BEA Liquid Data for WebLogic™

Getting Started

Version 8.1
Document Date: July 2003
Revised: July 2004

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Contents

About This Document

Getting Started with Liquid Data

About This Document

This document provides quick start steps that explain how to start the BEA Liquid Data for WebLogic server, bring up the WebLogic Server Administration Console and use it to configure some data sources, use the Data View Builder to map source and target schemas and construct a query, and run the query against the server.

What You Need to Know

This document is intended mainly for business users/developers and XML data designers who will be using Liquid Data to design distributed data solutions and construct queries. System Administrators will also find the initial steps useful as a snapshot example of the configuration process.

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation or go directly to the “e-docs” Product Documentation page at http://e-docs.bea.com.

How to Print the Document

You can print a copy of this document from a Web browser, one file at a time, by using the File—>Print option on your Web browser.

A PDF version of this document is available on the Liquid Data documentation Home page on the e-docs Web site (and also on the documentation CD). You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format. To access the PDFs, open the Liquid Data documentation Home page, click the PDF files button and select the document you want to print.

If you do not have the Adobe Acrobat Reader, you can get it for free from the Adobe Web site at http://www.adobe.com/.

Related Information

For more information in general about Java and XQuery, refer to the following sources.

- The Sun Microsystems, Inc. Java site at:

  `http://java.sun.com/`

- The World Wide Web Consortium XML Query section at:

  `http://www.w3.org/XML/Query`

For more information about BEA products, refer to the BEA documentation site at:

`http://edocs.bea.com/`

Contact Us!

Your feedback on the BEA Liquid Data documentation is important to us. Send us e-mail at **docsupport@bea.com** if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the Liquid Data documentation.

In your e-mail message, please indicate that you are using the documentation for the BEA Liquid Data for WebLogic 1.0 release.

If you have any questions about this version of Liquid Data, or if you have problems installing and running Liquid Data, contact BEA Customer Support through BEA WebSupport at **www.bea.com**. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
boldface text	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
italics	Indicates emphasis or book titles.

`monospace text`	Indicates code samples, commands and their options, data structures and their members, data types, directories, and file names and their extensions. Monospace text also indicates text that you must enter from the keyboard.
	Examples:
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `( )`
`monospace italic text`	Identifies variables in code. *Example*: `String` *`expr`*
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Example*s: LPT1 SIGNON OR

Convention	Item
`{ }`	Indicates a set of choices in a syntax line. The braces themselves should never be typed.

`[ ]`	Indicates optional items in a syntax line. The brackets themselves should never be typed.
	Example:
	`buildobjclient [-v] [-o name ] [-f file-list]...`
	`[-l file-list]...`

|	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.

`...`	Indicates one of the following in a command line:
	• That an argument can be repeated several times in a command line
	• That the statement omits additional optional arguments
	• That you can enter additional parameters, values, or other information
	The ellipsis itself should never be typed.
	Example:
	`buildobjclient [-v] [-o name] [-f file-list]...`
	`[-l file-list]...`

. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

Getting Started with Liquid Data

You can use this 10-step tutorial guide to learn how to:

- Configure data sources in Liquid Data
- Associate data sources with a target schema
- Generate and run queries in the Data View Builder
- Build a BEA WebLogic Workshop application around Liquid Data queries
- Display formatted query results as part of a web application

From Multiple Data Sources to a Web Application in 60 Minutes (or Less)

This chapter illustrates how you can use Liquid Data to create a query that treats multiple disparate data sources as a single data source. It assumes a large Internet company that has grown through mergers or acquisitions and, as a consequence, finds that its real-time order data must be accessed from two very separate data sources: Wireless order data from an RDBMS system; BroadBand order data via an XML document.

As you work through this Order Report example, keep in mind that each action you take generates source code, in some cases significant amounts. When you are done, you will find that your application contains nearly 500 lines of Java code, 97% of which was automatically generated. (For details see "Generated Files and Code" on page 54.)

Getting Started contains numerous illustrations in the form of screen captures. The goal is to enable you to create a query accessing multiple data sources and use that query when building and running a simple but complete web application — in less than 60 minutes!

The 10 steps in building up the Order Report web application are:

- Step 1. Start the Liquid Data Samples Server
- Step 2. Start the WebLogic Administration Console
- Step 3. Configure Data Sources
- Step 4. Start Data View Builder and Verify Data Sources
- Step 5. Construct Your Query (View a Demo)
- Step 6. View and Test Your Query
- Step 7. Save Your Liquid Data Project
- Step 8. Deploy Your Query
- Step 9. Creating a Liquid Data Control in WebLogic Workshop
- Step 10. Creating a Liquid Data-Powered Web Application

Example directories

Examples used in this chapter assume that the Liquid Data 8.1 sample domain is located at:

```
<WL_HOME>/samples/domains/liquiddata/
```

where `WL_HOME` is the root location of the WebLogic Server 8.1.

You can find many other examples and samples in *Liquid Data by Example*, including the Avitek Customer Self-Service Sample Application which also combines the back-end data retrieval capabilities of Liquid Data with WebLogic Workshop to create a more fully-functional customer self-service web application.

Step 1. Start the Liquid Data Samples Server

The Liquid Data Samples server is simply a WebLogic Server 8.1 that contains Liquid Data samples, including a Liquid Data repository. It can be started from Windows or UNIX.

Windows

To start the Liquid Data samples server under Windows choose the following menu option:

```
Start —> Programs —> BEA WebLogic Platform 8.1 —>
  BEA Liquid Data for WebLogic 8.1 SP2 —> Liquid Data Samples —> Samples Server
```

A terminal window called Samples Server will appear. When the Samples server is ready, you will see `Server started in RUNNING mode`.

UNIX

To start the Liquid Data samples server under UNIX run the `startWeblogic.sh` shell command in the following domain:

```
<WL_HOME>/samples/domains/liquiddata/startWeblogic.sh
```

When the Samples server is ready you will see `Server started in RUNNING mode`.

Note: For more information on setting up and starting the Samples server, see Post-Installation Tasks in the Liquid Data *Installation Guide*. For more information on starting Liquid Data servers in the various preconfigured domains, see Starting and Stopping the Server in the *Liquid Data Administration Guide*.

Step 2. Start the WebLogic Administration Console

The Liquid Data repository is managed and configured through the Liquid Data node of the WebLogic Administration Console. It is here that data sources are defined, security policies established, and so forth.

Starting the Console

The WebLogic Administration Console can be accessed from Windows or UNIX.

Windows

To start the WebLogic Administration Console under Windows choose the following menu option:

```
Start —> Programs —> BEA WebLogic Platform 8.1 —> BEA Liquid Data for WebLogic
8.1 SP2 —> Liquid Data Samples —> Admin Console
```

UNIX

To start the WebLogic Administration Console for the Liquid Data Samples server running on your local machine, enter the following URL in a web browser:

```
http://localhost:7001/console
```

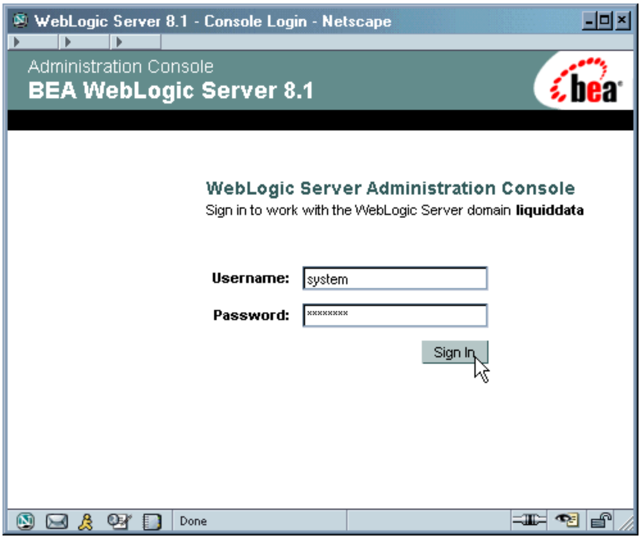
Logging Into the Administration Console

Log in to the console by providing the following default username and password for the Samples server:

Table 1 User Name and Password required by the Liquid Data node of the WebLogic Administration Console

Field	Defaults
Username	`system`
Password	`security`

Figure 1 Entering WebLogic Administration Console Login Information



WebLogic Server Administration Console

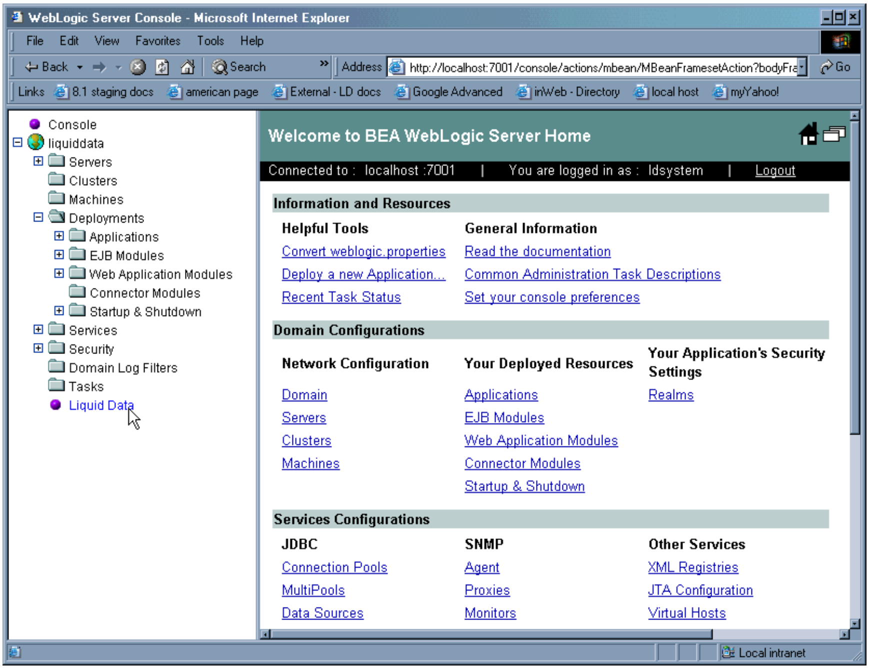
Sign in to work with the WebLogic Server domain **liquiddata**

Username: system

Password: ********

After you log in, the Console home page is displayed. Note that Liquid Data is the bottom element in the left pane (Figure 2).

Figure 2 WebLogic Administration Console



Note: For more information on using the Liquid Data node of the WebLogic Administration Console, see the *Liquid Data Administration Guide*.

Step 3. Configure Data Sources

This section describes how to use the WebLogic Administration Console to configure the two sample data sources used in the tutorial customer order report:

- A relational database data source description for customer and order information.
- An XML data source description for order information.

Configuring a Relational Data Source Description

A first step in making a data source available to Liquid Data is to register your data source with the Administration Console. To add a description of your RDBMS data source to the Liquid Data Server registry follow these steps:

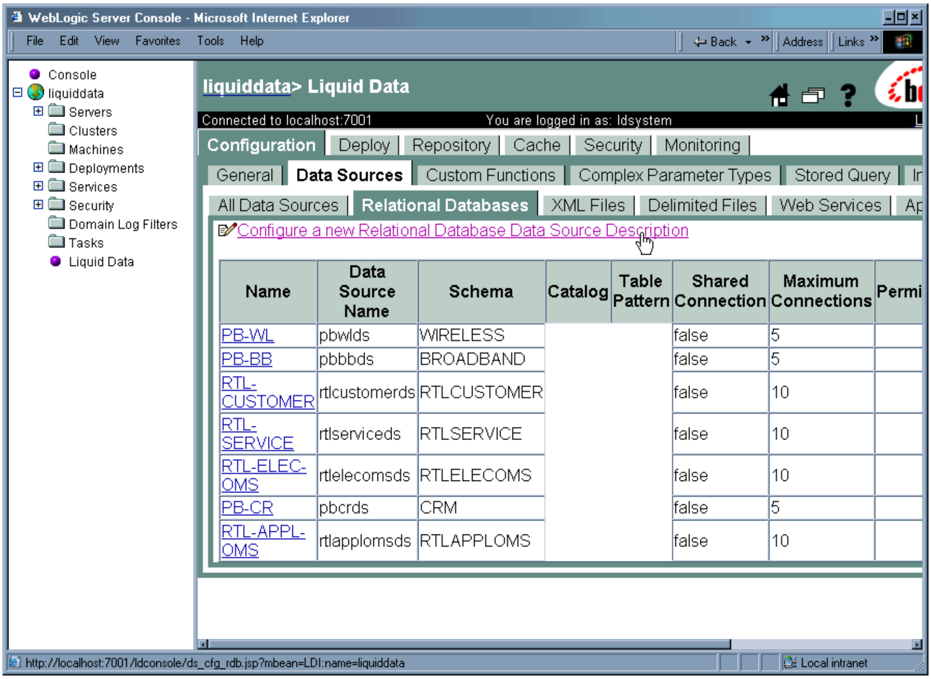
1. In the left pane, click the Liquid Data node (bottom icon in the left pane; see Figure 2).

2. In the right pane, click the Data Sources tab (under Configuration).

3. Click the Relational Databases tab.

4. Click the `Configure a New Relational Database Data Source Description` link to display the Configure tab (Figure 4).

Figure 3 Creating a New Relational Database Data Source



Name	Data Source Name	Schema	Catalog	Table Pattern	Shared Connection	Maximum Connections	Permi
PB-WL	pbwlds	WIRELESS			false	5	
PB-BB	pbbbds	BROADBAND			false	5	
RTL-CUSTOMER	rtlcustomerds	RTLCUSTOMER			false	10	
RTL-SERVICE	rtlserviceds	RTLSERVICE			false	10	
RTL-ELEC-OMS	rtlelecomsds	RTLELECOMS			false	10	
PB-CR	pbcrds	CRM			false	5	
RTL-APPL-OMS	rtlapplomsds	RTLAPPLOMS			false	10	

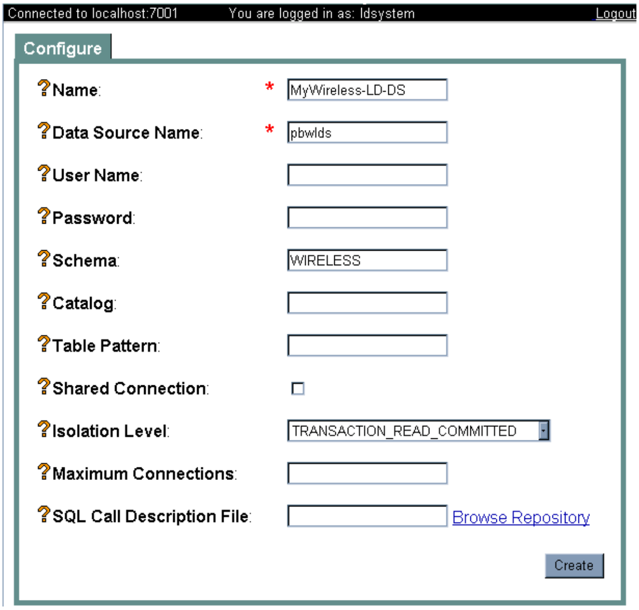
5. Fill in the fields for your data source as described in Table 2:

Table 2 Liquid Data Source Description for PB-WL Data Source

Field	Value to Enter
Name	`MyWireless-LD-DS` This is the logical name of the data source. (For this example, it is highly recommended that you use suggested names.)
Data Source Name	`pbwlds` This is the data source name; it matches a JNDI name for a JDBC data source in the Samples server.
Schema	`WIRELESS` The schema name must match the name of a schema stored in your data source. In this case, we are using the WIRELESS schema provided in the Samples database.
Namespace URI	Not needed for this basic example.

6. Click Create. If you look in the list of relational database sources, you will find your newly created data source definition. (For additional information see Configuring Access to Relational Databases in the *Liquid Data Administration Guide*.)

