# Novell exteNd

5.0

www.novell.com

GUIDED TOUR

Novell®

## Legal Notices

Copyright © 2003 Novell, Inc. All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of the publisher. This manual, and any portion thereof, may not be copied without the express written permission of Novell, Inc.

Novell, Inc. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc. makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to makes changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

Copyright © 2000, 2001, 2002, 2003 SilverStream Software, LLC. All rights reserved.

Title to the Software and its documentation, and patents, copyrights and all other property rights applicable thereto, shall at all times remain solely and exclusively with SilverStream and its licensors, and you shall not take any action inconsistent with such title. The Software is protected by copyright laws and international treaty provisions. You shall not remove any copyright notices or other proprietary notices from the Software or its documentation, and you must reproduce such notices on all copies or extracts of the Software or its documentation. You do not acquire any rights of ownership in the Software.

Patent pending.

Novell, Inc.

1800 South Novell Place

Provo, UT 85606

www.novell.com

exteNd *Guided Tour*

December 2003

**Online Documentation:** To access the online documemntation for this and other Novell products, and to get updates, see www.novell.com/documentation.

## Novell Trademarks

ConsoleOne is a registered trademark of Novell, Inc.
eDirectory is a trademark of Novell, Inc.
GroupWise is a registered trademark of Novell, Inc.
exteNd is a trademark of Novell, Inc.
exteNd Composer is a trademark of Novell, Inc.
exteNd Director is a trademark of Novell, Inc.
iChain is a registered trademark of Novell, Inc.
jBroker is a trademark of Novell, Inc.
NetWare is a registered trademark of Novell, Inc.
Novell is a registered trademark of Novell, Inc.
Novell eGuide is a trademark of Novell, Inc.

## SilverStream Trademarks

SilverStream is a registered trademark of SilverStream Software, LLC.

## Third-Party Trademarks

Acrobat, Adaptive Server, Adobe, AIX, Autonomy, BEA, Cloudscape, DRE, Dreamweaver, EJB, HP-UX, IBM, Informix, iPlanet, JASS, Java, JavaBeans, JavaMail, JavaServer Pages, JDBC, JNDI, JSP, J2EE, Linux, Macromedia, Microsoft, MySQL, Navigator, Netscape, Netscape Certificate Server, Netscape Directory Server, Oracle, PowerPoint, RSA, RSS, SPARC, SQL, SQL Server, Sun, Sybase, Symantec, UNIX, VeriSign, Windows, Windows NT

All third-party trademarks are the property of their respective owners.

## Third-Party Software Legal Notices

**The Apache Software License, Version 1.1**

Copyright (c) 2000 The Apache Software Foundation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (http://www.apache.org/)."

Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The names "Apache" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org.

5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

**Autonomy**

Copyright ©1996-2000 Autonomy, Inc.

### Castor

Copyright 2000-2002 (C) Intalio Inc. All Rights Reserved.

Redistribution and use of this software and associated documentation ("Software"), with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain copyright statements and notices. Redistributions must also contain a copy of this document.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The name "ExoLab" must not be used to endorse or promote products derived from this Software without prior written permission of Intalio Inc. For written permission, please contact info@exolab.org.

4. Products derived from this Software may not be called "Castor" nor may "Castor" appear in their names without prior written permission of Intalio Inc. Exolab, Castor and Intalio are trademarks of Intalio Inc.

5. Due credit should be given to the ExoLab Project (http://www.exolab.org/).

THIS SOFTWARE IS PROVIDED BY INTALIO AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INTALIO OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

### Indiana University Extreme! Lab Software License

Version 1.1.1

Copyright (c) 2002 Extreme! Lab, Indiana University. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Indiana University Extreme! Lab (http://www.extreme.indiana.edu/)."

Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The names "Indiana University" and "Indiana University Extreme! Lab" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact http://www.extreme.indiana.edu/.

5. Products derived from this software may not use "Indiana University" name nor may "Indiana University" appear in their name, without prior written permission of the Indiana University.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS, COPYRIGHT HOLDERS OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

### JDOM.JAR

Copyright (C) 2000-2002 Brett McLaughlin & Jason Hunter. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the disclaimer that follows these conditions in the documentation and/or other materials provided with the distribution.

3. The name "JDOM" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact license@jdom.org.

4. Products derived from this software may not be called "JDOM", nor may "JDOM" appear in their name, without prior written permission from the JDOM Project Management (pm@jdom.org).

In addition, we request (but do not require) that you include in the end-user documentation provided with the redistribution and/or in the software itself an acknowledgement equivalent to the following: "This product includes software developed by the JDOM Project (http://www.jdom.org/)."

Alternatively, the acknowledgment may be graphical using the logos available at http://www.jdom.org/images/logos.

THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE JDOM AUTHORS OR THE PROJECT CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

### Phaos

This Software is derived in part from the SSLavaTM Toolkit, which is Copyright ©1996-1998 by Phaos Technology Corporation. All Rights Reserved. Customer is prohibited from accessing the functionality of the Phaos software.

### Sun

Sun Microsystems, Inc.

Sun, Sun Microsystems, the Sun Logo Sun, the Sun logo, Sun Microsystems, JavaBeans, Enterprise JavaBeans, JavaServer Pages, Java Naming and Directory Interface, JDK, JDBC, Java, HotJava, HotJava Views, Visual Java, Solaris, NEO, Joe, Netra, NFS, ONC, ONC+, OpenWindows, PC-NFS, SNM, SunNet Manager, Solaris sunburst design, Solstice, SunCore, SolarNet, SunWeb, Sun Workstation, The Network Is The Computer, ToolTalk, Ultra, Ultracomputing, Ultraserver, Where The Network Is Going, SunWorkShop, XView, Java WorkShop, the Java Coffee Cup logo, Visual Java, and NetBeans are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

### W3C

W3C® SOFTWARE NOTICE AND LICENSE

This work (and included software, documentation such as READMEs, or other related items) is being provided by the copyright holders under the following license. By obtaining, using and/or copying this work, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions.

Permission to copy, modify, and distribute this software and its documentation, with or without modification, for any purpose and without fee or royalty is hereby granted, provided that you include the following on ALL copies of the software and documentation or portions thereof, including modifications:

1. The full text of this NOTICE in a location viewable to users of the redistributed or derivative work.

2. Any pre-existing intellectual property disclaimers, notices, or terms and conditions. If none exist, the W3C Software Short Notice should be included (hypertext is preferred, text is permitted) within the body of any redistributed or derivative code.

3. Notice of any changes or modifications to the files, including the date changes were made. (We recommend you provide URIs to the location from which the code is derived.)

THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR DOCUMENTATION.

The name and trademarks of copyright holders may NOT be used in advertising or publicity pertaining to the software without specific, written prior permission. Title to copyright in this software and any associated documentation will at all times remain with copyright holders.

# Contents

**About This Guide** . . . . . **9**

**1 Getting Started** . . . . . **11**
About Novell exteNd 5 . . . . . 11
About the Guided Tour . . . . . 12
Overview of the application . . . . . 12
Objective . . . . . 12
How you will proceed . . . . . 13
About this lesson . . . . . 13
Objective . . . . . 13
What you will do . . . . . 13
EXERCISE 1-1: Create a project area folder . . . . . 14
EXERCISE 1-2: Create the MySQL database . . . . . 14
EXERCISE 1-3: Start the exteNd Application Server . . . . . 16
EXERCISE 1-4: Create a JDBC connection pool . . . . . 16
EXERCISE 1-5: Start the exteNd LDAP directory server . . . . . 19

**2 Completing the exteNd Composer Project** . . . . . **21**
About this lesson . . . . . 21
What is exteNd Composer? . . . . . 21
Objective . . . . . 21
What you will do . . . . . 22
EXERCISE 2-1: Start exteNd Composer and open the project . . . . . 22
EXERCISE 2-2: Review the LDAP component . . . . . 23
EXERCISE 2-3: Sidebar: the exteNd Composer Component Editor Layout . . . . . 24
EXERCISE 2-4: View the input and output documents for the LDAP component . . . . . 27
EXERCISE 2-5: View the LDAP connection resource . . . . . 28
EXERCISE 2-6: Review the Action Model . . . . . 29
EXERCISE 2-7: Test the component . . . . . 34
EXERCISE 2-8: Complete the Action Model for the service . . . . . 35
EXERCISE 2-9: Create the Web Services Definition Language (WSDL) . . . . . 41

**3 Deploying and Testing the Web Service** . . . . . **45**
About this lesson . . . . . 45
Objective . . . . . 45
What you will do . . . . . 45
EXERCISE 3-1: Start exteNd Director and open the project . . . . . 46
EXERCISE 3-2: Create a SOAP-based service trigger . . . . . 48
EXERCISE 3-3: Sidebar: the exteNd Director development environment . . . . . 51
EXERCISE 3-4: Deploy the application . . . . . 53
EXERCISE 3-5: Test the Web Service using exteNd Composer . . . . . 54

**4 Completing the exteNd Director Project . . . . . 59**
About this lesson . . . . . 59
What is exteNd Director? . . . . . 59
Objective . . . . . 59
What you will do . . . . . 60
EXERCISE 4-1: Create a Web Service pageflow using a wizard . . . . . 60
EXERCISE 4-2: Sidebar: the resource set in an exteNd Director application . . . . . 64
EXERCISE 4-3: Test the default pageflow you created . . . . . 66
EXERCISE 4-4: Edit additional XForms . . . . . 73
EXERCISE 4-5: Add a new form to the pageflow . . . . . 79
EXERCISE 4-6: Change links to include the new form in the pageflow . . . . . 83
EXERCISE 4-7: Test the revised pageflow . . . . . 85
Summary . . . . . 87

# About the Guided Tour

### Purpose

This *Guided Tour* introduces you to Novell® exteNd™ and shows you how to develop a Web Service and build it into a Web application.

### Audience

This *Guided Tour* is for application developers familiar with building applications that provide a graphical user interface.

### Prerequisites

**Experience** This *Guided Tour* assumes you are familiar with visual development tools and comfortable with building forms that bind to data sources such as relational databases.

**Required software** If you installed the Novell exteNd Professional Suite 5.0 or Enterprise Suite 5.0, everything you need is on your computer.

This includes:

- Novell exteNd Application Server 5.1
- Novell exteNd Composer™ 5.0
- Novell exteNd Director™ 5.0

Plus:

- MySQL as the database
- exteNd LDAP as the LDAP directory
- The exteNd Director template project, which includes an exteNd Composer project (these projects are included in the GuidedTour.zip file in Docs/GuidedTour in your exteNd installation directory)

If you don’t have the required software, you can download the trial version of Novell exteNd 5 from the download Web site.

## Organization

Here's a summary of the lessons you'll find in the *Guided Tour*:

| Lesson | | Description |
|---|---|---|
| 1 | Getting Started | Provides an overview of the *Guided Tour* application and shows you how to set up the *Guided Tour* project and database |
| 2 | Completing the exteNd Composer Project | Shows you how to complete an exteNd Composer application that extracts information from an LDAP directory |
| 3 | Deploying and Testing the Web Service | Shows you how to deploy and test the exteNd Composer Web Service component you have created |
| 4 | Completing the exteNd Director Project | Shows you how to complete an exteNd Director portlet application that invokes the deployed Web Service |

## Latest updates

For the latest updates to this *Guided Tour* and the source files used, go to the documentation Web site.

# 1 Getting Started

## About Novell exteNd 5

Novell exteNd 5 is a comprehensive suite of development tools that gives you the ability to create end-to-end Web Services that are secure with little to no coding.

- With exteNd Composer you can produce Web Services that use non-XML data sources such as directories and relational databases.
- With exteNd Director you can write applications that allow end users to consume Web Services. You do this by building portlets that can then be deployed as part of portal Web applications.
- By also using exteNd Director to create forms that are bound to the data in Web Services you can craft truly complete business solutions with minimal coding.
- Finally, by deploying your application to the exteNd Application Server you are deploying to a standards-based server that provides a robust environment to support your applications.

Novell exteNd 5 provides a service-oriented architecture (SOA) that is standards based. exteNd 5 gives you all the tools you need to develop Web Services and Web applications to meet the ever-changing needs of your business—while using current and evolving standards that do not lock you into a specific platform.

**For more information**

| For | See |
|---|---|
| More information about Novell exteNd | The Web sites listed in the exteNd Help Library menu at the top of this page |
| More information about the technologies discussed in this *Guided Tour* | The Learning Resources page in *Welcome to Novell exteNd* in this help system |
| Latest updates to this *Guided Tour* | Check the exteNd documentation Web site frequently for updates to this *Guided Tour* |

# About the *Guided Tour*

## Overview of the application

The application (it is more like a part of a larger application) you will build in this *Guided Tour* will retrieve basic contact information from an LDAP directory (such as Novell eDirectory™). You will retrieve basic information such as full name, title, phone, and e-mail address. Then the information will be displayed in a portlet that is part of a Web application. While this is a simple read application, it is the start of a more complete solution—where users can maintain contact information about themselves or other directory users.

The Web Service you will create could also be used by other portlets to tailor their functionality to give a customer or other user of your application a personalized look. In developing the Web applications of tomorrow, developers need to provide users the information they need. The simple example you will build leads you in that direction by showing how to retrieve information specific to a user based on a centralized directory repository.

## Objective

This *Guided Tour* is not intended to be a complete coding exercise—but rather an introduction to exteNd and how you can use it to create Web Services and use them in a Web application. Your starting point will be a partially completed project. You will write some code to complete the application, deploy the application to the exteNd Application Server, and test the code you have completed by running the application in your favorite Web browser.

The code provided for you is well commented, and it would be instructive to review it as you work through the *Guided Tour*. You will become familiar with some of the key artifacts. You will also learn how exteNd Composer and exteNd Director help you build applications that solve business problems while minimizing the code you need to write.

### How you will proceed

You will begin by performing the setup of the template project and database required by the application.

Then you will complete an exteNd Composer service that will extract the user information from the LDAP directory, ultimately deploying this as a Web Service.

After completing the exteNd Composer part you will go into exteNd Director to complete the user interface. The Web application will request a User ID and invoke the Web Service, returning either the information about the user or an error message. The returned information will then be displayed.

*Now you are ready to start the first lesson.*

## About this lesson

### Objective

In this lesson you will extract the template project to create your project work area and create a database required for the exteNd Director application. You will set up a connection pool in the exteNd Application Server for the exteNd Director database. You will also start the exteNd Application Server and the exteNd LDAP server that you will use for deployment and testing.

### What you will do

1. Create a project area folder
2. Create the MySQL database
3. Start the exteNd Application Server
4. Create a JDBC connection pool
5. Start the exteNd LDAP directory server

## EXERCISE 1-1: Create a project area folder

The template project is located in the `\Docs\GuidedTour directory` under your exteNd 5 installation root directory. (The default installation root directory is `C:\Program Files\Novell\exteNd5`. This is the path name that will be used in the examples in this *Guided Tour*.) The project is in a zip file called **GuidedTour.zip**.

- Extract the file **template.zip** from GuidedTour.zip, and unzip template.zip into a working directory on your computer, such as D:\GuidedTour. When extracting the file, make sure you use an option such as Use Folder Names, which will keep the folder structure of the files in the template.zip during the extract. This will be your working copy of the *Guided Tour* project.

  The folder template with the folder **RequestUserInfoProject** is created in your working directory. RequestUserInfoProject is the exteNd Director project folder.

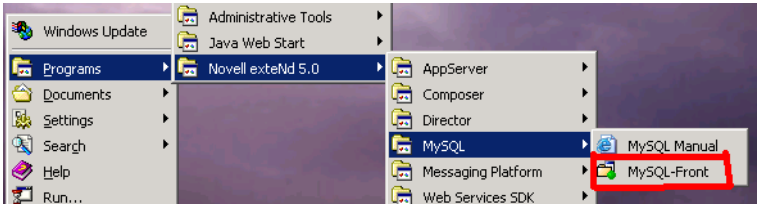
  **TIP:** All references to these projects throughout the *Guided Tour* use the base location of D:\GuidedTour\template.

## EXERCISE 1-2: Create the MySQL database

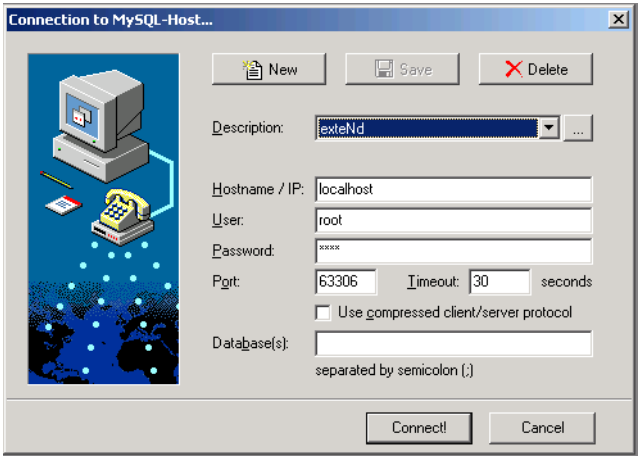
An exteNd Director application requires a database to store the data specific to exteNd Director. For the application you are building you will need to create a database and set up a connection pool in the exteNd Application Server. Now you will create a MySQL database.

**TIP:** You can actually use any database supported by exteNd Director. For information on the supported databases, see the Release Notes for exteNd 5.

1 Start MySQL-Front by going to the **Start>Programs>Novell exteNd 5.0>MySQL** menu and clicking **MySQL-Front**.

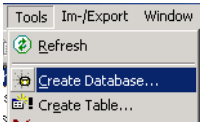


2 Using your MySQL root user, connect to the MySQL server. In the Password text box, type the password you provided for MySQL during the exteNd 5 installation process. Verify that the port for the connection is **63306**. You must use the version of MySQL that was installed with exteNd.

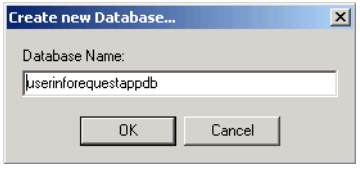



3 Click the **Connect** button.

4 Select **Tools>Create Database** from the menu.




5 Type `userinforequestappdb` in the Database Name text box.



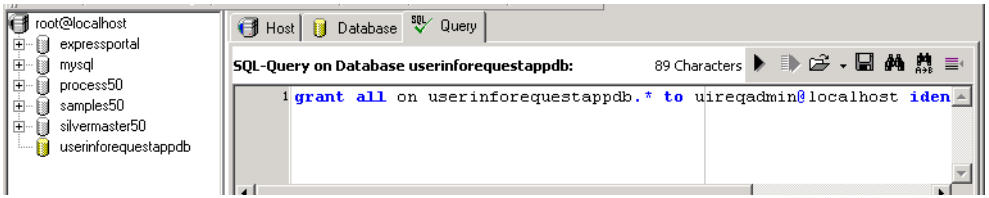

6 Click **OK**.

It is good practice to assign an administrative ID other than `root` to a database. This ensures that users of your applications are only running with the restricted privileges (instead of the unrestricted access allowed to the root user).

