

Developing Web Services

Exercises



Contents

1. Module	3
1.1 Objectives.....	3
1.2 Overview.....	3
2. About Developing Web Services.....	4
2.1 Introduction.....	4
2.2 References	5
3. Exercises	6
3.1 Prerequisites.....	6
3.2 Configuring Design and Deployment Structure.....	6
3.2.1 Creating Design time Folder Structure	6
3.2.2 Creating Deployment Structure	7
3.3 Generating Web Services Operation on Process Models.	8
3.3.1 Prerequisites.....	8
3.3.2 Creating the Web Service.....	8
3.4 Exploring Web Service Components	9
3.4.1 Exploring the Web Service Definition	9
3.4.2 Exploring the Web Service Interface	10
3.4.3 Exploring the Web Service Operations.....	11
3.4.4 Testing a Web Service Operation.....	12
3.5 Working with External Web Services	15
3.5.1 Prerequisites.....	15

3.5.2 Configuring Connectivity	15
3.5.3 Generating the Web Service.....	18
3.5.4 Testing a Web Service Operation	21
3.6 Generating Web Service Operations on Databases	22
3.6.1 Configuring Connectivity.....	22
3.6.2 Creating Database Metadata.....	25
3.6.3 Creating Design Folder Structure	26
3.6.4 Creating Deployment Structure	27
3.6.5 Generate the WS-AppServer Package.....	27
3.6.6 Publishing and Testing WS-AppServer Package.....	33
3.7 Adding Web Service Operations Manually	33
3.7.1 Create the web service operation	33
3.8 Web Services Security.....	37
3.8.1 Prerequisites.....	37
3.8.2 Define Runtime Security	37
3.9 Make Changes Available to SCM	39
4. Optional Exercises	40
4.1 Exploring Web Service Operation Usage - Optional.....	40
4.2 Custom WS Operations - GetAllEmployees	41
5. Learning Report	42

1. Module

1.1 Objectives

After completing this course module, you will be able to:

- Work with External Web Services
- Make Business Process Models Web Services Enabled
- Generate basic services on database content
- Generate basic services on database content with java layer logic

1.2 Overview

This module introduces you to the concept of web services and how to make back ends web service enabled by developing your own web services. After setting up connectivity to relevant backend(s), you need web services to perform the operations that you want to be executed e.g. retrieve and update data, send mail, download files and start processes.

2. About Developing Web Services

2.1 Introduction

In this module you will learn how to build web services based on:

- Working with existing, external (outside your Cordys organization) web services.
- Generate web service operations on a business process that you have developed and implemented.
- Generate web service operations on existing applications to expose functionality from these applications.

This requires connectivity to those applications and/or back ends. In addition, you can use web service interfaces, included with the application connector to generate web services on these applications.

Web Services

A web service offers certain services over the web. A web service has a WSDL (Web Service Description Language) that describes the web service; the WSDL contains information like address (URI), operations, messages and schemas. Through this description, information about these web services can be exchanged and how to invoke these services.

When creating services in Cordys, we will follow the naming conventions as dictated by the WSDL definition of W3(C).

The *Web Service Definition* is the WSDL of a Web Service. Within a WSDL you have at least one, but possibly several *Web Service Interfaces*. The *Web Service Interface* groups together *Web Service Operations*. In this way, you can group similar operations together in an interface targeted for specific audiences e.g. finance or sales department. These web service interfaces are then defined in the same WSDL to distinguish them easily.

A *Web Service Operation* is the actual action or function that you can call by executing a SOAP message providing the appropriate request. The web service operation serves as an abstraction layer to the backend/application that has become Web Services enabled now.

The *Web Service Operation* contains two attributes:

1. *Implementation*

The implementation contains the execution statement for the backend and is structured in the way the corresponding application connector dictates.

2. *Interface*

The Interface contains the WSDL part for this web service operation, which exists of the web service address, the operation name and the corresponding messages (i.e. the input and output message) and the schema definition for these messages. The schema contains the XSD (XML Schema definition) that describes the xml structure for the input and output message, for example, which elements are used and their corresponding data types.

2.2 References

More information about this subject is available:

- Online Cordys Documentation
Developing Applications → Working with Web Services
- <http://community.cordys.com>
- <http://www.w3.org>

3. Exercises

3.1 Prerequisites

Before you can start with this module, take a note of the following prerequisites. The exercises are written based on successful completion of those prerequisites.

You must have completed the following modules

- Application Management.
- Web Services and the SOA Grid
- (Developing Processes for some exercises)

You must have ONLY the following roles assigned to yourself

- Administrator
- Developer
- Cordys Fundamentals Trainee

You require internet access for some exercises

- From your client to generate services on external web services
- From the Cordys server to execute the generated web services operations


3.2 Configuring Design and Deployment Structure

In this exercise you will setup the folder structure for the design time and deployment time components of the web services operations and the “metadata” for generation of those web services operations.

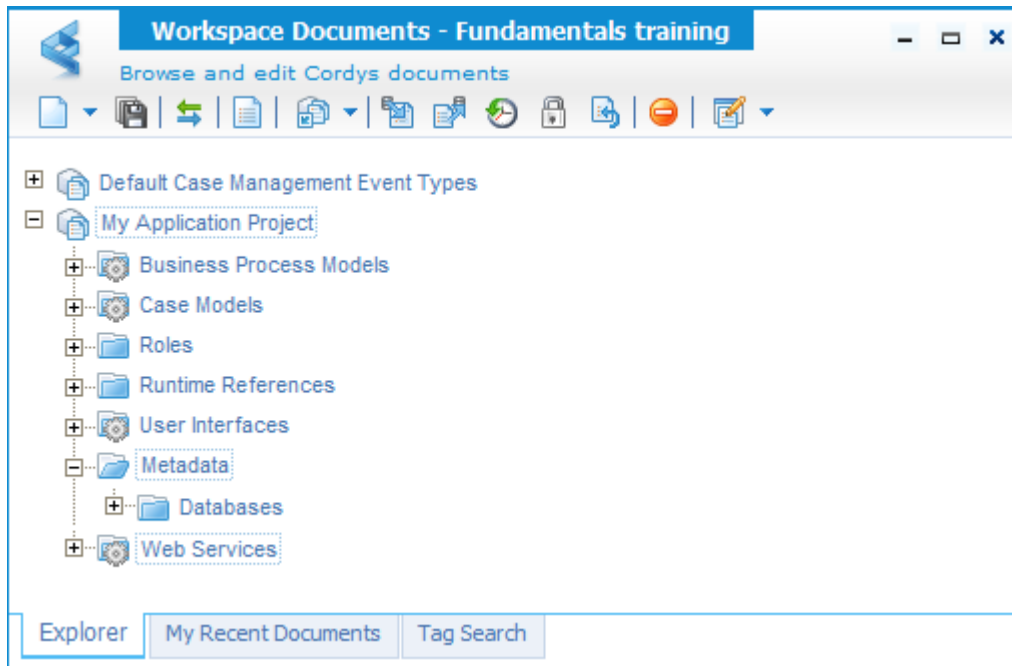
For detailed information about setting up the CWS structure, see the module *Application Management*.