Figure 4 A Relational Database Data Source Defined



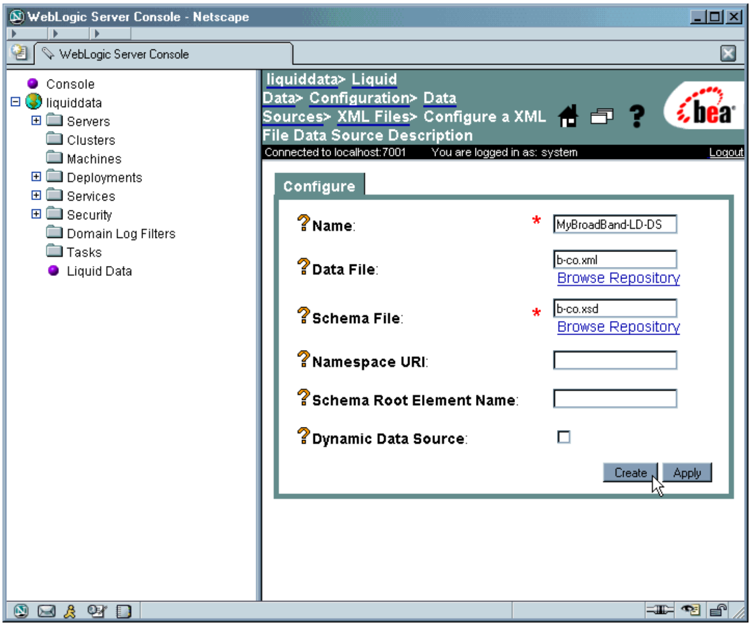
?Name:	*	MyWireless-LD-DS
?Data Source Name:	*	pbwlds
?User Name:		
?Password:		
?Schema:		WIRELESS
?Catalog:		
?Table Pattern:		
?Shared Connection:		□
?Isolation Level:		TRANSACTION_READ_COMMITTED
?Maximum Connections:		
?SQL Call Description File:		

Configure an XML File Data Source Description

To create a description of your XML data source, follow these steps:

1. In the left pane, click the Liquid Data node.
2. Click the XML Files tab (next to Relational Databases).
3. Click the `Configure a New XML Data Source Description` link to access the Configure dialog box (Figure 5).

Figure 5 Defining a XML Data Source Definition


WebLogic Server Console - Netscape
WebLogic Server Console
Console
liquiddata
Servers
Clusters
Machines
Deployments
Services
Security
Domain Log Filters
Tasks
Liquid Data
liquiddata> Liquid Data> Configuration> Data Sources> XML Files> Configure a XML File Data Source Description
bea
Connected to localhost:7001 You are logged in as: system
Logout
Configure

Name: * MyBroadBand-LD-DS

Data File: b-co.xml Browse Repository

Schema File: * b-co.xsd Browse Repository

Namespace URI:

Schema Root Element Name:

Dynamic Data Source: □

4. Fill in the fields for your XML data source as described in Table 3.

Table 3 Liquid Data Data Source Description for My-PB-WL Data Source

Field	Value to Enter
Name	`MyBroadBand-LD-DS` This is the name of the data source description used to register the XML data source with the Liquid Data server.
Data File	`b-co.xml` This is the XML data file name. It must correspond to an XML data file located in the Liquid Data server repository. (The example file name provided above resides in the Samples server repository.)
Schema File	`b-co.xsd` Schema associated with the XML file. The schema you specify here must be in the Liquid Data server repository.
Namespace URI	Not needed for this basic example.

5. Click Create. If you look in the list of XML data sources, you will find your newly created data source definition. (For additional information see Configuring Access to XML Files in the *Liquid Data Administration Guide*.)

Step 4. Start Data View Builder and Verify Data Sources

Next, you are ready to start the Windows-based Data View Builder. This is where you will create and test your query. You can start the Data View Builder two ways:

1. Choose:

```
Start ——> Programs ——> BEA WebLogic Platform 8.1 ——> BEA Liquid Data for WebLogic
8.1 SP2 ——> Data View Builder
```

2. Or you can start the Data View Builder by double-clicking on the file:

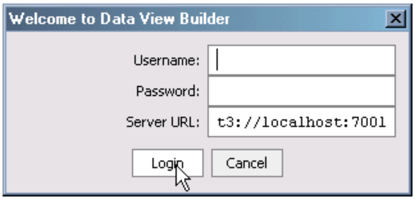
```
<WL_HOME>\liquiddata\DataViewBuilder\bin\DVBuilder.cmd
```

If you are not already connected to the Liquid Data Samples server, enter the following address in the Server URL field in the Welcome... dialog box (Figure 6).

```
t3://localhost:7001
```

No username or password is needed; leave those fields blank.

Figure 6 Data View Builder Login Dialog



Welcome to Data View Builder

Username:

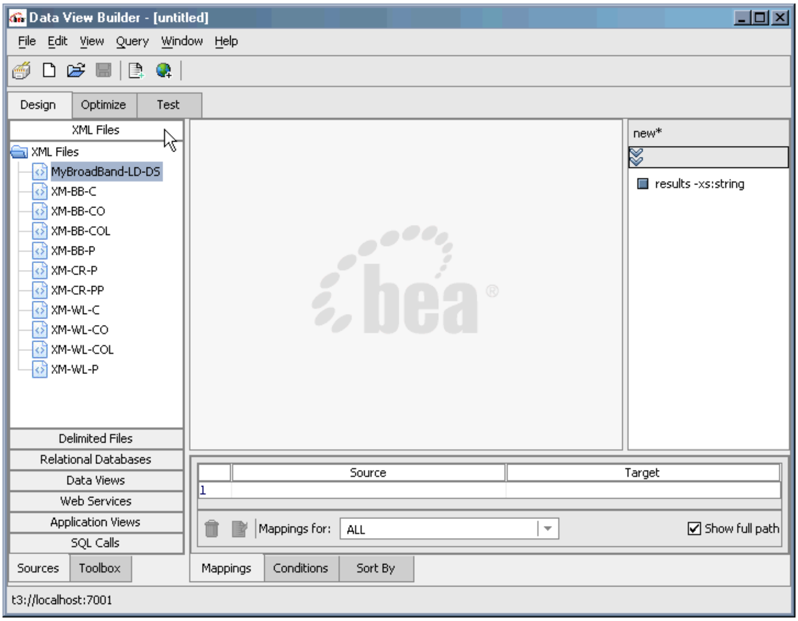
Password:

Server URL: t3://localhost:7001

3. Click the Login button.

   The Data View Builder workspace and tools appear. You can click on the buttons on the left Navigation panel to view the various data sources available from the Liquid Data server to which the Data View Builder is connected.

Figure 7 Initial View of Data View Builder Workspace



Mappings for: ALL

☑ Show full path

4. Once configured, your data sources are immediately available. In the Sources area:
   - Click Relational Databases to view available RDBMS data sources. This list should include your new relational database data source: `MyWireless-LD-DS`
   - Click XML Files to view available XML data sources. This list should include your new XML data source: `MyBroadBand-LD-DS`

Step 5. Construct Your Query

The Data View Builder allows you to build queries (and *data views*) through graphical representations of your data and any query conditions.

You first define your query by dragging and dropping (*mapping*) elements from source to target schemas. Constants, parameters, and functions are available to fully express a query statement.

The customer order report query you are building assumes that you have customer orders stored in two data sources (Wireless in a relational database and BroadBand in an XML file). The query will return the order information for any particular customer.

To construct the example customer order report query, you need to:

- Open Relevant Data Source Schemas
- Select a Target Schema From the Sample Liquid Data Repository
- Map Source Elements to the Target Schema
- Define Query Conditions
- Create a Query Parameter

View a Demo

Constructing the "Order Query" Demo... If you are looking at this documentation online, you can click the Demo button to view an animated demonstration of how to build up the conditions and create the mappings that are similar to this example. With minor variations, the Order Query demo previews Steps 1-7 of this chapter.

Open Relevant Data Source Schemas

To start building a query, first select the schemas for the data sources you need.

1. Open the data source schemas from the Builder toolbar as follows:
   - In the XML Files area double-click on `MyBroadBand-LD-DS` to move the associated schema into your work area.
   - Click Relational Databases and double-click on `MyWireless-LD-DS` to open the associated schema.

   The schemas for the each of the data sources appear.

Note: Move data sources around on the workspace as needed.

Select a Target Schema From the Sample Liquid Data Repository

A target schema represents a structure of your query results. (For additional information on target schemas see "Understanding Target Schemas" in the Schemas and Namespaces in Liquid Data chapter of *Building Queries and Data Views*.)

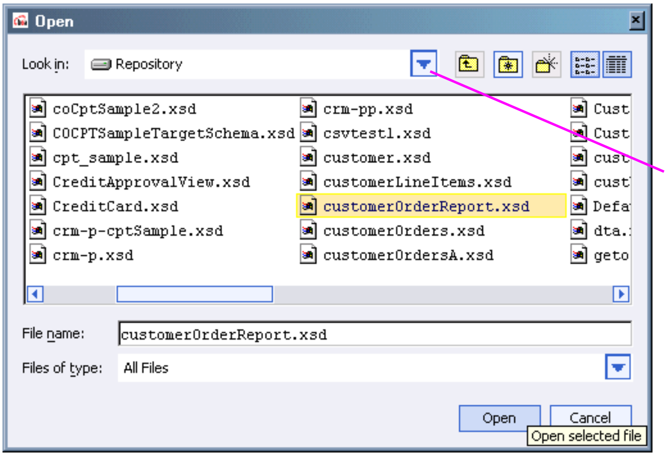
For this example, use `customerOrderReport.xsd` as your target schema.

To add the target schema:

1. Choose the menu command File —> Set Target Schema. This brings up the Liquid Data file browser.

2. Scroll to the Liquid Data Repository, the topmost item in the scroll list.

Figure 8 Selecting a Target Schema from the Liquid Data Repository



Look in: Repository

coCptSample2.xsd
COCPTSampleTargetSchema.xsd
cpt_sample.xsd
CreditApprovalView.xsd
CreditCard.xsd
crm-p-cptSample.xsd
crm-p.xsd
crm-pp.xsd
csvtest1.xsd
customer.xsd
customerLineItems.xsd
customerOrderReport.xsd
customerOrders.xsd
customerOrdersA.xsd
Cust
Cust
cust
cust
Defa
dta.:
geto

File name: customerOrderReport.xsd

Files of type: All Files

Click the scroll button to select the top item, the Liquid Data Repository.

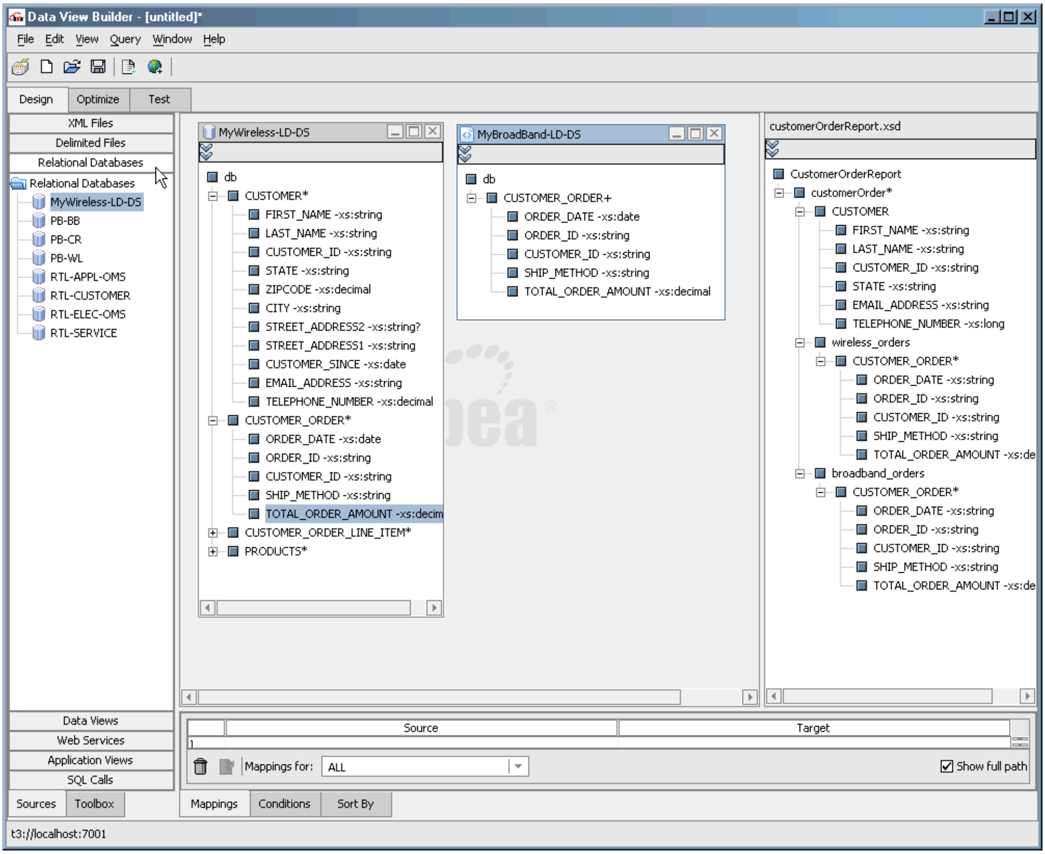
3. Select `customerOrderReport.xsd` and click Open to set this as your target schema.

4. In the target schema panel of the Data View Builder right-click on the topmost node of the target schema (CustomerOrderReport) and choose Expand to display all the schema elements (Figure 9).

Note: Schemas are located in the following Liquid Data repository directory:

```
<WL_HOME>/samples/domains/liquiddata/ldrepository/schemas
```

Figure 9 Data View Builder Showing Source and Target Schemas



At this point, your work area contains two data source schemas and a target schema.

Map Source Elements to the Target Schema

The Data View Builder generates queries based on your mapping of source elements to your target schema.

The CustomerOrderReport target schema includes a CUSTOMER_ORDER node for both Wireless and BroadBand. Map CUSTOMER_ORDER elements from each data source to the corresponding CUSTOMER_ORDER elements in the target schema as follows:

- Drag and drop all elements in the Wireless data source CUSTOMER_ORDER into the corresponding elements in the target schema's wireless_orders: CUSTOMER_ORDER node (Figure 10).
- Drag and drop all elements contained in the BroadBand data source CUSTOMER_ORDER into the corresponding elements in the target schema's wireless_orders: CUSTOMER_ORDER node (Figure 11).