7 Make sure the database you just created is selected. Then click the **SQL Query** tab.

8 Type the following SQL statement into the Query tab:

```
grant all on userinforequestappdb.* to uireqadmin@localhost
identified by 'uireqadmin';
```

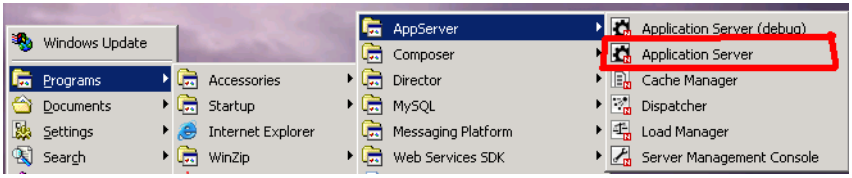



9 Click the Execute SQL icon or press **F9**.

10 Close the MySQL-Front application by selecting **File>Exit**.

## EXERCISE 1-3: Start the exteNd Application Server

- To start the Application Server, go to the **Start>Programs>Novell exteNd 5.0>AppServer** menu and select **Application Server**. (Do not select the debug option.)



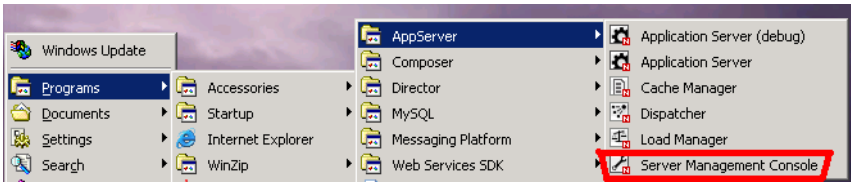

The application server has completed its startup initialization when you see the following text on the server console (the build number and server/port will be specific to your install):

```
exteNd application server [Build Number:Release5.1.0 (031203_1)] serving at:
http://jgdext5testvm:80 (runtime, admin)
```

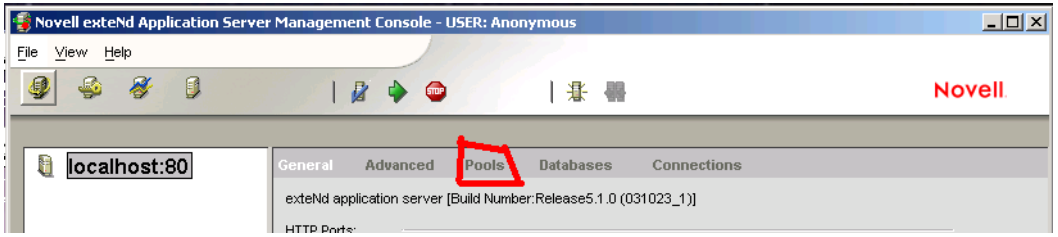
## EXERCISE 1-4: Create a JDBC connection pool

The database you created will be accessed as a connection pool in the Application Server. This improves the performance of the application and provides flexibility in deployment by leaving the specification of which database to use until deployment time. Now you will define a pool in the exteNd Application Server.

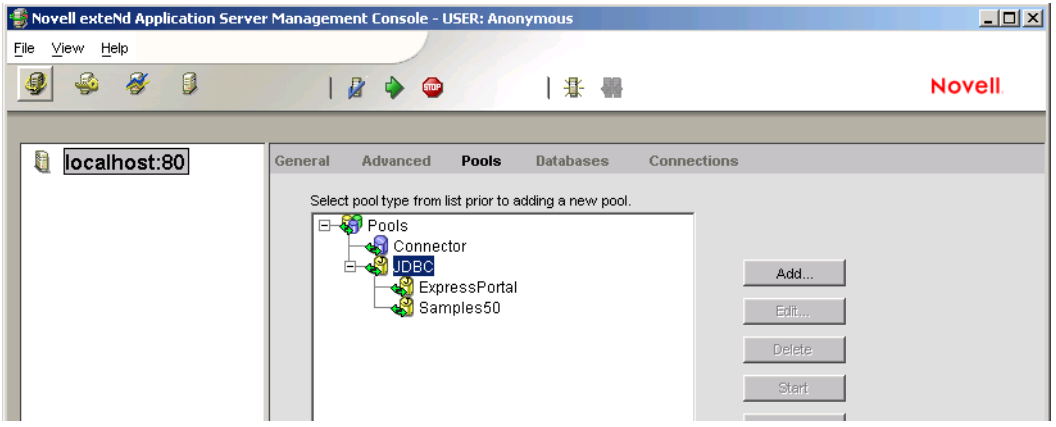
1 When the Application Server is ready, start the Server Management Console (SMC) by going to the **Start>Programs>Novell exteNd 5.0>AppServer** menu and selecting **Server Management Console**.




2 Select the **Pools** tab in the **Configuration** section of the SMC.



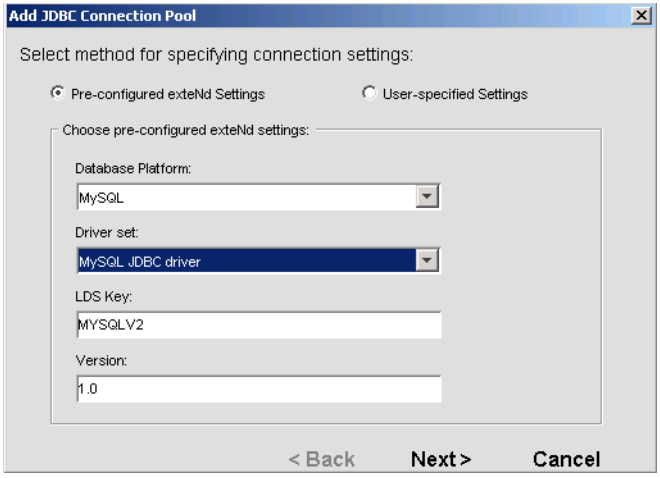

3 Select **JDBC** as the Pools Type.




4 Click **Add**.

5 Select **MySQL** as the Database Platform from the dropdown list.

6 If not already selected, select the **MySQL JDBC** driver from the Driver Set dropdown list.

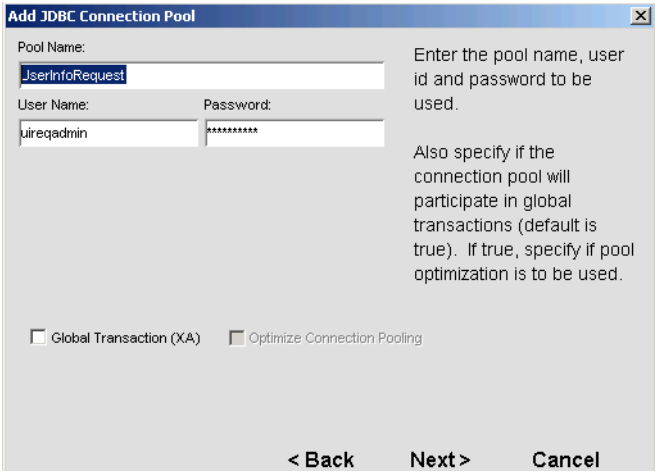



7 Click **Next**.

8 Type `UserInfoRequest` in the **Pool Name** text box.

The complete pool name for a JNDI lookup will be `JDBC/UserInfoRequest`.

9 Type `uireqadmin` in both the **User Name** and **Password** text boxes.




10 Click **Next**.

The JDBC URL template is:

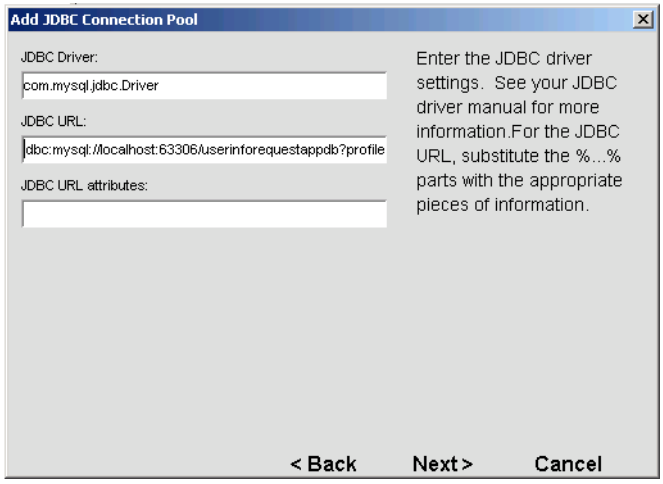
```
jdbc:mysql://%HOST%:%PORT%/%DBNAME%?profileSql=false&maxRows=0
```

11 Change the variables in the template as follows:

- Change **%HOST%** to **localhost**.
- Change **%PORT%** to **63306**.
- Change **%DBNAME%** to **userinforequestappdb**.

The URL should read:

```
jdbc:mysql://localhost:63306/userinforequestappdb?profileSql=false&maxRows=0
```




12 Click **Next**.

13 Click **Next** on the Optional Connection Pool Properties dialog.

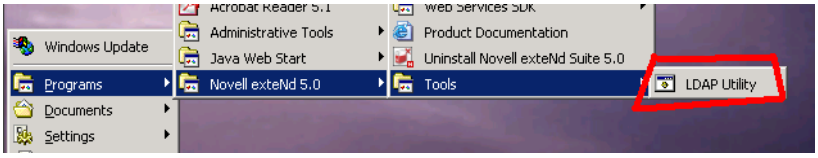
14 Accept the default connection pool manager properties by clicking **Finish**.

15 Close the SMC application by selecting **File>Exit**.

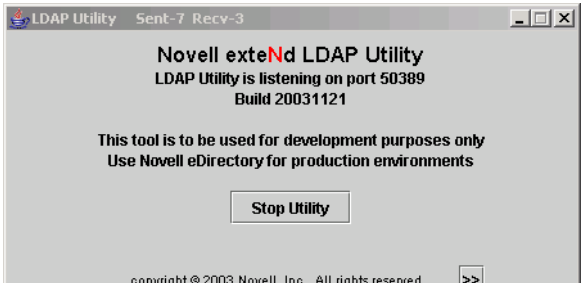
## EXERCISE 1-5: Start the exteNd LDAP directory server

The Novell exteNd LDAP directory server is the data source that will be used by the exteNd Composer Web Service you will complete in the next lesson. exteNd LDAP is a simple LDAP directory implementation that provides an LDAP server to be used by developers for development and testing.

1 Start the LDAP server by going to the **Start>Programs>Novell exteNd 5.0>Tools** and clicking **LDAP Utility**.



2 Minimize the Novell exteNd LDAP Utility window by clicking the **Minimize** icon.




**Next lesson** In the next lesson you will complete an exteNd Composer service that you will later deploy as a Web Service.

# 2 Completing the exteNd Composer Project

## About this lesson

### What is exteNd Composer?

Novell exteNd Composer is a development (and runtime) environment for the rapid design and deployment of Web Services and XML integration to applications—applications that can connect to diverse back-end (legacy) systems and data sources. exteNd Composer offers a flexible, intuitive, point-and-click GUI (graphical user interface), giving the business analyst or application developer a powerful tool for creating robust XML integration solutions in minimum time.

### Objective

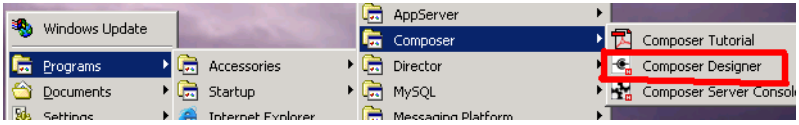
This project provides an exteNd Composer component that extracts information from an LDAP directory. The component (using the LDAP Connect) provides basic user information such as name, phone, and title. In this lesson you will complete the application by completing an exteNd Composer service for later deployment as a Web Service. The Web Service will receive a request with a user ID and provide the information supplied by the component.

## What you will do

1 Start exteNd Composer and open the project
2 Review the LDAP component
3 Sidebar: the exteNd Composer Component Editor Layout
4 View the input and output documents for the LDAP component
5 View the LDAP connection resource
6 Review the Action Model
7 Test the component
8 Complete the Action Model for the service
9 Create the Web Services Definition Language (WSDL)

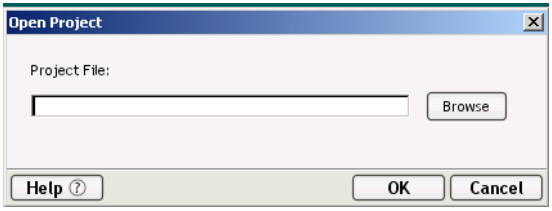
### EXERCISE 2-1: Start exteNd Composer and open the project

1 Start exteNd Composer by going to the **Start>Programs>Novell exteNd 5.0>Composer** menu and selecting **Composer Designer**.

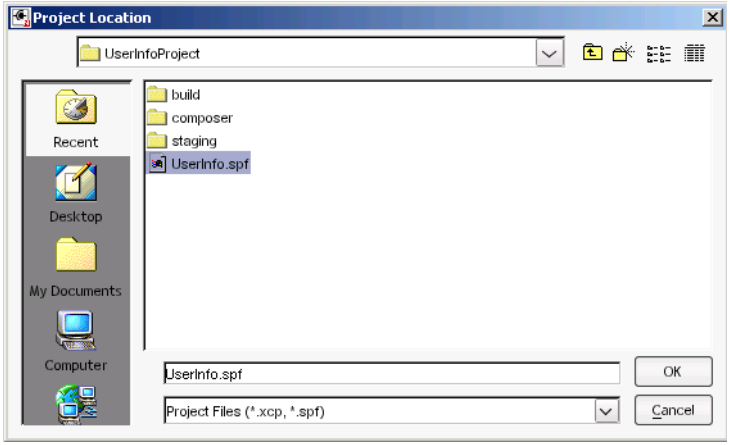



In the interest of time, you will use a template project that has some of the artifacts you need already coded. You will review some of the artifacts and complete the steps required to finish this part of the application.

2 Open the project by selecting menu **File>Open Project**.



3 Navigate to the project folder by clicking the **Browse** button. The project folder is: D:\GuidedTour\template\RequestUserInfoProject\UserInfoProject.




4 Select the project file **UserInfo.spf** and click **OK**.

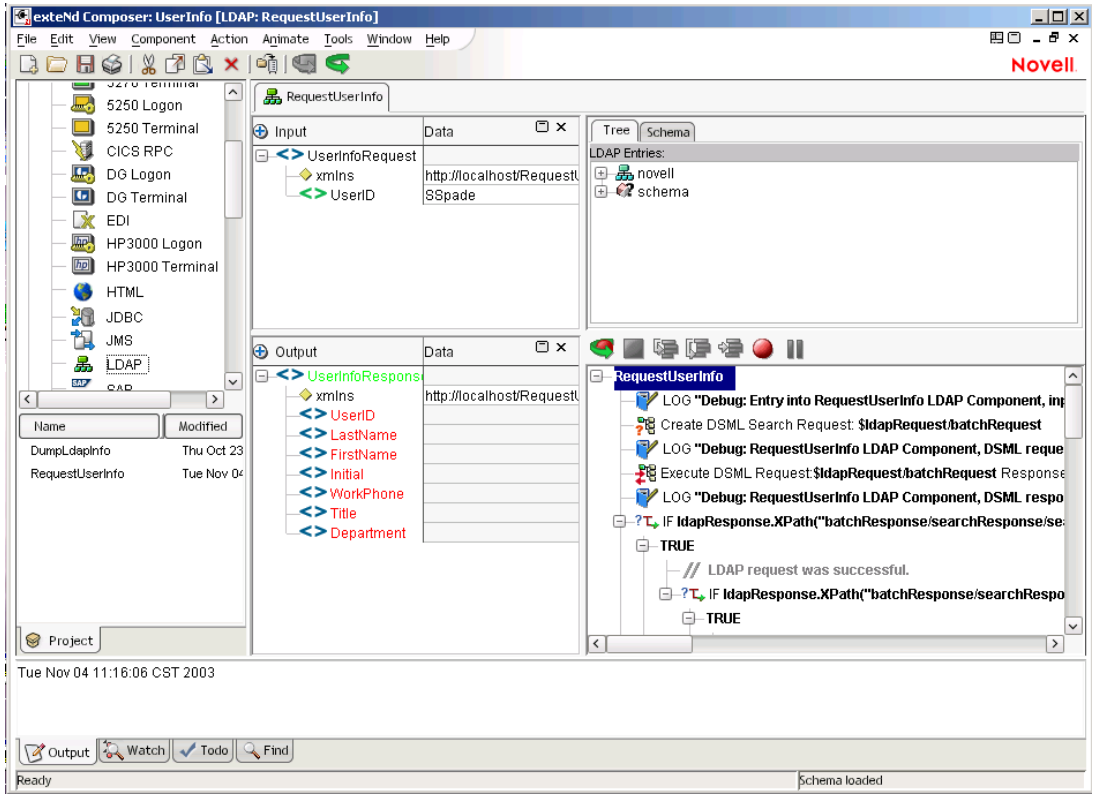
5 Click **OK**.

## EXERCISE 2-2: Review the LDAP component

In this exercise you will review the component that retrieves the information from the exteNd LDAP server. This component was created using exteNd Composer LDAP Connect.

1 In the Category Pane of the Navigation Pane, select **Component** and then **LDAP**.

2 In the Detail Pane of the Navigation Pane, double-click **RequestUserInfo**. This opens the component in the Editor Pane.

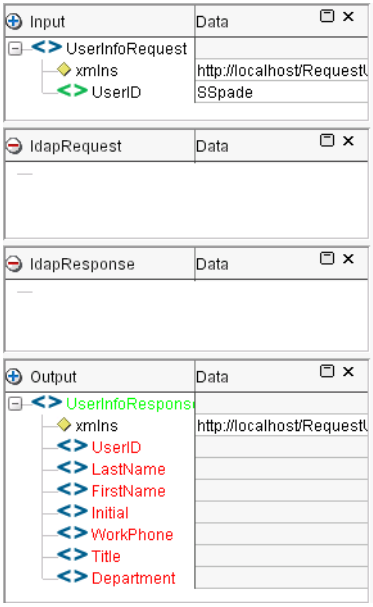



## EXERCISE 2-3: Sidebar: the exteNd Composer Component Editor Layout

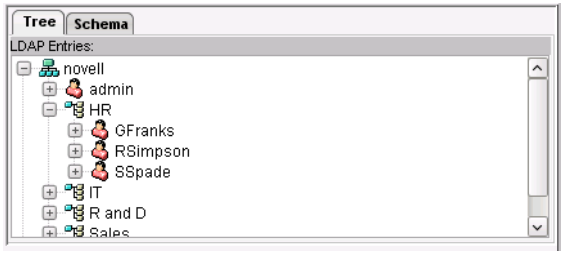
Now you will take a look at the exteNd Composer functionality that allows you to quickly build and test solutions that access legacy data sources.

Step 2 of the previous exercise shows the exteNd Composer Designer with an LDAP Component open in the Editor panel. The Editor Pane is divided into three subpanes:

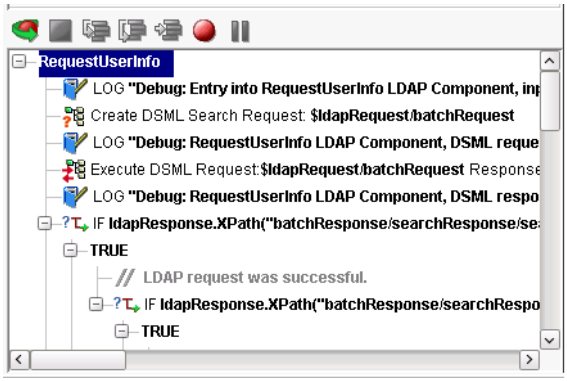
- In the **XML Documents or DOM** Pane (DOM stands for Document Object Model), you can view and edit the XML documents being manipulated by the component. To make it easier to work with the exteNd Composer Designer, you can control which of these DOMs are visible at any time.