3.2.1 Creating Design time Folder Structure

Here you will setup the design time folder structure for Web Services according to the standard for this training.

1. Open the *Workspace Documents* ( **Workspace Documents**).
2. Open the *Fundamentals training* workspace.

3. In the next steps, you will create the following structure:



Metadata

1. Right click the *My Application Project* and select *New → Folder*.
2. Enter the name: **Metadata**.
3. Add a subfolder named **Databases**.

Web Services

1. Right click the *My Application Project* and select *New → Folder*.
2. Enter the name: **Web Services**.

Ws-AppServer

You will configure the Design time folder structure for WS-AppServer later, in the corresponding exercise *Creating Design Folder Structure* on page 26.

3.2.2 Creating Deployment Structure

In this part of the exercise you will create the deployment structure.

Metadata

The Metadata is only used at design time, so no additional actions need to be performed.

Web Services

When you deploy web services, the corresponding web service interface is deployed. The name of the web service is built using the following elements:

- The folder structure where the web service is located, separated by dots.
- The name of web service definition followed by a dot.
- The name of the web service interface.

For example, folder.folder.webservice definition.webservice interface,
com.company1.mobility.crm.customer

Effectively the combination of web service name and application name is used to make the web service interface unique.

1. Right click the *Web Services* folder and select *Set Start Point of Qualified Name*.



Ws-AppServer

You will configure the deployment structure for WS-AppServer later; in the corresponding exercise *Creating Deployment Structure* on page 27.

3.3 Generating a Web Services Operation on Process Models

In this exercise you will generate a web services operation on a business process model that has been designed and implemented. You will use the *GetProductInformation* process that you designed and implemented in module *Developing Processes*.

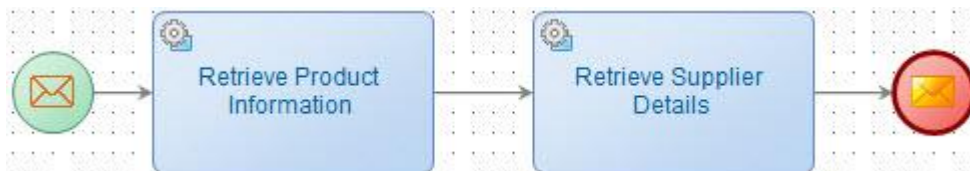
By making a BPM web service enabled, you are able to launch the business process from anywhere.

3.3.1 Prerequisites

1. You must have completed the module *Developing Processes* in order to complete this exercise.
2. Make sure the *Cordys Services* Service container is started.

3.3.2 Creating the Web Service

1. If closed, open the Workspace Documents.
2. In the *My Application* project, navigate to *Business Process Models* → *com* → *companyX* → *myapplication*.
3. Open the process *GetProductInformation*.



4. Right click in the process design area and select *Execution* → *Generate Web Service*.

5. Provide the following values:

Field	Value
Web Service	New
Web Service Name	TradingProcess
Location	My Application project/Web Services
Web Service Interface Name	Sales
Web Service Operation Name	
Checked	GetProductInformation

GetProductInformation - Web Service Generation Wizard

Select Web Service

Web service
☒ New ☐ Existing

Web Service Name
 TradingProcess

Location
 My Application Project/Web Services

Select Web Service Operation(s)

Web Service Interface Name
 Sales

<input checked="" type="checkbox"/> Web Service Operation Name	Activity Name	Monitoring	Execution Priority
<input checked="" type="checkbox"/> GetProductInformation	Start		

< Previous Next > **Finish** Cancel

6. Click **Finish**.
7. Close the process.

3.4 Exploring Web Service Components

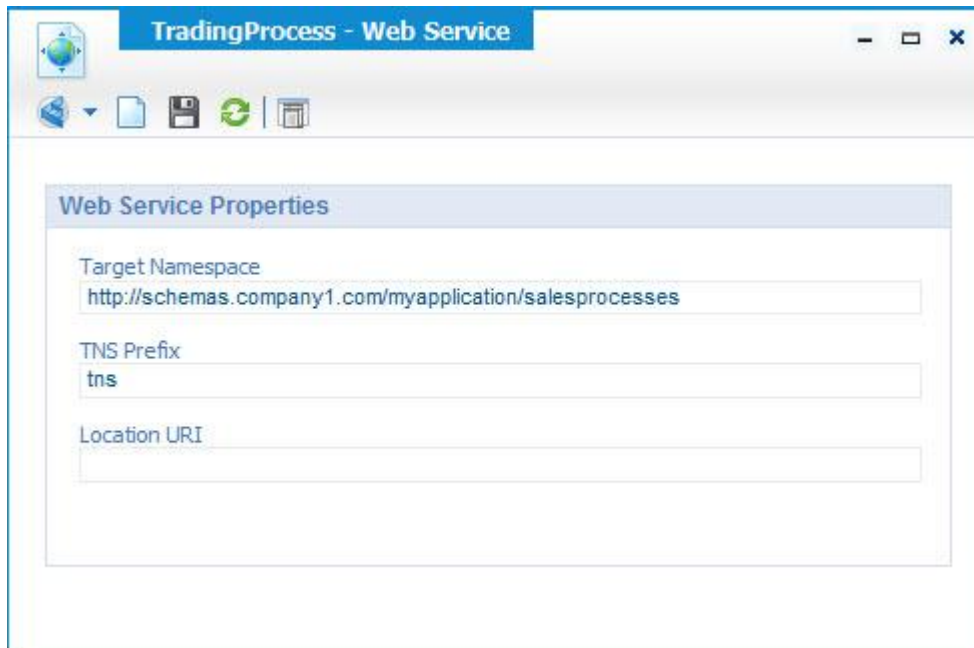
In this exercise you will have a closer look at the components that are created when you generate a Web Service. You will look into the web service you generated from the business process model.