Figure 10 Mapping Elements in Wireless : CUSTOMER_ORDER to the Target Schema

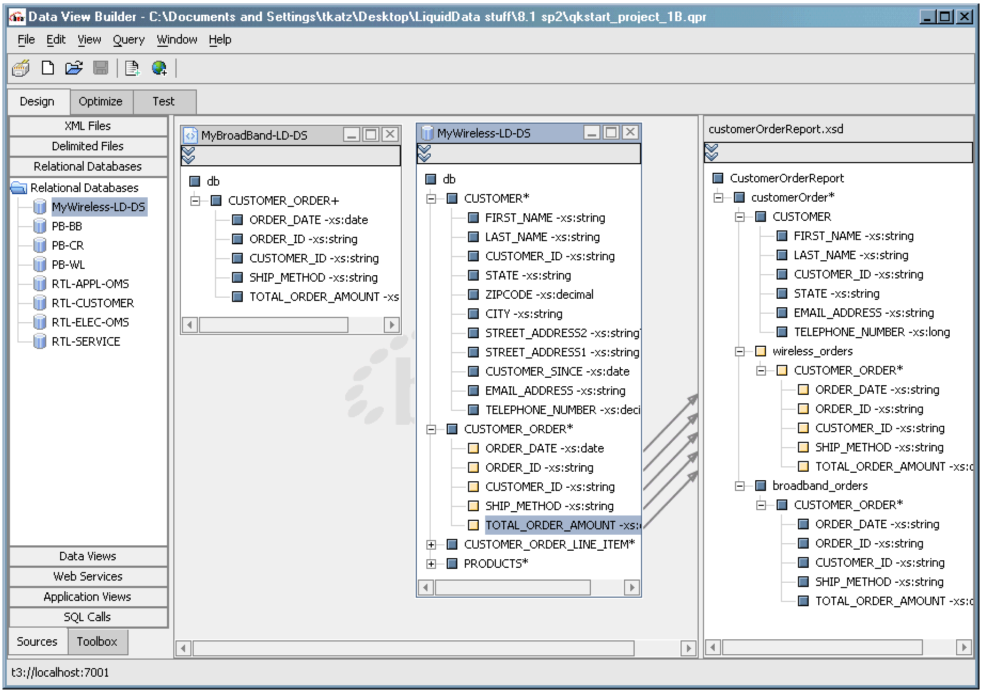
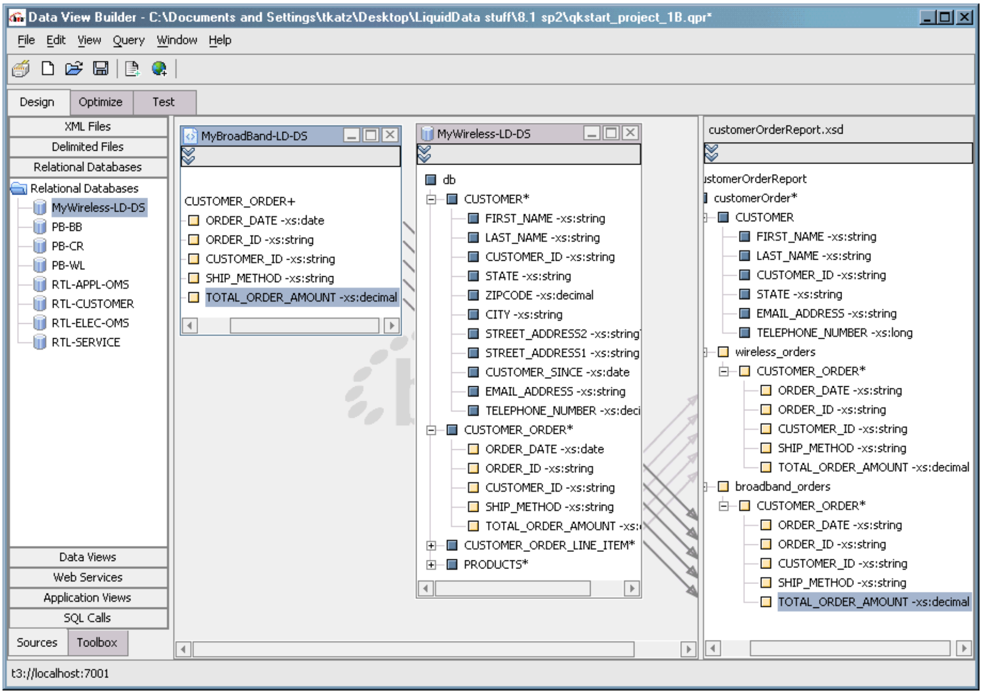


Figure 11 Mapping Elements in Broadband : CUSTOMER_ORDER to the Target Schema



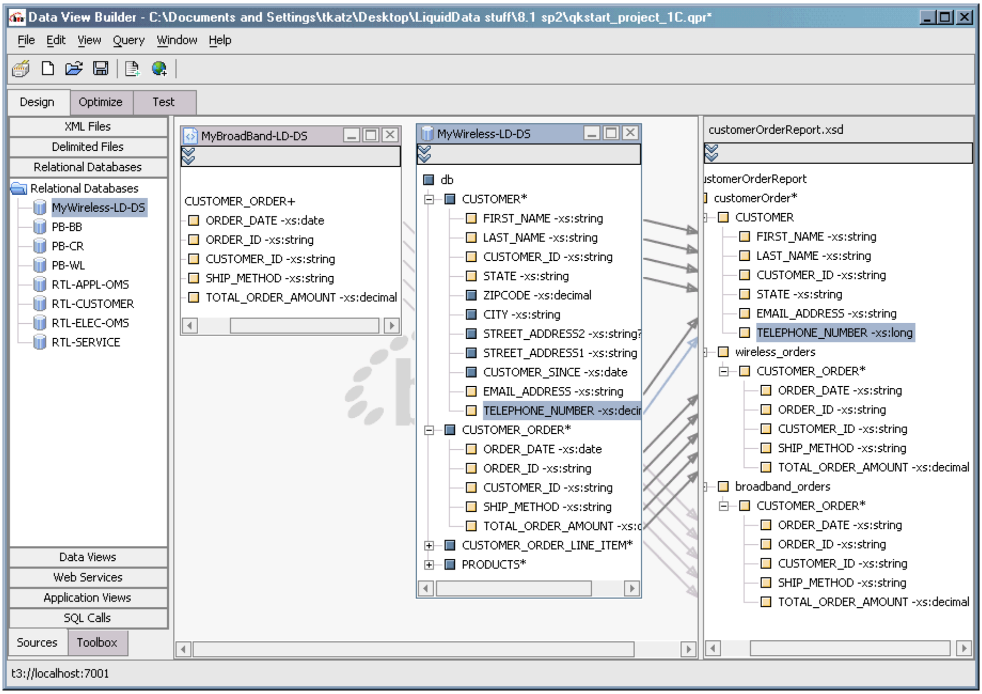
In both cases, you mapped the sub-elements: ORDER_DATE, ORDER_ID, CUSTOMER_ID, SHIP_METHOD, and TOTAL_ORDER_AMOUNT to corresponding elements in your target schema.

Note: As you map elements from source to target, descriptions appear in the lower part of the workspace (Figure 13). If you need to delete an incorrect mapping, click in the Mapping ID field to highlight the row you wish to delete. Then click the Trash icon on your lower left.

Use the Wireless data source schema to populate the CUSTOMER element in the target schema.

– Drag and drop the sub-elements contained in the Wireless data source CUSTOMER into the corresponding sub-elements in the target schema CUSTOMER node. As you can see, not all elements in a source schema need to be mapped to the target.

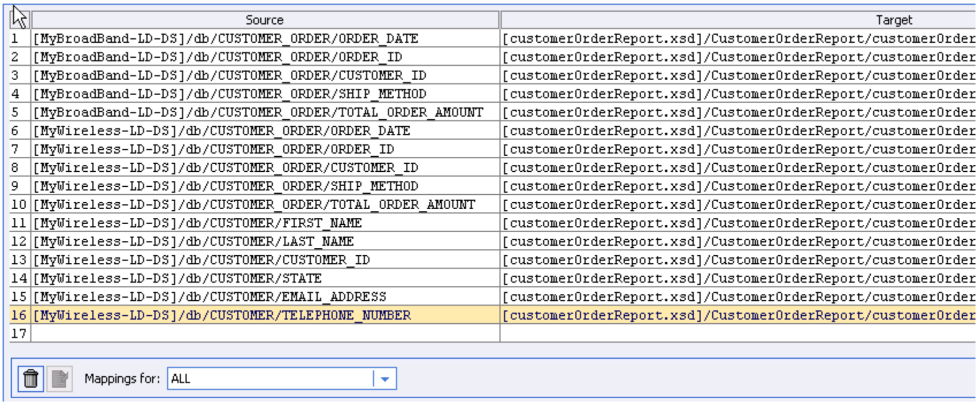
Figure 12 Mapping Elements in Wireless : CUSTOMER to the Target Schema



At this point, you have completed the source-to-target mappings and your target schema is ready.

The 16 source-to-target mappings you created appear on the Mappings tab (Figure 13).

Figure 13 Complete List of Source-to-Target Mappings



	Source	Target
1	[MyBroadBand-LD-DS]/db/CUSTOMER_ORDER/ORDER_DATE	[customerOrderReport.xsd]/CustomerOrderReport/customerOrder
2	[MyBroadBand-LD-DS]/db/CUSTOMER_ORDER/ORDER_ID	[customerOrderReport.xsd]/CustomerOrderReport/customerOrder
3	[MyBroadBand-LD-DS]/db/CUSTOMER_ORDER/CUSTOMER_ID	[customerOrderReport.xsd]/CustomerOrderReport/customerOrder
4	[MyBroadBand-LD-DS]/db/CUSTOMER_ORDER/SHIP_METHOD	[customerOrderReport.xsd]/CustomerOrderReport/customerOrder
5	[MyBroadBand-LD-DS]/db/CUSTOMER_ORDER/TOTAL_ORDER_AMOUNT	[customerOrderReport.xsd]/CustomerOrderReport/customerOrder
6	[MyWireless-LD-DS]/db/CUSTOMER_ORDER/ORDER_DATE	[customerOrderReport.xsd]/CustomerOrderReport/customerOrder
7	[MyWireless-LD-DS]/db/CUSTOMER_ORDER/ORDER_ID	[customerOrderReport.xsd]/CustomerOrderReport/customerOrder
8	[MyWireless-LD-DS]/db/CUSTOMER_ORDER/CUSTOMER_ID	[customerOrderReport.xsd]/CustomerOrderReport/customerOrder
9	[MyWireless-LD-DS]/db/CUSTOMER_ORDER/SHIP_METHOD	[customerOrderReport.xsd]/CustomerOrderReport/customerOrder
10	[MyWireless-LD-DS]/db/CUSTOMER_ORDER/TOTAL_ORDER_AMOUNT	[customerOrderReport.xsd]/CustomerOrderReport/customerOrder
11	[MyWireless-LD-DS]/db/CUSTOMER/FIRST_NAME	[customerOrderReport.xsd]/CustomerOrderReport/customerOrder
12	[MyWireless-LD-DS]/db/CUSTOMER/LAST_NAME	[customerOrderReport.xsd]/CustomerOrderReport/customerOrder
13	[MyWireless-LD-DS]/db/CUSTOMER/CUSTOMER_ID	[customerOrderReport.xsd]/CustomerOrderReport/customerOrder
14	[MyWireless-LD-DS]/db/CUSTOMER/STATE	[customerOrderReport.xsd]/CustomerOrderReport/customerOrder
15	[MyWireless-LD-DS]/db/CUSTOMER/EMAIL_ADDRESS	[customerOrderReport.xsd]/CustomerOrderReport/customerOrder
16	[MyWireless-LD-DS]/db/CUSTOMER/TELEPHONE_NUMBER	[customerOrderReport.xsd]/CustomerOrderReport/customerOrder
17		

Mappings for: ALL

Define Query Conditions

The purpose of the query is to retrieve BroadBand and Wireless transactions for a specified customer. The following sections describe how to define conditions for the project you are building:

- Create a Join Between Wireless : CUSTOMER and Wireless : CUSTOMER_ORDER
- Create a Join between Wireless : CUSTOMER_ID and Broadband : CUSTOMER_ID
- Create a Query Parameter

Create a Join Between Wireless : CUSTOMER and Wireless : CUSTOMER_ORDER

In the Wireless data source drag and drop the element CUSTOMER : CUSTOMER_ID to CUSTOMER_ORDER : CUSTOMER_ID to create a join between CUSTOMER and CUSTOMER_ORDER:

Join Element	Join Element
`[myWireless-LD-DS]/db/CUSTOMER/`**`CUSTOMER_ID`**	`[myWireless-LD-DS]/db/CUSTOMER_ORDER/`**`CUSTOMER_ID`**

The syntax of this join appears on the first row of the Conditions tab.

Note: Whenever you create a query condition, the Conditions tab is automatically displayed in the lower part of the workspace (Figure 14).

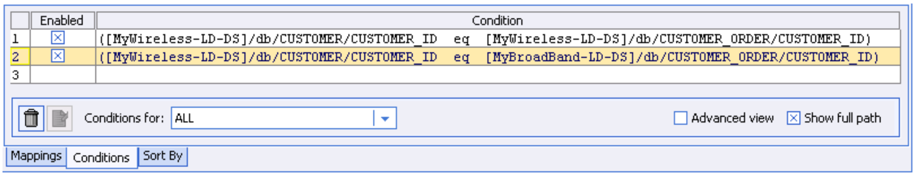
Create a Join between Wireless : CUSTOMER_ID and Broadband : CUSTOMER_ID

Drag and drop the Wireless element CUSTOMER : CUSTOMER_ID onto the BroadBand data source element CUSTOMER_ORDER : CUSTOMER_ID:

Join Element	Join Element
`[myWireless-LD-DS]/db/CUSTOMER/`**`CUSTOMER_ID`**	`[myBroadBand-LD-DS]/db/CUSTOMER_ORDER/`**`CUSTOMER_ID`**

The syntax of this join is displayed on the next row on the Conditions tab.

Figure 14 Query Conditions After Selection of Two Joins



	Enabled	Condition
1	☒	([MyWireless-LD-DS]/db/CUSTOMER/CUSTOMER_ID eq [MyWireless-LD-DS]/db/CUSTOMER_ORDER/CUSTOMER_ID)
2	☒	([MyWireless-LD-DS]/db/CUSTOMER/CUSTOMER_ID eq [MyBroadBand-LD-DS]/db/CUSTOMER_ORDER/CUSTOMER_ID)
3		

Conditions for: ALL

☐ Advanced view ☒ Show full path

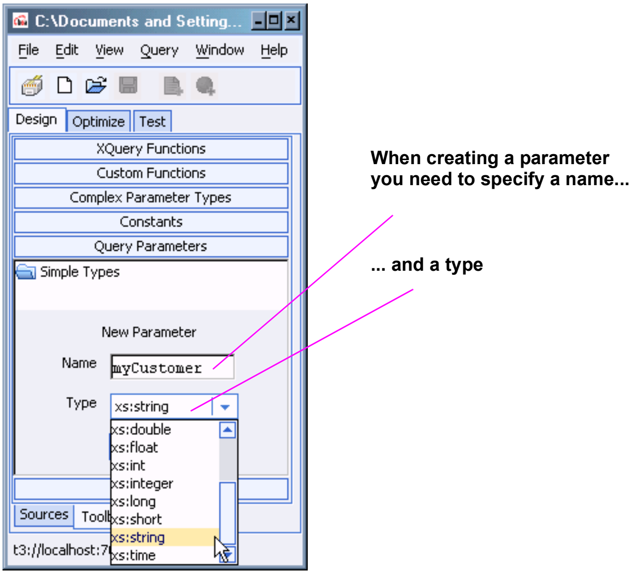
Note: If you run the query at this point, your results retrieve information on all customers in your customer database.

Create a Query Parameter

Although you could use a constant to “hard-code” your query for a particular customer, creating a query parameter is a more flexible approach. Using a query parameter, you can retrieve information on any customer in your data set.

1. On the Builder Toolbar (left navigation panel) click the Toolbox tab.
2. Select Query Parameters.
3. You need to name the parameter and specify a type. In this example, use `myCustomer` as the name and select `string` from the Type pulldown menu (see Figure 14).
4. Click Create.

Figure 15 Creating a New Parameter of Type String



Name myCustomer

Type xs:string

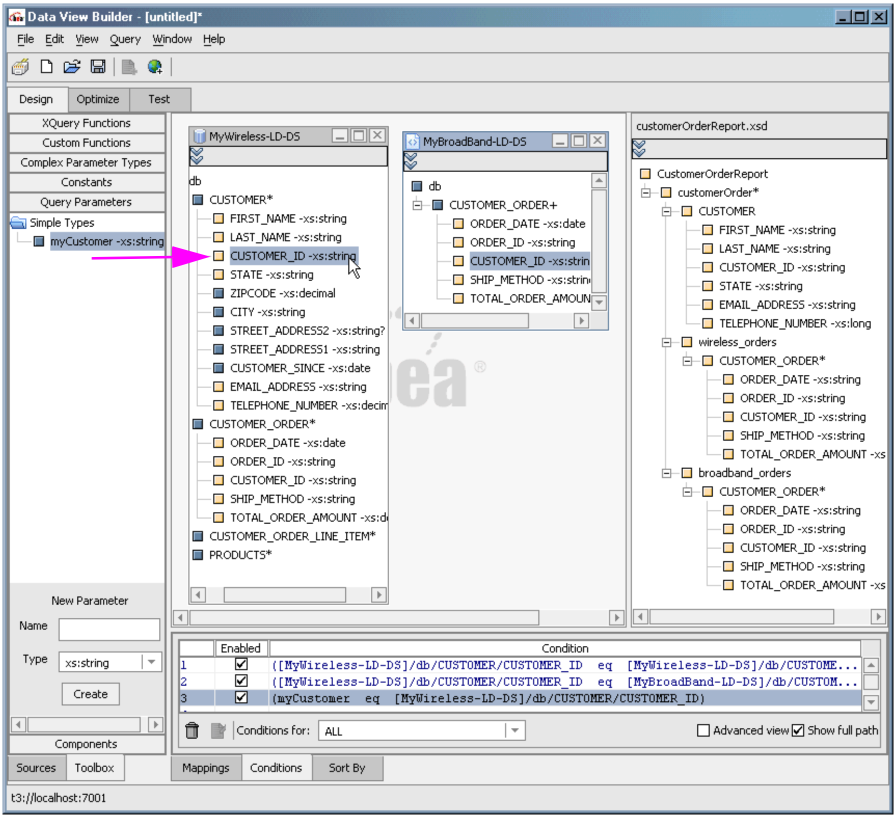
When creating a parameter you need to specify a name...

