples/videocentral/services/CustomerInfraction.java
- COM/ibm/db2/samples/videocentral/services/CustomerRegistration.java
- COM/ibm/db2/samples/videocentral/services/CustomerRentedList.java
- COM/ibm/db2/samples/videocentral/services/WishList.java
- COM/ibm/db2/samples/videocentral/services/MovieSearch.java

Once you have deployed all the Web services, you will see two WSDL files for each service in your Web Project (see Figure 28):

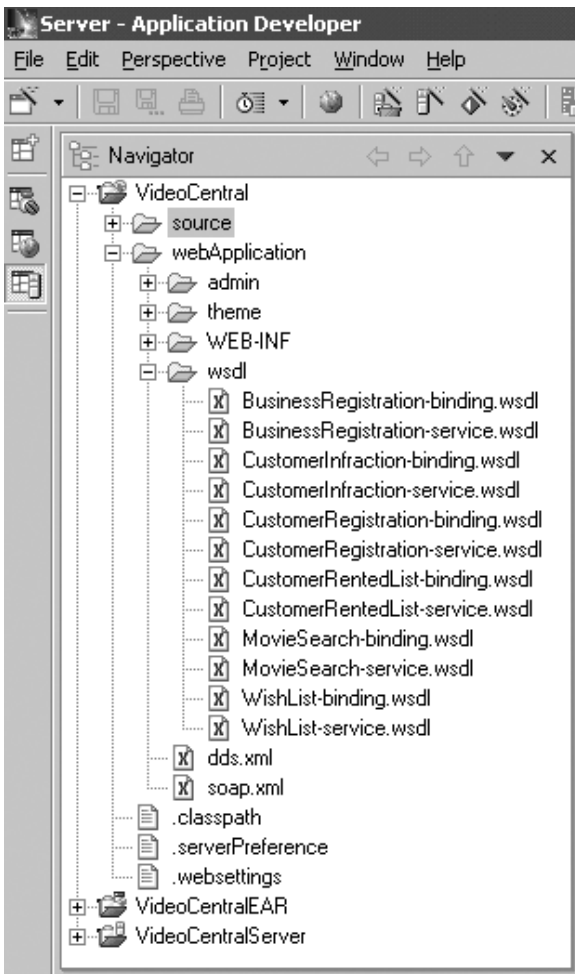


Figure 28. All Web services deployed

Lesson summary

In this chapter you have:

- Verified the environment for deploying the Video Central
- Deployed the six Video Central Web services from WebSphere Studio Application Developer to WebSphere Application Server
- Generated proxy classes to be used to access the Web services of the Video Central
- Generated WSDL files for each service. The WSDL files could be published to the UDDI registry (chapter 2) or provided to a service requester

Next step

Next, we will create the Web based sample client to invoke the Video Central services in order to see the Web services in use.

Chapter 7. Creating and running a Web based sample client for Video Central

Duration

This lesson should take approximately 25 minutes to complete.

Prerequisites

Steps in "Chapter 4. Getting started in WebSphere Studio Application Developer" and "Chapter 6. Deployment of Video Central" must have been completed successfully.

Overview

This chapter describes the design of a Web based sample client (service requester) for Video Central and the steps required to use the Web services provided. These steps include regenerating the proxy classes and deploying the client to the WebSphere Application Server. In Chapter 6, we generated the proxy classes and WSDL files from Java Beans. We will regenerate the proxy classes from the WSDL files in this chapter.

The design of Video Central sample client

The next section will be performed on the Service Requester's (Video Client) application server. The client creates the proxy classes for the Web services offered by the Web service provider (see Figure 29).

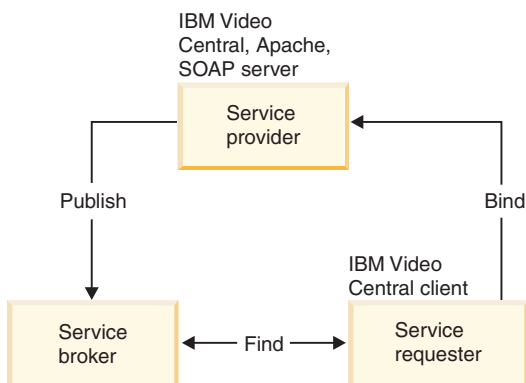


Figure 29. Components of Web services

To this point, all the files have been located on the Video Central server (the Service provider). The next section of the tutorial will deal with the Video Client server (the Service requester). For our tutorial, we are using the same WebSphere Application Server, but in reality Video Client would be on a different application server located at each of the video stores.

Class and sequence diagrams for Video Central sample client

Figure 30 shows the complete class diagram for Video Central sample client, Video Client:

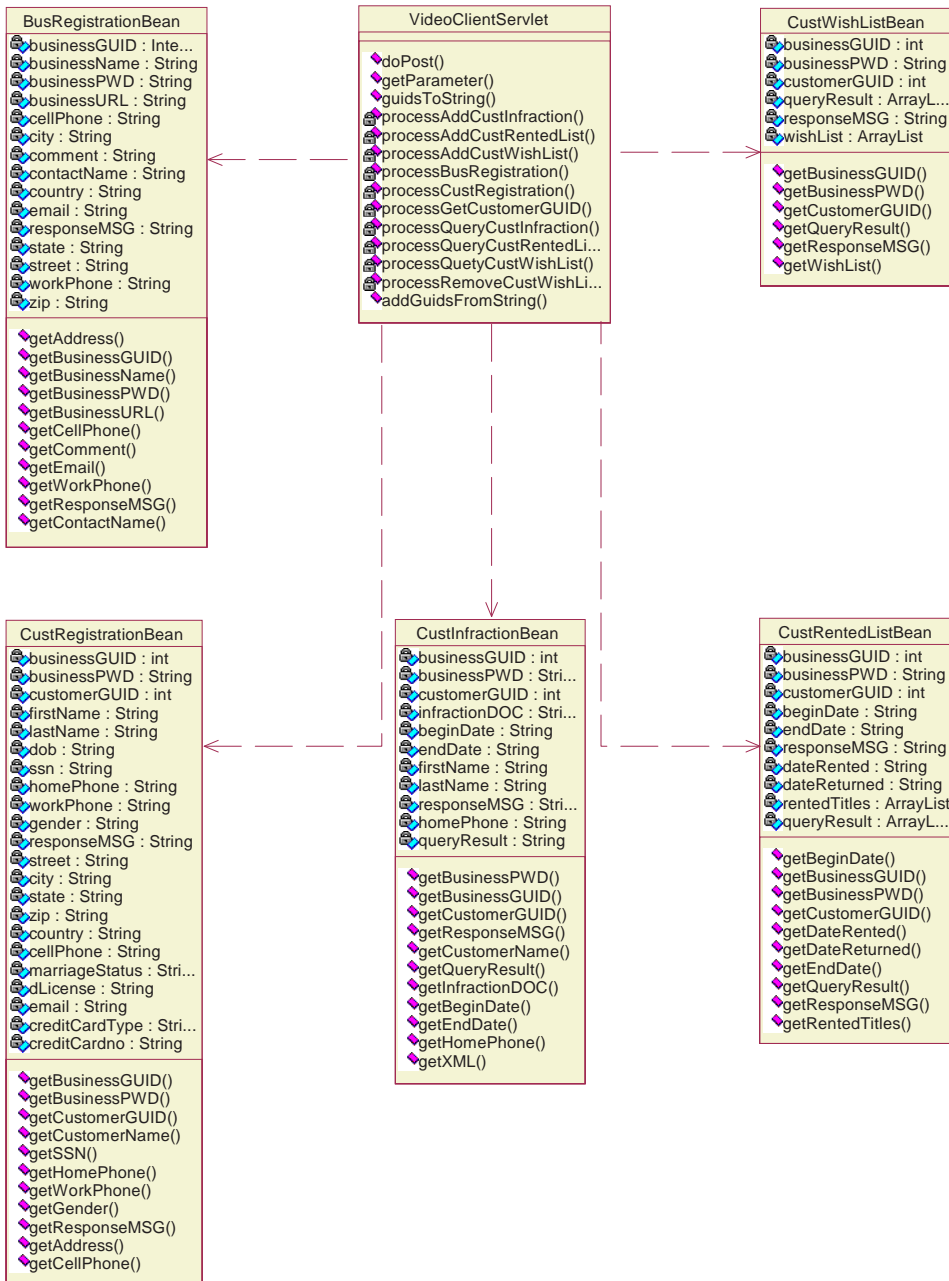


Figure 30. Class diagram for Video Central sample client, Video Client

Figure 31 shows a complete sequence diagram for Video Central sample client, Video Client:

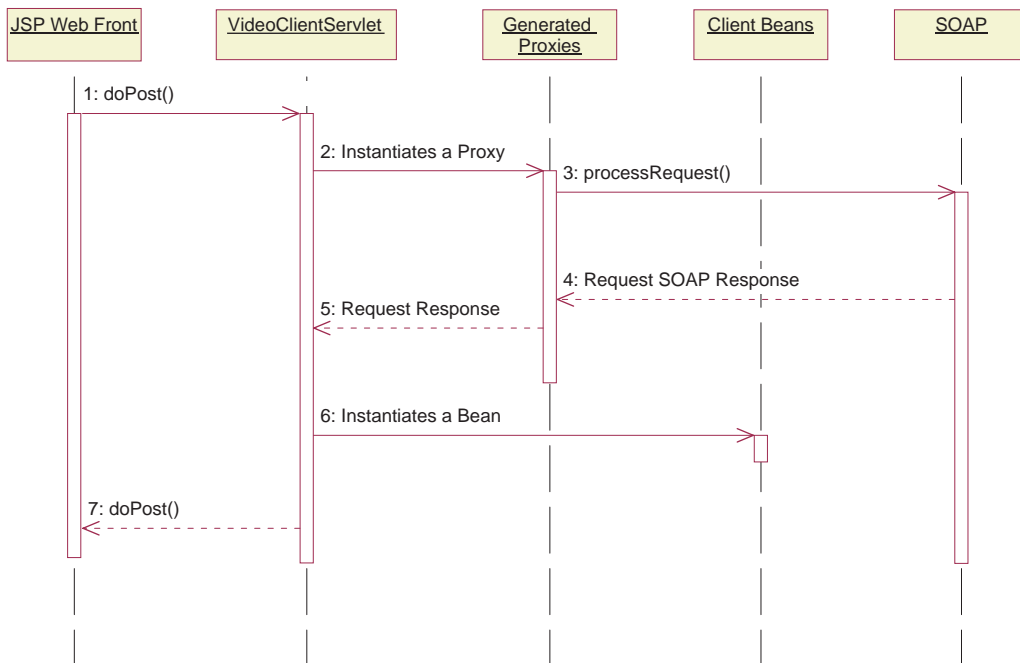


Figure 31. Sequence diagram for Video Central sample client, Video Client

In the Video Client, all requests are sent to VideoClientServlet. When processing each request, VideoClientServlet invokes the corresponding proxy request, which then invokes the remote Web service. By using the information returned in the response, VideoClientServlet instantiates a client Bean. Finally, the results are displayed to the end-user using JavaServer Page.

Video Central tasks

You will now learn how to:

- Create the required proxy classes
- Deploy the client code to WebSphere Application Server

Task 1: Preparing the client project

Within WebSphere Studio Application Developer perform the following steps:

1. Create the VideoClient Web project:
 - a. In the Web perspective, highlight the **VideoCentral** Web project (This is the highest level in the tree in WebSphere Studio Application Developer).
 - b. Create a Web project by clicking on **File** → **New** → **Web Project**.
 - c. Name the Web project VideoClient and the Enterprise Application Project name VideoClientEAR.

- d. Click **Next**.
- e. You will now see the Web Settings screen. Keep all the defaults and Click **Next**.
- f. You should now be at the **Java Settings** screen. Select Libraries panel. Click on **Add External JARs** and in the pop-up **JAR Selection** window add the following external jars in this order:
 - <WAS_install_dir>\lib\xerces.jar
 - <WSAD_install_dir>\plugins\com.ibm.etools.webservice\runtime\soap.jar

xerces.jar is required by the customer infraction Web service requester because infractions are manipulated as XML objects in memory and xerces is an XML Document Object Model (DOM) parser provided by IBM.

You should now see the following (see Figure 32).