3.4.1 Exploring the Web Service Definition

1. If closed, open the Workspace Documents.
2. In the *My Application project*, expand the folder *Web Services*.



3. Open the web service definition *TradingProcess*:



With what web service related component can you compare the web service definition?

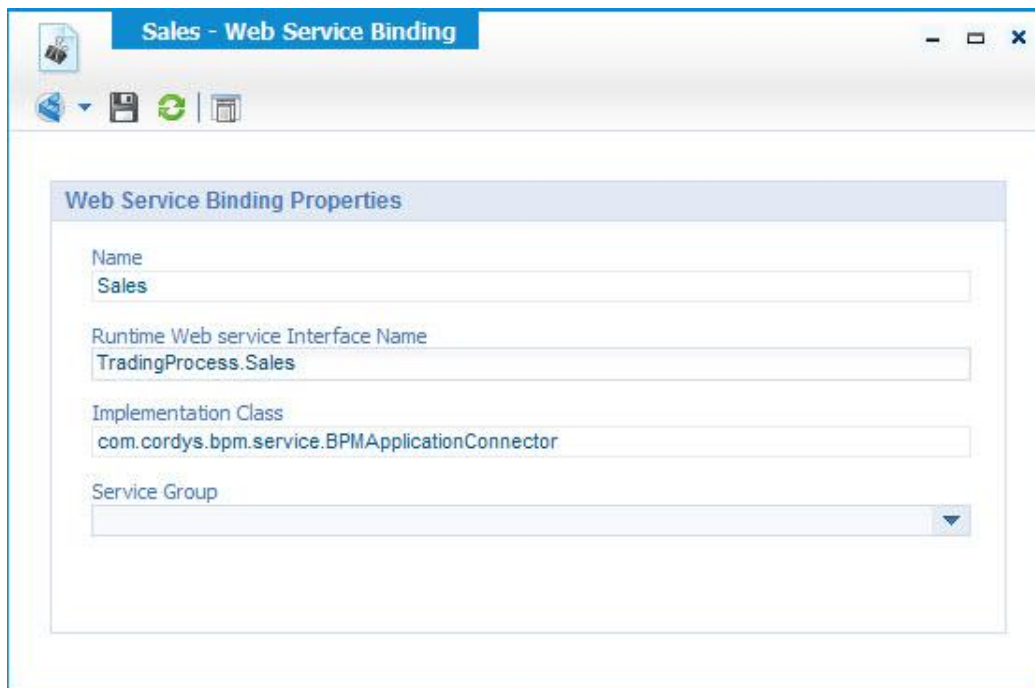
4. Close the *web service definition set* document.

3.4.2 Exploring the Web Service Interface

1. Expand the web service definition *TradingProcess*.



2. Open the web service interface document *Sales*:



How is the Runtime Web Service Interface Name constructed?

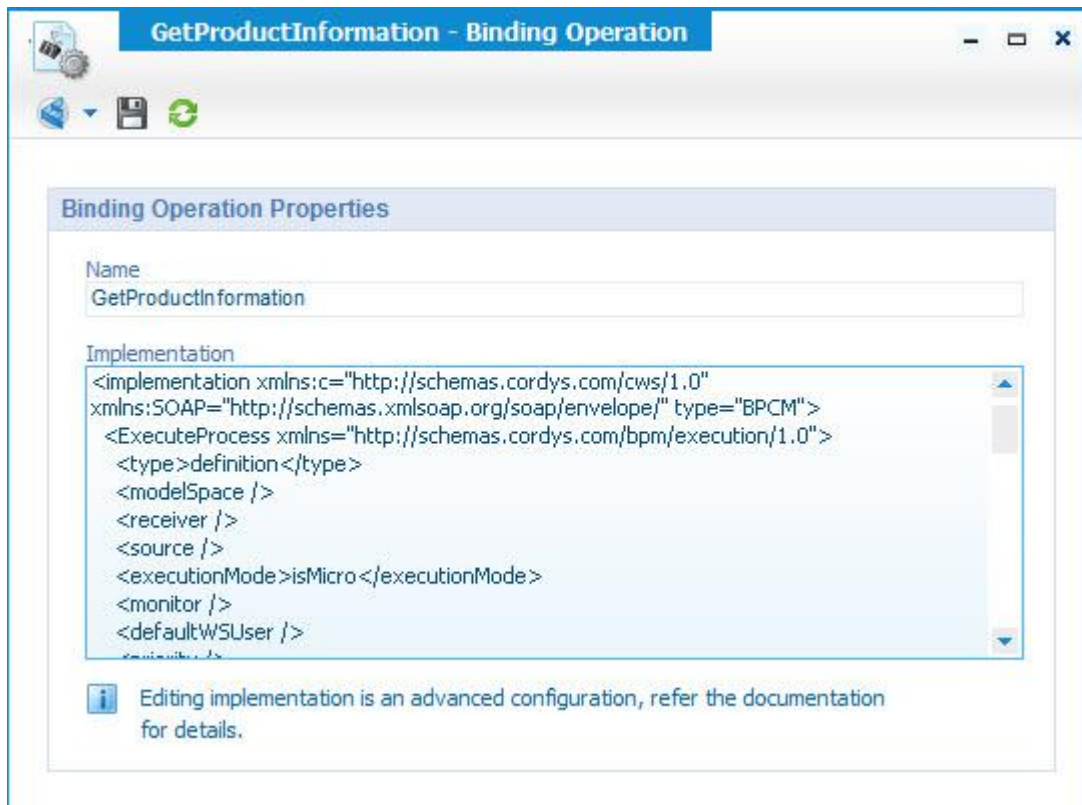
3. Close the web service interface document *Sales*.