... and a type

5. Next you need to associate your new query parameter with a data element. Do this by dragging the `myCustomer` query parameter to the `CUSTOMER : CUSTOMER_ID` element of MyWirelessLD-DS database schema:

Join Element	Join Element
`[Query Parameter]` **`myCustomer`**	`[myWireless-LD-DS]/db/CUSTOMER/`**`CUSTOMER_ID`**

Figure 16 Mapping a Parameter to a Source Schema Element

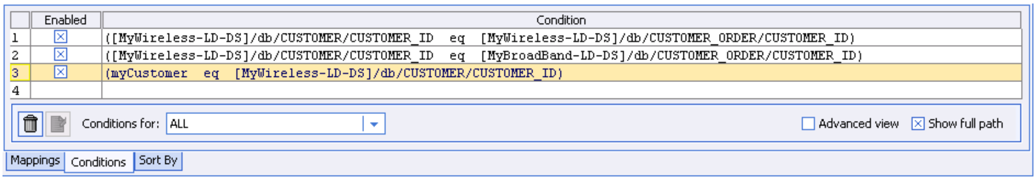


	Enabled	Condition
1	☑	([MyWireless-LD-DS]/db/CUSTOMER/CUSTOMER_ID eq [MyWireless-LD-DS]/db/CUSTOME...
2	☑	([MyWireless-LD-DS]/db/CUSTOMER/CUSTOMER_ID eq [MyBroadBand-LD-DS]/db/CUSTOM...
3	☑	(myCustomer eq [MyWireless-LD-DS]/db/CUSTOMER/CUSTOMER_ID)

Conditions for: ALL □ Advanced view ☑ Show full path

You have now set up all the source conditions needed for your query.

Figure 17 Complete Set of Conditions



	Enabled	Condition
1	☒	([MyWireless-LD-DS]/db/CUSTOMER/CUSTOMER_ID eq [MyWireless-LD-DS]/db/CUSTOMER_ORDER/CUSTOMER_ID)
2	☒	([MyWireless-LD-DS]/db/CUSTOMER/CUSTOMER_ID eq [MyBroadBand-LD-DS]/db/CUSTOMER_ORDER/CUSTOMER_ID)
3	☒	(myCustomer eq [MyWireless-LD-DS]/db/CUSTOMER/CUSTOMER_ID)
4		

Conditions for: ALL □ Advanced view ☒ Show full path

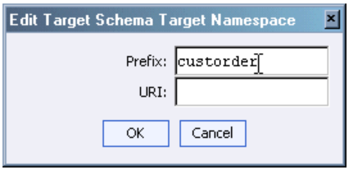
Set the Target Namespace

Although not necessary in all cases, it is good practice to set a namespace for your target schema. See "Understanding XML Namespaces" in the Overview and Key Concepts chapter of *Building Queries and Data Views.*

To set a target namespace for the customerOrderReport target schema follow these steps:

1. Click on the top node of the target schema (CustomerOrderReport).
2. Select Target Namespace from the Data View Builder Query menu (Query —> Target Namespace).

Figure 18 Specifying a Target Namespace



Edit Target Schema Target Namespace

Prefix: custorder

URI:

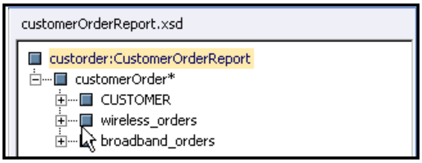
3. Enter values shown in Table 4 for the sample target namespace prefix and URI (Figure 18).

Table 4 Namespace Prefix and URI Values

Field	Defaults
Prefix	`custorder`
URI	`urn:schemas-bea-com:ld-custorder`

4. Click Ok. Your new target namespace setting will appear in the topmost node of your target schema (Figure 19).

Figure 19 Target Schema with New Namespace



customerOrderReport.xsd

custorder:CustomerOrderReport
customerOrder*
CUSTOMER
wireless_orders
broadband_orders

5. Using the File —> Save Target Schema command, navigate to the Liquid Data Server repository directory and save your schema to the new filename: `custOrdRpt.xsd`

Step 6. View and Test Your Query

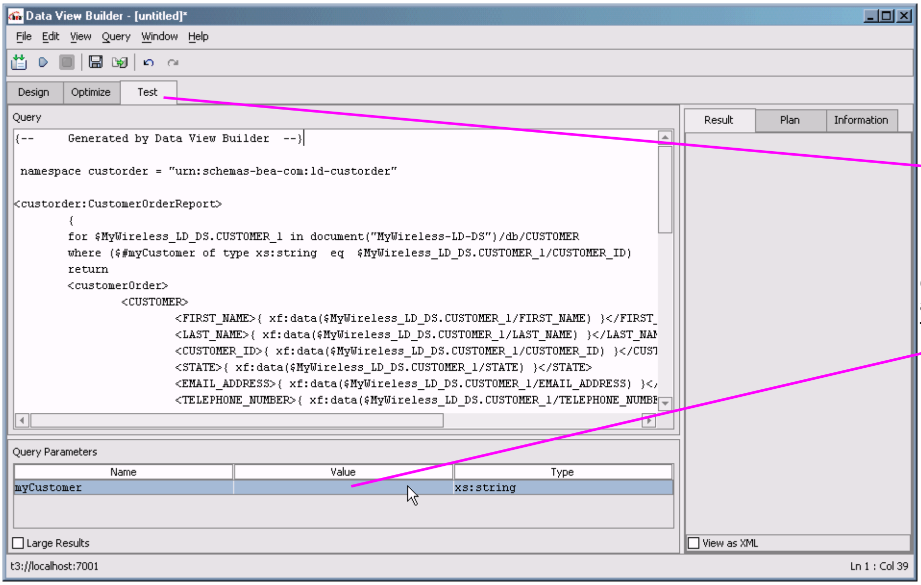
At this point you have assembled all the query elements. Specifically you have mapped data source elements to your target schema, you have created the necessary joins, and you have specified a query parameter functioning similarly to a SQL WHERE clause.

Entering Test Mode

When you enter Test mode, your query is automatically generated.

Click the Test tab at the top of the workspace (Figure 20). After a few moments your query will automatically appear.

Figure 20 Supplying a Query Parameter



```
{--     Generated by Data View Builder  --}

 namespace custorder = "urn:schemas-bea-com:ld-custorder"

<custorder:CustomerOrderReport>
        {
        for $MyWireless_LD_DS.CUSTOMER_1 in document("MyWireless-LD-DS")/db/CUSTOMER
        where ($#myCustomer of type xs:string  eq  $MyWireless_LD_DS.CUSTOMER_1/CUSTOMER_ID)
        return
        <customerOrder>
                <CUSTOMER>
                        <FIRST_NAME>{ xf:data($MyWireless_LD_DS.CUSTOMER_1/FIRST_NAME) }</FIRST_
                        <LAST_NAME>{ xf:data($MyWireless_LD_DS.CUSTOMER_1/LAST_NAME) }</LAST_NAM
                        <CUSTOMER_ID>{ xf:data($MyWireless_LD_DS.CUSTOMER_1/CUSTOMER_ID) }</CUST
                        <STATE>{ xf:data($MyWireless_LD_DS.CUSTOMER_1/STATE) }</STATE>
                        <EMAIL_ADDRESS>{ xf:data($MyWireless_LD_DS.CUSTOMER_1/EMAIL_ADDRESS) }</
                        <TELEPHONE_NUMBER>{ xf:data($MyWireless_LD_DS.CUSTOMER_1/TELEPHONE_NUMBE
```

Name	Value	Type
myCustomer		xs:string

Test tab

Enter CUSTOMER_1 as parameter for myCustomer

The Automatically Generated Query

The automatically generated XQuery is shown in Listing 5.

Listing 5 Generated customerOrderReport Query

```
namespace custorder = "urn:schemas-bea-com:ld-custorder"

<custorder:CustomerOrderReport>
   {
   for $MyWireless_LD_DS.CUSTOMER_1 in document("MyWireless-LD-DS")/db/CUSTOMER
   where ($#myCustomer of type xs:string  eq 
$MyWireless_LD_DS.CUSTOMER_1/CUSTOMER_ID)
   return
   <customerOrder>
      <CUSTOMER>
         <FIRST_NAME>{ xf:data($MyWireless_LD_DS.CUSTOMER_1/FIRST_NAME) 
}</FIRST_NAME>
         <LAST_NAME>{ xf:data($MyWireless_LD_DS.CUSTOMER_1/LAST_NAME) 
}</LAST_NAME>
         <CUSTOMER_ID>{ xf:data($MyWireless_LD_DS.CUSTOMER_1/CUSTOMER_ID) 
}</CUSTOMER_ID>
         <STATE>{ xf:data($MyWireless_LD_DS.CUSTOMER_1/STATE) }</STATE>
         <EMAIL_ADDRESS>{ xf:data($MyWireless_LD_DS.CUSTOMER_1/EMAIL_ADDRESS) 
}</EMAIL_ADDRESS>
         <TELEPHONE_NUMBER>{ 
xf:data($MyWireless_LD_DS.CUSTOMER_1/TELEPHONE_NUMBER) }</TELEPHONE_NUMBER>
      </CUSTOMER>
      <wireless_orders>
         {
         for $MyWireless_LD_DS.CUSTOMER_ORDER_2 in 
document("MyWireless-LD-DS")/db/CUSTOMER_ORDER
         where ($MyWireless_LD_DS.CUSTOMER_1/CUSTOMER_ID  eq 
$MyWireless_LD_DS.CUSTOMER_ORDER_2/CUSTOMER_ID)
         return
         <CUSTOMER_ORDER>
            <ORDER_DATE>{ 
xf:data($MyWireless_LD_DS.CUSTOMER_ORDER_2/ORDER_DATE) }</ORDER_DATE>
            <ORDER_ID>{ xf:data($MyWireless_LD_DS.CUSTOMER_ORDER_2/ORDER_ID) 
}</ORDER_ID>
            <CUSTOMER_ID>{ 
xf:data($MyWireless_LD_DS.CUSTOMER_ORDER_2/CUSTOMER_ID) }</CUSTOMER_ID>
            <SHIP_METHOD>{ 
xf:data($MyWireless_LD_DS.CUSTOMER_ORDER_2/SHIP_METHOD) }</SHIP_METHOD>
            <TOTAL_ORDER_AMOUNT>{ 
xf:data($MyWireless_LD_DS.CUSTOMER_ORDER_2/TOTAL_ORDER_AMOUNT) 
}</TOTAL_ORDER_AMOUNT>
         </CUSTOMER_ORDER>
         }
      </wireless_orders>
      <broadband_orders>
         {
```

```
         for $MyBroadBand_LD_DS.CUSTOMER_ORDER_3 in
document("MyBroadBand-LD-DS")/db/CUSTOMER_ORDER
         where ($MyWireless_LD_DS.CUSTOMER_1/CUSTOMER_ID  eq
$MyBroadBand_LD_DS.CUSTOMER_ORDER_3/CUSTOMER_ID)
         return
         <CUSTOMER_ORDER>
            <ORDER_DATE>{
xf:data($MyBroadBand_LD_DS.CUSTOMER_ORDER_3/ORDER_DATE) }</ORDER_DATE>
            <ORDER_ID>{ xf:data($MyBroadBand_LD_DS.CUSTOMER_ORDER_3/ORDER_ID)
}</ORDER_ID>
            <CUSTOMER_ID>{
xf:data($MyBroadBand_LD_DS.CUSTOMER_ORDER_3/CUSTOMER_ID) }</CUSTOMER_ID>
            <SHIP_METHOD>{
xf:data($MyBroadBand_LD_DS.CUSTOMER_ORDER_3/SHIP_METHOD) }</SHIP_METHOD>
            <TOTAL_ORDER_AMOUNT>{
xf:data($MyBroadBand_LD_DS.CUSTOMER_ORDER_3/TOTAL_ORDER_AMOUNT)
}</TOTAL_ORDER_AMOUNT>
         </CUSTOMER_ORDER>
         }
      </broadband_orders>
   </customerOrder>
   }
</custorder:CustomerOrderReport>
```

Testing Your Query

Once generated, your query is ready to be tested. However, you need to supply the customer ID parameter you set up in "Create a Query Parameter" on page 19.

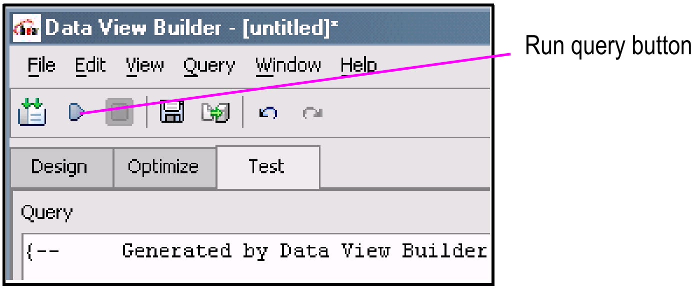
Click in the Value field next to myCustomer and directly enter `CUSTOMER_1`, as (see Figure 20).

Note: Query parameters are case sensitive.

Running Your Query

When you are ready to run the query, click the Run query button on the toolbar in the upper left (Figure 21).

Figure 21 Run Query Button

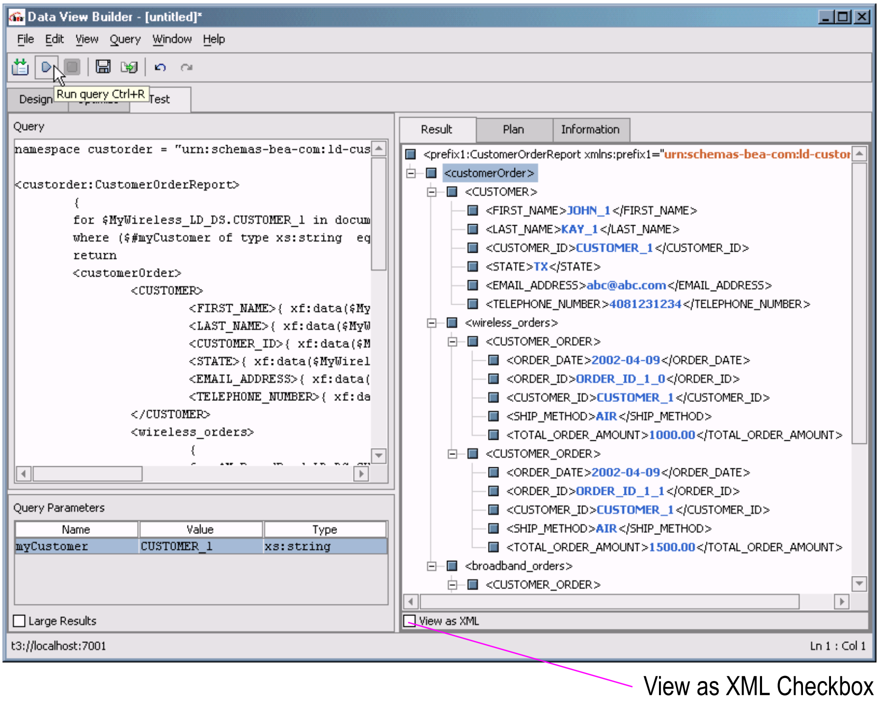


Data View Builder - [untitled]*
File Edit View Query Window Help
Design Optimize Test
Query
Run query button

```
{--      Generated by Data View Builder
```

The Liquid Data server processes the query and returns results. Right-click —> Expand on the top line of the generated results (`customerOrder`) to view your data in an outline-like arrangement.