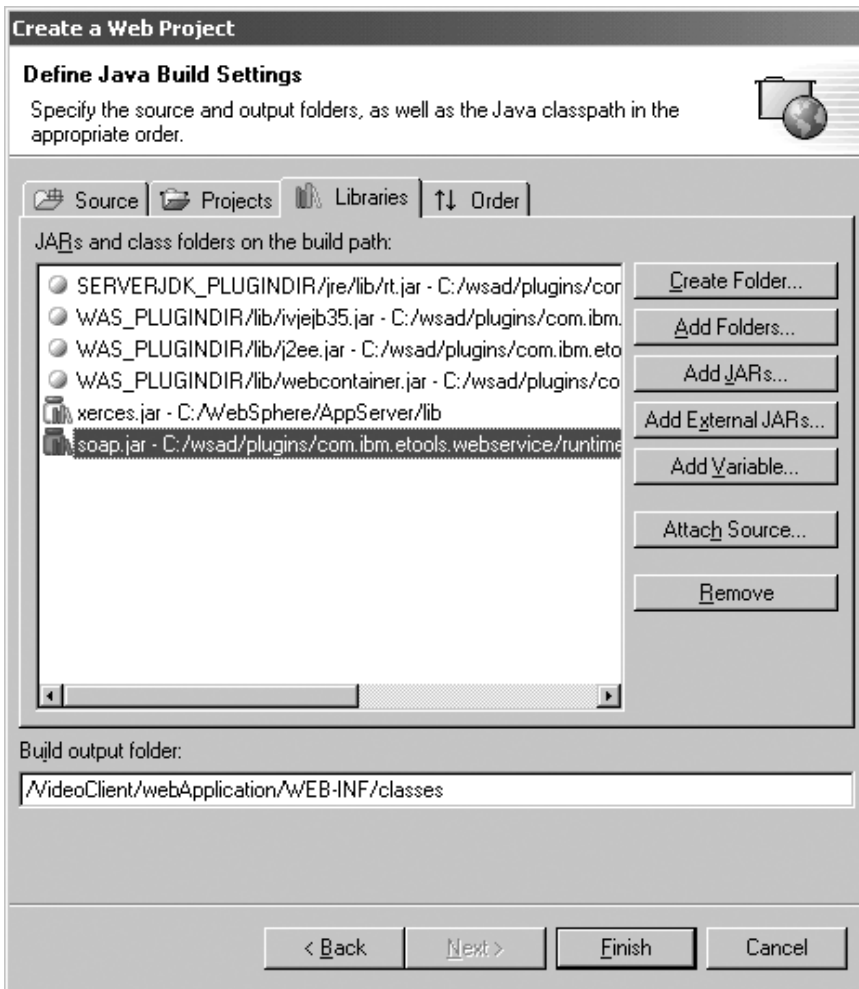


Figure 32. Add external JAR files

- g. Click **Finish**.
2. Add VideoClient to the server:
 - a. In the Server Configuration panel, expand **Server Configurations** and right click on **VideoCentralServer**. Select **Add Project** → **VideoClientEAR** (see Figure 33).

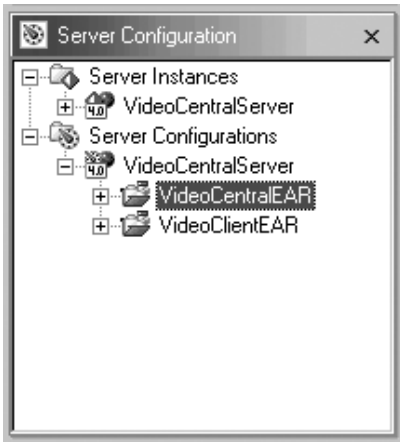


Figure 33. Adding the client Web project to the server

- b. In the Navigator panel, right click on the **VideoClient** project and select **Properties** → **Server Preference**. Highlight the **VideoCentralServer** in the box.
 - c. Click **Apply** and then click **OK**.
 3. Import the client code to VideoClient:
 - a. Expand your newly created VideoClient Web project and highlight **source** for the Web project (see Figure 34).

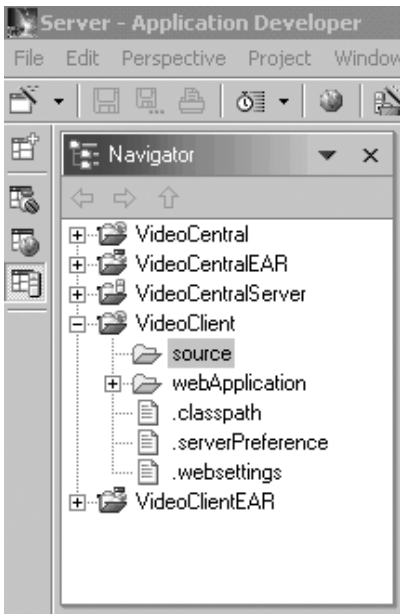


Figure 34. Highlight source

- b. Click **File** → **Import**.
- c. For the Import Source, select **ZIP file**. Click **Next**.
- d. Browse for
<DB2_install_dir>\tutorials\VideoCentral\lib\VideoClient.jar.
Click **OK**. You are now on the **ZIP file** screen (see Figure 35).

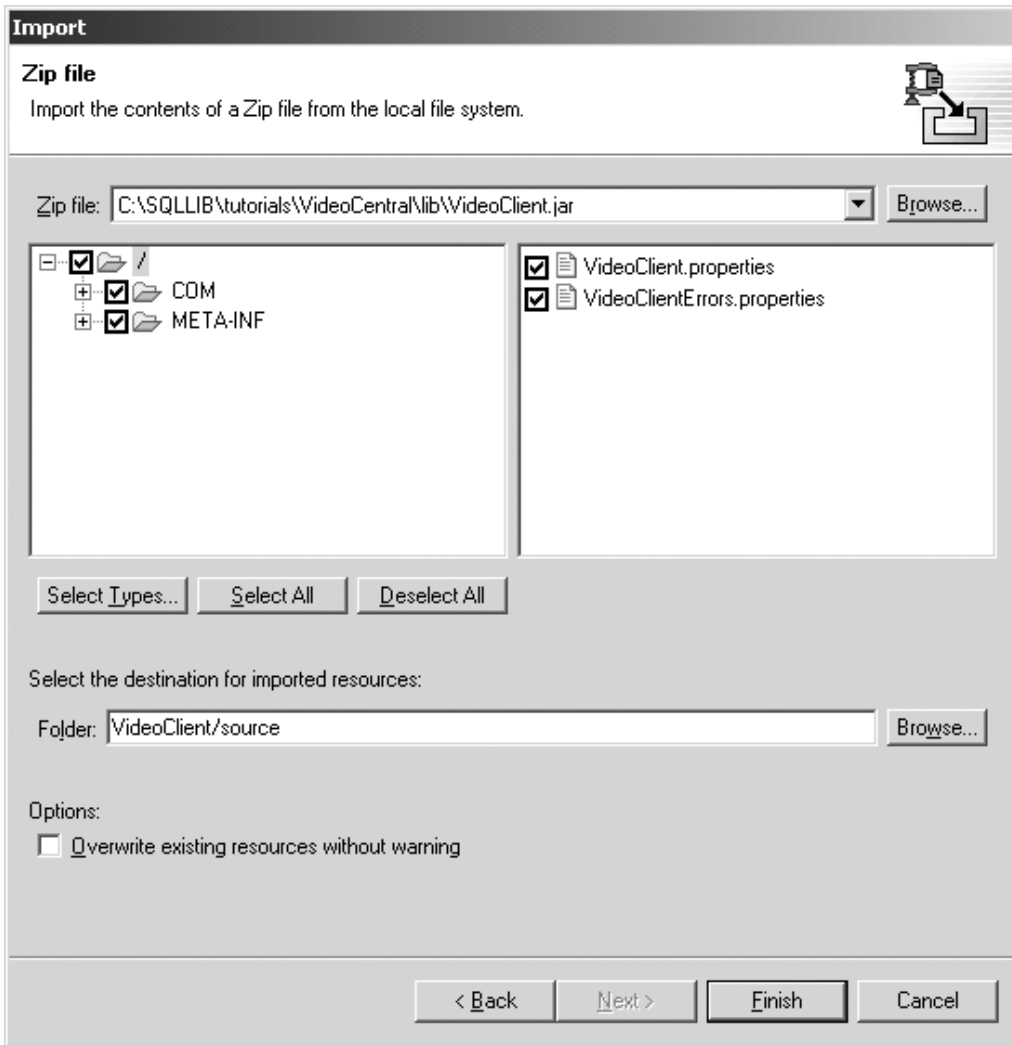


Figure 35. Importing the client code from a JAR file into VideoClient project

- e. Click **Finish**. The files will now be imported into the WebSphere Studio Application Developer and you will see on the action bar that the files are being compiled.
4. Import the client JSPs:
 - a. Expand your VideoClient Web project and highlight **webApplication** for the Web project (see Figure 36).