- The **Native Environment** Pane is used by the Novell exteNd Connect products to provide a native view into the data source for which the exteNd Connect product provides the integration interface. So in exteNd LDAP Connect you see a graphical view of the Directory tree.



- In the **Action Model** pane, you work with the exteNd Composer actions associated with the current component.




An exteNd Composer action is similar to a programming statement. It takes input in the form of parameters and performs specific tasks.

An exteNd Composer component consists of a set of instructions for processing XML documents or communicating with non-XML data sources. This set of instructions is called an Action Model. In exteNd Composer components and services, the Action Model performs all data mapping, data transformation, and data transfer tasks.

**Action Model toolbar** The toolbar above the Action Model gives you access to exteNd Composer's **animation tools**. These tools allow you to test and troubleshoot the actions associated with your services and components. You can step through a service or component Action Model one step at a time and see the result of each action. By doing this, you will become aware of any action failures as well as be able to verify that the output of the component is what you expected.

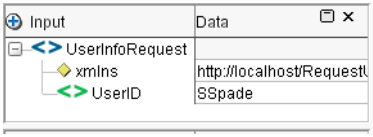
The animation tools allow you to set breakpoints so you can test individual sections of an Action Model. Using breakpoints, you can easily troubleshoot an Action Model by stopping execution just before any action that is causing trouble, and stepping through that and subsequent actions one at a time.

| Button | Tool | What it does |
|---|---|---|
| | Start Animation | Starts the animation process. |
| | End Animation | Stops the animation process. |

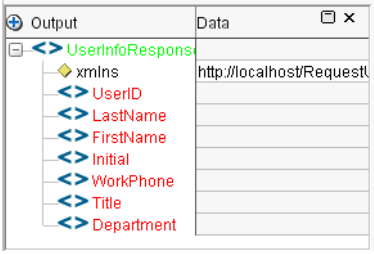
| Button | Tool | What it does |
|---|---|---|
| | Step Into | Executes the currently selected action and highlights the next sequential action. If the currently selected action is a Component, Repeat, Decision, or Try/On Error action, the next highlighted action becomes the details of those actions. |
| | Step Over | Executes the currently selected action and highlights the next sequential action. Unlike the Step Into button, clicking this button does not highlight and execute the details of Component, Repeat, Decision, or Try/On Fault actions. |
| | Toggle Breakpoint | Sets the highlighted action in the Action Model as a breakpoint. You may set more than one breakpoint. |
| | Run To Breakpoint/End | Runs the animation to the next breakpoint or to the end of the Action Model. |
| | Pause Animation | Pauses the animation. |

## EXERCISE 2-4: View the input and output documents for the LDAP component

The Input XML document is an XML template containing a sample of the data required to make a request for user information. You use this template as a guide for writing the Action Model and testing the execution of the model.

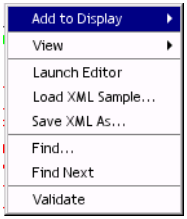



The Output XML document is also an XML template. It contains a sample of the data that will be provided to fulfill the request.



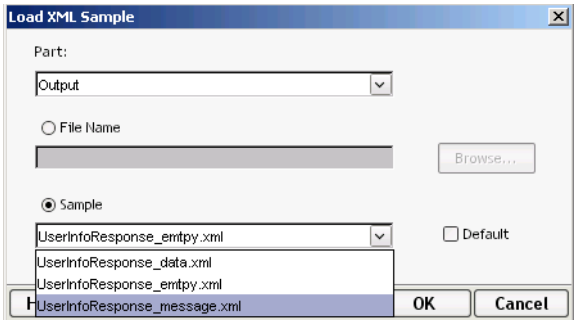
The sample of the Output XML is what is sent if the component is successful in connecting to the directory and finding the requested user information. If the user does not exist, an error message is sent. You will now view the sample data and the XML structure in the Output XML document.

1 In the Output XML document in the Component Editor, right-click to open the context menu.

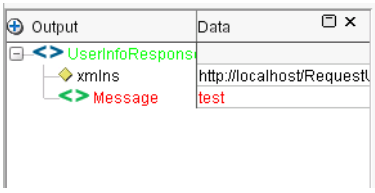



2 Select **Load XML Sample**.

3 Click the **Sample** dropdown list then select **UserInfoResponse_message.xml**.




4 Click **OK**.

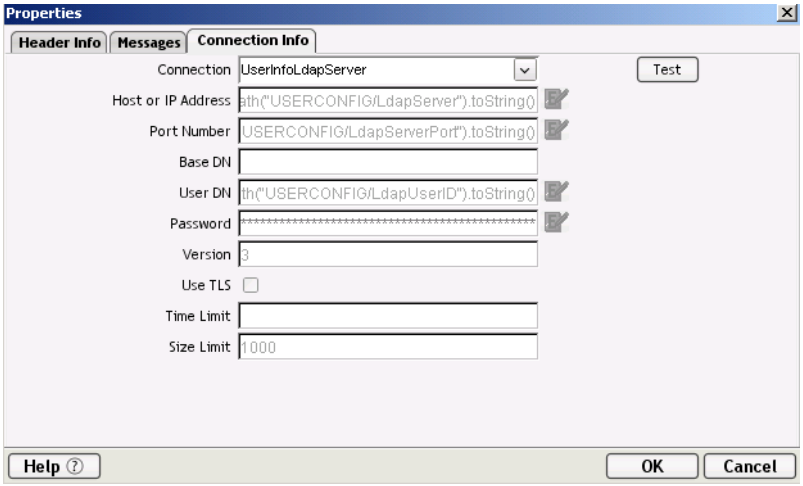



## EXERCISE 2-5: View the LDAP connection resource

The LDAP component requires a connection resource to connect to an LDAP server. The connection resource provides the specific information required to establish a connection to the exteNd LDAP directory server you are using. Now you will view the resource used by this component.

1 From the menu bar select **File>Properties**.

2 Select the **Connection Info** tab.

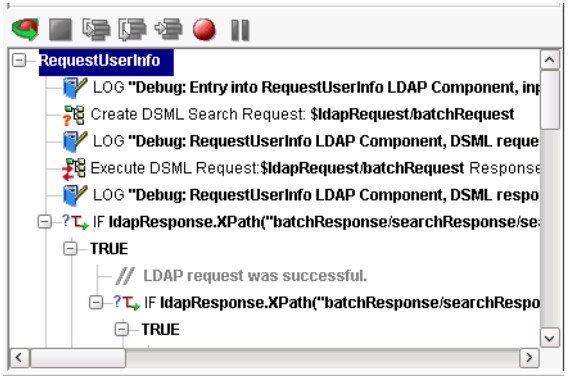



The Connection Info tab provides the information for the connection resource being used in a read-only mode. By clicking the Test button you can verify that a connection to the exteNd LDAP server can be established. Note the icon next to some of the field values: this indicates that an ECMAScript expression is used to create the required value. The actual values are stored as project variables in your project, where you can easily change them in one place or even dynamically before the component is executed.

3 Click **OK** to close the dialog.

## EXERCISE 2-6: Review the Action Model

The Action Model is the collection of instructions that are executed in sequence to process the input to produce the output. In this exercise you will review some of the key actions in the LDAP component that will be executed by the Web Service you are going to complete.

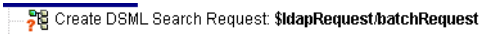



## Log actions

The first, third, and fifth actions are **Log** actions used to report debug-level information. This logging can be turned on by setting exteNd Composer's Designer or Server Log threshold. After deployment to an application server, these Log actions may be helpful in tracking down problems with a deployed application. It is good development practice to be proactive about adding them to your code. If you want to see the results of the debug Log actions in this project, you must set the Log Threshold in the Preferences to 4 or lower.

## Create DSML Search Request action

The second action is a **Create DSML Search Request** action, used to create an LDAP request as a DSML XML document. This action creates the actual query details (the where clause) and identifies the attributes that are to be returned.

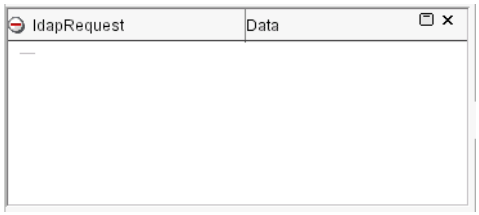



When you select this action, the Native Environment Pane allows you to review and edit the details of the search request. The UserID in the Input XML Document is used to query the uid attribute in the directory.

The result of executing this action is an XML document that is placed in a temporary XML DOM.

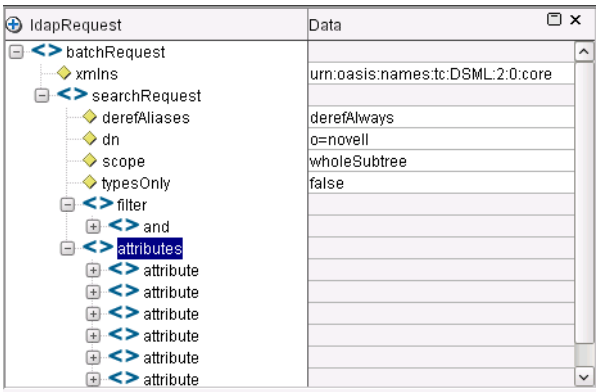
Now you will view this document in the XML Documents panel.

1 Right-click in the XML Document panel, select the **Add to Display** submenu, and select **ldapRequest**. This is the temporary XML DOM used to hold the XML created by the Create DSML Action.




2 In the Action Model, select the **Create DSML Search** action.

3 Click the **Execute Current Action** icon on the toolbar. This executes the action and displays the XML created for the request in the XML Document panel.



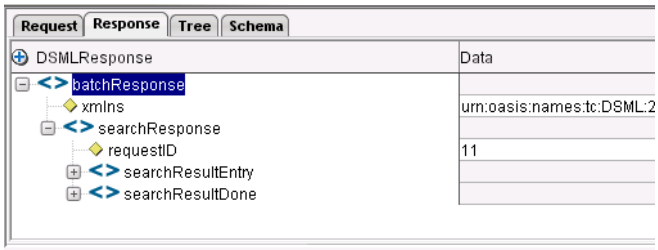

4 Now click the **Execute All** icon on the toolbar. This causes the entire component to be executed, providing you the results of a successful query.

5 Click **OK** at the Execution Completed message dialog.

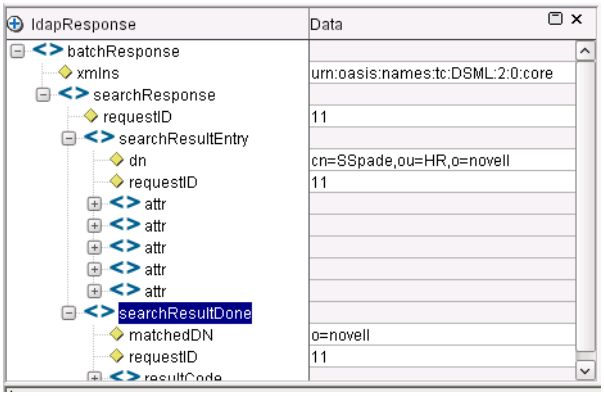
### Execute DSML Request action

The fourth action is an **Execute DSML Request** action. This action takes the request created by the Create DSML Search Request action and connects to the server to execute the request. The result is placed into another temporary XML document called **ldapResponse**.

Here is what the Native Environment Pane looks like when the action is selected, after execution:




Here is the ldapResponse XML document.

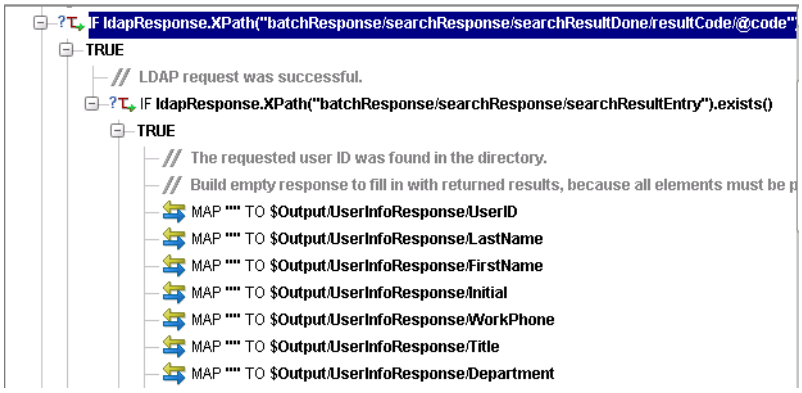



- Use the steps you learned above to make the **ldapResponse XML document** visible.

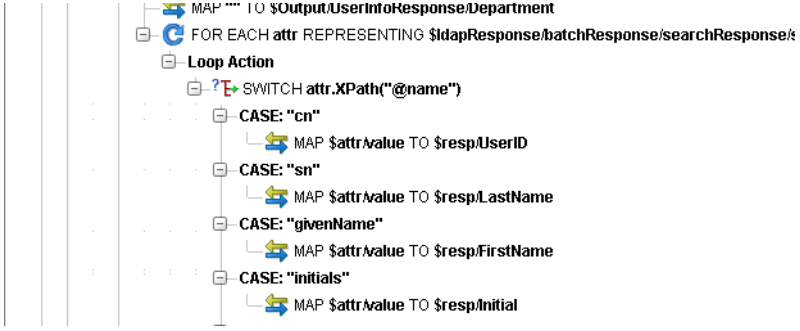
### Decision action

The sixth action is a **Decision** (IF) action. This action checks for a successful request. If the request is successful, the TRUE block of actions will be executed. This sequence of actions is used to produce the required output you saw in the Output XML Document earlier.

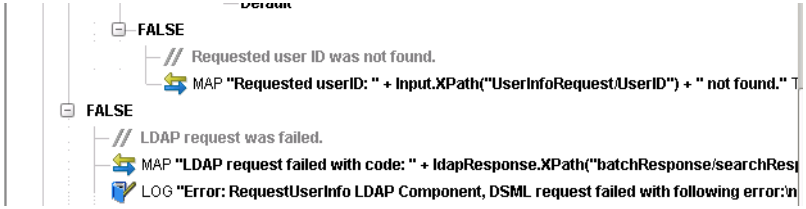
First the XML document is created with no data to ensure that the structure of the XML document conforms to the schema. exteNd Composer will always create well-formed XML, but it is your responsibility to ensure that the XML is valid when an XML Schema or DTD is specified.




The next step is to handle each of the attributes returned in the result and place the data in the XML document. The DSML response will contain a repeating element for each of the attributes that was requested and is available for that specific entry in the directory. The Action Model must loop through each of these repeating elements and check the attribute name, placing the value into the correct element in the output XML document.

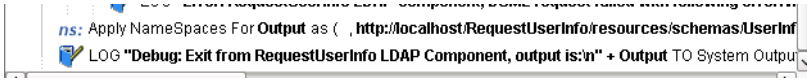


The FALSE blocks of the Decision actions contain **Map** actions that create message structures used in the event of failure, and a **Log** action used to log the failure of an LDAP request.




### Apply Namespace action

The final action is an **Apply Namespace** action that adds a reference to the XML Schema namespace that can be used to validate this document.



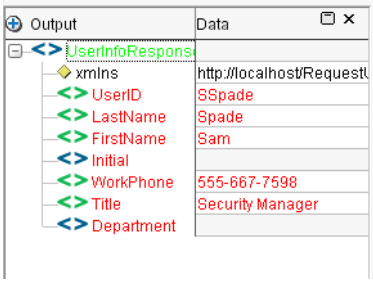

## EXERCISE 2-7: Test the component

With the exteNd Composer Designer, you can test the execution of the component without having to deploy to a J2EE application server. You will now test the component—first with a **valid** user ID and then with an **invalid** ID.

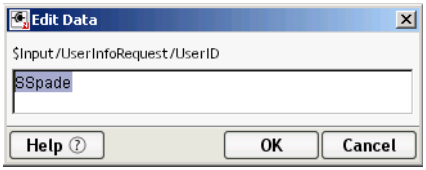
1 The valid user ID is already provided for in the default XML Input sample. Click the **Execute All** icon on the toolbar.

2 Click **OK** for the Execution Completed message dialog.

3 Review the Output XML Document.



4 To enter an invalid User ID you need to change the Input sample. Double-click the value **SSpade**.




5 Type **xxx** replacing SSpade.

6 Click **OK**.

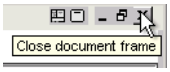
7 Click the **Execute All** icon on the toolbar.

8 Click **OK** for the Execution Completed message dialog.

9 Review the Output XML Document.

You have now finished reviewing the LDAP component. To close the LDAP component:

10 Click on the **Close** icon in the editor to close the component.




11 If prompted to save changes, Click **No**.

## EXERCISE 2-8: Complete the Action Model for the service

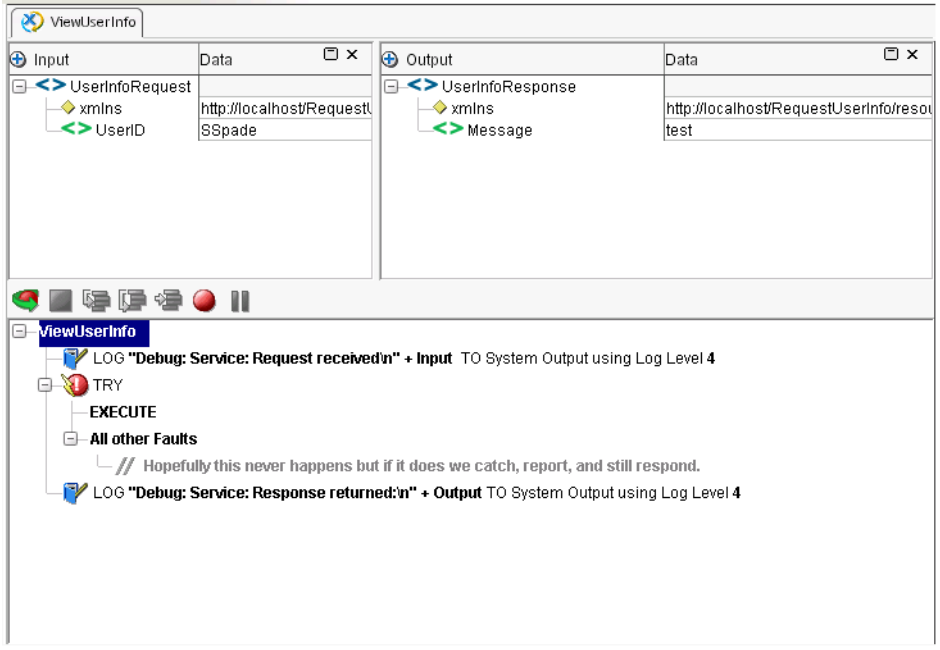
With exteNd Composer, a service is required to act as the interface to the outside world. An application that uses the LDAP component you reviewed must invoke a service through a Service Trigger (more on this later) that will execute the component or even possibly other components. A service has its own Action Model. In this exercise you will complete the service that accepts the requested user ID and returns the information supplied by the RequestUserInfo component.