Figure 22 Test Mode Displaying Query and Result in Tree Format



```
namespace custorder = "urn:schemas-bea-com:ld-cus

<custorder:CustomerOrderReport>
        {
        for $MyWireless_LD_DS.CUSTOMER_1 in docum
        where ($#myCustomer of type xs:string  eq
        return
        <customerOrder>
                <CUSTOMER>
                        <FIRST_NAME>{ xf:data($My
                        <LAST_NAME>{ xf:data($MyW
                        <CUSTOMER_ID>{ xf:data($M
                        <STATE>{ xf:data($MyWirel
                        <EMAIL_ADDRESS>{ xf:data(
                        <TELEPHONE_NUMBER>{ xf:da
                </CUSTOMER>
                <wireless_orders>
                        {
```

Name	Value	Type
myCustomer	CUSTOMER_1	xs:string

```
<prefix1:CustomerOrderReport xmlns:prefix1="urn:schemas-bea-com:ld-custor
  <customerOrder>
    <CUSTOMER>
      <FIRST_NAME>JOHN_1</FIRST_NAME>
      <LAST_NAME>KAY_1</LAST_NAME>
      <CUSTOMER_ID>CUSTOMER_1</CUSTOMER_ID>
      <STATE>TX</STATE>
      <EMAIL_ADDRESS>abc@abc.com</EMAIL_ADDRESS>
      <TELEPHONE_NUMBER>4081231234</TELEPHONE_NUMBER>
    <wireless_orders>
      <CUSTOMER_ORDER>
        <ORDER_DATE>2002-04-09</ORDER_DATE>
        <ORDER_ID>ORDER_ID_1_0</ORDER_ID>
        <CUSTOMER_ID>CUSTOMER_1</CUSTOMER_ID>
        <SHIP_METHOD>AIR</SHIP_METHOD>
        <TOTAL_ORDER_AMOUNT>1000.00</TOTAL_ORDER_AMOUNT>
      <CUSTOMER_ORDER>
        <ORDER_DATE>2002-04-09</ORDER_DATE>
        <ORDER_ID>ORDER_ID_1_1</ORDER_ID>
        <CUSTOMER_ID>CUSTOMER_1</CUSTOMER_ID>
        <SHIP_METHOD>AIR</SHIP_METHOD>
        <TOTAL_ORDER_AMOUNT>1500.00</TOTAL_ORDER_AMOUNT>
    <broadband_orders>
      <CUSTOMER_ORDER>
```

If you click the `View as XML` checkbox in the lower part of the Test view area of the Data View Builder, query results are displayed in XML (see Listing 6).

Listing 6 Query Result in XML

```
<prefix1:CustomerOrderReport xmlns:prefix1="urn:schemas-bea-com:ld-custorder">
  <customerOrder>
    <CUSTOMER>
      <FIRST_NAME>JOHN_1</FIRST_NAME>
      <LAST_NAME>KAY_1</LAST_NAME>
      <CUSTOMER_ID>CUSTOMER_1</CUSTOMER_ID>
      <STATE>TX</STATE>
      <EMAIL_ADDRESS>abc@abc.com</EMAIL_ADDRESS>
      <TELEPHONE_NUMBER>4081231234</TELEPHONE_NUMBER>
    </CUSTOMER>
    <wireless_orders>
      <CUSTOMER_ORDER>
        <ORDER_DATE>2002-03-06</ORDER_DATE>
        <ORDER_ID>ORDER_ID_1_0</ORDER_ID>
        <CUSTOMER_ID>CUSTOMER_1</CUSTOMER_ID>
        <SHIP_METHOD>AIR</SHIP_METHOD>
        <TOTAL_ORDER_AMOUNT>1000</TOTAL_ORDER_AMOUNT>
      </CUSTOMER_ORDER>
      <CUSTOMER_ORDER>
        <ORDER_DATE>2002-03-06</ORDER_DATE>
        <ORDER_ID>ORDER_ID_1_1</ORDER_ID>
        <CUSTOMER_ID>CUSTOMER_1</CUSTOMER_ID>
        <SHIP_METHOD>AIR</SHIP_METHOD>
        <TOTAL_ORDER_AMOUNT>2000</TOTAL_ORDER_AMOUNT>
      </CUSTOMER_ORDER>
    </wireless_orders>
    <broadband_orders>
      <CUSTOMER_ORDER>
        <ORDER_DATE>2002-04-09</ORDER_DATE>
        <ORDER_ID>ORDER_ID_1_0</ORDER_ID>
        <CUSTOMER_ID>CUSTOMER_1</CUSTOMER_ID>
        <SHIP_METHOD>AIR</SHIP_METHOD>
        <TOTAL_ORDER_AMOUNT>1000.00</TOTAL_ORDER_AMOUNT>
      </CUSTOMER_ORDER>
      <CUSTOMER_ORDER>
        <ORDER_DATE>2002-04-09</ORDER_DATE>
        <ORDER_ID>ORDER_ID_1_1</ORDER_ID>
        <CUSTOMER_ID>CUSTOMER_1</CUSTOMER_ID>
        <SHIP_METHOD>AIR</SHIP_METHOD>
        <TOTAL_ORDER_AMOUNT>1500.00</TOTAL_ORDER_AMOUNT>
      </CUSTOMER_ORDER>
    </broadband_orders>
  </customerOrder>
</prefix1:CustomerOrderReport>
```

Summary

You have successfully built and run a query using Liquid Data.

Note: To switch back to Data View Builder Design mode, click the Design View tab on the toolbar.

Step 7. Save Your Liquid Data Project

Now that you have successfully built and run a query, you can save your Data View Builder project. The next time you need to run or modify your query or Data View, you can simply reopen the project.

To save your project:

1. Choose the menu option File —> Save Project.
2. Navigate to the directory where you want to save your project. Enter `customerReport` in the File name field.
3. Click Save.

   The file is saved with a `.qpr` extension.

See “Working With Projects” in the Starting the Builder and Touring the GUI chapter of *Building Queries and Data Views.*

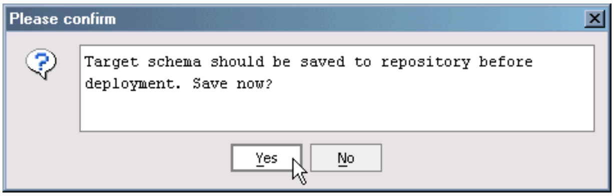
Step 8. Deploy Your Query

To use of the results of your query in an application such as WebLogic Workshop, you first need to associate a saved query with a saved target schema. This is known as *deploying a query* and provides the *metadata* necessary to create an XMLBean that provides data access services in Workshop. (See “Deploying a Query” in the Testing Queries chapter of *Building Queries and Data Views.*)

Follow these steps to deploy your query:

1. In Test mode, select Deploy Query from the Query menu (Query —> Deploy Query). A dialog box appears where you can choose to save your current schema to the Liquid Data repository.

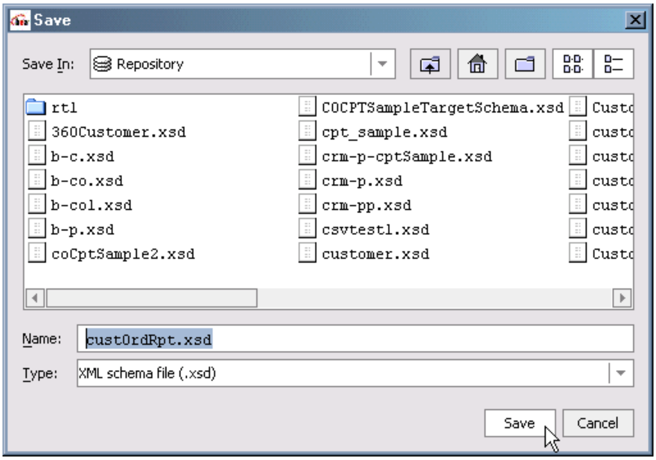
Figure 23 Deploy Query Save Target Schema Dialog Box



Target schema should be saved to repository before deployment. Save now?

2. Choose Yes. When the repository file browser appears you can use the supplied target schema name of `custOrdRpt.xsd`

Figure 24 Deploy Query Save Target Schema Dialog Box



Save In: Repository

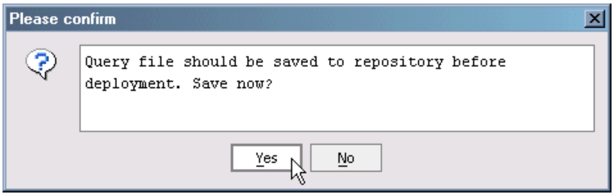
rtl
360Customer.xsd
b-c.xsd
b-co.xsd
b-col.xsd
b-p.xsd
coCptSample2.xsd
COCPTSampleTargetSchema.xsd
cpt_sample.xsd
crm-p-cptSample.xsd
crm-p.xsd
crm-pp.xsd
csvtest1.xsd
customer.xsd
Custo
custo
custo
custo
custo
custo
Custo

Name: custOrdRpt.xsd

Type: XML schema file (.xsd)

3. Next, you will be asked if you want to save the current query. Again select Yes.

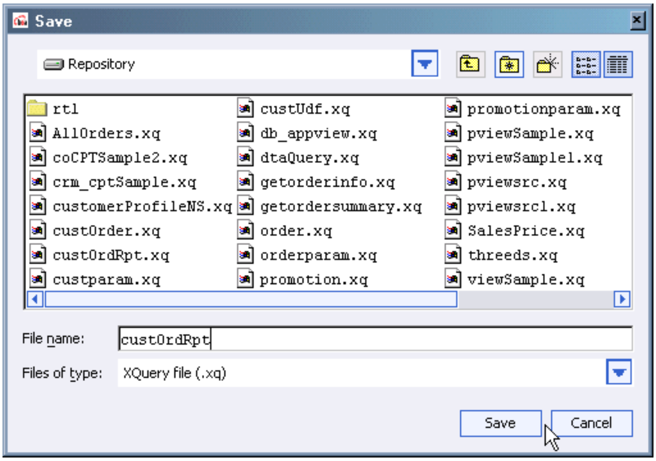
Figure 25 Deploy Query Save Current Query Dialog Box



Query file should be saved to repository before deployment. Save now?

4. Save your query to the name `custOrdRpt`.

Figure 26 Creating a Stored Query



Repository

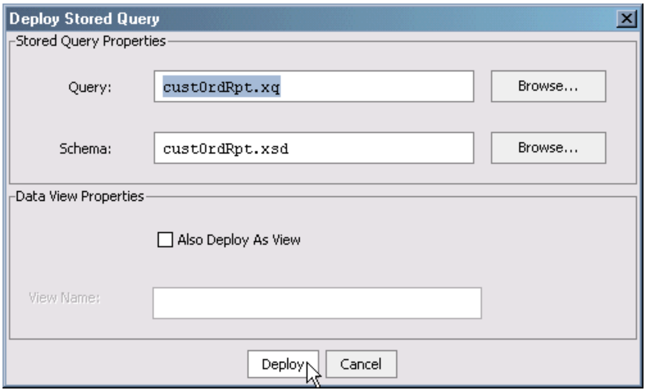
rtl
AllOrders.xq
coCPTSample2.xq
crm_cptSample.xq
customerProfileNS.xq
custOrder.xq
custOrdRpt.xq
custparam.xq
custUdf.xq
db_appview.xq
dtaQuery.xq
getorderinfo.xq
getordersummary.xq
order.xq
orderparam.xq
promotion.xq
promotionparam.xq
pviewSample.xq
pviewSample1.xq
pviewsrc.xq
pviewsrc1.xq
SalesPrice.xq
threeds.xq
viewSample.xq

File name: custOrdRpt

Files of type: XQuery file (.xq)

5. The Deploy Stored Query dialog box appears, populated with the name of your query and target schema.

Figure 27 Deploying custOrdRpt Query with custOrdRpt Schema



Query: custOrdRpt.xq

Schema: custOrdRpt.xsd

☐ Also Deploy As View

View Name:

6. Click Deploy. You should see a Deployment Successful! message.

Step 9. Creating a Liquid Data Control in WebLogic Workshop

Once you have deployed a query, you can quickly make it available to WebLogic Workshop. (For details see "Using WebLogic Workshop Controls to Develop Liquid Data Applications" in the *Application Developer's Guide*.)

In the following two sections you will undertake:

- Creating a Liquid Data Control
- Generating a Page Flow File
- Obtaining Preliminary Query Results
- Modifying Page Flow Source
- Adding Data to the Application
- Testing Your Results

Creating a Liquid Data Control

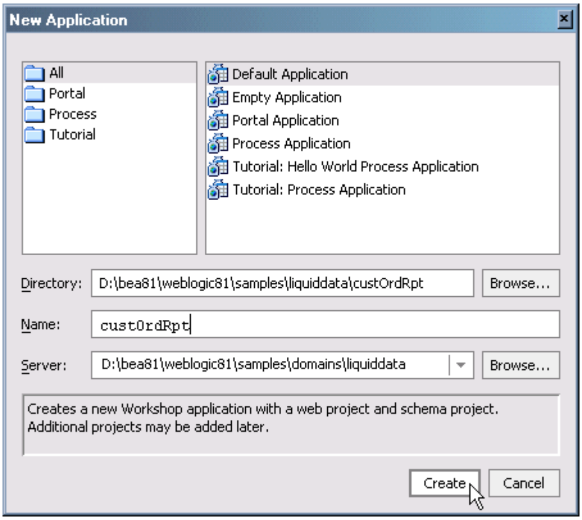
A Liquid Data control provides access from WebLogic Workshop to queries that have been deployed to a Liquid Data server.

1. Open WebLogic Workshop from the Windows Start menu:

   Start —> Programs —> BEA WebLogic Platform 8.1 —> WebLogic Workshop 8.1

2. Create a new WebLogic Workshop application (File —> New —> Application). Use `custOrdRpt` as the application name. Notice that the selected application type is Default Application.

Figure 28 Creating a New WebLogic Workshop Application



New Application

Directory: D:\bea81\weblogic81\samples\liquiddata\custOrdRpt

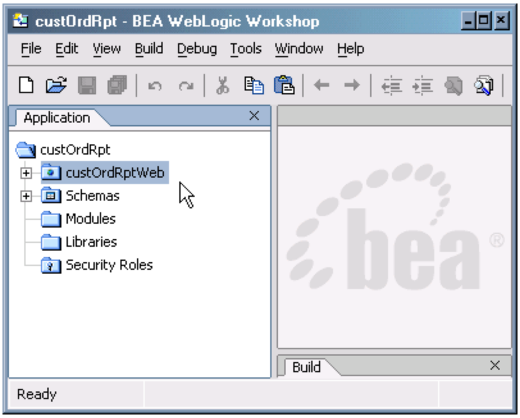
Name: custOrdRpt

Server: D:\bea81\weblogic81\samples\domains\liquiddata

Creates a new Workshop application with a web project and schema project. Additional projects may be added later.

When you create a default application, a project of type Web Project is automatically built using the name of the application plus `Web`. In this case the project is `custOrdRptWeb`.