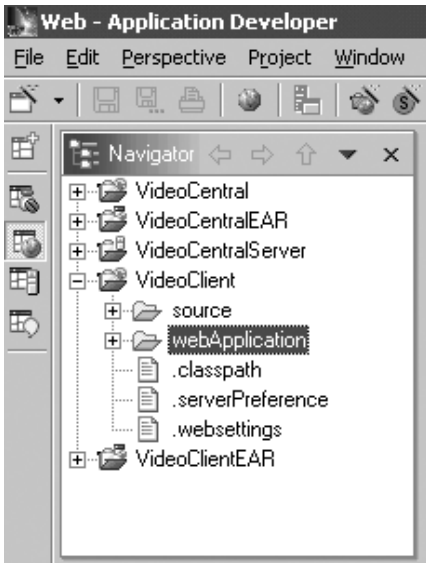


Figure 36. Highlight webApplication

- b. Click **File** → **Import**.
- c. For the Import Source, select **File System**. Click **Next**.
- d. Browse for <DB2_install_dir>\tutorials\VideoCentral\jsp and then click **OK**. You are now on the File System screen. Click **Select All** (see Figure 37).

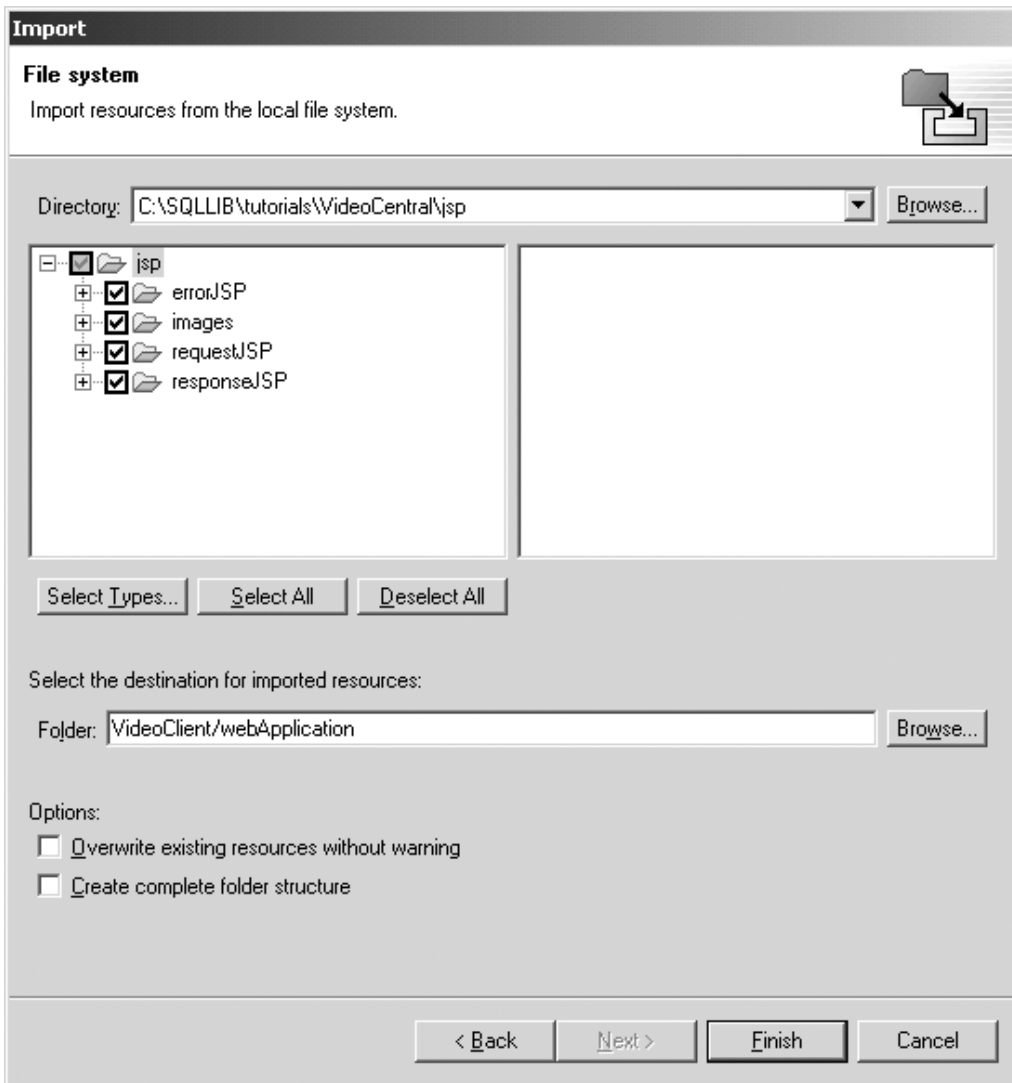


Figure 37. File system screen

- e. Click **Finish**.

The files will now be imported into WebSphere Studio Application Developer and you will see on the action bar that the files are being compiled.