1 In the Category Pane of the Navigation Pane, select **Service** and then **Web Service**.

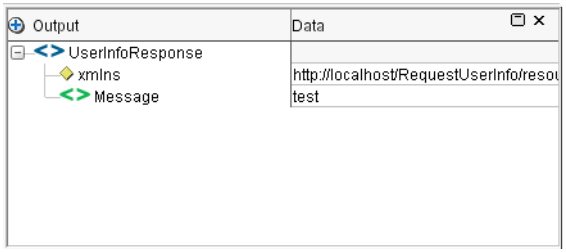
Although the service you selected is classified as a Web Service, it does not necessarily mean that it has a WSDL and SOAP binding. It is a **Web Service** only in the sense that it will normally be invoked as part of a Web application and that it responds to the request/response model used with HTTP.

The service trigger (or triggers) for invoking the service are created when it is deployed to a J2EE application server (like the Novell exteNd Application Server). One of the deployment options available to you is to use a Web Service with a WSDL and SOAP binding.

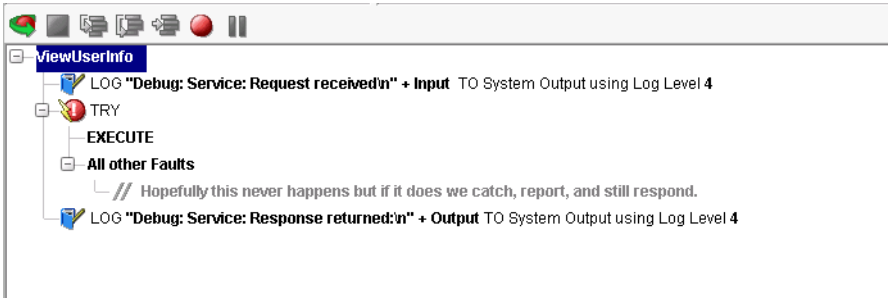
2 In the Detail Pane of the Navigation Pane, double-click **ViewUserInfo**. This opens the service in the Editor Pane.



Note that the XML Output sample used for the service is the message response. This was set as the default because the service will normally take the output produced by the LDAP component as its output. But in the event that the component fails to execute for any reason, the service will respond with a message.




The Action Model for the service has been partially completed for you. The **Try On Fault** action is a wrapper for the execution of the component; if it fails, the Fault section will be executed and a Log action will send a log message to the System Output. You will add the action to execute the RequestUserInfo component and a Map action to create the XML output in the event of a fault.

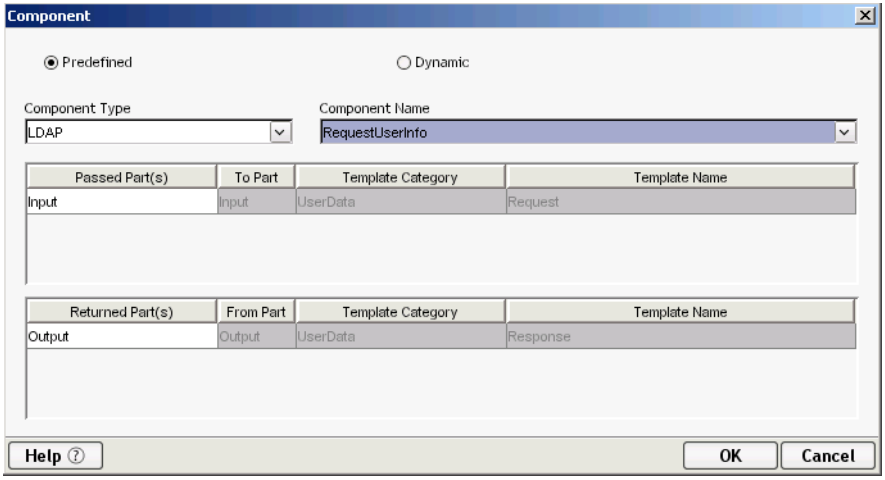



3 Select the **EXECUTE** in the TRY action.

4 Right-click to display the context menu. Then select **New Action>Component**.

5 In the Component dialog, enter the following:

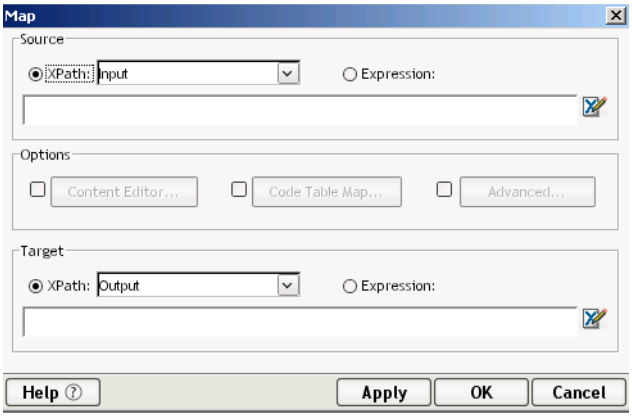
- Component Type: **LDAP**
- Component Name: **RequestUserInfo**
- Passed Part: **Input**
- Returned Part: **Output**



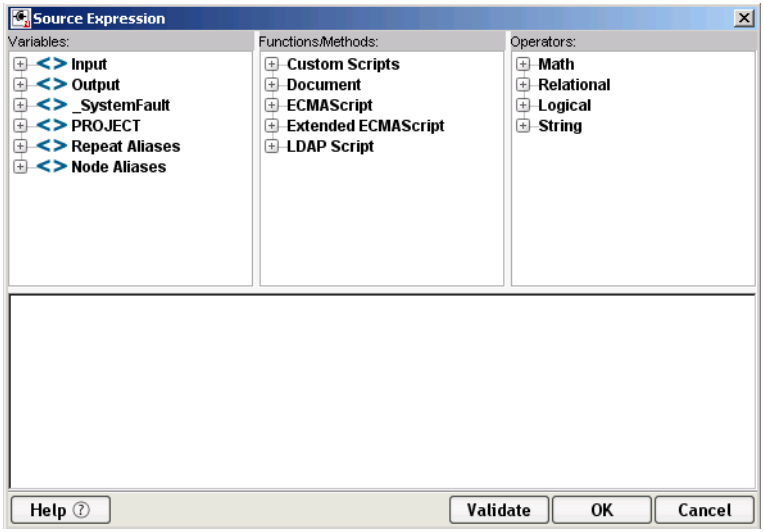

6 Click **OK**.

7 Select **All other Faults** of the Try/On Fault action.

8 Right-click to display the context menu. Then select **New Action>Map** (or use the keyboard shortcut **Ctrl+M**).



9 In the Source area click the **Expression** radio button, then click the ECMA Expression icon to open the expression editor.




10 Type the following (include the quotes):

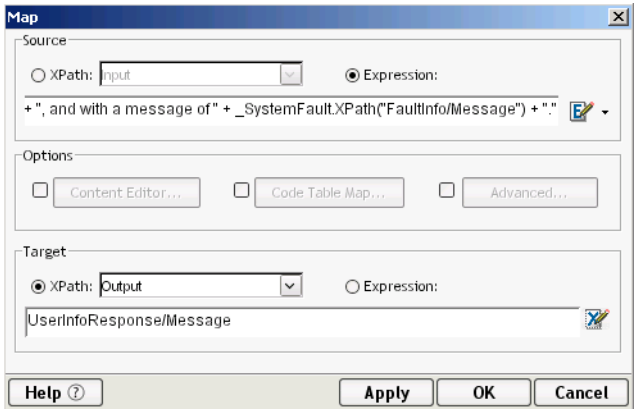
```
"Error in component " +
_SystemFault.XPath("FaultInfo/ComponentName") + " at " +
_SystemFault.XPath("FaultInfo/DateTime") + " with MainCode " +
_SystemFault.XPath("FaultInfo/MainCode") + ", SubCode " +
_SystemFault.XPath("FaultInfo/SubCode") + ", and with a message
of " + _SystemFault.XPath("FaultInfo/Message") + "."
```

_SystemFault is a Composer DOM variable that holds an XML document in the event that a system fault occurs. If you had forced a fault during the execution of the service (by stopping your exteNd LDAP server, for example), you could view the structure of this document in the expression editor. You would do this by clicking the icon next to _SystemFault.

When you double-click an XML element in a DOM variable such as _SystemFault, the expression editor writes the appropriate ECMAScript expression—for example: _SystemFault.XPath("FaultInfo/DateTime").

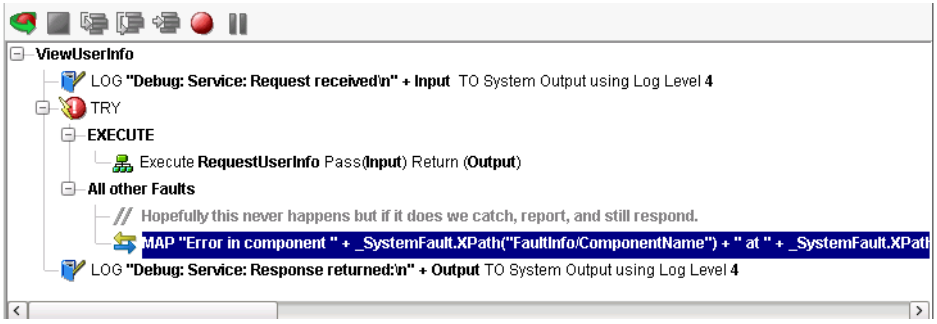
11 Click **OK**.

12 In the Target text box, type `UserInfoResponse/Message` (or use the expression editor by clicking the icon.




13 Click **OK**.

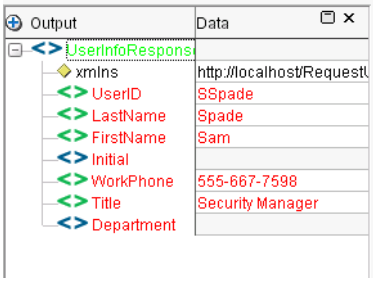
14 To place the Map action after the Comment action, you can drag and drop. Left-click the **Map Action**, hold it and drag it down to the **Comment Action**. Then release the left mouse button.



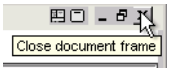

15 Click the **Execute All** icon on the toolbar.

16 Click **OK** for the Execution Completed message dialog.

17 Review the Output XML Document.



18 Click the **Close** icon in the editor to close the service.

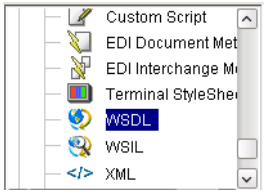



19 If prompted to save changes, click **Yes**.

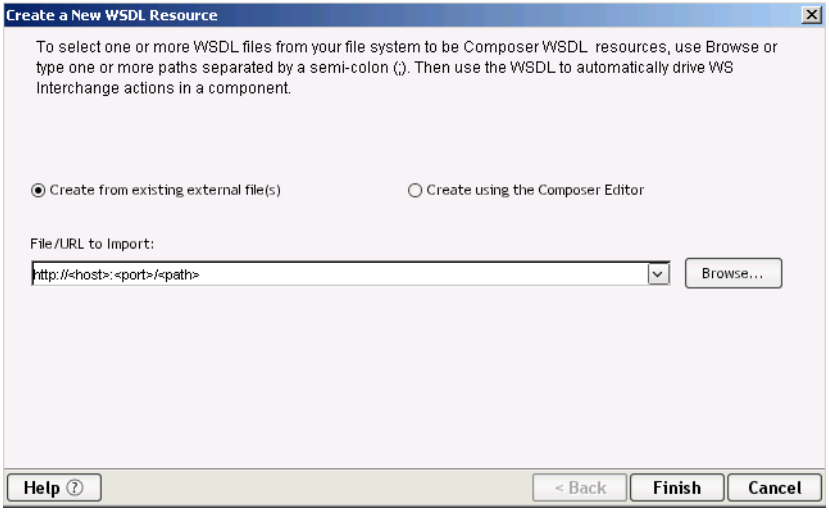
## EXERCISE 2-9: Create the Web Services Definition Language (WSDL)

Now that the service is completed and tested, your next task is to create a WSDL for the service. You need to do this because you will be deploying this service as a Web Service, and the WSDL is required for the service trigger that will be generated. The WSDL will also be used to help build the user interface for the application in Lesson 4.

1 In the Category Pane of the Navigation Pane, select **Resource** and then **WSDL**.



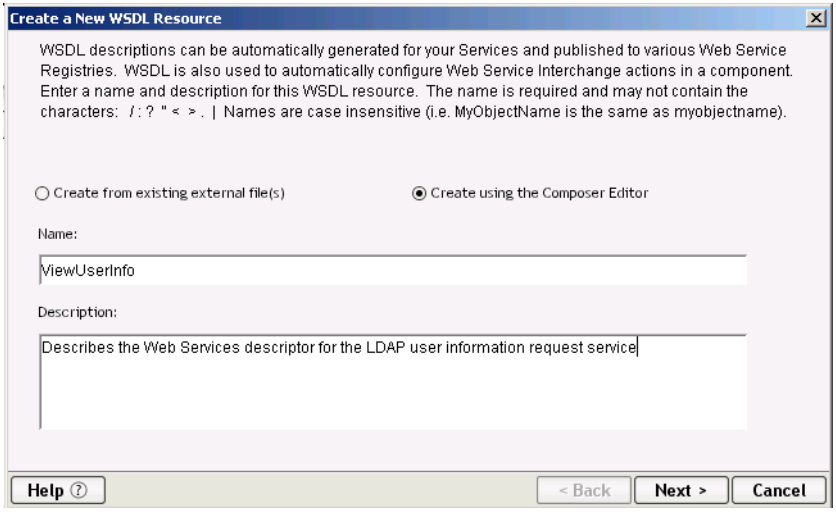

2 Right-click to display the context menu, then select **New**.



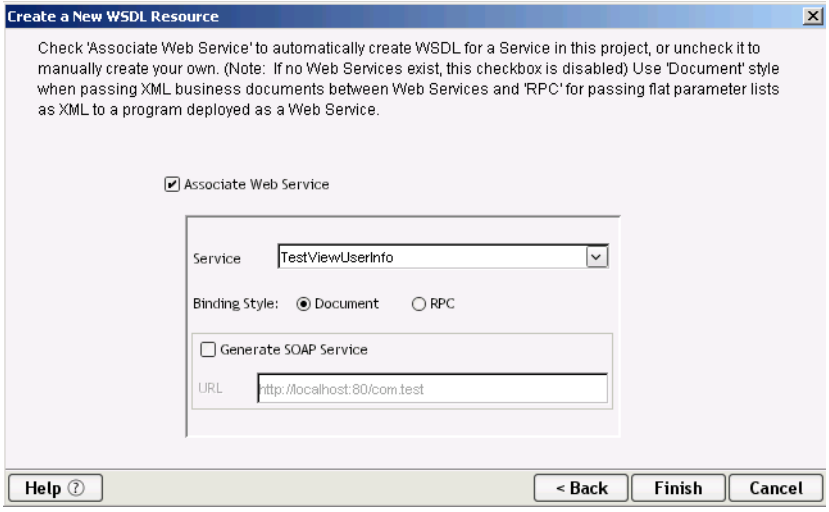

3 Select the **Create using Composer Editor** radio button.

4 Type **ViewUserInfo** in the Name text box.

5 Type **Describes the Web Services descriptor for the LDAP user information request service** in the Description text box.



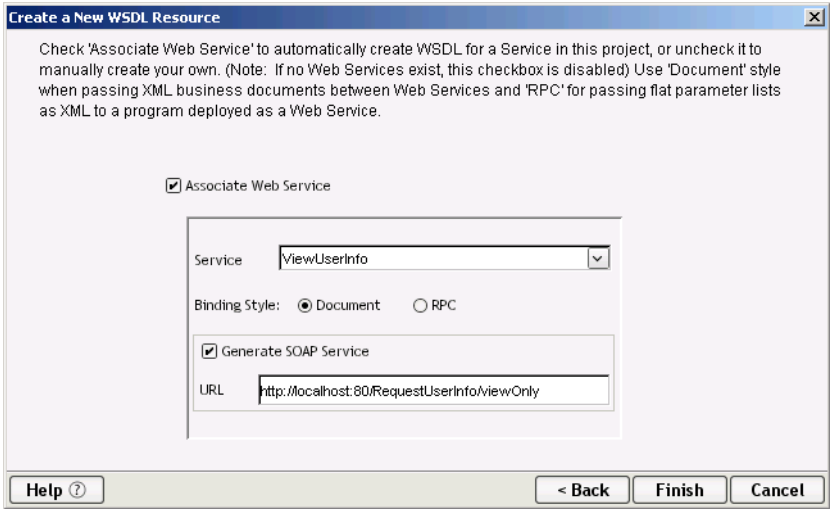

6 Click **Next**.




7 Select **ViewUserInfo** from the Service dropdown list.

8 Select the **Generate SOAP Binding** check box.

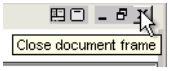
9 Type **http://localhost:80/RequestUserInfo/viewOnly** in the URL text box.




10 Click **Finish**.

11 If prompted with a message, click **OK**.

12 Click the Close document frame icon in the editor to close the WSDL.




You have now finished creating the exteNd Composer Web Service.

**Next lesson** In the next lesson you will create a service trigger for the service, then deploy and test the service.

# 3 Deploying and Testing the Web Service

## About this lesson

### Objective

The exteNd Composer project you just worked in Lesson 2 is part of an exteNd Director project. exteNd Composer is being used as the data layer of the application, and exteNd Director will be used for the presentation layer. Through cleanly separating the two layers, changes may be made to one without necessarily requiring changes in the other.

In this lesson you are going to create a service trigger for the exteNd Composer service you completed, deploy the application, then use exteNd Composer to test the SOAP interface to the service.

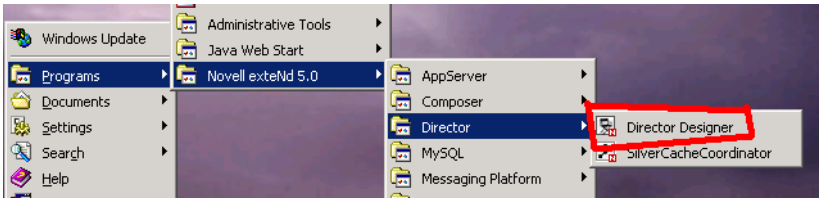
### What you will do


1 Start exteNd Director and open the project
2 Create a SOAP-based service trigger
3 Sidebar: the exteNd Director development environment
4 Deploy the application
5 Test the Web Service using exteNd Composer

## EXERCISE 3-1: Start exteNd Director and open the project

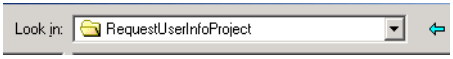
In the interest of time, you will use a template project that has some of the artifacts you need already completed, including the addition of the exteNd Composer project. In Lesson 4, “Completing the exteNd Director Project”, on page 59, you will be reviewing some of these artifacts. When you have completed the required artifacts, you will test the completed application.