3.4.3 Exploring the Web Service Operations



1. Expand the *Sales* web service interface, to view the available web service operations:



2. Open the *GetProductInformation* operation, to view the implementation of the operation:



When is the implementation of a Web Service operation used?

3. Click **Menu** () in the toolbar.
4. Click **Used By** ( Used By).
At this moment, the *GetProductInformation* operation is not used by any Cordys document.

NOTE

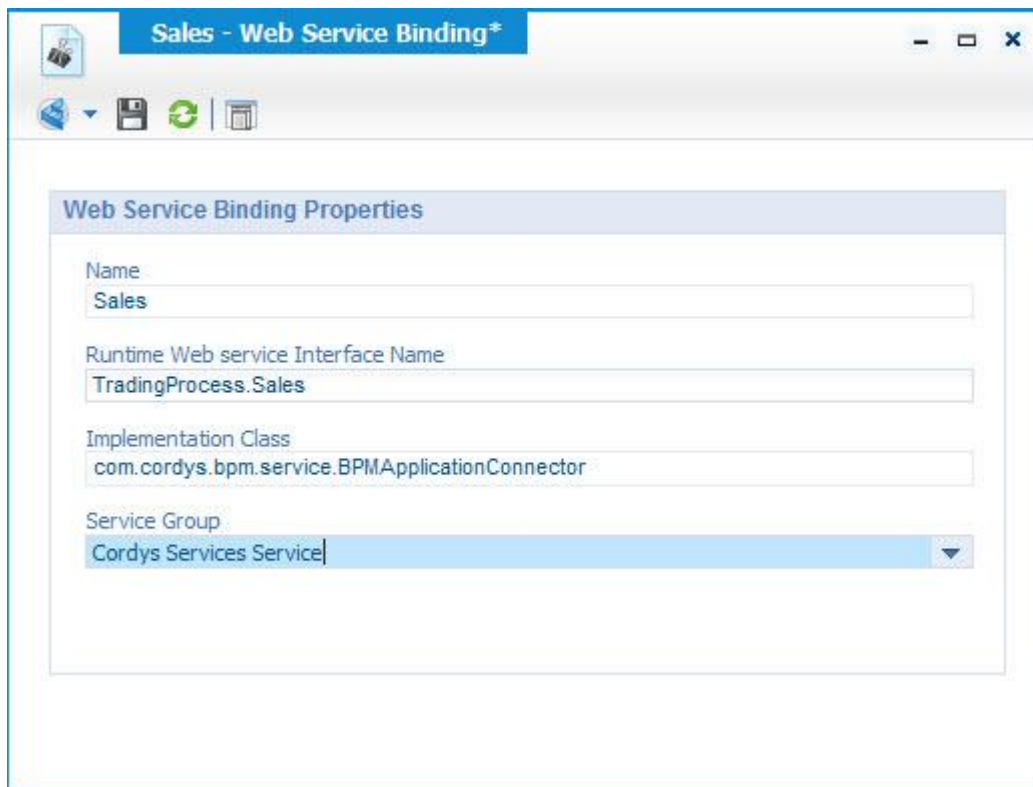
For all Cordys documents in CWS, you can see where a document is used and you can launch the parent document from the list.

5. Close the *Quick Access Menu*.
6. Close the web service operation document.
7. Right click *GetProductInformation* operation and select *Show WSDL*.
8. Have a look at the WSDL and close the *WSDL* window.

3.4.4 Testing a Web Service Operation

1. In *My Application* project, navigate to: **Web Services** → *TradingProcess*.
2. Open the *Sales* web service interface document.

3. Select the *Service Group: Cordys Services Service*.

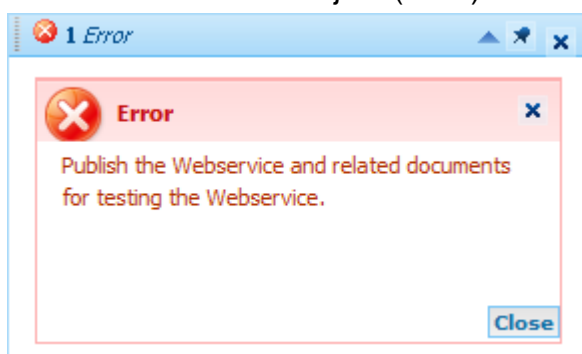


NOTE

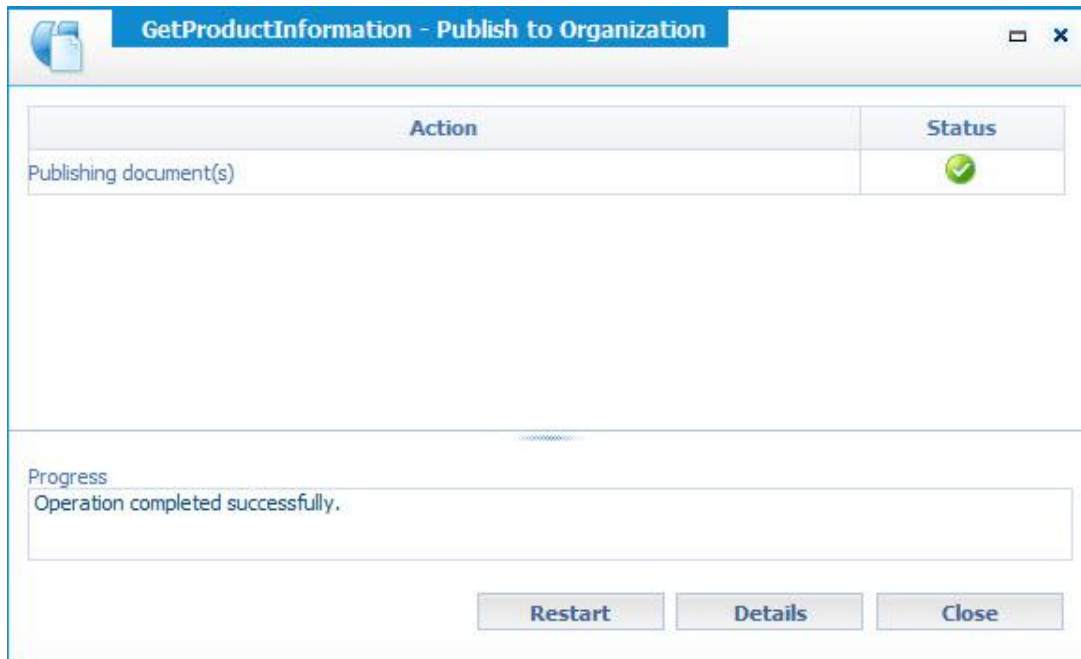
The Cordys Services Service is already setup in the training environment and is configured to execute business process model related components in your organization, like this BPM Web Service Operation.