Once the files are imported into the VideoClient project, you should have a structure like the following (see Figure 38):

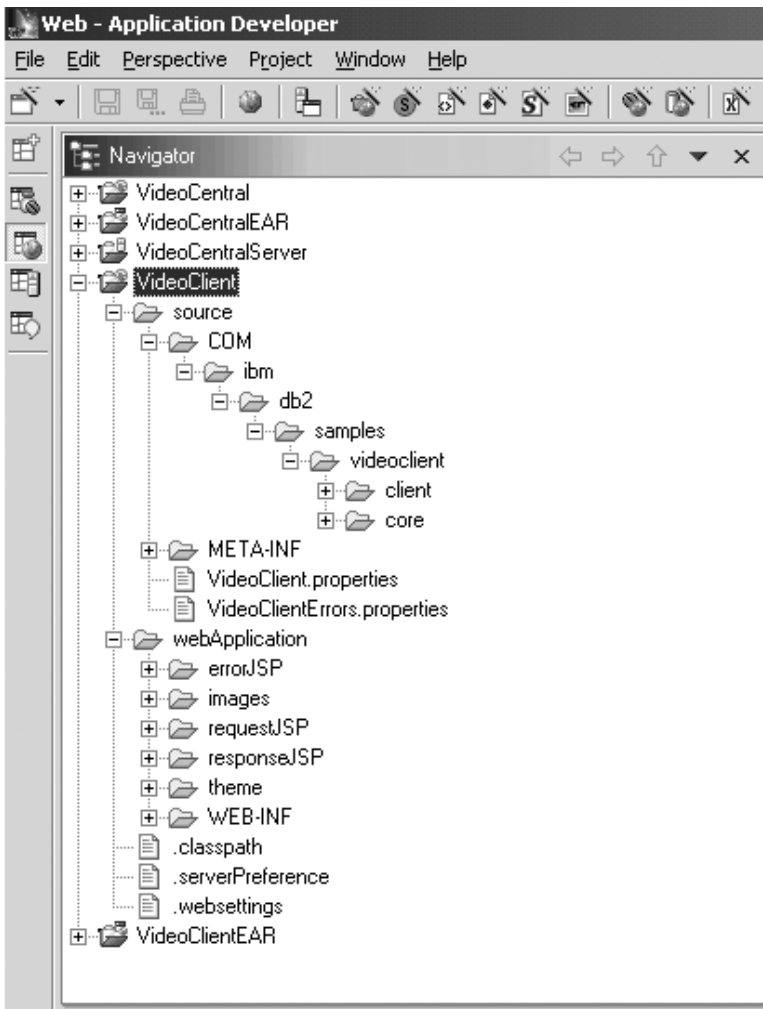


Figure 38. Successful import of Video Central client, VideoClient

Task 2: Creating the proxies using the WebSphere Studio Application Developer

You must have completed Chapter 4 to accomplish this section. The WSDL files created in the previous section are required to perform this task.

1. From the Web perspective, under the Web project VideoClient, right click **source/COM/ibm/db2/samples/videoclient**. Select **New** → **Folder**, and create a new folder named proxies. The container should show VideoClient/source/COM/ibm/db2/samples/videoclient/proxies. Click **Finish**.

2. Generate the proxy for the BusinessRegistration service.
 - a. From the Web perspective, expand **webApplication\wsdl** under the VideoCentral Web project and right click **BusinessRegistration-service.wsdl** → **New** → **Other**. In the pop-up window select **Web Services** → **Web Service Client**. Click **Next**.
 - b. Click **Next** again until you reach the **Web Service Binding Proxy Generation** pop-up window.
 - c. In the **Folder** field, replace `/VideoCentral/source` with `/VideoClient/source`.
 - d. In the **Class** field, replace `proxy.soap.BusinessRegistrationProxy` with:
`COM.ibm.db2.samples.videoclient.proxies.BusinessRegistrationProxy`

Note: For your convenience during the creation of the remaining proxies, copy this path into your clipboard. You can use it to paste into the path for the subsequent proxies.

You should now see the following (see Figure 39).

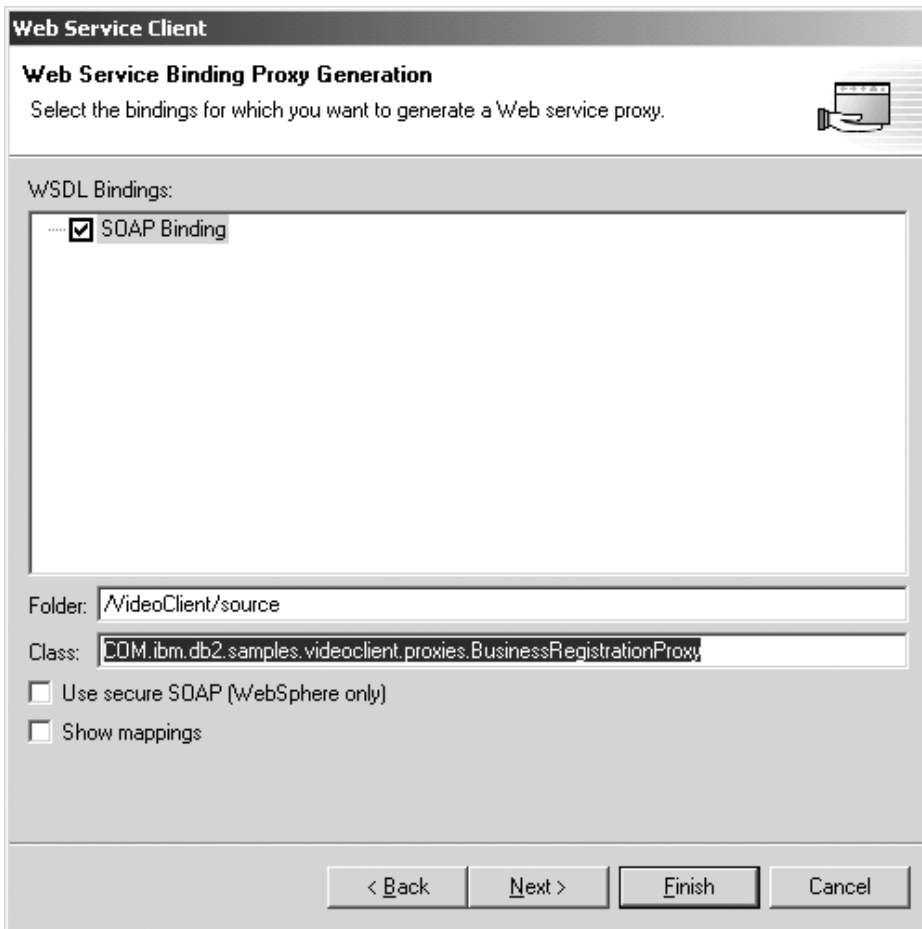


Figure 39. Generating the BusinessRegistration proxy

e. Click **Finish**.

Repeat **Task 2, Step 2** for the remaining five WSDL files:

- CustomerRegistration-service.wsdl
- CustomerInfraction-service.wsdl
- CustomerRentedList-service.wsdl
- WishList-service.wsdl
- MovieSearch-service.wsdl

You should see the proxy Java files in your VideoClient Web project. You should no longer see any compiler errors in WebSphere Studio Application

Developer except for some possible warnings, which are acceptable (see Figure 40).