1 Start exteNd Director by going to the **Start>Programs>Novell exteNd 5.0>Director** menu and selecting **Director Designer**:

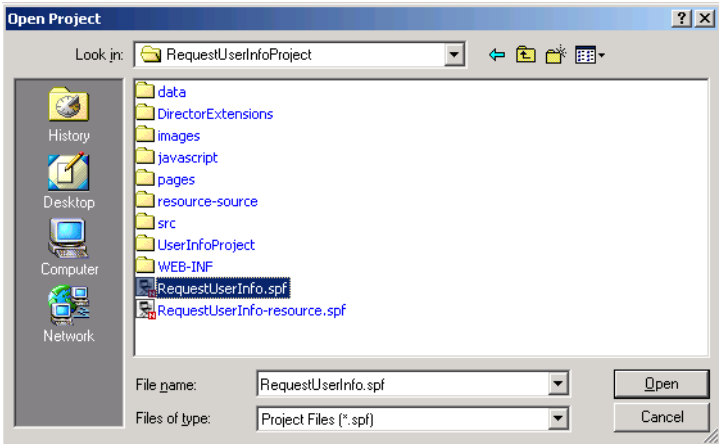



2 Open the project by selecting **File>Open Project**.

3 Navigate to the project folder by clicking on the **Look in** dropdown list. The project folder is **D:\GuidedTour\template\RequestUserInfoProject**:




4 Select the project file **RequestUserInfo.spf**, then click **Open**:



5 Change the view

in the Navigation Pane to the **archive layout** view.

**What you see in archive layout view** This view is a structural view of the archive that will be deployed to the server. It shows you where the various artifacts in your project will be placed in the archive. The default view is **source layout**, which shows you where the artifacts are actually stored in the development file system. In J2EE, where things are placed in the archive is important.

All the artifacts of the exteNd Composer and exteNd Director projects are stored as files in folders located relative to the project file you opened.

**The resource set** One of the features of exteNd Director that helps cut down on the amount of developer time required in testing—by reducing the number of deployments required—is the resource set. (For more details on the resource set, see the chapter on using the resource set in an exteNd Director application in *Developing exteNd Director Applications*.)

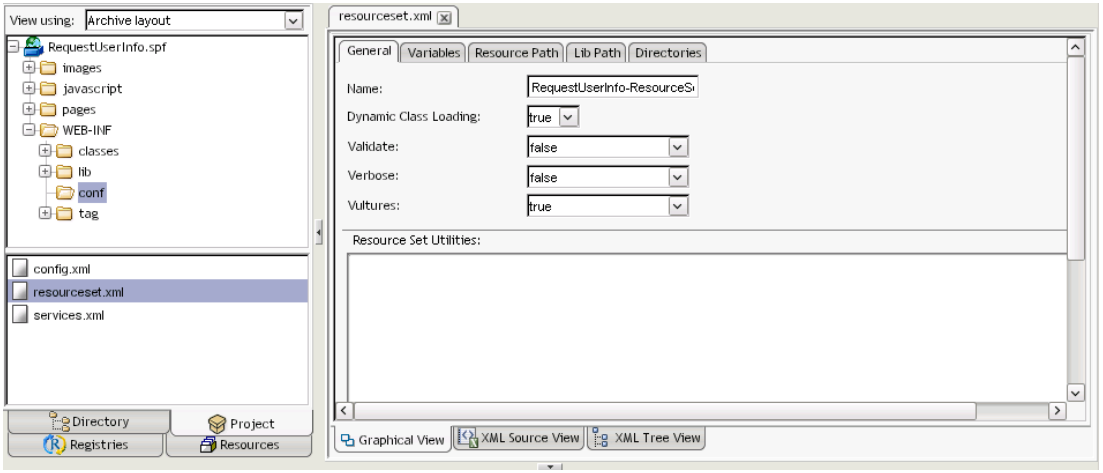
An exteNd Director project has a configuration file called **resourceset.xml**, and for the resource set to be used properly in exteNd Director and the exteNd Application Server (or any other application server), a variable in the file must point to the project's physical location on your drive.

Now you will adjust this variable, because the current location of your project is most likely different from the location used to create the project initially.

6 In the Folder Pane of the Navigation Pane, click the ⊞ icon next to the WEB-INF folder.

7 Select the **conf** folder.

8 Double-click the **resourceset.xml** file in the Detail Pane of the Navigation Pane to open the file in the Editor Pane:



9 In the Editor Pane, click the **Variables** tab.

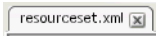
10 Locate the **WARLOCATION** key and change its Value to point to the physical location where you placed the template project—for example: D:\GuidedTour\template\RequestUserInfoProject.

11 Press **Enter** to save the change:

12 Click the toolbar icon to save your changes.

13 In the Editor pane, click the **General** tab.

14 Scroll down until you see the **Restart** button. Click the **Restart** button.

15 Close the open editor by clicking the editor title tab 

.

## EXERCISE 3-2: Create a SOAP-based service trigger

An exteNd Composer service requires a Java service trigger that will interact with the exteNd Composer runtime on behalf of the users of the service.The exteNd Composer service you completed will be invoked as a Web Service using SOAP.

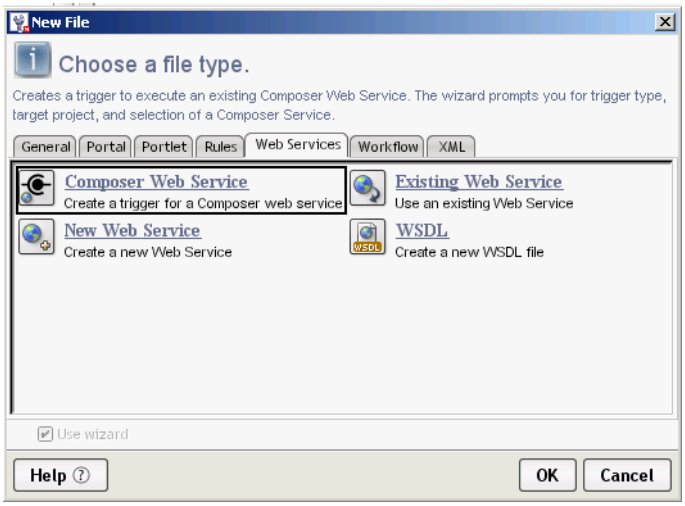
For working with Web Services, you need to have a Java servlet that is listening at the URL specified when you created the WSDL (you did this in EXERCISE 2-9: “Create the Web Services Definition Language (WSDL)”).

In this exercise, you are going to configure a generic servlet, provided by exteNd Composer, to listen at your specified URL.

1 Click the **New File** icon .

2 Select the **Web Services** tab.

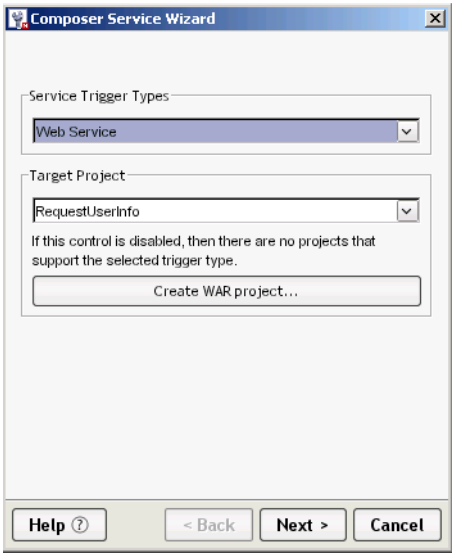
3 Select the **Composer Web Service Wizard**:




4 Click **OK**.

5 Select **Web Service** as the Service Trigger Types.

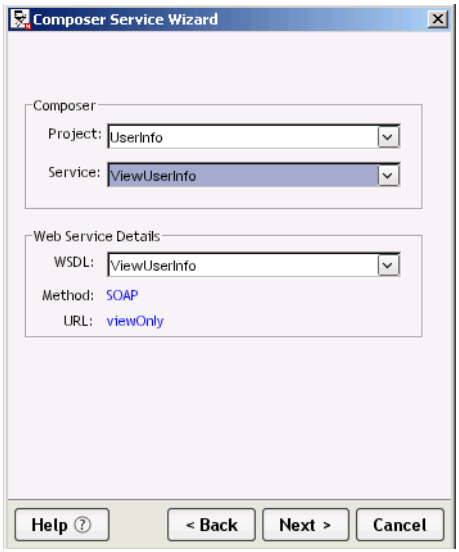
6 Verify that **RequestUserInfo** is the target project:




7 Click **Next**.

8 Verify that **UserInfo** is the project selected.

9 Select **ViewUserInfo** as the service:

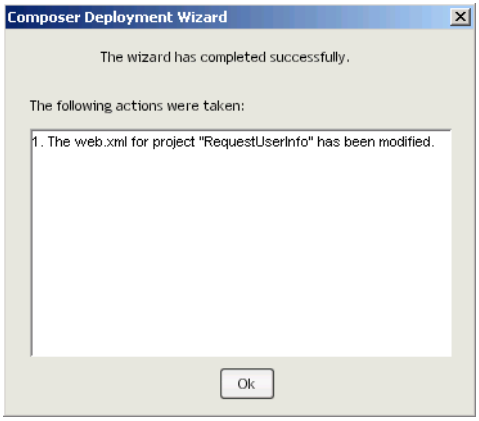



10 Verify that **ViewUserInfo** is selected as the WSDL in the Web Service Details block. Click **Next**.

11 Accept the defaults:

12 Click **Finish**.

13 Click **OK** for the Composer Deployment Wizard confirmation dialog.



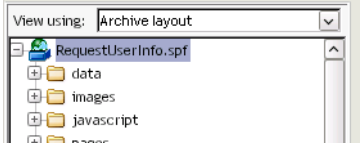

## EXERCISE 3-3: Sidebar: the exteNd Director development environment

The exteNd Director development environment, along with providing you with the ability to visually create exteNd Director artifacts such as portlets and XForms, is also used to keep track of the J2EE-specific artifacts and to build and deploy the J2EE archives that make up your project. In this exercise you will take a slight detour to look at some of the features of exteNd Director that are useful in managing the development of a Web-based solution.

In the last exercise, when you created an exteNd Composer service trigger, you may have noticed that no source file was opened. So you may wonder whether you actually have a service trigger or not.

The answer is yes—the exteNd Composer runtime provides a servlet implementation that accepts the SOAP request, invokes the exteNd Composer service, and formats a SOAP response. This servlet is configured to invoke the correct exteNd Composer service by setting initialization parameters in the **web.xml** (J2EE Web application descriptor) file of the Web archive. Now you will view these entries.

1 Scroll up the Folder Pane of the Navigation Pane until you see the RequestUserInfo.spf entry for the project, then select it.




2 Right-click to open the Context menu, then select **Open Deployment Descriptor**.

3 If you get the Select Build Option dialog and do not want to see this dialog again, select No, don't build now, Never automatically build my project—then click **OK**:

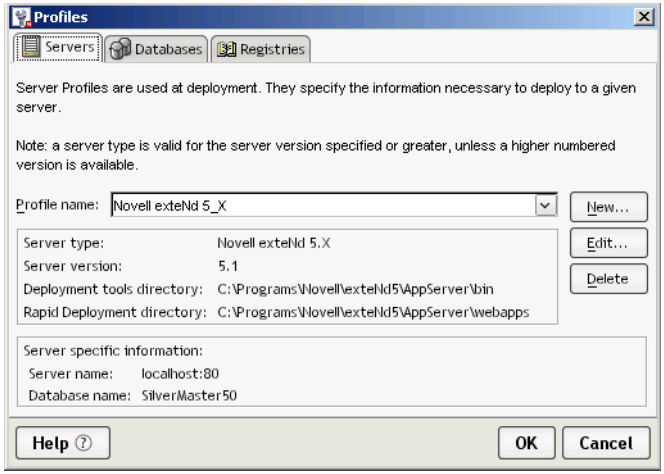
- It is usually better to explicitly build your project rather than having it done automatically, especially when all you want to do at this moment is simply view the descriptor file. When you deploy your project in a later exercise, the exteNd Director development environment will check whether a build is required at that point.
- You could also have opened this file by navigating to the WEB-INF folder and selecting it in the details. The descriptor file is always called **web.xml** and will always be located in the **WEB-INF** folder of the archive.

4 In the Editor Pane, scroll down until you see an entry called **com.novell.composer.userinfo.ViewUserInfoSoapService**. This is the servlet configuration.

- The web.xml file is XML, and the exteNd Director development environment gives you access to the actual XML. But in most situations you will use the Visual Editor to edit the contents of this file, providing a more intuitive way to maintain the file. You may view or change elements in the file by using the right-mouse button to bring up a Property Inspector. The Property Inspector will show you the valid values for the element you are working on or allow you to select its values from other entries in the file when you are entering a cross-reference.
- If you scroll further down you also find a Servlet Mapping entry for this servlet that configures the proper URL pattern that this instance will use.

5 Close the open editor by clicking on the editor title tab.

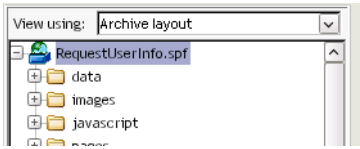
For each type of application server, the exteNd Director development environment uses a separate server profile, which defines the details required for deploying to that type of application server. In this *Guided Tour* you are using the default profile, which will deploy to an exteNd Application Server running locally on your computer. If you want to review this profile or create a new one, use the profile editor available from **Tools>Profiles**:




## EXERCISE 3-4: Deploy the application

Now you are ready to deploy the application to the exteNd Application Server so that you can test the exteNd Composer service as a Web Service.

1 Scroll up the Folder Pane of the Navigation Pane until you see the **RequestUserInfo.spf** entry for the project, then select it:




2 Right-click to open the context menu, then select **Open Deployment Plan**.

The deployment plan defines the server-specific details necessary for successfully deploying the application to the exteNd Application Server. It is created and maintained based on the information in the web.xml descriptor file. You should always review and save the deployment plan before doing a deployment.

**TIP:** If you are deploying to a different server, you will need to change the server profile used by this plan.

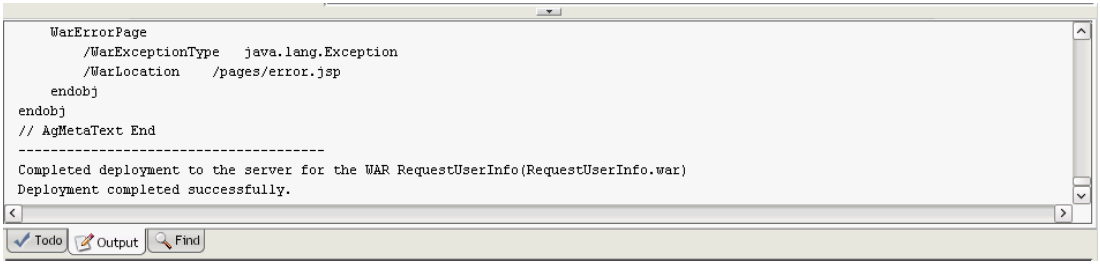
3 Click the toolbar icon to save your changes.

4 Close the open editor by clicking the on the editor title tab.

5 Click the **Deploy** toolbar icon .

When you see **Deployment completed successfully** in the Output Pane of the exteNd Director development environment, your deployment has completed.

This first deployment took a little bit longer than is usual, because not only was your application deployed to the server and then started by the server—but the database was also initialized. This was the database you created and defined a connection pool for in EXERCISE 1-4: “Create a JDBC connection pool”.




## EXERCISE 3-5: Test the Web Service using exteNd Composer

In the exteNd Composer project there is a service (TestViewUserInfo) that was created to test the deployed ViewUserInfo as a SOAP service. This service has one action (the WS Interchange) that is used by exteNd Composer to consume a Web Service that uses SOAP binding. In this exercise you will use this service to test the application you just deployed.

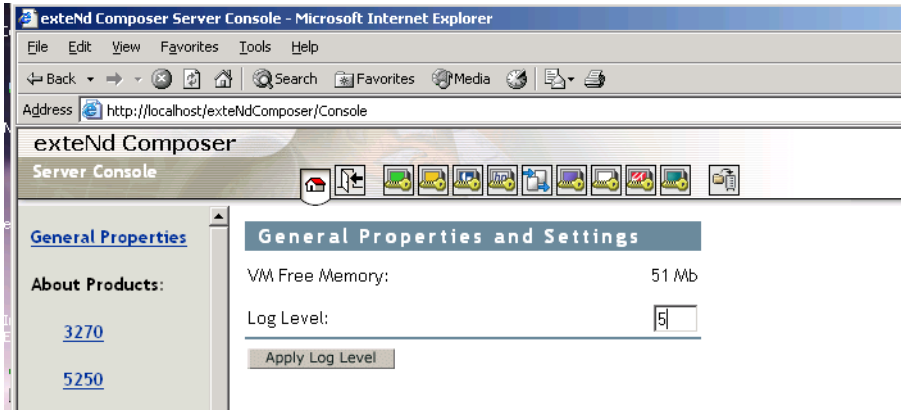
### Set the log threshold

As part of your testing you want to see the output from the Log actions in the Composer Component and Service to appear on the server console. For this to happen you need to set the log threshold on the server using the Composer Administrator Console.

1 Open your favorite browser.

2 In the address bar, type **http://localhost/exteNdComposer/** and press **Enter**.

3 Type the exteNd Application Server administrator user ID and password you supplied when you installed the exteNd 5 Suite, then click OK:




4 Change the Log Level to **4**, then click **Apply Log Level**.

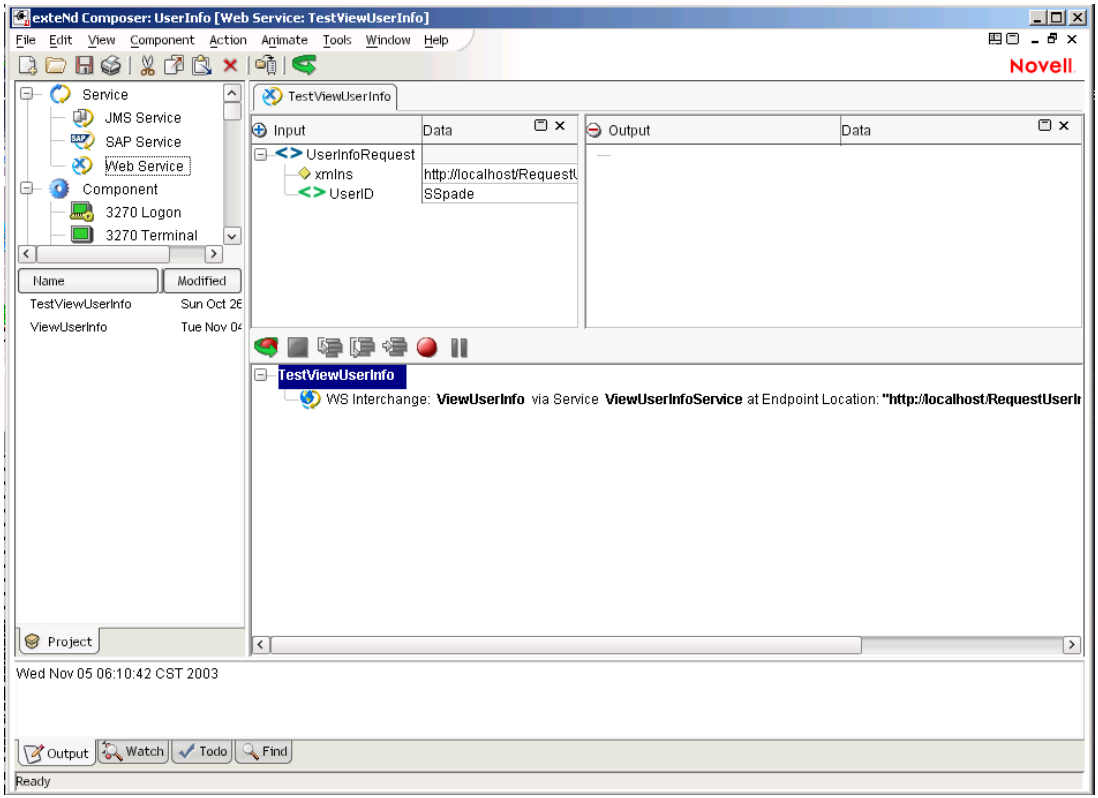
5 Close the browser application by using **File>Close**.