Why is it important to select the service group and what is an alternative way to achieve the same result?

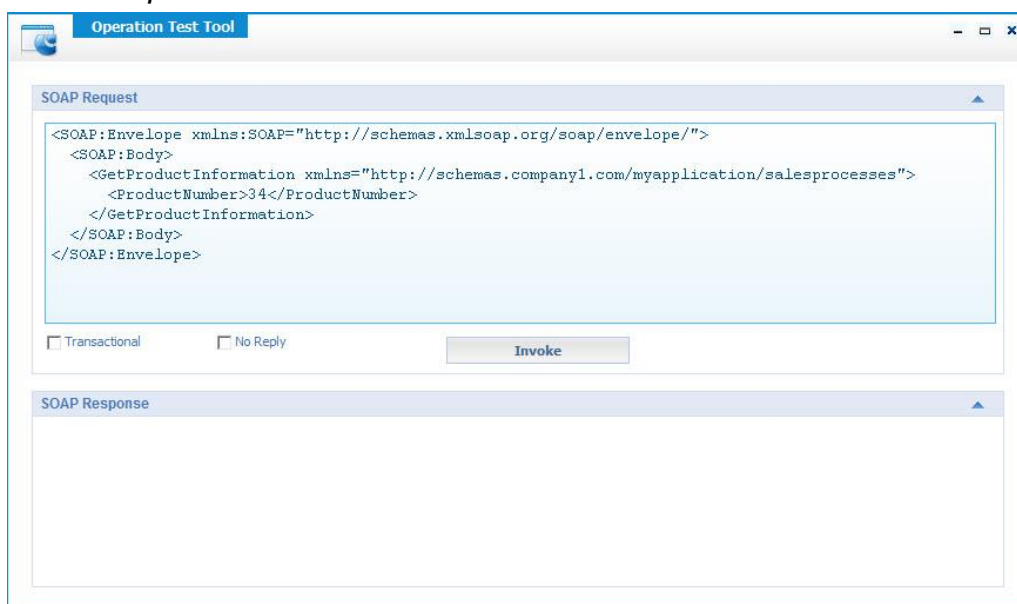
4. **Save** the web service interface document and close the window.
5. Right click the *GetProductInformation* operation and select *Test Web Service Operation*.
6. A Unified Feedback Object (UFO) indicates the service is not published yet:



7. Right click the *GetProductInformation* operation and select *Publish to Organization*.

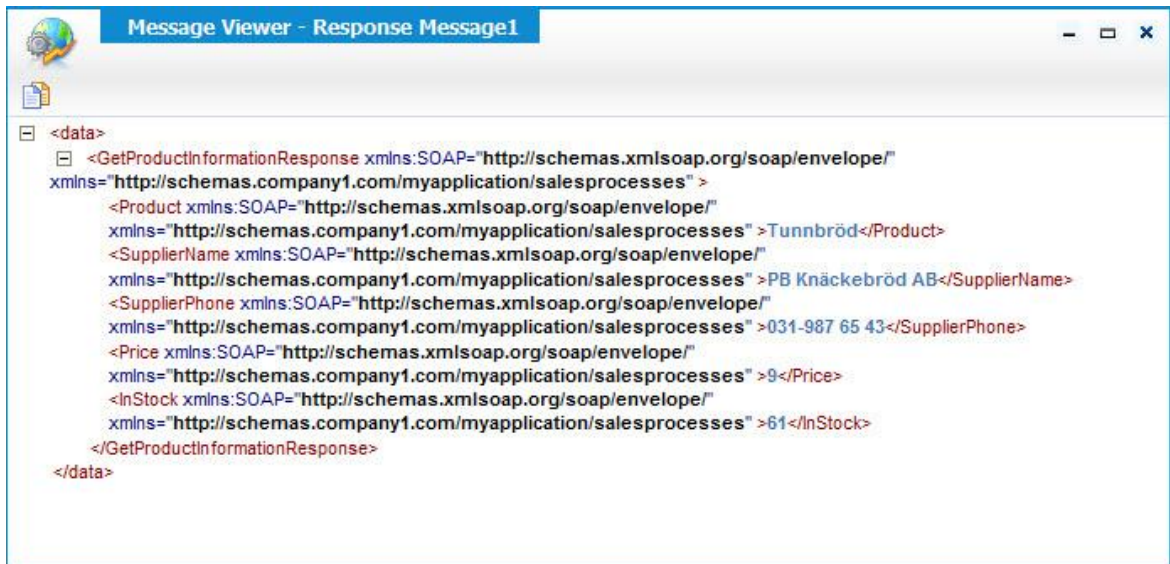


8. Close the *Publish to Organization* window.
9. Right click the *GetProductInformation* operation and select *Test Web Service Operation* again.
10. Provide a *product number* between 1 and 77.



11. Click **Invoke**.

12. Check the response message, which is similar to executing the BPM.



When you published the Web Service to your organization what was actually done?

3.5 Working with External Web Services

In this exercise you will generate web services operations in Cordys that invoke External Web Services. The service in Cordys functions as bridge between those service and can then directly be used in User Interfaces and Business Process Models. In addition, security can also be set when the external service requires authentication.


3.5.1 Prerequisites

NOTE

For this exercise internet access is required in order to work with the external web service. If you do not have internet access you can safely ignore this exercise.

3.5.2 Configuring Connectivity

In order to generate web services in Cordys you need to have a Service Group configuration towards the backend from which you will generate the web service. In this case the external web service. This service group can also directly be used to test the web service. Custom web service exists only in your organization and therefore requires the service group as well in your organization.