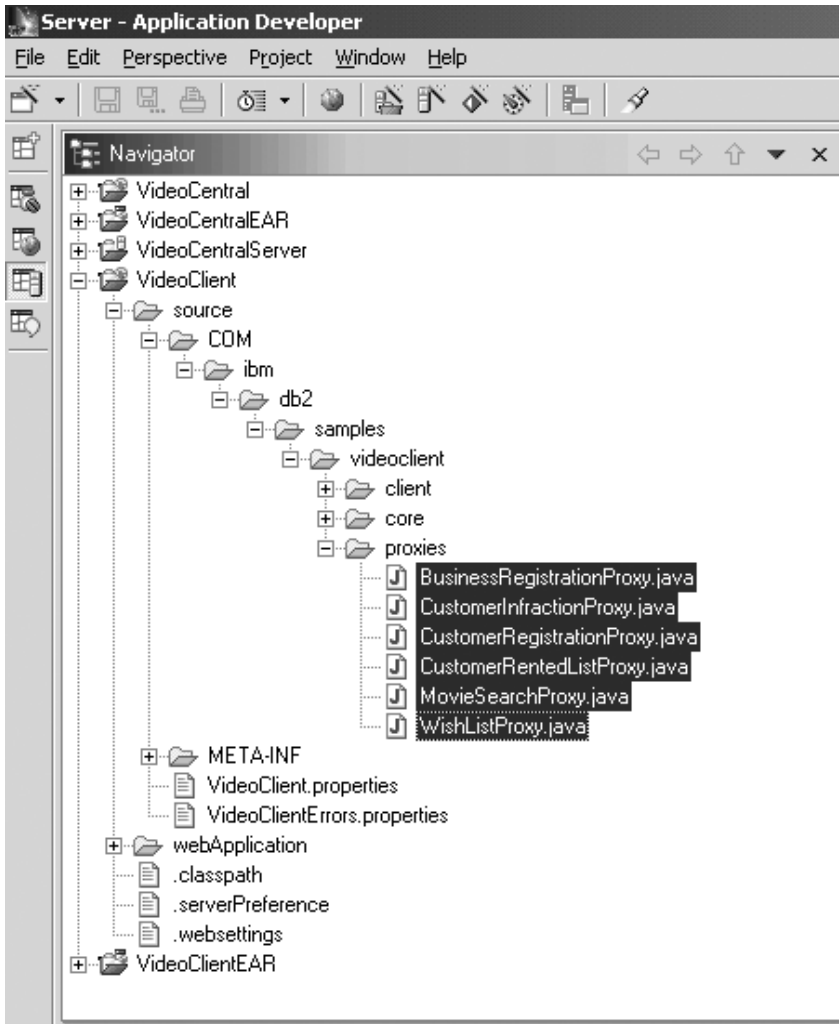


Figure 40. Proxies created for all WSDL files

Task 3: Deploying the sample client to the WebSphere Application Server

1. In the Web perspective, expand the **VideoClient\webApplication\WEB-INF** folder.
2. In the **Web-INF** folder, open the **web.xml** file.
3. Select the **Servlets** tab.
4. Add the **VideoClientServlet**:
 - a. Click the **Add** button and select the **VideoClientServlet** and click **Ok**.

- b. Highlight the **VideoClientServlet**, and in the **URL mappings** field click **Add** and type in `servlet/VideoClient` where it says (New URL). You should now see the following (see Figure 41).

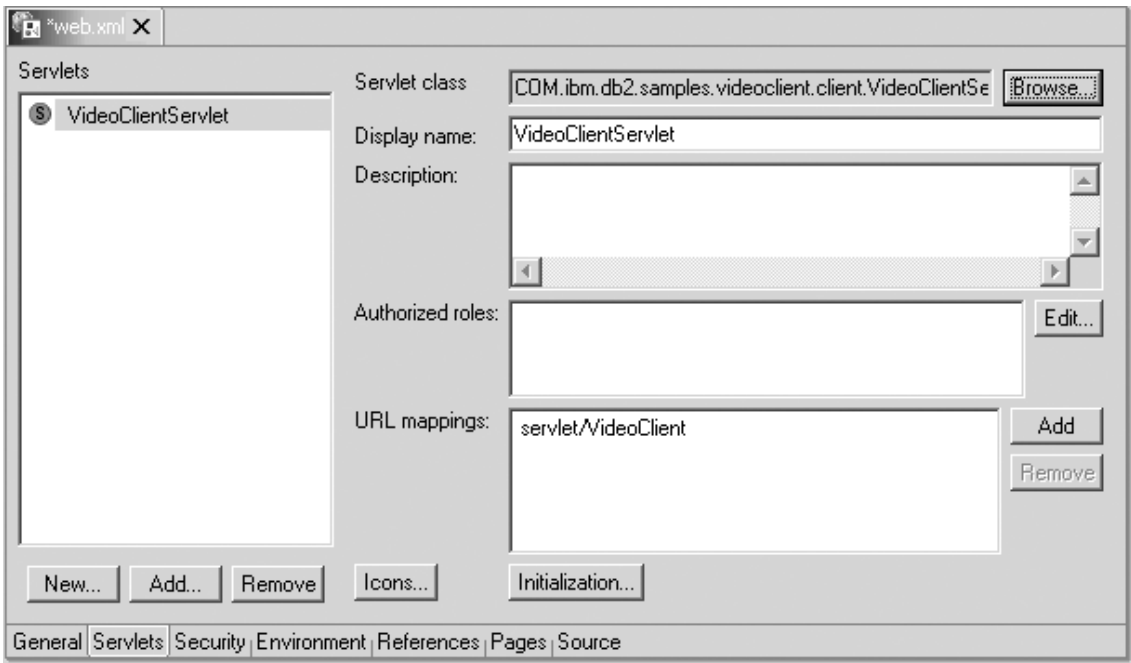


Figure 41. Adding the client servlets

5. Click **File** → **Save web.xml**.
6. Close **web.xml** window.

Task 4: Running the sample client

1. To access the client, open a Web browser, and go to `http://localhost:9080/VideoClient/requestJSP/VideoClientRequest.jsp`
2. Now you can proceed and use the client. To start using the Video Central services, a business must be registered. Since you do not have any customers registered, the next step is to perform a Customer Registration. Adding a Customer infraction will use the XML Extender. After adding a customer infraction, you can perform a query of infractions. If you are using an XML enabled Web browser, the XML document containing infractions will be displayed in the browser. Now, feel free to try out the other services as you will.

Note: The file `info.vc` will be created in your `c:\` directory. The current design of Video Central reads and writes this file to register the client businesses. We recognize that this is not the best approach and will be

modifying this design in future phases. Since Video Central is a business-to-business application, there is no context available between Web services invocation. Using the `info.vc` file enabled us to perform requests from a business (client) without asking for an ID and password for each request. Ideally, a trusted security authorization token would be encoded for each of the Web services requests.