## Test the Web Service

1 Open the exteNd Composer Designer. (It should still be open, and using Alt+Tab should get you to it.)

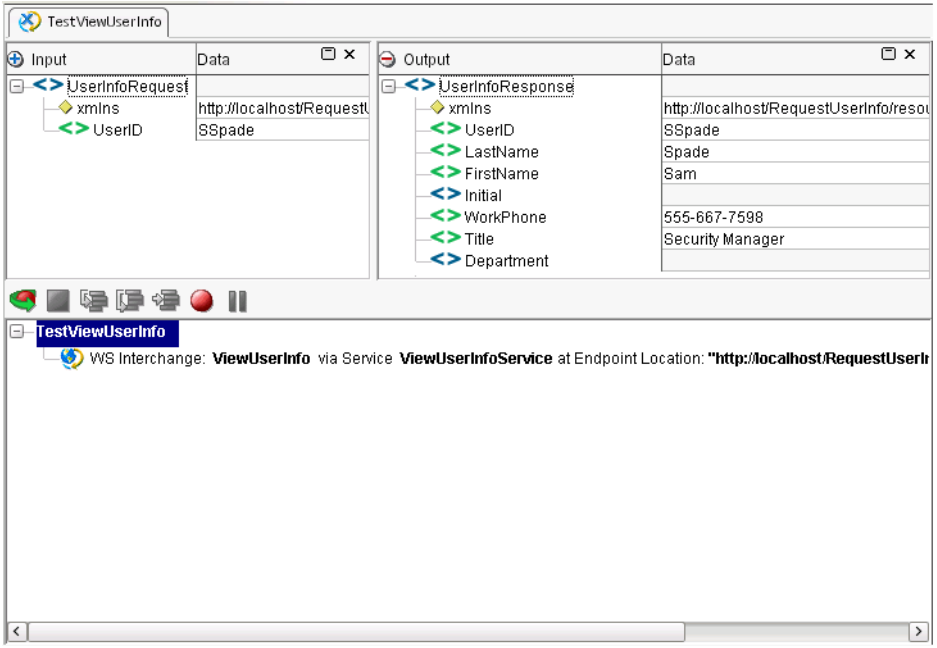
2 In the Category Pane of the Navigation Pane, select **Web Service** under the Service category.

3 Double-click **TestViewUserInfo** to open the service in the Editor Pane.

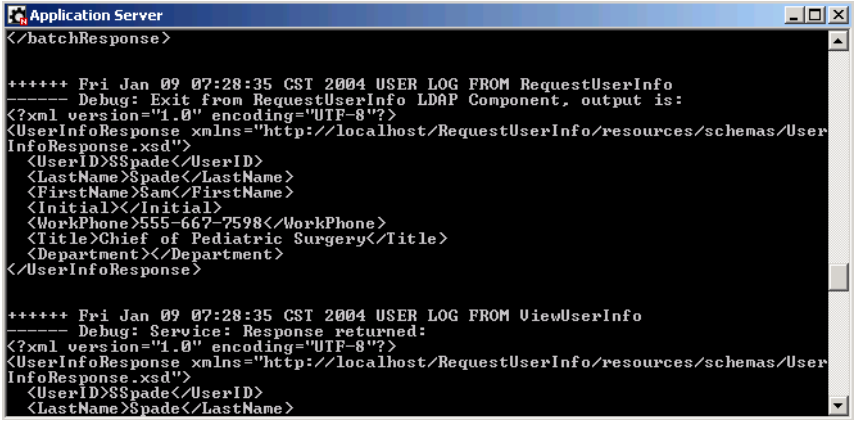



4 Click the Execute All icon on the toolbar.

5 Click **OK** on the Execution Completed dialog. You should see results returned by the SOAP service in the Output DOM of the editor.




When you look at the server console, you will see the log results from the exteNd Composer runtime:




Congratulations. You have successfully completed an exteNd Composer service and deployed it with a SOAP service trigger.

6 Close the exteNd Composer Designer application by using **File>Exit**. You will not need exteNd Composer for the remainder of this *Guided Tour*.

7 When prompted to confirm closing exteNd Composer, click **Yes**.

**Next lesson** In the next lesson you will complete an exteNd Director portal application that allows end users to consume the service.

# 4 Completing the exteNd Director Project

## About this lesson

### What is exteNd Director?

exteNd Director is an interaction portal server that enables IT organizations to rapidly deliver and easily maintain rich, personalized Web applications. exteNd Director provides a comprehensive development environment to help you create state-of-the-art interactive applications that users access through a portal:

- Powerful J2EE capabilities for the rapid implementation of sophisticated, services-oriented business Web sites
- Dynamic content, data, and Web Services
- Build interactive form-based applications that consume Web Services
- Model business processes using workflow, rules, and content management
- User personalization and profiling
- Deliver to any front end (browser, PDA, phone, kiosk)
- Multiplatform, multivendor combinations of application servers, databases, and operating systems

### Objective

In this lesson you will build a simple application that will present an initial form to the user requesting a user ID and then will invoke the Web Service (Composer ViewUserInfo) you deployed. The application will then display the results returned using one of two forms depending on whether user data or a message was returned. This application will run as a portlet in a personal page of a portal application. As you complete the application you will also have the opportunity to test your application.

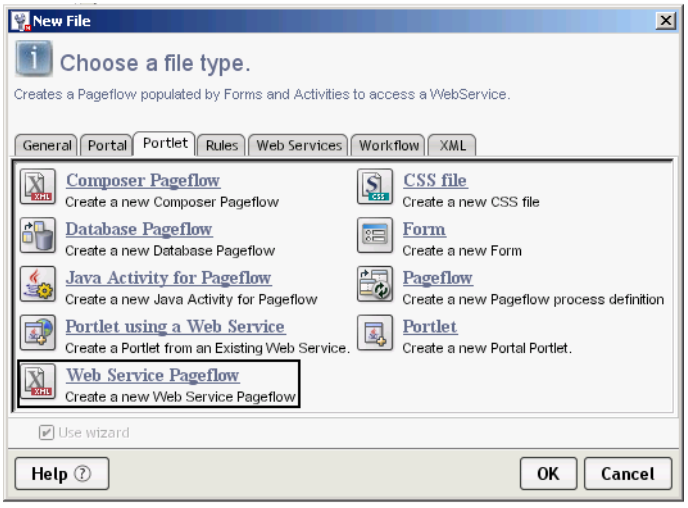
## What you will do

1 Create a Web Service pageflow using a wizard
2 Sidebar: the resource set in an exteNd Director application
3 Test the default pageflow you created
4 Edit additional XForms
5 Add a new form to the pageflow
6 Change links to include the new form in the pageflow
7 Test the revised pageflow

### EXERCISE 4-1: Create a Web Service pageflow using a wizard

In this exercise you will use an exteNd Director wizard to create a pageflow—an exteNd Director artifact that links together XForms (presentation and data binding) to the invocation of a Web Service. The wizard builds XForms based on the schema in the Web Service and ties it all together to invoke the Web Service—and includes default exception handling. All this without you writing any code.

1 Click the **New File** icon .
2 In the New file dialog, select the **Portlet** tab.
3 Select the **Web Service Pageflow** Wizard.



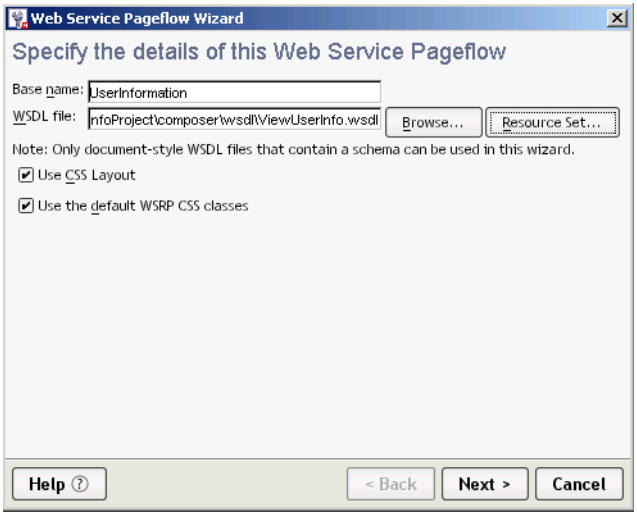

4 Click **OK**. The Web Service Pageflow Wizard displays.
5 Type **UserInformation** for the Base Name.

6 You must select the WSDL document for which you are building a user interface. There are two ways you can do this:

- The **Browse** button allows you to select a WSDL that is not currently part of your project.
- The **Resource Set** button lets you select a WSDL that is stored in your project, for example, the WSDL for the exteNd Composer service you will be using.

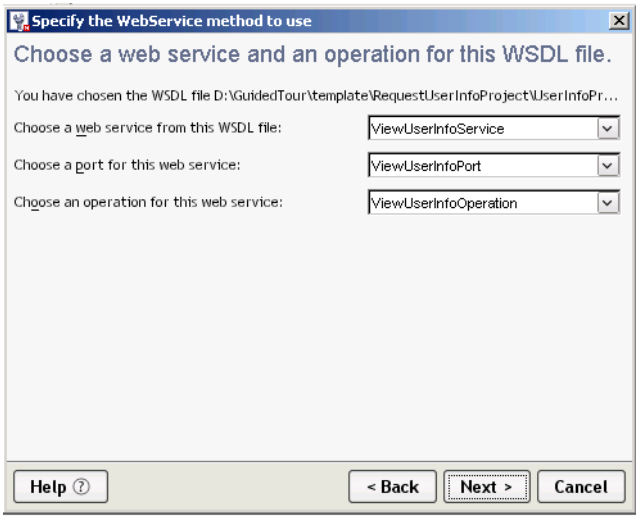
Use the **Resource Set** button to navigate to the location of the WSDL in the exteNd Composer project. In the **Choose file** dialog expand the folders until you see the **ViewUserInfo.wsdl** file. Select the **ViewUserInfo.wsdl** file. Click **OK**.

7 Accept the default checks on **Use CSS Layout** and **Use the default WSRP CSS classes**:

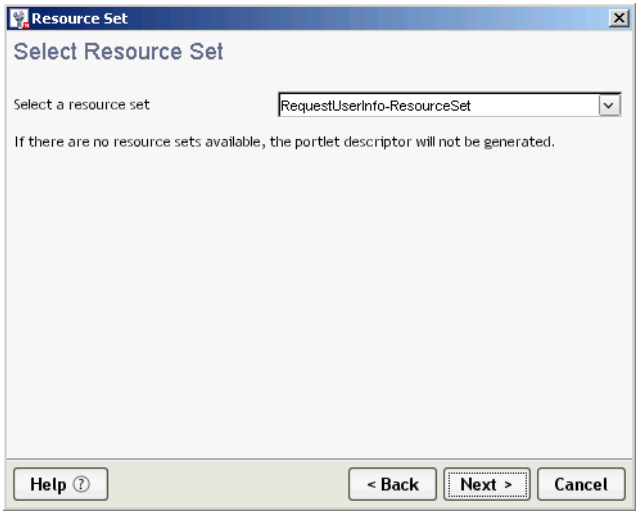



8 Click **Next**.

9 The WSDL has only one operation, so take the defaults for the Select the Web Service Method dialog by clicking **Next**:

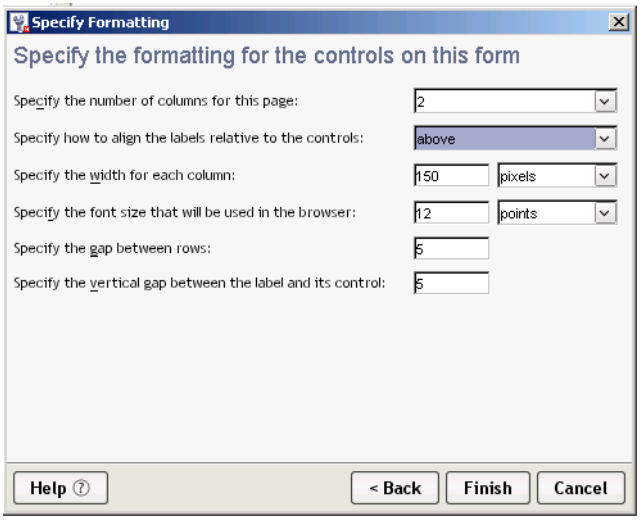



10 The exteNd Director project only has one resource set, so click **Next** in the Resource Set dialog:



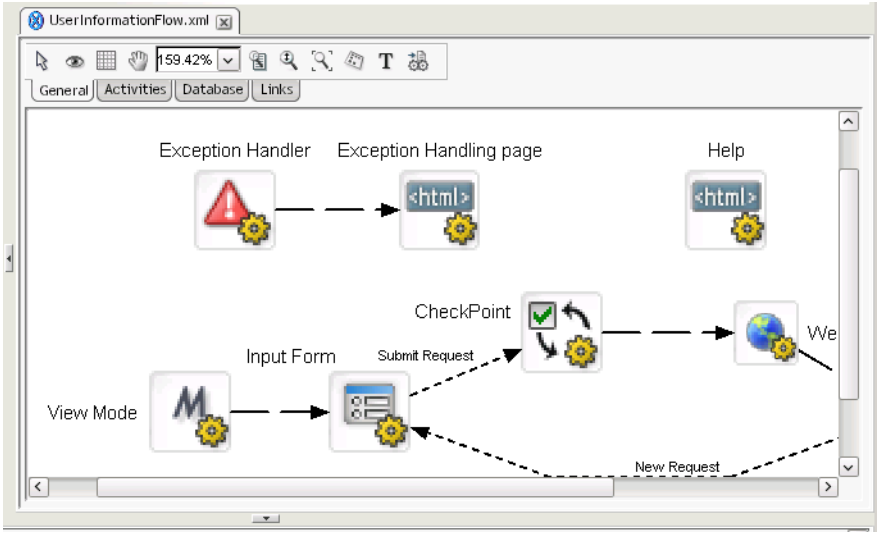

11 On the Formatting Controls tab, select **2** for number of columns.

12 Select **above** for the label alignment.



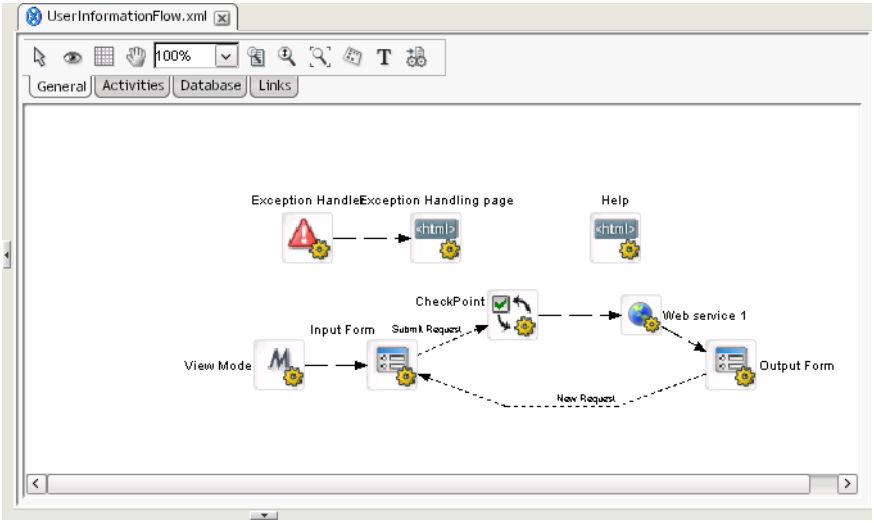

13 Click **Finish**.

14 Click **OK** on the **Done creating forms and pageflow** message:



15 Use the percentage dropdown list

to adjust the size of the pageflow diagram in the editor. 100% is a good compromise between size and readability.



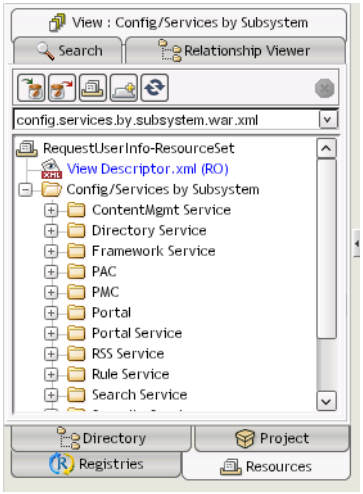

## EXERCISE 4-2: Sidebar: the resource set in an exteNd Director application

An exteNd Director resource set organizes descriptors and other files used by exteNd Director subsystems and provides for dynamic loading during development, avoiding frequent redeployments and speeding up your testing. Each custom Web application can include a resource set.

A resource set holds application-defined resources and classes. Some of these resources are templates or definitions for using a subsystem's features—such as a rule, XForms, pageflows, or a portlet descriptor. Others specify how subsystems work together—such as bindings between rules and users. Resources are usually XML files; some are accompanied by Java classes.

A resource set organizes your application's resources in a known directory structure and within exteNd Director you may use the Resources tab in the Navigation Pane to find the various artifacts in your project. For example, if you wanted to find all the files created in the previous exercise you would do the following:

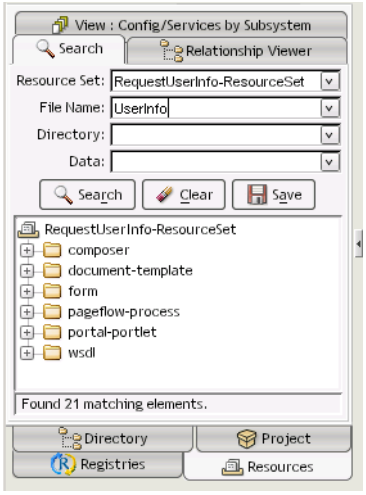
1 Select the **Resources** tab in the Navigation Pane.




2 Select the **Search** tab.

3 In the File Name text box, type **UserInfo**.

4 Click the **Search** button:




5 You can save this search as a view to reuse the search criteria at a later time. Click the **Save** button.

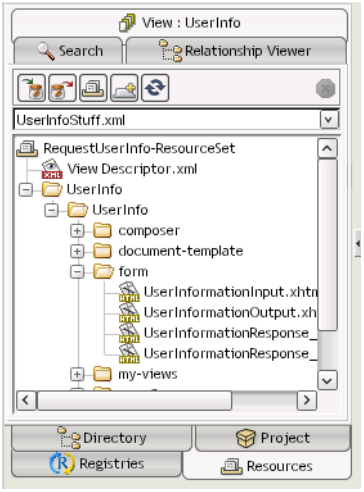
6 Name the file **UserInfoStuff.xml** and click **Save**.

The view is an XML file that describes the search criteria and display structure, and you may edit it to server your specific needs.

7 Close the open editor by clicking on the **editor title** tab to close the **UserInfoStuff.xml** file.

8 Select the **View** tab.

9 In the dropdown list find **UserInfoStuff**, then select it:




You may expand the folders to see the various artifacts you created—and by double-clicking a particular entry the appropriate editor for working with that artifact will be opened.

📖 For details, see the chapter on using the resource set in an exteNd Director application in *Developing exteNd Director Applications*.