1. Open the *System Resource Manager*.
2. Click **New** ().

Application Connector

3. Select the *UDDI Service* application connector.

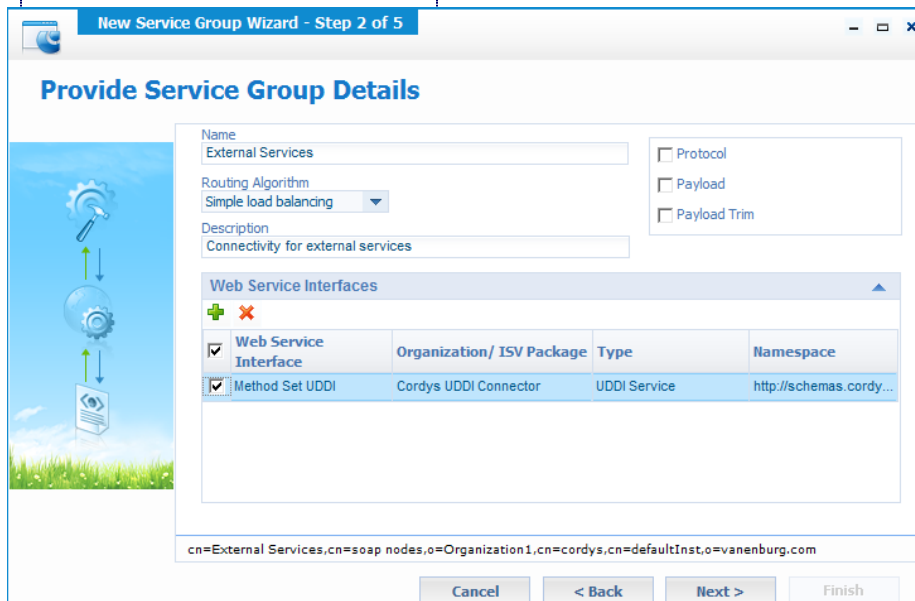


4. Click **Next**.

Service Group

5. Provide the following values for the Service Group:

Field	Value
Name	External Services
Routing algorithm	Simple load balancing
Description	Connectivity for external services
Web Service Interfaces check	Method Set UDDI



6. Click **Next**.

Service Container

7. Provide the following values for the Service Container:

Field	Value
Name	External Services
Startup Type	Manual

8. Click **Next**.

Application Connector Properties

9. No actions required.

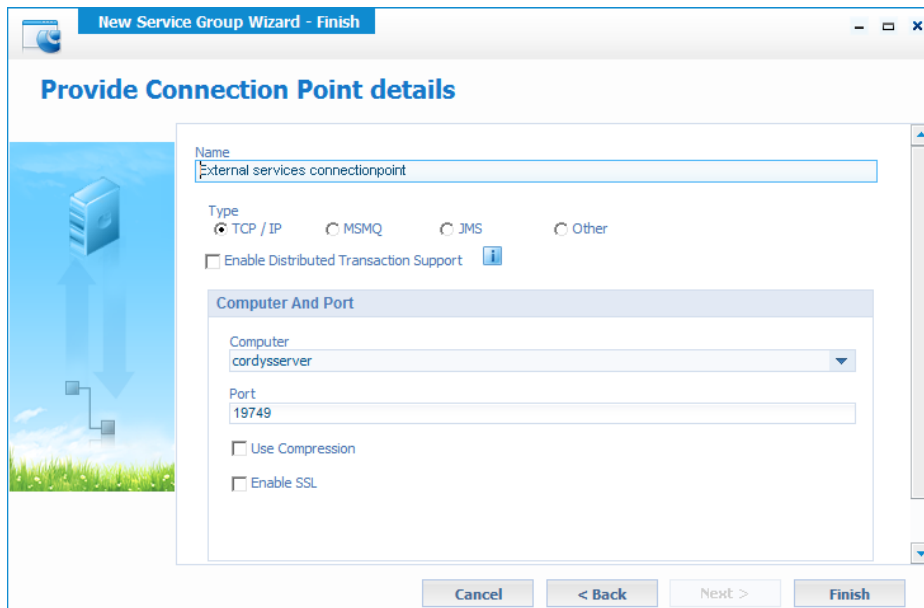
NOTE

When performing this exercise at your company network it is possible that you have to provide proxy details. Ask your system administrator.

10. Click **Next**.

Connection point

11. Provide the name: **External Services connectionpoint**.
12. Accept the random *TCP/IP* port for this training.



13. Click **Finish**.
14. Start the *External Services* service container.

3.5.3 Generating the Web Service

In this exercise you will generate a web service that contains operations that work on the related external web services.

1. If closed, open the *Fundamentals workspace*.
2. Click **New** in the *Workspace Document* toolbar.



Web Service

Create a Web Service to access business logic from various sources like Database, WS-AppServer Package, External WSDL, Java Classes and custom

3. Select *Web Service* ().

4. Enter the following values:

Field	Value
Select the Source	Import WSDL
Name	MicrosoftTerraServer
Description	Microsoft TerraServer USA
Location	My Application project/Web Services

MicrosoftTerraServer - Web Service*

Select the source
Import WSDL

Name
MicrosoftTerraServer

Description
Microsoft TerraServer USA

Location
Web Services

< Previous Next > Finish Cancel

5. Click **Next**.

6. Provide the following url:
<http://msrmaps.com/TerraService2.asmx?wsdl>

MicrosoftTerraServer - Web Service*

WSDL URL <http://msrmaps.com/TerraService2.asmx?wsdl>

☐ Authentication to read the WSDL ☐ Authentication to invoke the service

Type Basic Type Basic

Username Username

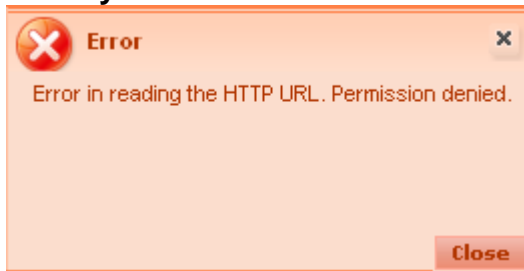
Password Password

☐ Interface WSDL

Show Services

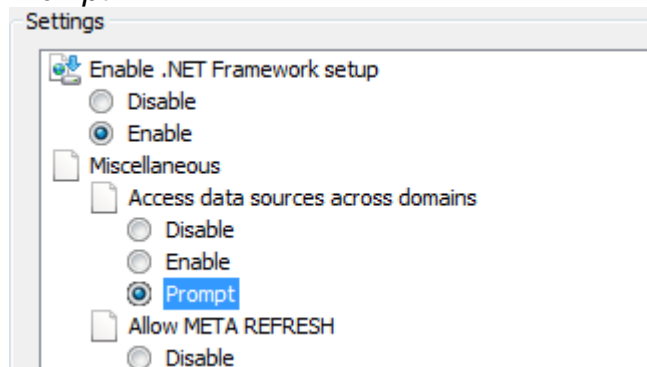
< Previous Next > Finish Cancel

7. Click **Show Services**.

When you have: URL Permission denied!