Lesson summary

In this chapter you have completed:

- Overview of Video Central sample client design
- Creation of the proxy classes
- Deployment of the client code
- Running the client

Chapter 8. Solving common problems

1. If the deployment of code from WebSphere Studio Application Developer to WebSphere Application Server is not successful, check the following: WebSphere installation path may be incorrect. The default on the **Java bean Web Service Deployment** screen in WebSphere Studio Application Developer is C:\WebSphere\AppServer. Change it to the proper installation directory for WebSphere Application Server in your environment.
2. If you get the error message "Launching the server failed: Trace service port 7000 is in use. ORB bootstrap port 900 is in use. Change each used port number to another unused port on the ports page of the server configuration editor. In case you have another WebSphere server running you can try to increase each used port number by one and try again", do the following:

In WebSphere Studio Application Developer, select the Servers panel in the Servers perspective. Right click on **VideoCentralServer** within this panel and select **Stop**.
3. When you get any error while you access the Web services provided by Video Central from the browser, the first thing you need to do is checking if the server instance's Server Status is "Server is synchronized". You may need to re-publish Web projects within WebSphere Studio Application Developer.
4. If you get error message "SOAPException: faultCode=SOAP-ENV:Client; msg=Error opening socket: Connection refused: no further information; targetException=java.lang.IllegalArgumentException: Error opening socket: Connection refused: no further information]" while you access Video Central Web services, check if the port number used by the corresponding proxy class to communicate with the service is same as the port number that the SOAP runtime is listening to (in this tutorial, the port number is 9080).
5. If you get the error message like "DES7300N Index has not been activated, SQL0438N", check the entry for Net Search indexes location, `mvnets.inxdir`, in `VideoCentral.properties` to make sure that it has the correct path. You may also need to execute the following commands:
 - `db2nx update index name database sample`
 - `db2nx update index plot database sample`
6. To debug any application error in WebSphere Studio Application Developer, the server console log in the Servers panel provides useful information for problem resolution.

Chapter 9. Additional information

- IBM DB2: <http://www.ibm.com/software/data/db2/>
- XML Extender documentation:
<http://www.ibm.com/software/data/db2/extenders/xmlext/library.html>
- Net Search Extender documentation:
<http://www.ibm.com/software/data/db2/extenders/netsearch/download.html>
- SOAP : <http://www.w3.org/TR/SOAP/>
- Java programming: <http://www.javasoft.com>
- WSDL: <http://www.w3.org/TR/wsdl>
- Web services information:
<http://www.ibm.com/developerworks/webservices/>
- IBM DeveloperWorks - XML: <http://www.ibm.com/developerworks/xml/>
- WebSphere Developer Domain: <http://www7b.boulder.ibm.com/wsdd/>
- DB2 Developer Domain: <http://www7b.boulder.ibm.com/dmdd/>
- DB2 Web services: <http://www.ibm.com/software/data/webservices/>
- WebSphere/DB2 integration:
<http://www.ibm.com/software/data/integration/WebSphere>

Appendix A. Reference information

What is the DB2 XML Extender?

The XML Extender helps you integrate the power of IBM's DB2 Universal Database with the flexibility of XML. DB2's XML Extender provides the ability to store and access XML documents, generate XML documents from existing relational data, and shred (decompose, storing untagged element or attribute content) XML documents into relational data.

XML Extender provides new data types, functions, and stored procedures to manage your XML data in DB2. You also have the ability to decompose and store XML in its component parts as columns in multiple tables. Indexes can be defined over the element or attribute of an XML document for fast retrieval. In addition, text search and section search can be enabled on the XML column or its decomposed part using Text Extenders.

You can also formulate an XML document from your existing DB2 tables for data interchange in a business-to-business environment. Net.Data and XML Extender can be used to generate XML documents from DB2, and these documents can be distributed to consumers for viewing with a browser.

XML Extender provides the following features to help you manage and exploit XML data with DB2:

- Administration tools to help you manage the integration of XML data in relational tables.
- Storage and usage methods for your XML data: XML column, XML collection.
- A Document Type Definition (DTD) repository (DTD_REF) for you to store DTDs used to validate XML data.
- A mapping scheme called the Document Access Definition (DAD) file for you to map XML documents to relational data. DAD files are managed using the XML_USAGE table, created when you enable a database for XML.
- Powerful user-defined functions (UDFs) to store and retrieve XML documents in XML columns, as well as to extract XML element or attribute values. A UDF is a function that is defined to the database management system and can be referenced thereafter in SQL queries. The XML Extender provides the following types of UDFs:
 - Storage: Stores intact XML documents in XML-enabled columns as XML data types.

- Extract: Extracts XML documents, or the specified elements and attributes as base data types.
- Update: Updates entire XML documents or specified element and attribute values.

All the XML Extender's UDFs have the prefix `db2xml`, which is the schema name of the DB2 XML Extender UDFs.

What is the DB2 Net Search Extender?

The DB2 Net Search Extender adds the power of fast full-text retrieval engine to Net.Data, Java, or DB2 CLI applications integrated into the DB2 Universal Database.

Net Search Extender provides fast indexing of very large data volumes, allows search at high speed with a large number of concurrent users, and stores presorted table columns in main memory at indexing time to avoid expensive database access and paging at search time.

Net Search Extender provides the following features that compliment the functionality of DB2 Text Extender and DB2 Text Information Extender:

- Indexing:
 - One very fast index type. (Ngram)
 - Multiple indexes on the same text column are possible.
 - Indexing proceeds without locking data.
 - Dynamic updating of indexes, reflecting changes in the database.
- Search:
 - Provided a stored procedure on the instead of UDFs.
 - Allows word, phrase, stemmed, or fuzzy search.
 - Identifies and restricts searching to sections in the documents that have been marked by special tags.
 - Offers numeric search on a range of values.
 - Supports Boolean and wildcard operations.
- Search result:
 - Lets you specify how the search results are sorted at indexing time, or user supplied rank values for sorting.
 - Lets you specify search result subsets when large data volumes are searched and large result lists are expected.
 - Lets you set a limit on search terms with a high hit count.
 - Allows positioning (cursor setting) access on search results.

What is the IBM WebSphere Studio Application Developer?

IBM's WebSphere Studio Application Developer (We will also refer to this as WSAD in this document) is the follow-on technology for WebSphere Studio, Professional and Advanced Editions and VisualAge[®] for Java, Enterprise Edition. These products support end-to-end development, testing, and deployment of e-business applications.

The new WebSphere Studio products are designed from the ground up to meet the requirements for all new types of applications. These requirements include open standards, Java, XML, Web services, testing, varying levels of integration with other components and ISV products, pluggability, expandability, role-based development, increased usability for all users, enhanced team support, as well as increased speed to market. WebSphere Studio provides integrated development tools for all e-business development roles, from Web developers to Java developers to business analysts to architects to enterprise programmers.

WebSphere Studio brings together the most popular features of WebSphere Studio "Classic" and VisualAge for Java and combines them with the advantages of our latest technology, providing open standards, tool integration, more flexibility, and the ability to tie in existing applications.