## EXERCISE 4-3: Test the default pageflow you created

Because pageflows and XForms created by the Web Service Pageflow Wizard are stored in the resource set and you already deployed the application, you may immediately test the default portlet that was created. This is because by default during development, your exteNd Director project was configured to dynamically load the resource set artifacts.

1 Open a browser window.

2 Type the URL **http://localhost/RequestUserInfo/portal/portlet/LoginPortlet** in the address bar.

3 At the portal login page, click the **New User?** link:

4 Enter the information to create a new user—for example:

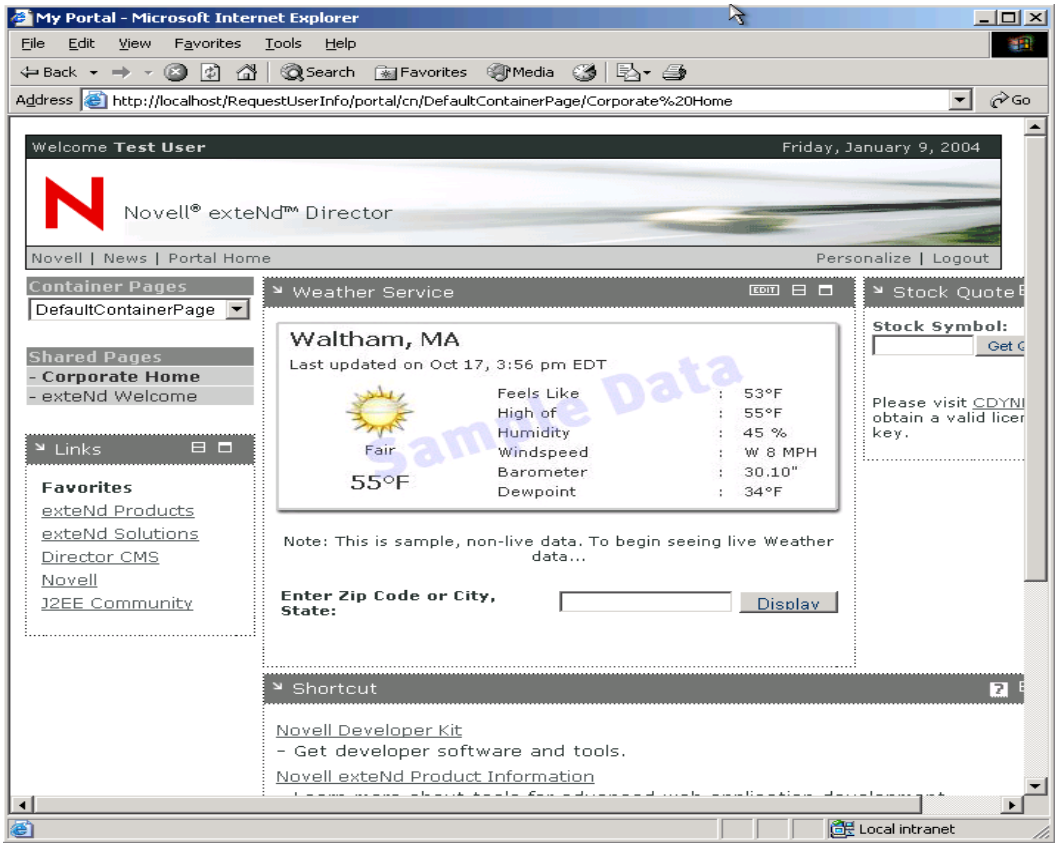
- User ID: **user1**
- Password: **password**
- Confirm Password: **password**
- First Name: **Test**
- Last Name: **User**
- Email: **user1@somewhere.com**

5 Click **Create**.

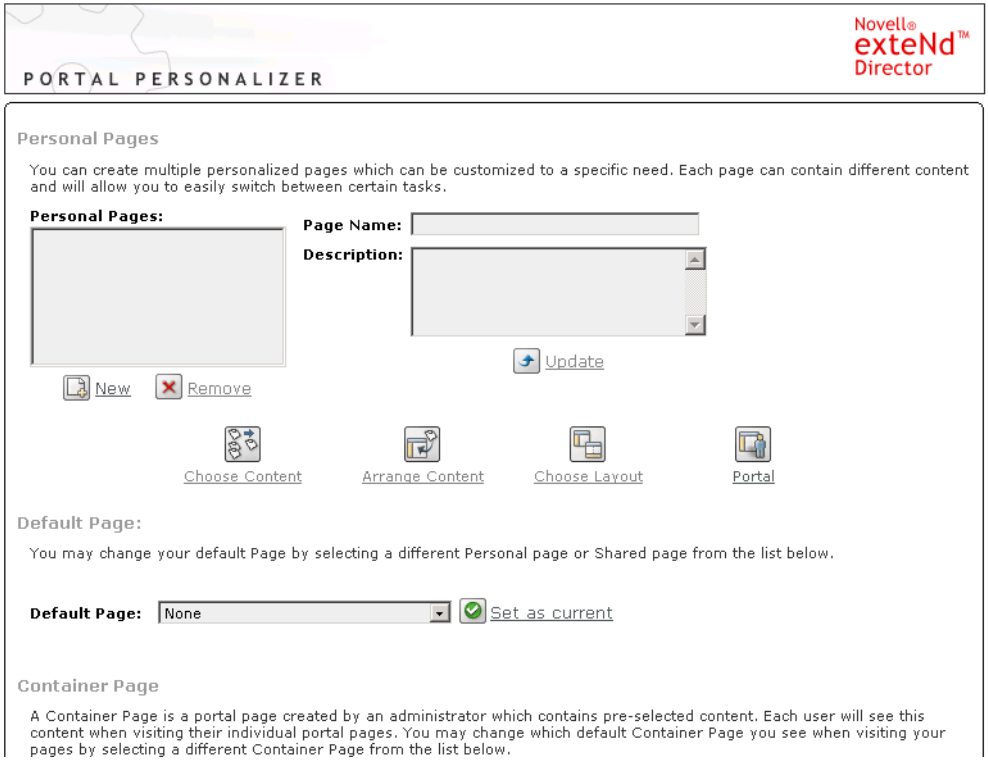
6 Click **Continue** to return to the login page.

7 Log in using the account you just created. Click **Login** or press **Enter**.

You will see the default portal shared page with some sample portlets. You will create a personal page where the portlet for the pageflow you created will be placed. You will use this page later in this lesson to test and review your changes.

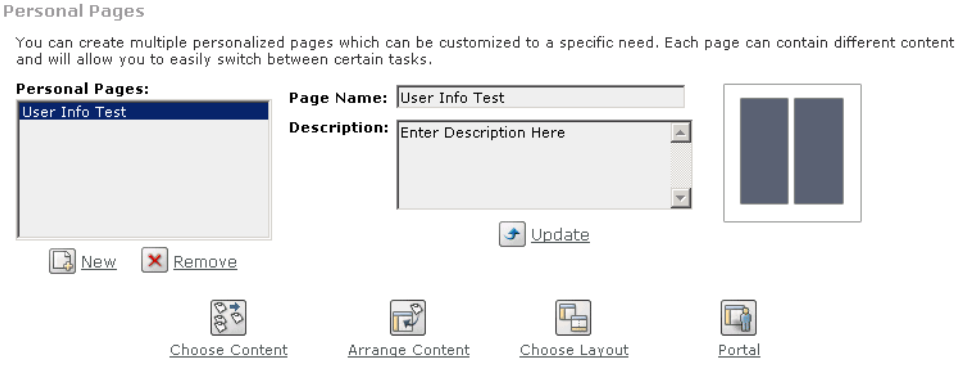


8 Click the **Personalize** link:




9 Click **New** under the Personal Pages list to create a new personal page.

10 Change the name of the page to **User Info Test**, then click **Update**:

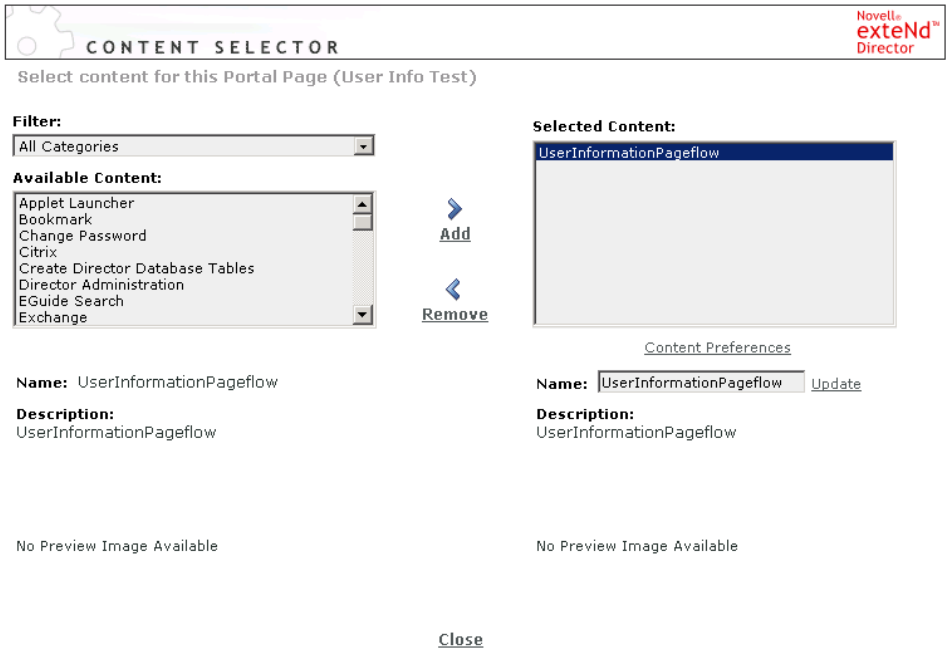



11 In the Default Page text box, select the page you just created.

12 Click the **Set as current** link. This allows you to quickly get to this page in the portal page.

13 Click the **Choose Content** link.

14 Scroll the Available Portlets list until you find **UserInformationPageflow**, then select it and click **Add**.

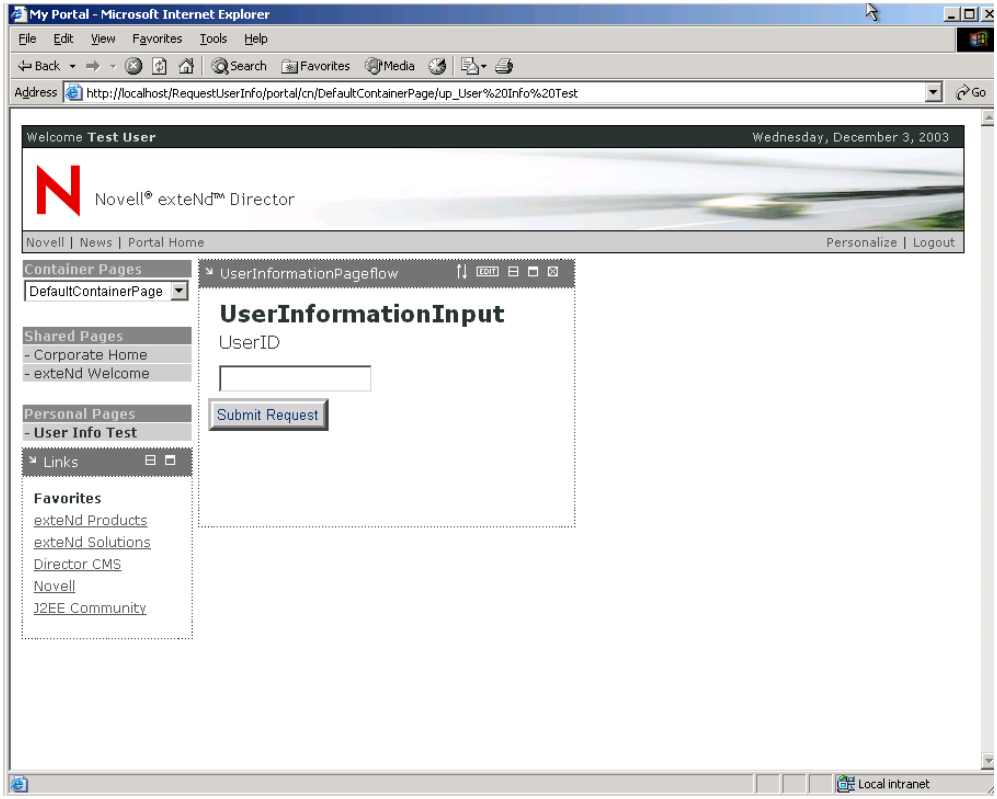



15 Click **Close**.

16 Click the **Portal** link.

17 At the top of the page, click **Portal Home**—or under Personal Pages click **User Info Test** (the page you just created)—to view your pageflow portlet.

18 If you see the message **Portlet did not respond in time**, click the link or reload the page (by clicking on **Portal Home** or under Personal Pages clicking **User Info Test**).



19 Type **SSpade** in the UserID text box of the UserInformationInput Portlet page, then click the **Submit Request** button.

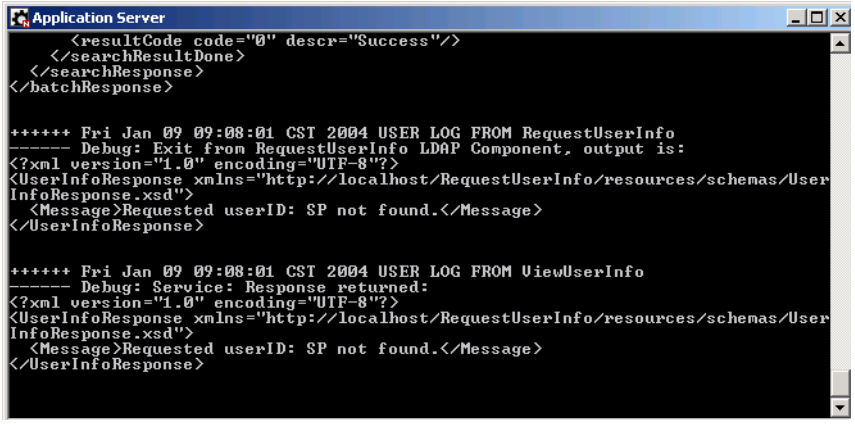
20 Click the **New Request** button to return to the first form.

21 Type **SP** in the UserID text box, then click the **Submit Request** button.

You will not see any message.

22 Click the **New Request** button to return to the first form.

If you take a peek at the exteNd Application Server console, you will see the Log output from the exteNd Composer service that shows the data returned to the application.




The forms and pageflow created by the wizard did not take this form into consideration, so you need to change the pageflow. You will add a conditional decision to the pageflow that will decide which form is displayed; the decision will be based on the data returned from the invoked Web Service. This is your task for the rest of this lesson.

## EXERCISE 4-4: Edit additional XForms

A form was already created that displays the message returned by the Web Service when an invalid request is made. You will review this form and make some slight modifications. In the next exercise you will insert this form into the pageflow you created earlier with the Web Service Pageflow Wizard.

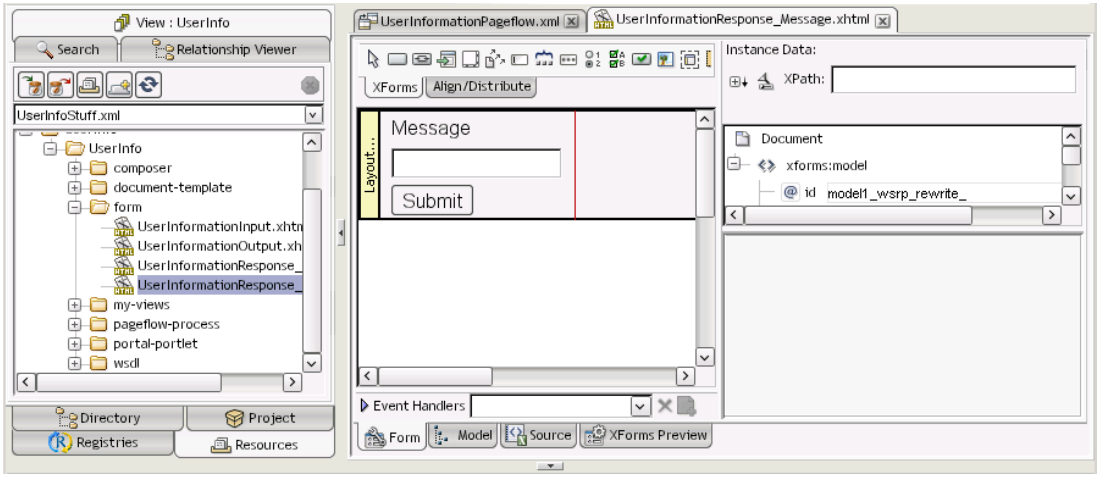
1 In exteNd Director, select the **Resources** tab

of the Navigation Pane.

2 Select the **View** tab in the Resources Navigation Pane view, then use the dropdown list to select **UserInfoStuff** (the view you created earlier).

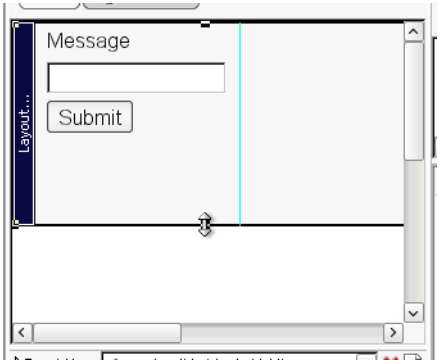
3 Expand the listed folders by clicking the ⊞ icon until you find the form folder.

4 Double-click **UserInformationResponse_Message.xhtml**, which will open the form in the XForms Editor.




You are going to change the look and feel of this form by replacing the input message text box with an output (display only) text box.

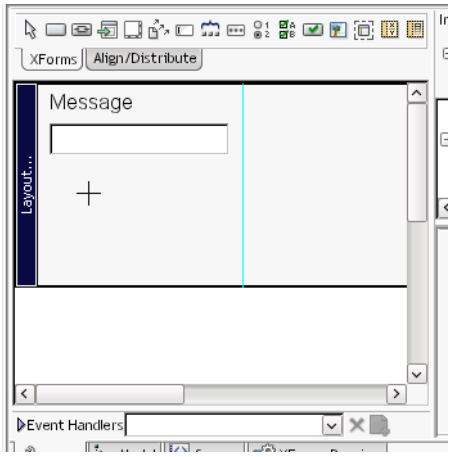
5 Select the **Layout Region** and, holding the left mouse button, increase its size by scrolling downward with the mouse.