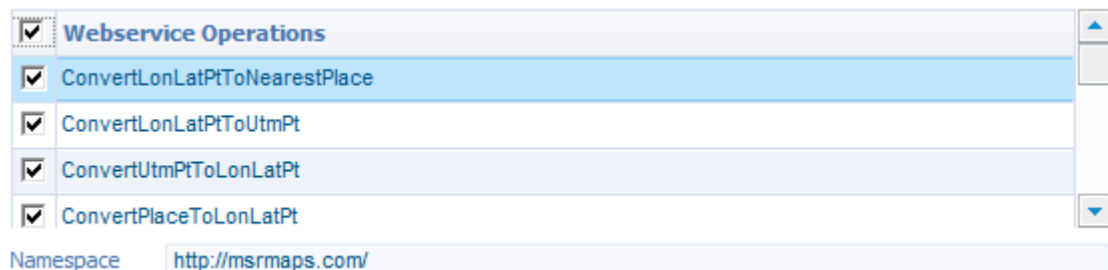
Most likely your browser does not allow “Access data sources across domains”. For Internet Explorer you can change it by:

- Internet Explorer menu *Tools, Internet Options*.
- Tab page *Security*, Click **Custom Level...**
- In group *Miscellaneous* set *Access data sources across domains* to *Prompt*.



- Click **OK** (and **Yes** to confirm).
- Click **OK**.

When you click **Show Services** again you are prompted to confirm access.

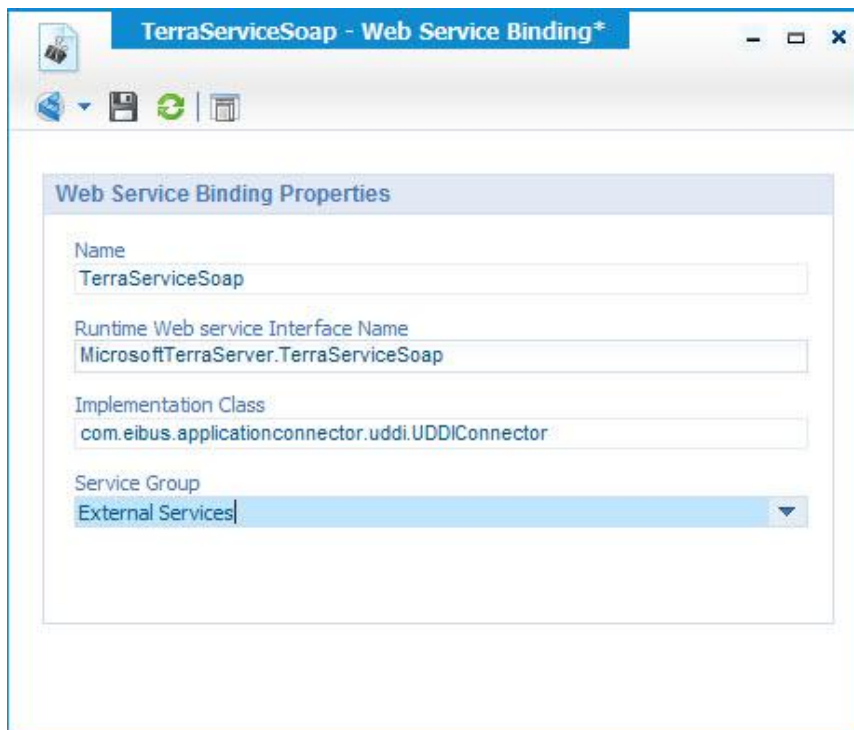
8. Select all *Webservice Operations*:**9.** Click **Finish**.**10.** Close the *MicrosoftTerraService Web Service* window.

Why would you use external web services?

What are the limitations of using external web services?

3.5.4 Testing a Web Service Operation

1. In *My Application* project, navigate to: [Web Services](#) → [MicrosoftTerraServer](#).
2. Open the TerraServiceSoap web service interface document.
3. Select the *Service Group: External Services*.



4. **Save** the web service interface document and close the window.
5. Publish the *TerraServiceSoap* Web Service Interface to the organization.
6. Right click the *GetPlaceList* operation and select *Test Web Service Operation*.
7. Provide the following input message:

```
<placeName>Amsterdam</placeName>
<MaxItems>5</MaxItems>
<imagePresence>false</imagePresence>
```


8. Click **Invoke**.
9. Check the response message.
10. Open the *System Resource Manager* and stop the *External Services* service container.

3.6 Generating Web Service Operations on Databases

In this exercise you will make the Northwind database web service enabled. You will create the required web service operations on top of the tables in this database with a business logic layer in between.

3.6.1 Configuring Connectivity

You will first setup a service group which is required to connect to the Northwind database. For this you will use the WS-AppServer application connector.