About the Web services environment

WebSphere Studio provides wizards and other tools to enable rapid development of Web services.

The Web services development tools provided in WebSphere Studio are based on open, cross-platform standards:

- ***Universal Description Discovery and Integration (UDDI)***: Enables businesses to describe themselves, publish technical specifications on how they want to conduct e-business with other companies, and search for other businesses that provide goods and services they need, all via online UDDI registries
- ***Simple Object Access Protocol (SOAP)***: Is a standard for reliably transporting electronic business messages from one business application to another over the Internet

SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML-based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses.

SOAP can potentially be used in combination with a variety of other protocols; however, the only bindings defined in this document describe how to use SOAP in combination with HTTP and HTTP Extension Framework.

For the SOAP Version 1.1 Specification and additional SOAP reference, go to <http://www.w3.org/TR/SOAP/>

- **Web Services Description Language (WSDL):** Describes programs accessible via the Internet (or other networks), and the message formats and protocols used to communicate with them.

WebSphere Studio facilitates the following processes to assist with building and deploying Web services-enabled applications:

- **Discover.** Browse the UDDI Business Registry to locate existing Web services for integration.
- **Create or Transform.** Create Web services from existing artifacts, such as beans, URLs that take and return data, DB2 XML Extender calls, DB2 stored procedures, and SQL queries.
- **Build.** Wrap existing artifacts as SOAP and HTTP GET/POST accessible services and describe them in WSDL. The Web services wizards assist you in generating a SOAP proxy to Web services described in WSDL and in generating bean skeletons from WSDL.
- **Deploy.** Deploy Web services in the WebSphere Application Server or Tomcat test environments using Server Tools.
- **Test.** Test Web services running locally or remotely to get instant feedback.
- **Develop.** Generate sample applications to assist you in creating your own Web Service client application.
- **Publish.** Publish Web services to the UDDI Business Registry, advertising your Web services so that other businesses can access them.

For more information on Web services tools, consult the online help that comes with the installation of WebSphere Studio Application Developer.

About the XML Development Environment

WebSphere Studio provides a comprehensive XML development environment that includes tools for building DTDs, XML schemas, and XML files. It also supports integration of relational data and XML.

The following XML tools are available:

The XML editor is a tool for creating, viewing, and validating XML files. You can use it to create new XML files, either from scratch, existing DTDs, or existing XML schemas. You can also use it to edit XML files, associate them with DTDs or schemas, and validate them.

The DTD editor is a tool for creating, viewing, and validating DTDs. Using the DTD editor, you can create and validate DTD elements, attributes, entities, and notations. You can generate XML schema files, and generate Java beans for creating XML instances of an XML schema. You can also use the DTD editor to generate a default HTML form based on the DTDs you create.

The XML schema editor facilitates creating, viewing, and validating XML schemas. You can use the XML schema editor to perform tasks such as creating XML schema components, importing and viewing XML schemas, generating DTDs and relational table definitions from XML schemas, generating Java beans for creating XML instances of an XML schema, and generating DDL from an XML schema.

The XSL trace editor allows you to apply an XSL stylesheet against an XML document to create a result document (HTML or XML). You can transform XML documents into HTML, text, or other XML document types. The editor displays the three documents (result, source XML, source XSL) and enables you to visually step through the XSL transformation script, examining the relationships between the three documents.

The XML to XML mapping editor maps one or more source XML documents to a single target XML document. You can provide a source file (DTD or XML) and a target file and define the mappings between the source and the target. Each mapping is a selection of a target field, a conversion function and source fields. Mappings can be edited, deleted or stored for later use. After defining the mappings you can generate an XSLT script, which can then be used to combine and transform any XML documents that conform to the source DTDs.

The XML and SQL query wizard is used to create an XML file from the results of an SQL query. You can optionally create an XML schema or DTD file that describes the structure that the XML file has for use in other applications. You can also use the XML and SQL Query wizard to create a DADX file that can be used with the Web services tool. The generated DADX file will contain your SQL query.

The RDB to XML mapping editor makes it easy to define the mapping between relational tables and a DTD file. You can map columns in one or more relational tables to elements and attributes in an XML document. You can generate a document access definition (DAD) script, used by IBM DB2 Extender, to either compose XML documents from existing DB2 data, or decompose XML documents into DB2 data. You can also create a test harness to test the generated DAD file.

For more information about XML tools, consult the online help that comes with the installation of WebSphere Studio Application Developer.

Appendix B. Glossary

DAD - Document Access Definition. Used to define the indexing scheme for a XML column or a mapping scheme of a XML collection.

DTD - Document Type Definition. A DTD is a file (or several files to be used together), written in XML's Declaration Syntax, which contains a formal description of a particular type of document. It sets out what names can be used for element types, where they may occur, and how they all fit together.

JDBC - Java Database Connectivity. An Application Programming Interface (API) that has the same characteristics as Open Database Connectivity (ODBC) but is specifically designed for use by Java database applications.

SOAP - Simple Open Access Protocol. A protocol specification that defines a uniform way of passing XML-encoded data.

UDDI - Universal Description, Discovery and Integration Service. A registry mechanism for clients to dynamically find other web services.

WSDL - Web Services Description Language. An XML format that describes what a Web service can do, where it resides, and how to invoke it.

XML - eXtensible Markup Language. It is the universal format for structured documents and data on the Web. It is extensible because it is not a fixed format like HTML. It is a subset of SGML, the Standard Generalized Markup Language.

XSL - eXtensible Style Language. Defines the standard stylesheet language of XML.

Appendix C. Notices

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

- DB2
- IBM
- VisualAge
- WebSphere

Microsoft and Windows NT are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

Contacting IBM

If you have a technical problem, please review and carry out the actions suggested by the Solving Common Problems in Section 8.0 and the DB2 Troubleshooting Guide before contacting DB2 Customer Support. The DB2 Troubleshooting Guide suggests information that you can gather to help DB2 Customer Support to serve you better.

For information or to order any of the DB2 Universal Database products contact an IBM representative at a local branch office or contact any authorized IBM software remarketer.

If you live in the U.S.A., then you can call one of the following numbers:

- 1-800-237-5511 for customer support
- 1-888-426-4343 to learn about available service options

Product information

DB2 Universal Database product information is available by telephone or by the World Wide Web at <http://www.ibm.com/software/data/db2/udb/winos2unix/support>

This site contains the latest information on the technical library, ordering books, client downloads, newsgroups, FixPaks, news, and links to web resources.

If you live in the U.S.A., then you can call one of the following numbers:

- 1-800-IBM-CALL (1-800-426-2255) to order products or to obtain general information.
- 1-800-879-2755 to order publications.

For information on how to contact IBM outside of the United States, go to the IBM Worldwide page at <http://www.ibm.com/planetwide>

In some countries, IBM-authorized dealers should contact their dealer support structure for information.



Printed in U.S.A.