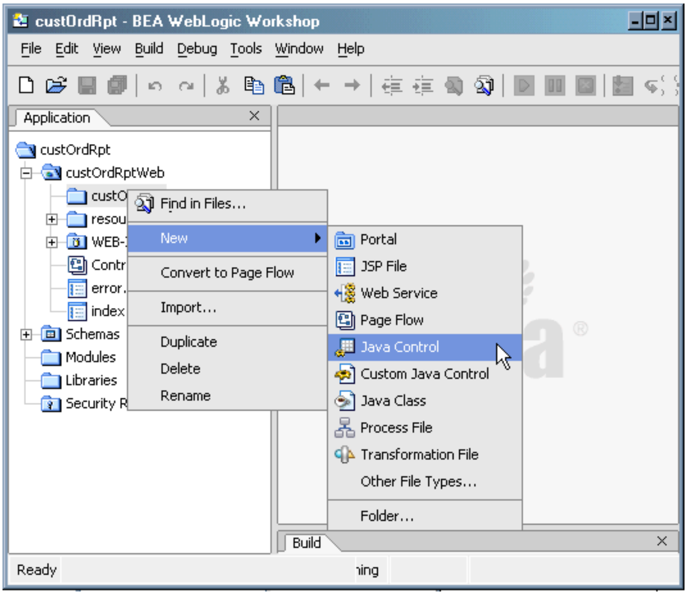
Figure 29 Creating a Web Project Within the New Application



3. Right click on `custOrdRptWeb`, then select New —> Folder. Name your new folder `custOrdRpt`. Click Ok.

4. Right-click on the `custOrdRpt` folder and select New —> Java Control.

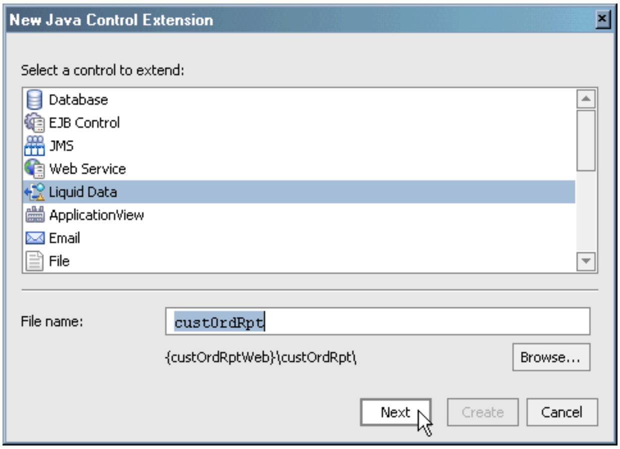
Figure 30 Creating a New Java Control



5. Select Liquid Data from the list of available Java control extensions.

   Enter `custOrdRpt` as the filename for the control (Figure 31). The Liquid Data control enables you to turn Liquid Data queries into XML Beans which can be used by a WebLogic Workshop application.

Figure 31 Selecting and Naming a Liquid Data Control

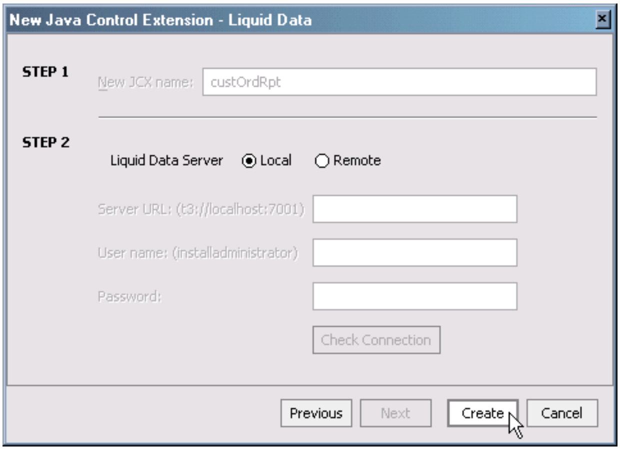


File name: custOrdRpt

{custOrdRptWeb}\custOrdRpt\

6. Click Next. If your server is remote, you will need to enter remote connection information. However, most likely Liquid Data samples are running on your local system; if so, select Local, the default option.

Figure 32 Creating a New Java Control Dialog



New Java Control Extension - Liquid Data

STEP 1

New JCX name: custOrdRpt

STEP 2

Liquid Data Server ● Local ○ Remote

Server URL: (t3://localhost:7001)

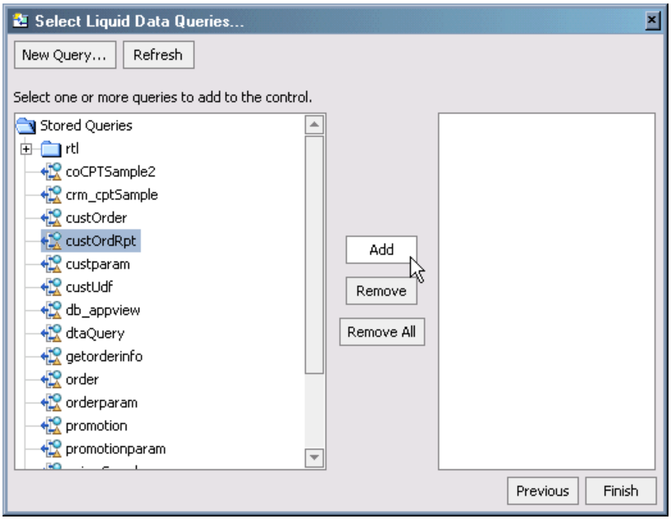
User name: (installadministrator)

Password:

7. Click Create.

8. From the list of available queries select `custOrdRpt`.

Figure 33 Adding a Deployed Query to a Liquid Data Control



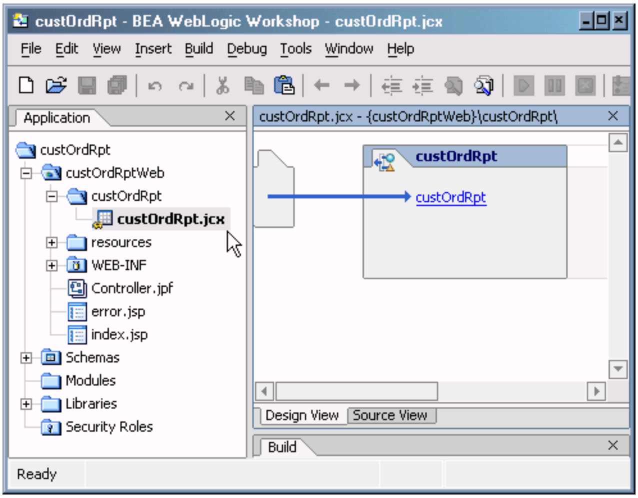
Select Liquid Data Queries...

Select one or more queries to add to the control.

- Stored Queries
  - rtl
  - coCPTSample2
  - crm_cptSample
  - custOrder
  - custOrdRpt
  - custparam
  - custUdf
  - db_appview
  - dtaQuery
  - getorderinfo
  - order
  - orderparam
  - promotion
  - promotionparam

9. Click the Add button.
10. When the query moves to the right-hand column, click Finish. A graphical depiction of the newly created Liquid Data control is shown in Figure 34.

Figure 34 Design View of Liquid Data Control



The final section takes you through the steps required to build a web application that display query results in automatically-generated java server pages (JSP).

Step 10. Creating a Liquid Data-Powered Web Application

Once a Liquid Data Control is available, its query elements can be built into JSP pages through creation of a WebLogic Workshop application.

Generating a Page Flow File

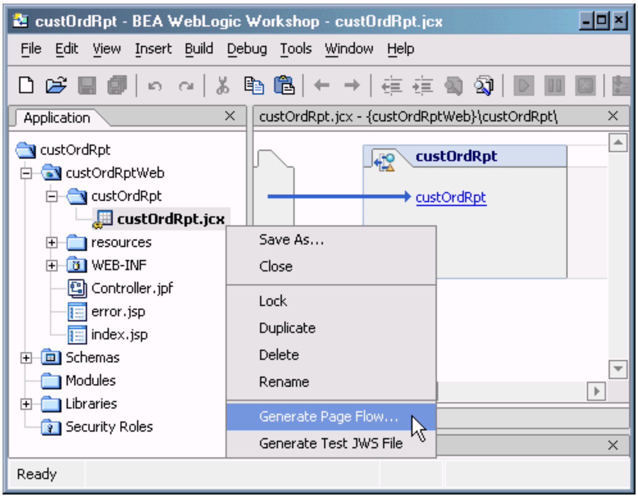
WebLogic Workshop applications are driven by a single page flow file, identified by its `.jpf` extension. When you generate a page flow, WebLogic Workshop creates the page flow file and several other files. (For details see "Generating a Page Flow From a Control" in the chapter Using Workshop Controls to Develop Liquid Data Applications in the *Application Developer's Guide*.)

Perform the following steps to generate a page file (`custOrdRpt.jpf`) from a Liquid Data java control.

1. Right-click on the `custOrdRpt.jcx` file.

2. Select Generate Page Flow (Figure 35).

Figure 35 Generating a Page Flow Document From a Liquid Data Java Control



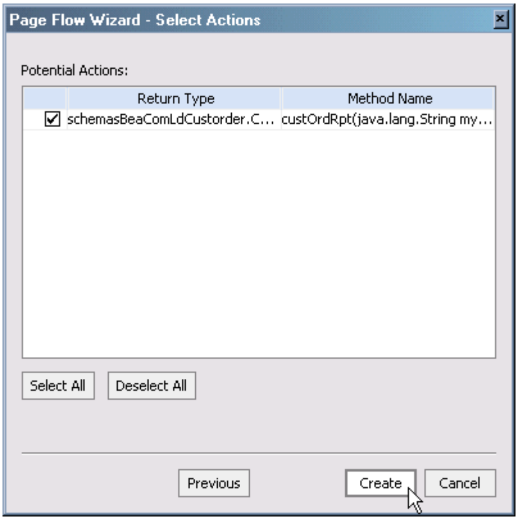
3. In the Page Flow Wizard, enter `custOrdRpt` as the page flow name.

4. Click Next.

5. In the Page Flow Wizard - Select Actions dialog box check the query methods you want available to your new application. In the case of `custOrdReport`, there only one method. Click the checkbox to the left of the method (Figure 36).

6. Select Create.

Figure 36 Identifying the Liquid Data Methods for the Page Flow



	Return Type	Method Name
☑	schemasBeaComLdCustorder.C...	custOrdRpt(java.lang.String my...

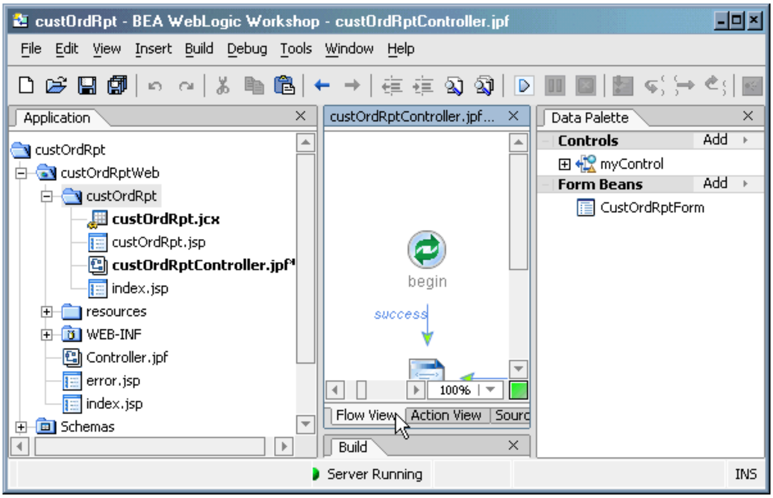
WebLogic Workshop generates:

– An application page flow file (`custOrdRptController.jpf`)

– A start page (`index.jsp`)

– A JSP file for method(s) you specified through the Liquid Data control (`custOrdRpt.jsp`).

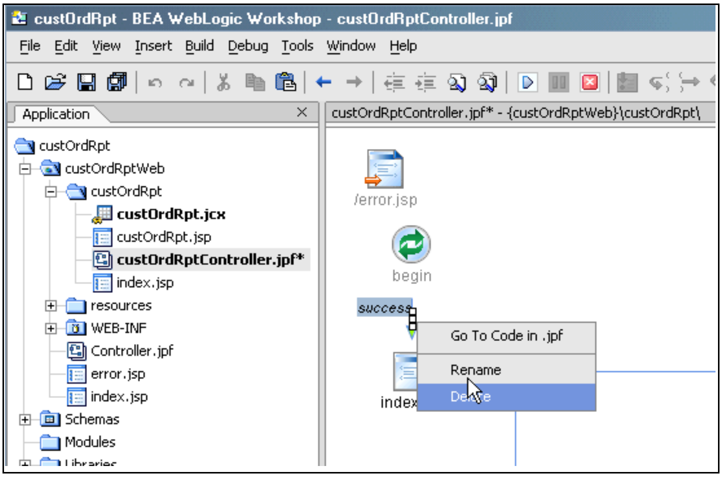
Figure 37 WebLogic Workshop View After a Page Flow is Generated



The application Flow View graphically depicts the relationship between the various pages of your application.

7. Delete the `success` link that connects the `begin` element to `index.jsp` by selecting the link (click on it) and then choose Right-click —> Delete (Figure 38). The change is needed because `index.jsp` will not be the application's start-up page.

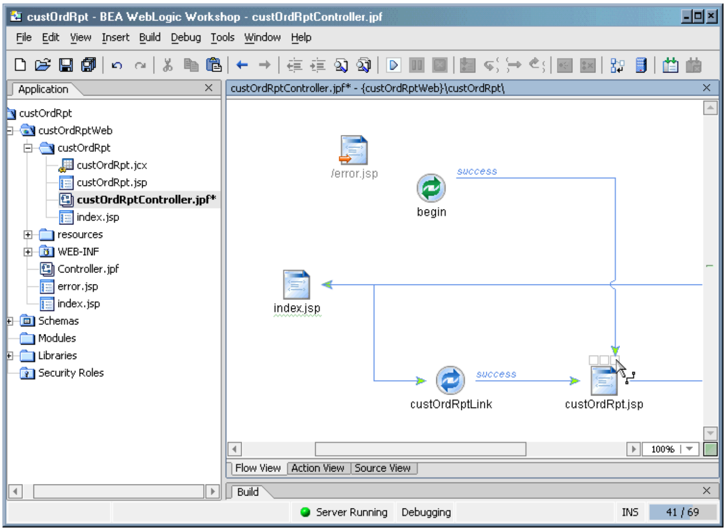
Figure 38 Removing a Page Flow Link



8. Create a new link from `begin` to `custOrdRpt.jsp`; click adjacent to `begin`. Then, holding your mouse button down, draw a link to `custOrdRpt.jsp` (Figure 39). It will automatically be labeled `success`.

   Based on the XMLBean that was created from the custOrdRpt deployed query, a customer ID parameter is required by the web application. In WebLogic Workshop the new initial application page (`custOrdRpt.jsp`) requests a Customer ID. (See "Create a Query Parameter" on page 19.)

Figure 39 Creating a New Page Flow Link



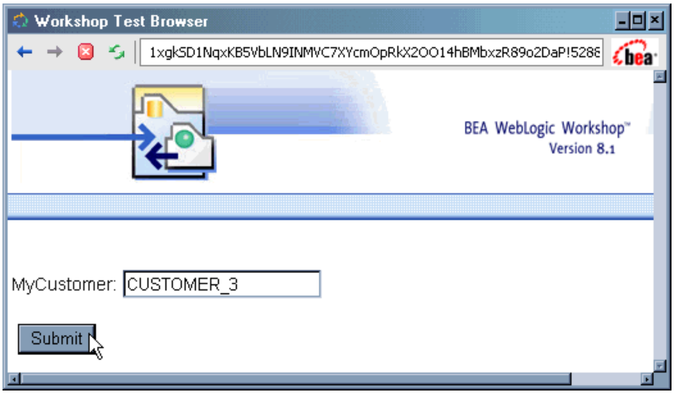
Obtaining Preliminary Query Results

You can now test your application for data access.

1. From the WebLogic Workshop menu Choose Debug —> Start Without Debugging.
2. After some moments the `custOrdRptController` page appears. Enter a query parameter such as `CUSTOMER_3` (Figure 40).

   Note: Case matters; `customer_3` will not succeed.