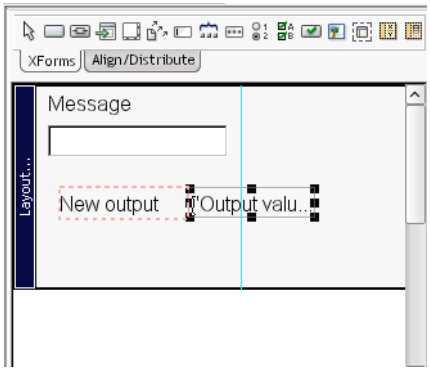



6 Click the **Submit** button on the form and press the **Delete** key to delete it.

7 Click the **Insert XForms output** button.

8 Click **below** the **Message** text box to add the **Output** control.



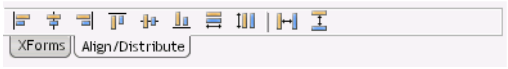



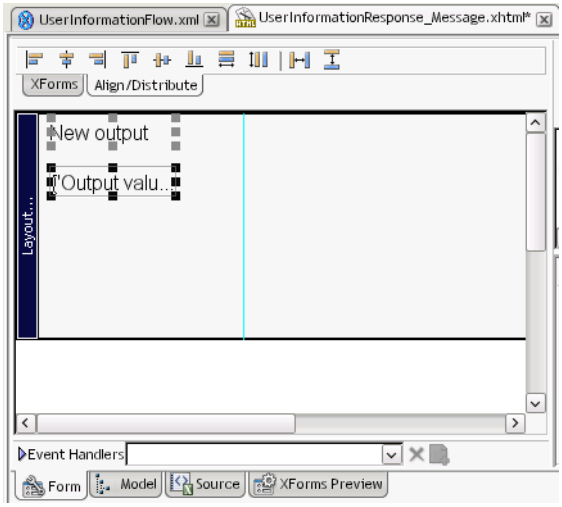

9 Select the **Input message** text box, then press the **Delete** key.

10 Select the **Output** text box and drag it below the **New output** label.

11 Select the label and text box while holding the **Ctrl** key, then drag both toward the top of the layout area.

12 Use the **Align/Distribute** tab to properly align the controls on the layout.



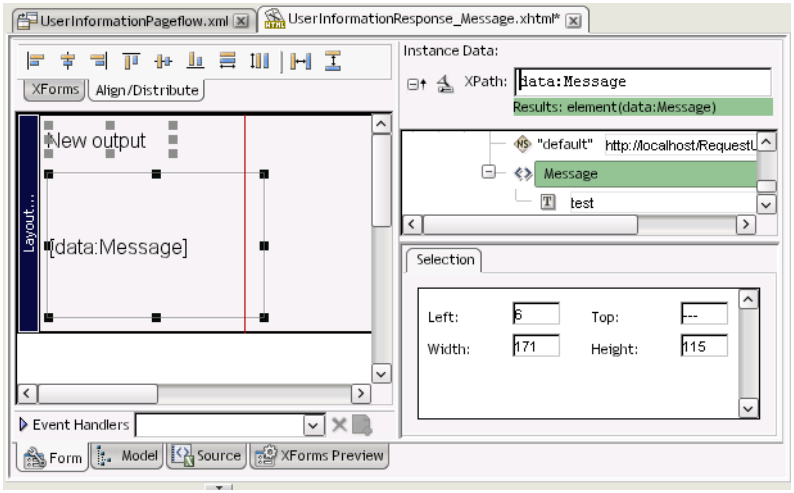




13 Click the **Save** icon to save your changes

14 Enlarge the Output text box by clicking it and then dragging the corner down and to the right.

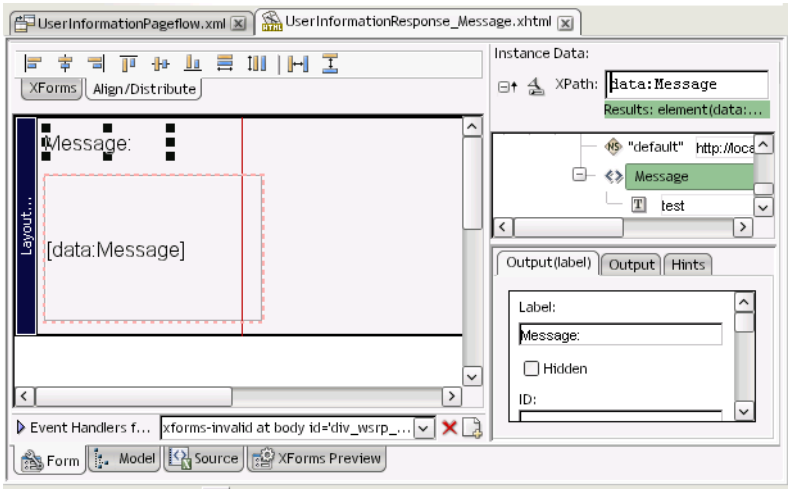
15 In the Instance Data Pane click the icon until you see the Message element (you may need to scroll).

16 Click the Message element, the hold the left mouse button and drag the element on the Output text box in the Form Pane.




17 Click the **Save** icon to save your changes.

18 Select the Output (label) tab in the Property Inspector, scroll to the top, and change the Label to read **Message:**.




19 Click the **Save** icon to save your changes.

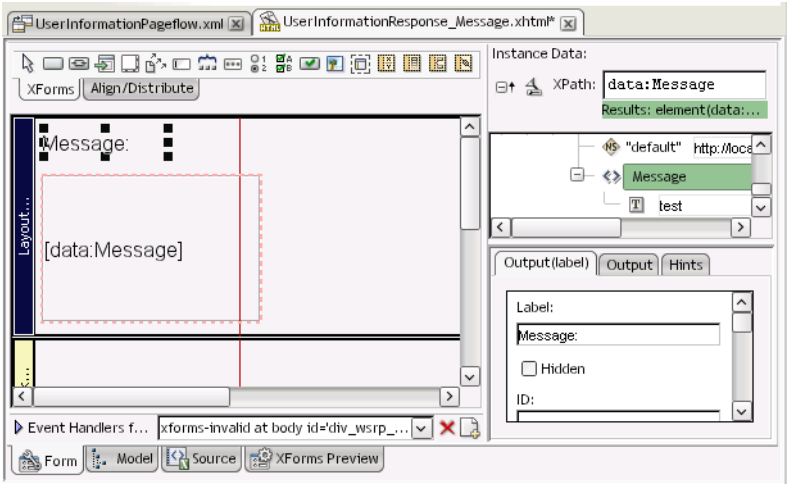
20 Click the divider between the Form Layout section and the Instance Property section of the XForms Editor to expand the Form Layout, showing you all the items in the tools tab in the Form Layout section.

**TIP:** If this does not work, hold the left mouse button and drag the divider to the right until you see all the XForms tools on the tab.

21 Click the Insert Pageflow link region button.

22 Click in the Form Layout to add the control to the form:

This region will be used by the Pageflow button link when you add this page to the existing pageflow.

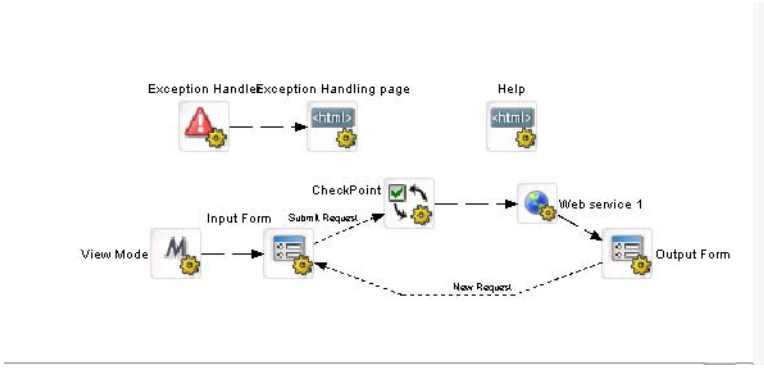



23 Click the **Save** icon to save your changes.

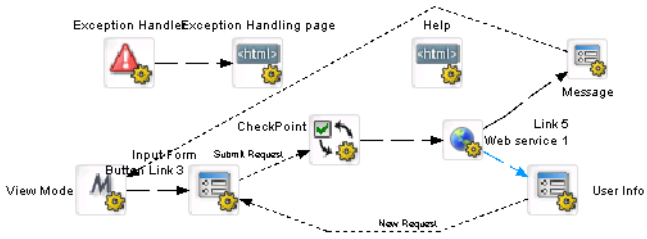
## EXERCISE 4-5: Add a new form to the pageflow

In this exercise you are going to add the Message Display page to the existing pageflow and then change the links on this page to include this page as part of the application flow. One of the links will have a condition that will check what result was received from the Web Service to determine which page should be displayed—the User Information page or the Message page. Your objective is to go from the flow pictured just below to the one pictured under it.

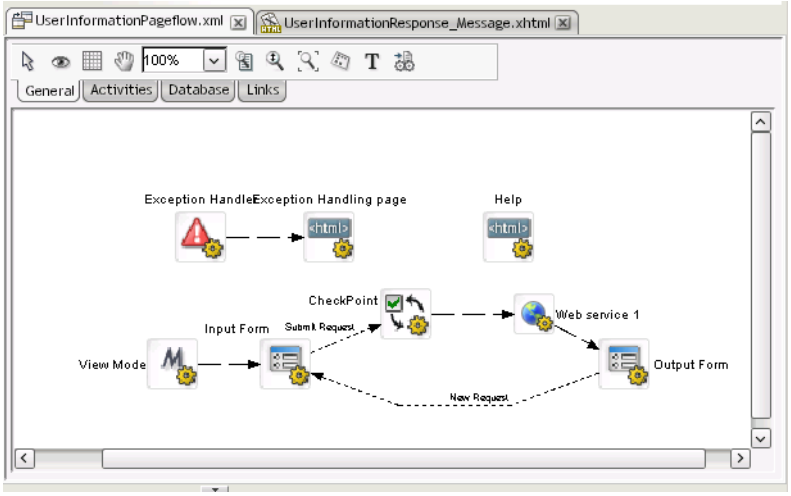
### Original pageflow




### Target pageflow



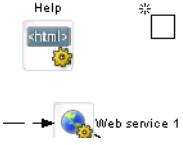
1 Select the **UserInformationPageflow.xml** file in the exteNd Director Edit Pane. UserInformationPageflow.xml opens in the Pageflow Modeler.




2 Select the **Activities** tab in the Pageflow Modeler.

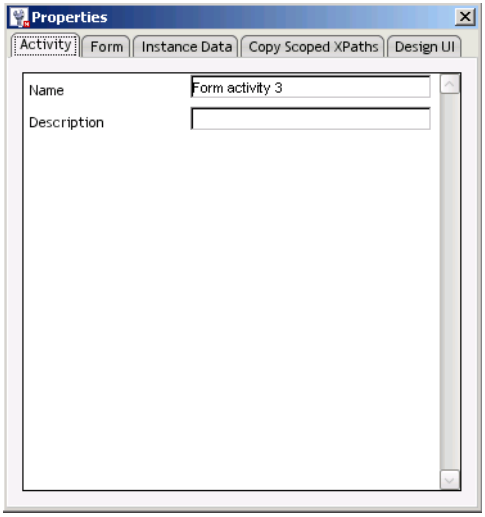
3 Click the **Form activity** toolbar button.

4 Add the form to the right of the Help HTML activity in the pageflow diagram by moving the mouse pointer to the location and clicking the left mouse button:




5 Click the label added under the Form activity you just placed on the diagram and edit it to read **Message**.

6 Double-click the **Form activity** you just placed on the diagram to open the Property Inspector for the activity:




You may drag this window to a more convenient location on you desktop so that it does not cover the diagram or other parts of the editor that you need to view.

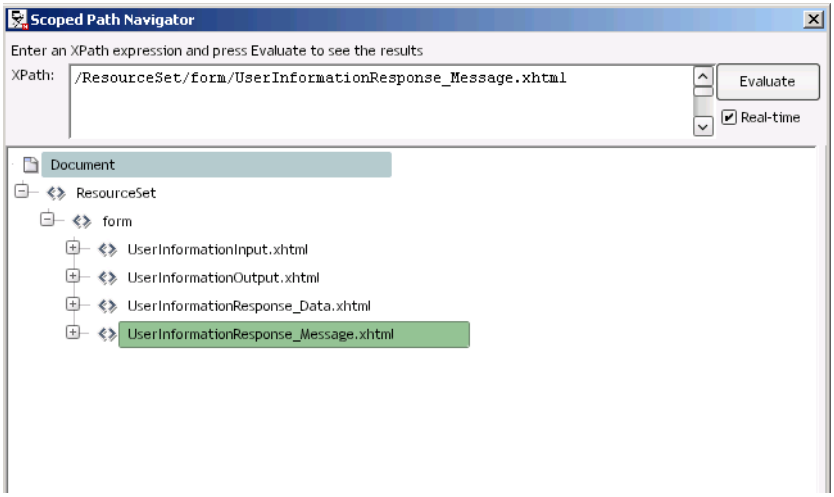
7 Type **Message** in the Name text box.

8 Click the **Form** tab.

9 Click the button next to the **XHTML** text box on the Form tab.

10 In the Choose The Scoped XPath dialog, select **ResourceSet** from the dropdown list and then click the **Browse** button.

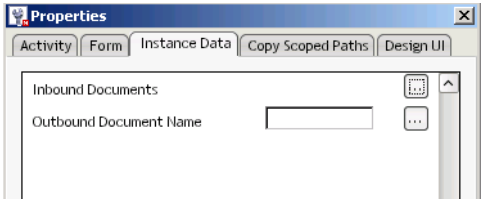
11 In the XPath Navigator, use the button to open the ResourceSet and form lists and then select the **UserInformationResponse_Message.xhtml** file:



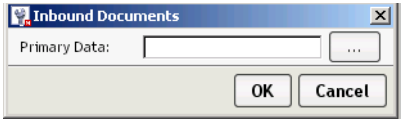

12 Click **OK**.

13 Click the **Save** icon to save your changes.

14 Click the **Instance Data** tab.




15 Click the button next to **Inbound Documents**.



16 Click the ... button next to the **Primary Data** text box.

17 Click the Scoped XPaths dropdown list to select **/Flow/document/RESPONSE**.

18 Click **OK**.

19 Click the **Save** icon to save your changes.

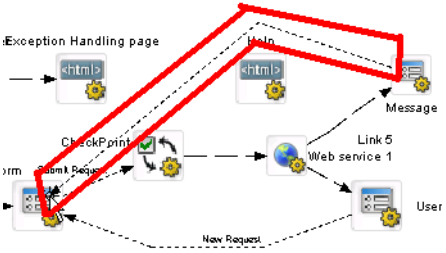
## EXERCISE 4-6: Change links to include the new form in the pageflow

Now you will create links for the new form to include it as part of the normal flow of the application. The Web Service will go to both forms with a condition added to one of the links to determine which one. The new Message form will link to the Input form completing the loop in the application.

1 Select the **Links** toolbar in the Pageflow Modeler:

2 Click the **Button Link** button .

3 Click the **Message Form** activity, drag to the **Input Form** activity, then left-click to create a link between the two activities:



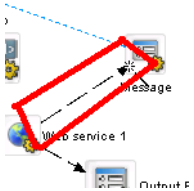

4 Click the link you just added to select it.

5 In the Property Inspector in the Button Link tab, change the Description to **New Request**.

This is the value that will appear on the button when the page is rendered in the browser.

6 Click the **Link** button .

7 Click the Web Service activity, drag to the Message Form activity, then left-click to create a link between the two activities:



8 Change the label next to the Output Form to read **User Info**.

9 Click the **Save** icon to save your changes.

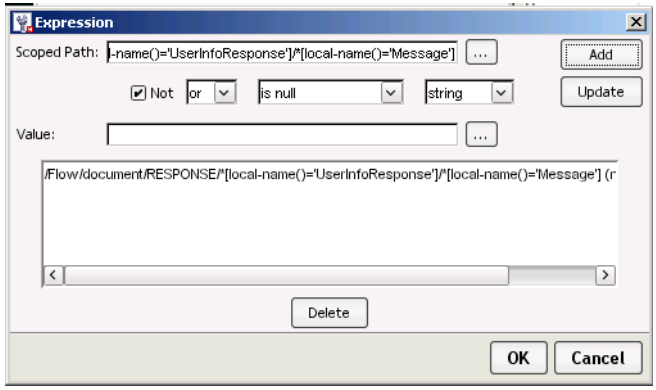
10 Click on the link between the Web Service activity and the Message Form activity.

11 In the Property Inspector, click on the **Edit Expression** link Edit Expression... in the Link tab.

12 In the Scoped Path text box of the Expression dialog, type the following expression (be careful with the punctuation and case):

```
/Flow/document/RESPONSE/*[local-
name()='UserInfoResponse']/*[local-name()='Message']
```

13 In the operator dropdown list, select **is null**, check the **Not** check box, then click the **Add** button:




14 Click **OK**.

15 In the Link tab of the Property Inspector, make sure that the Precedence text box has a value of **1**.

16 Click on the link between the Web Service activity and the User Info Form activity.

17 In the Link tab of the Property Inspector, make sure that the Precedence text box has a value of **2**.

18 Click the **Save** icon to save your changes.

## EXERCISE 4-7: Test the revised pageflow

Now that you have completed the changes to the pageflow to incorporate the new page, the last step is to test your changes in the application server. Because of the dynamic loading feature of exteNd Director, your changed forms and pageflow have been automatically loaded into the application server—and all that is needed to test the changes is to open a browser window and log in to the Portal application.

1 Open a browser window.

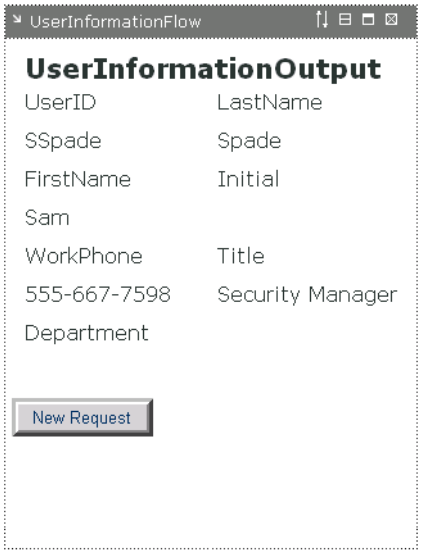
2 Type **http://localhost/RequestUserInfo/portal/portlet/LoginPortlet** in the address bar, then press **Enter**.

3 Log in as:

- User: **user1**
- Password: **password**

4 Under the Personal Pages list, click the **User Info Test** link.

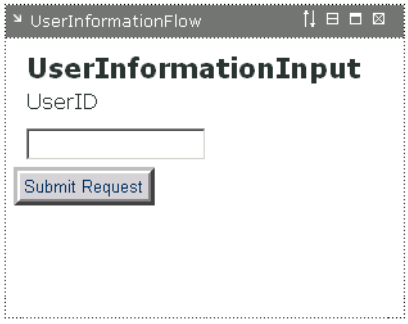
5 In the UserID text box of the UserInformationInput page, type **SSpade** and then click **Submit Request** button.




6 Click the **New Request** button.

7 Type **sp** in UserID text box, then click the **Submit Request** button.

8 Click the **New Request** button.




9 Close exteNd Director by using **File>Exit**.

Congratulations. You have successfully completed the *Guided Tour*.

# Summary

In this *Guided Tour* through the development features of the exteNd 5 Suite, you have:

- Used exteNd Composer to create a service that extracts information from an LDAP directory.

  This showed you how you can use exteNd Composer to XML-enable your legacy systems and use the information in them to provide new services.
- Deployed the service as a SOAP service.

  This made it possible to take a standards-based approach to accessing this information.
- Used exteNd Director to build a browser-based interface that can use this service by accepting input from the user and then displaying information to the user.

In short, you have gone from producing a Web Service to consuming a Web Service, all with one integrated set of tools. The entire application was provided by a robust J2EE application server. *And all this without writing one line of Java code*.

If you were to continue development of your Web Service application, the next step would be to go from a query-only application to a maintenance-and-query application. But that is a challenge for another time.

**Learning more** To learn more about Novell exteNd, see the Learning Resources page in *Welcome to Novell exteNd* in this help system.