Figure 40 Supplying a Query Parameter From the New Application

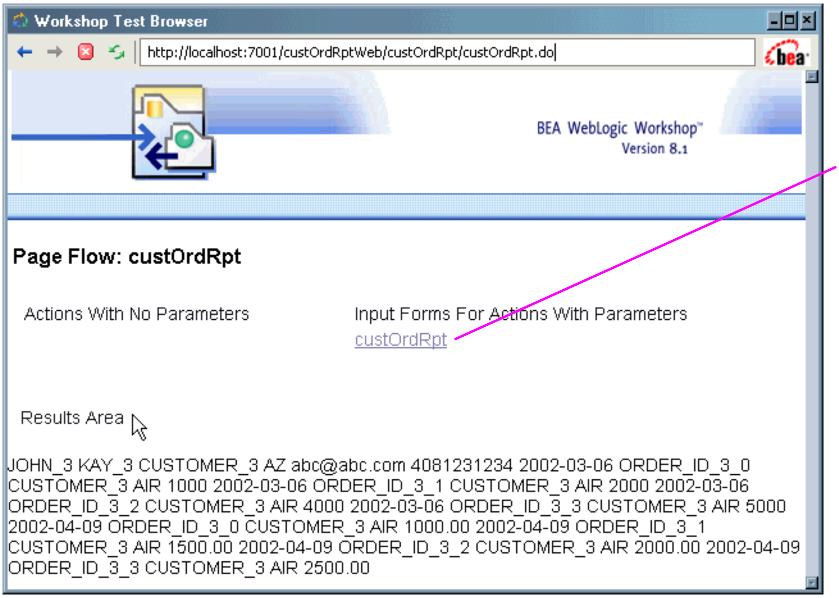


1xgkSD1NqxKB5VbLN9INMVC7XYcmOpRkX2OO14hBMbxzR89o2DaP!5288

MyCustomer: CUSTOMER_3

3. Click Submit to test your query. After a few moments raw results such as is shown in Figure 41 appears.

Figure 41 Raw Query Results



Workshop Test Browser
BEA WebLogic Workshop™
Version 8.1
Actions With No Parameters
Input Forms For Actions With Parameters
custOrdRpt

http://localhost:7001/custOrdRptWeb/custOrdRpt/custOrdRpt.do

Page Flow: custOrdRpt

Results Area

```
JOHN_3 KAY_3 CUSTOMER_3 AZ abc@abc.com 4081231234 2002-03-06 ORDER_ID_3_0
CUSTOMER_3 AIR 1000 2002-03-06 ORDER_ID_3_1 CUSTOMER_3 AIR 2000 2002-03-06
ORDER_ID_3_2 CUSTOMER_3 AIR 4000 2002-03-06 ORDER_ID_3_3 CUSTOMER_3 AIR 5000
2002-04-09 ORDER_ID_3_0 CUSTOMER_3 AIR 1000.00 2002-04-09 ORDER_ID_3_1
CUSTOMER_3 AIR 1500.00 2002-04-09 ORDER_ID_3_2 CUSTOMER_3 AIR 2000.00 2002-04-09
ORDER_ID_3_3 CUSTOMER_3 AIR 2500.00
```

Click application form name to enter a different query parameter

Modifying Page Flow Source

While the application's Java Page Flow (JPF) file is automatically generated, a few lines of Java code are needed to add XMLBean variables to your page flow. (For details see the topics "Adding XMLBean Variables to the Page Flow" and "To Initialize the Variable in the Page Flow" in the chapter Using Workshop Controls to Develop Liquid Data Applications in the *Application Developer's Guide*.)

Follow these steps to copy several lines of Java code into the correct section of `custOrdRptController.jpf`:

1. Open the `custOrdRptController.jpf` file in WebLogic Workshop.
2. Click the Source View tab in WebLogic Workshop.
3. Immediately after the following line (near the beginning of the file):

   ```
   private custOrdRpt myControl;
   ```

   insert the following three lines of java code:

```
public
schemasBeaComLdCustorder.CustomerOrderReportDocument.CustomerOrderReport.Custo
merOrder.BroadbandOrders.CUSTOMERORDER[] BroadBand;

public
schemasBeaComLdCustorder.CustomerOrderReportDocument.CustomerOrderReport.Custo
merOrder.WirelessOrders.CUSTOMERORDER[] Wireless;

public
schemasBeaComLdCustorder.CustomerOrderReportDocument.CustomerOrderReport.Custo
merOrder Customers;
```

These instructions create three variables. Two represent the `CUSTOMERORDER[]` arrays for BroadBandOrders and WirelessOrders data sources, respectively. The third, Customers, represents data associated with a particular customer.

4. A few lines down locate the section of code that ties the query to the `custOrdRpt` form. An easy way to do this is to search. From the WebLogic Workshop menu select Edit —> Find (or Control-F). Search for a section of code beginning with the string:

   ```
   Forward custOrdRpt
   ```

   After the line:

   ```
   getRequest().setAttribute( "results", var )
   ```

   enter (or copy/paste) the following:

```
Wireless =
var.getCustomerOrderReport().getCustomerOrderArray(0).getWirelessOrders().getC
USTOMERORDERArray() ;

BroadBand =
var.getCustomerOrderReport().getCustomerOrderArray(0).getBroadbandOrders().get
CUSTOMERORDERArray() ;

Customers = var.getCustomerOrderReport().getCustomerOrderArray(0);
```

These instructions make the three public variables available to the application. The relevant code is shown in Figure 7 (newly entered lines are in boldface).

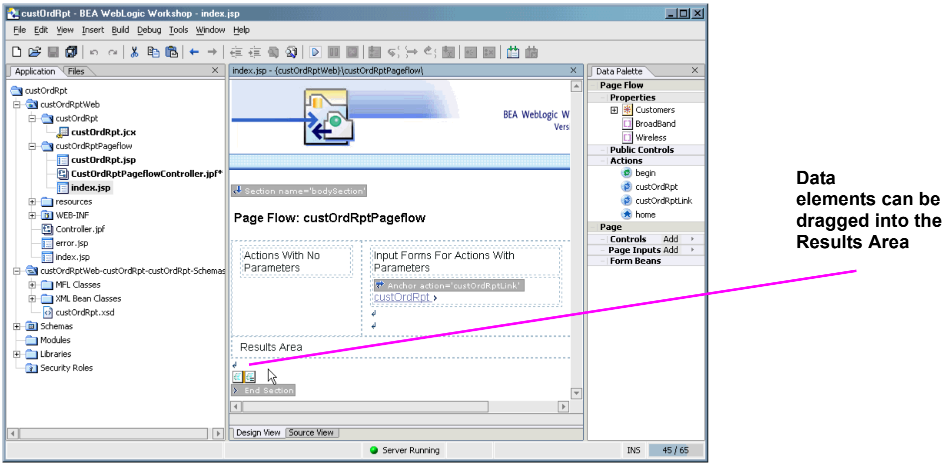
Listing 7 CustOrdRpt Page Flow File (custOrdRptController.jpf).

```
package custOrdRpt;
import com.bea.wlw.netui.pageflow.FormData;
import com.bea.wlw.netui.pageflow.Forward;
import com.bea.wlw.netui.pageflow.PageFlowController;

import custOrdRpt.custOrdRpt;

...

public class custOrdRptController extends PageFlowController
{
    /**
     * This is the control used to generate this pageflow
     *  @common:control
     */
    private custOrdRpt myControl;

public
schemasBeaComLdCustorder.CustomerOrderReportDocument.CustomerOrderReport.Custo
merOrder.BroadbandOrders.CUSTOMERORDER[] BroadBand;

public
schemasBeaComLdCustorder.CustomerOrderReportDocument.CustomerOrderReport.Custo
merOrder.WirelessOrders.CUSTOMERORDER[] Wireless;
public

schemasBeaComLdCustorder.CustomerOrderReportDocument.CustomerOrderReport.Custo
merOrder Customers;

    // Uncomment this declaration to access Global.app.
    //
    //     protected global.Global globalApp;
    //

    // For an example of page flow exception handling see the example "catch"
and "exception-handler"
    // annotations in {project}/WEB-INF/src/global/Global.app

    /**
     * This method represents the point of entry into the pageflow
     * @jpf:action
     */
    protected Forward begin()
    {
        return new Forward( "success" );
    }

    /**
     * Action encapsulating the control method :custOrdRpt
     * @jpf:action
     * @jpf:forward name="success" path="index.jsp"
     * @jpf:catch method="exceptionHandler" type="Exception"
     */
    public Forward custOrdRpt( CustOrdRptForm aForm )
        throws Exception
    {
        schemasBeaComLdCustorder.CustomerOrderReportDocument var =
myControl.custOrdRpt( aForm.myCustomer );
        getRequest().setAttribute( "results", var );

Wireless =
var.getCustomerOrderReport().getCustomerOrderArray(0).getWirelessOrders().getC
USTOMERORDERArray() ;
BroadBand =
var.getCustomerOrderReport().getCustomerOrderArray(0).getBroadbandOrders().get
CUSTOMERORDERArray() ;
Customers = var.getCustomerOrderReport().getCustomerOrderArray(0);

        return new Forward( "success" );
    }

    /**
     * @jpf:action
     * @jpf:forward name="success" path="custOrdRpt.jsp"
     */
    public Forward custOrdRptLink()
        throws Exception
    {
        return new Forward( "success" );
    }
    /**
     * @jpf:exception-handler
     * @jpf:forward name="errorPage" path="/error.jsp"
     */
    protected Forward exceptionHandler( Exception ex, String actionName, String 
message, FormData form )
    {
        String displayMessage = "An exception occurred in the action " + 
actionName;
        getRequest().setAttribute( "errorMessage", displayMessage );
        return new Forward( "errorPage" );
    }

    /**
     * FormData class CustOrdRptForm
     * FormData get and set methods may be overwritten by the Form Bean editor.
     */
    public static class CustOrdRptForm extends FormData
    {
        private java.lang.String myCustomer;
        public void setMyCustomer( java.lang.String myCustomer )
        {
            this.myCustomer = myCustomer;
        }
        public java.lang.String getMyCustomer()
        {
            return myCustomer;
        }
    }
}
```

Adding Data to the Application

Now that variables are available to hold query results, the last step is to modify `index.jsp` to display your results. In the `custOrdRpt` folder double-click on `index.jsp` (Figure 42).

Figure 42 WebLogic Workshop Design View of index.jsp



Data elements can be dragged into the Results Area

Three things need to be added to `index.jsp`:

- Field labels.
- Data fields, to display information about the current customer.
- NetUI *repeater* fields, to contain data from the BroadBand and Wireless data sources.

The following sections walk you through the `index.jsp` editing process.

Adding Field Labels

First, several labels need to be added to `index.jsp`.

Go to the source view. Open the `index.jsp` file. Near the end of that file find and replace the long line beginning with:

```
<%Object res = request.getAttribute ...
```

With the following:

```
<b>CUSTOMER</b>

<br>

<b>BroadBand Orders</b>

<br>

<b>Wireless Orders</b>
<br>
```

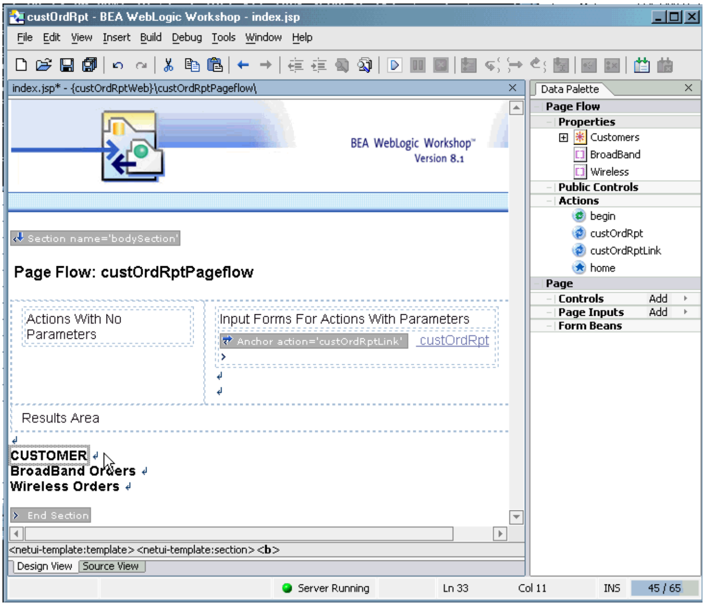
Adding Data Fields

Next, a HTML table needs to be added to the page to hold customer information (in an RDBMS application, customer information would represent the *master* and customer orders the *detail*).

1. Return to Design View.

2. Using your mouse, click on CUSTOMER (Figure 43).

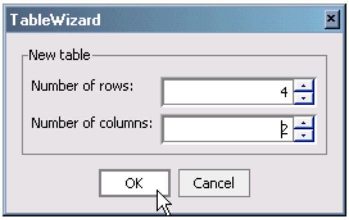
Figure 43 Java Server Page (index.jsp) After Adding Field Labels


custOrdRpt - BEA WebLogic Workshop - index.jsp
File Edit View Insert Build Debug Tools Window Help
index.jsp* - {custOrdRptWeb}\custOrdRptPageflow\
BEA WebLogic Workshop™ Version 8.1
Section name='bodySection'
Page Flow: custOrdRptPageflow
Actions With No Parameters
Input Forms For Actions With Parameters
Anchor action='custOrdRptLink'
custOrdRpt
Results Area
CUSTOMER
BroadBand Orders
Wireless Orders
End Section
<netui-template:template> <netui-template:section> <b>
Design View
Source View
Data Palette
Page Flow
Properties
Customers
BroadBand
Wireless
Public Controls
Actions
begin
custOrdRpt
custOrdRptLink
home
Page
Controls Add
Page Inputs Add
Form Beans
Server Running
Ln 33
Col 11
INS
45 / 65

3. From the WebLogic Workshop menu select Insert —> HTML —> Table.

4. When the TableWizard appears, set the number of rows to 4 and the number of columns to 2.

Figure 44 Creating a HTML Table in WebLogic Workshop



TableWizard

New table

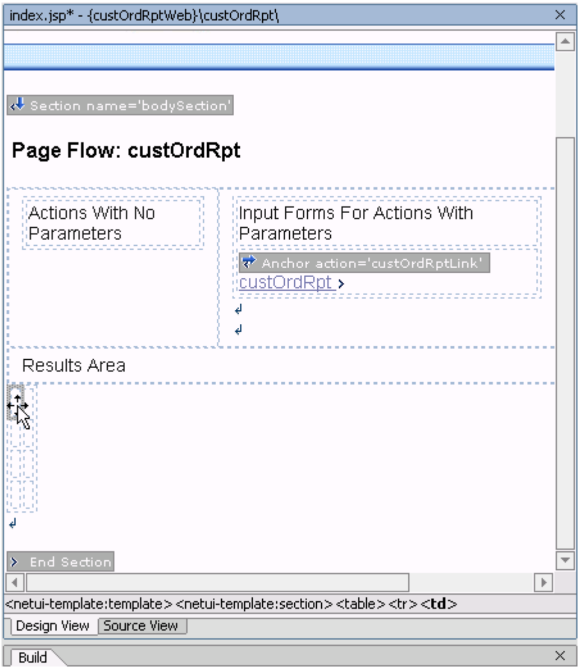
Number of rows: 4

Number of columns: 2

5. Click Ok. Notice that the table is placed in the Results Area below the CUSTOMER label.

6. Double-click in the upper-left field of the generated grid (Figure 45). This will open to the relevant section of `index.jsp` in Source View.

Figure 45 Selecting a Cell in WebLogic Workshop Table Grid


index.jsp* - {custOrdRptWeb}\custOrdRpt\
Section name='bodySection'
Page Flow: custOrdRpt
Actions With No Parameters
Input Forms For Actions With Parameters
Anchor action='custOrdRptLink'
custOrdRpt
Results Area
End Section
<netui-template:template> <netui-template:section> <table> <tr> <td>
Design View
Source View
Build

7. Replace the first table detail of each row with the following descriptions: First name, Last name, Customer ID, and Email Address. Replacement text is highlighted (Figure 48).

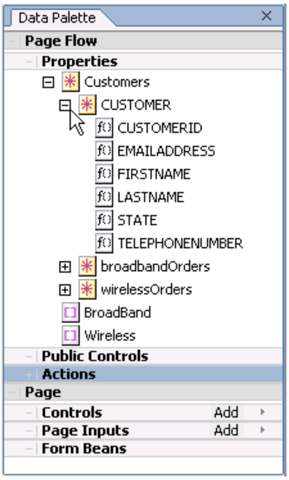
```
<table border="1">
        <tr>
            <td>First Name</td>
            <td>&nbsp;</td>
        </tr>
        <tr>
            <td>Last Name</td>
            <td>&nbsp;</td>
        </tr>
        <tr>
            <td>Customer ID</td>
            <td>&nbsp;</td>
        </tr>
```

```
    <tr>
        <td>Email Address</td>
        <td>&nbsp;</td>
    </tr>
</table>
```

8. Switch back to Design View.

9. In the WebLogic Workshop Data Palette, expand PageFlow —> Properties —> Customers—>CUSTOMER to see the available Customer elements (Figure 46).

Figure 46 Data Palette Page Flow Data Components



10. Drag-and-drop data elements to their corresponding text labels (Figure 47).

Figure 47 Populating WebLogic Workshop HTML Table with Customer Data Element

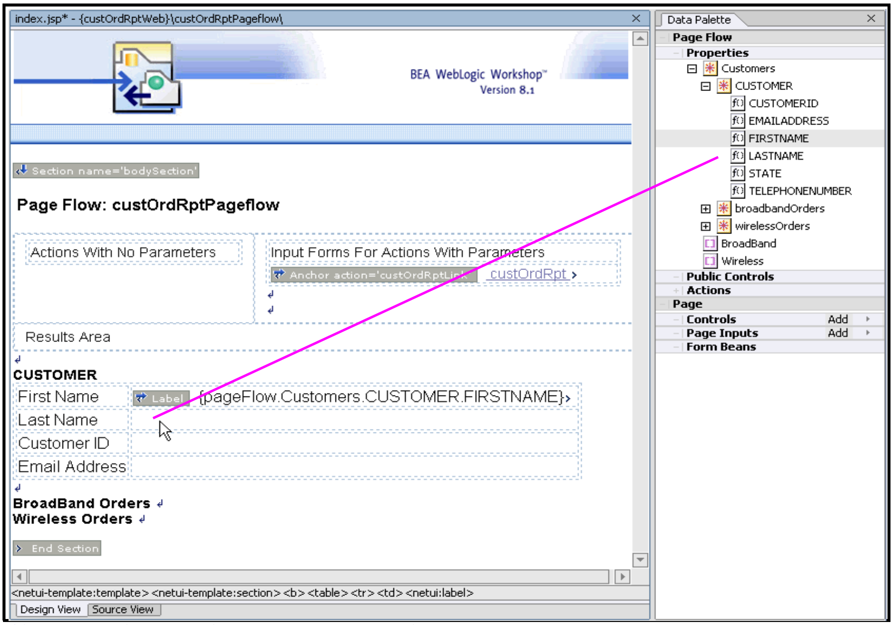
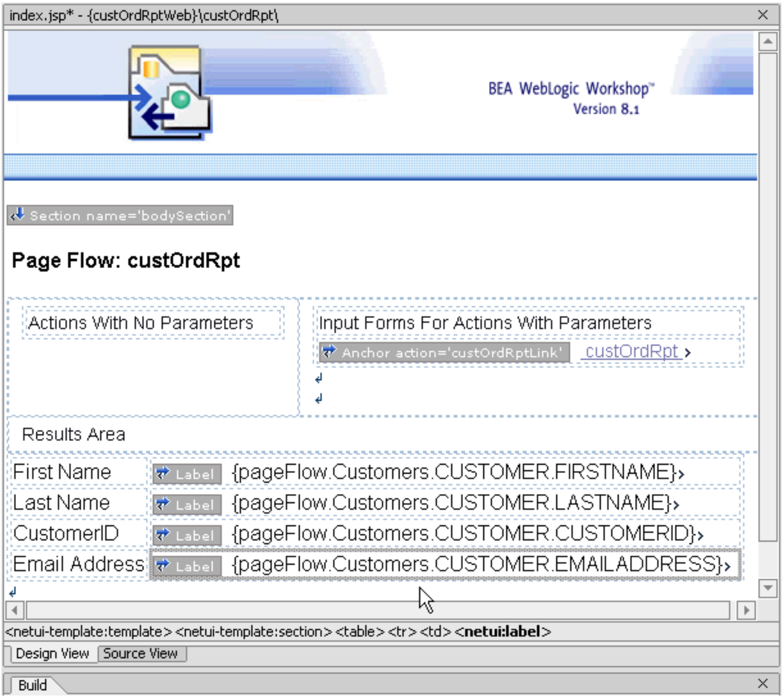


Figure 48 shows a fully configured customer information table.

Figure 48 Design View of Customer ID Table in index.jsp


index.jsp* - {custOrdRptWeb}\custOrdRpt\
BEA WebLogic Workshop™
Version 8.1
Section name='bodySection'
Actions With No Parameters
Input Forms For Actions With Parameters
Anchor action='custOrdRptLink'
custOrdRpt
Design View
Source View
Build

Page Flow: custOrdRpt

Results Area

First Name	Label	{pageFlow.Customers.CUSTOMER.FIRSTNAME}
Last Name	Label	{pageFlow.Customers.CUSTOMER.LASTNAME}
CustomerID	Label	{pageFlow.Customers.CUSTOMER.CUSTOMERID}
Email Address	Label	{pageFlow.Customers.CUSTOMER.EMAILADDRESS}

```
<netui-template:template> <netui-template:section> <table> <tr> <td> <netui:label>
```

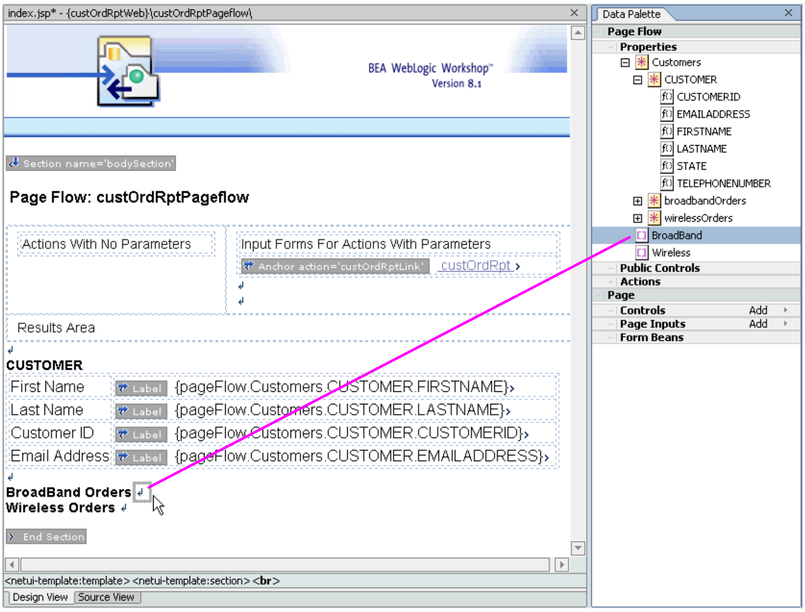
Add Repeater Fields For Order Arrays

WebLogic Workshop repeater fields make it very easy to insert arrays of data such as Wireless and BroadBand order information. (For details see the topic “To Add a Repeater to a JSP File” in the chapter Using Workshop Controls to Develop Liquid Data Applications in the *Application Developer’s Guide*.)

Using your cursor, highlight the return after BroadBand (Figure 49).

1. From the Data Palette, just above the Public Controls section, drag the BroadBand array symbol to the highlighted area (also Figure 49).

Figure 49 Populating a JSP With an Array Containing BroadBand Order Data Elements

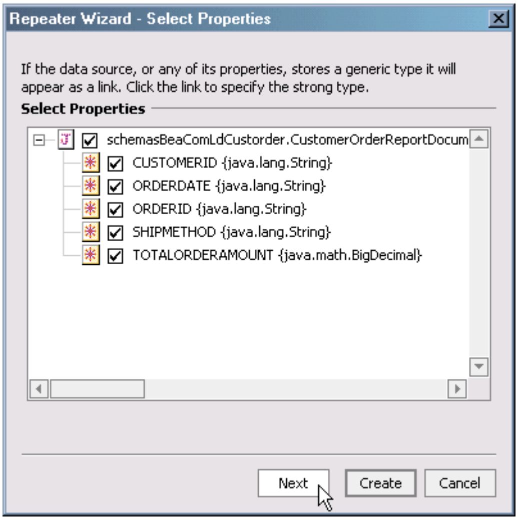


First Name	Label	{pageFlow.Customers.CUSTOMER.FIRSTNAME}
Last Name	Label	{pageFlow.Customers.CUSTOMER.LASTNAME}
Customer ID	Label	{pageFlow.Customers.CUSTOMER.CUSTOMERID}
Email Address	Label	{pageFlow.Customers.CUSTOMER.EMAILADDRESS}

<netui-template:template> <netui-template:section>

2. Accept the defaults (meaning all the elements in the array) by clicking Next. No title is needed.

Figure 50 Selecting Fields for NetUI Repeater Element



Repeater Wizard - Select Properties

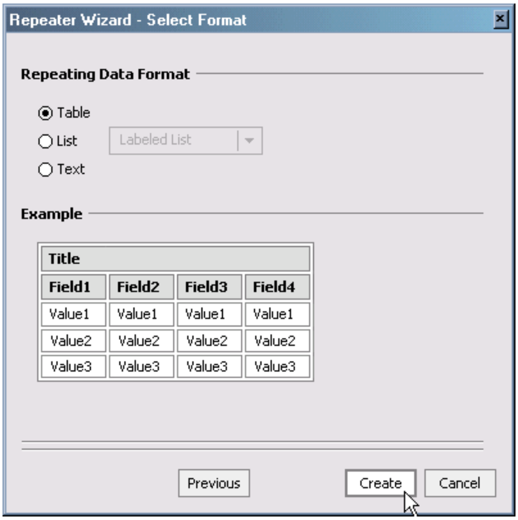
If the data source, or any of its properties, stores a generic type it will appear as a link. Click the link to specify the strong type.

Select Properties

- ☑ schemasBeaComLdCustorder.CustomerOrderReportDocum
  - ☑ CUSTOMERID {java.lang.String}
  - ☑ ORDERDATE {java.lang.String}
  - ☑ ORDERID {java.lang.String}
  - ☑ SHIPMETHOD {java.lang.String}
  - ☑ TOTALORDERAMOUNT {java.math.BigDecimal}

3. Click Create to accept the default repeater data format of Table.

Figure 51 Setting Format for NetUI Repeater Element



Repeater Wizard - Select Format

Repeating Data Format

● Table
○ List Labeled List
○ Text

Example

Title			
Field1	Field2	Field3	Field4
Value1	Value1	Value1	Value1
Value2	Value2	Value2	Value2
Value3	Value3	Value3	Value3

4. Repeat Steps 1 and 2 for Wireless, dragging the Wireless

5. array icon just to the right of the Wireless label. As necessary, drag-and-drop text labels so they appear in the right location.

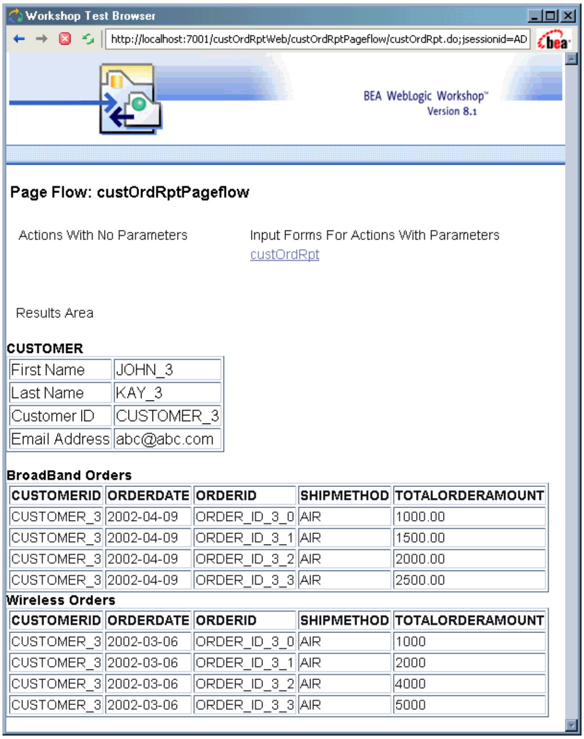
Testing Your Results

Congratulations. Your application is completed. To test it:

1. Run your application using the Start Without Debugging command.
2. When the initial screen appears enter, in uppercase, a valid customer ID such as `CUSTOMER_3`.

When finished, your results page contain the same data as is shown in Figure 52.

Figure 52 Results Page After Query is Run


Workshop Test Browser
http://localhost:7001/custOrdRptWeb/custOrdRptPageflow/custOrdRpt.do;jsessionid=AD
bea
BEA WebLogic Workshop™
Version 8.1

Page Flow: custOrdRptPageflow

Actions With No Parameters

Input Forms For Actions With Parameters
custOrdRpt

Results Area

CUSTOMER

First Name	JOHN_3
Last Name	KAY_3
Customer ID	CUSTOMER_3
Email Address	abc@abc.com

BroadBand Orders

CUSTOMERID	ORDERDATE	ORDERID	SHIPMETHOD	TOTALORDERAMOUNT
CUSTOMER_3	2002-04-09	ORDER_ID_3_0	AIR	1000.00
CUSTOMER_3	2002-04-09	ORDER_ID_3_1	AIR	1500.00
CUSTOMER_3	2002-04-09	ORDER_ID_3_2	AIR	2000.00
CUSTOMER_3	2002-04-09	ORDER_ID_3_3	AIR	2500.00

Wireless Orders

CUSTOMERID	ORDERDATE	ORDERID	SHIPMETHOD	TOTALORDERAMOUNT
CUSTOMER_3	2002-03-06	ORDER_ID_3_0	AIR	1000
CUSTOMER_3	2002-03-06	ORDER_ID_3_1	AIR	2000
CUSTOMER_3	2002-03-06	ORDER_ID_3_2	AIR	4000
CUSTOMER_3	2002-03-06	ORDER_ID_3_3	AIR	5000

Summary

This concludes the Liquid Data Quick Start tutorial. If you followed along with the instructions, you successfully:

- identified disparate data sources
- Created a parameterized query

- Deployed your query to the Liquid Data server
- Created a Liquid Data control
- Used a query from the control in a web application

Generated Files and Code

All the files associated with the Liquid Data query and web application are generated automatically. Table 8 provides some details. Notice that of the nearly 500 lines of code required to produce this rather simple application, only 13 lines were created by hand.

Table 8 Statistics on Generated Files and Code Used to Create the custOrdRpt Web Application

Generated File	Source / Purpose	Total lines of code	Hand coded
`custOrdRpt.xq`	Data View Builder	49	0
`custOrdRpt.jcx`	Liquid Data control, includes target schema	96	0
`custOrdRptController.jpf`	WebLogic Workshop Page Flow Control File	180	6
`index.jsp`	Web application home page	121	7
`custOrdRpt.jsp`	Web application input parameter page	23	0
	TOTAL	**469**	**13**

Topics of Further Interest

Now that you are somewhat familiar with the basics of creating Liquid Data queries using the Data View Builder, the following may be of increased interest:

- *Liquid Data by Example* contains numerous samples and examples beginning starting with a comprehensive survey of Liquid Data Retail Sample Application. That Sample illustrates the data retrieval capabilities of Liquid Data with WebLogic Workshop to create a customer self-service web application.

- Liquid Data *Application Developer's Guide* provides detailed instructions on using results from Liquid Data queries in WebLogic Workshop, through Enterprise Java Beans (EJBs), and in JSP clients.

- Liquid Data *Concepts Guide* provides an end-to-end overview of the product including a survey of the product's capabilities, architecture and role in a WebLogic-centered application development environment.

- *Building Queries and Data Views* provides a detailed GUI overview and description of Data View Builder features.

- The Liquid Data *Administration Guide* provides information about how to configure the data source types supported by Liquid Data, how to configure security, how to set up monitoring and reporting.

Index