1. Open the *System Resource Manager*.
2. Click **New** ()

Application Connector

3. Select the *WS-AppServer* application connector.
4. Click **Next**.

Service Group

5. Provide the following values for the service group:

Field	Value
Name	Northwind Service
Routing algorithm	Simple load balancing
Description	Northwind database connection
Web Service Interfaces check	WS-AppServer Development Method Set WS-AppServer Runtime Method Set

6. Click **Next**.

Service Container

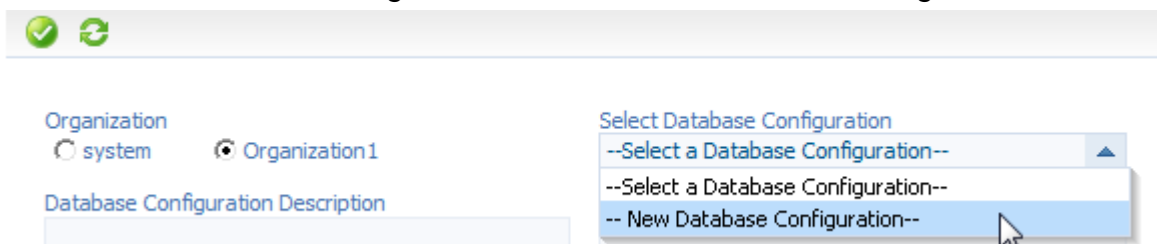
7. Provide the following values for the Service Container:

Field	Value
Name	Northwind
Startup Type	Manual

8. Click **Next**.

Application Connector Properties

9. Select *New database Configuration* in the *Select Database Configuration* field




10. Provide the following values for the *Database Configuration*:

Field	Value
Name	Northwind Test
Description	Northwind test database

The screenshot shows the 'New Database Configuration' dialog box. The 'Name' field contains 'Northwind Test' and the 'Description' field contains 'Northwind Test Database'.

For the values of the remaining fields:

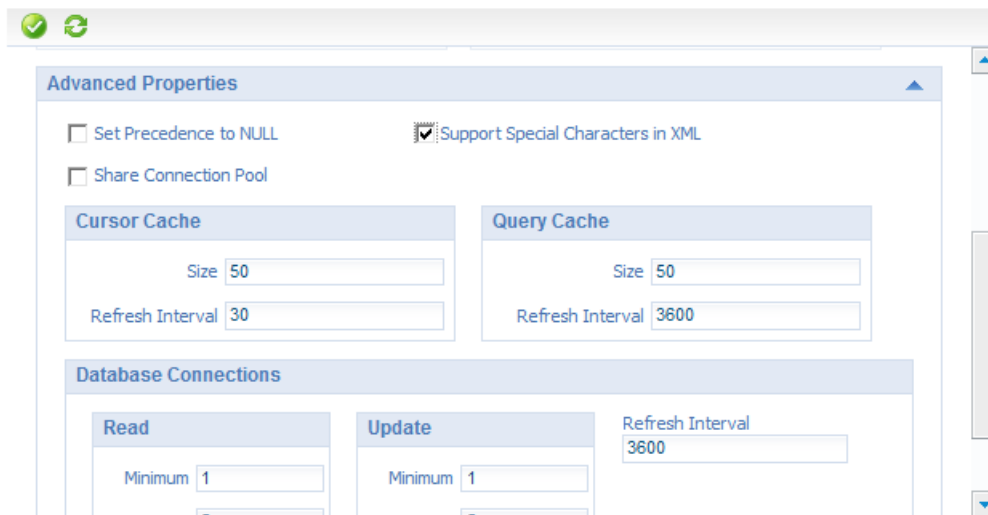
11. Open *Fundamentals Configuration* ().
12. Go to the tab *Developing Services*.
13. Use these values to fill the details for the *Northwind Test* database configuration.
14. Click **Test Connectivity** to check whether you have the correct details:

The screenshot shows the 'New Database Configuration' dialog box with the 'JDBC' tab selected. The 'JDBC Driver' is set to 'MySQL', 'Driver Class' is 'com.mysql.jdbc.Driver', and 'Connection String' is 'jdbc:server:3306?useUnicode=true&characterEncoding=UTF-8'. The 'Default Database' is 'Northwind', 'DB User' is 'nw', and 'Password' is masked. An 'Information' dialog box is open, stating 'Test Connectivity was successful.'

NOTE

The database configuration shown in the above image is used in combination with the Self Study Training Environment. The correct configuration details for classroom training can be found in the Fundamentals Configuration artifact.

15. Click **Save**.
16. Expand the *Advanced Properties* box.

17. Check *Support Special Characters in XML*.**NOTE**

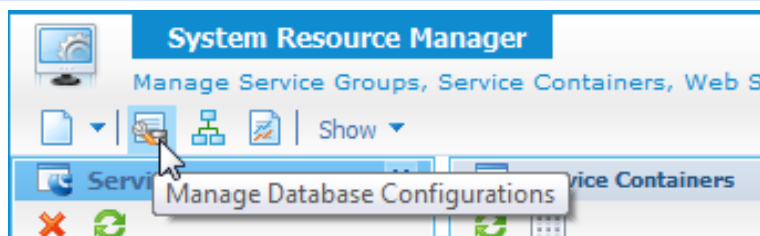
When you expect that in a database, a table or field name will contain characters like spaces “ ”, dollars “\$” etc you should check this option. As XML does not allow these characters, this setting will make sure that the application connector replaces these with a valid string.

18. Click **Next**.Connection point

- 19.** Provide the name: **Northwind connectionpoint**.
- 20.** Accept the random TCP/IP port for this training.
- 21.** Click **Finish**.
- 22.** Start the *Northwind* service container.

NOTE

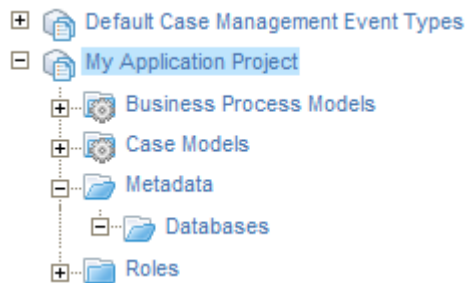
You can manage database configurations afterwards via the System Resource Manager.



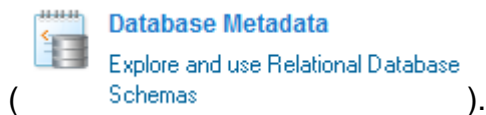
3.6.2 Creating Database Metadata

To build the Java logic and the Web Service operations on top of a database, you first have to import the database metadata into your project. The imported metadata contains information like tables, fields, key constraints etc.

1. If closed, open the *Workspace Documents*.
2. In *My Application Project* navigate to *Metadata* → *Databases*.



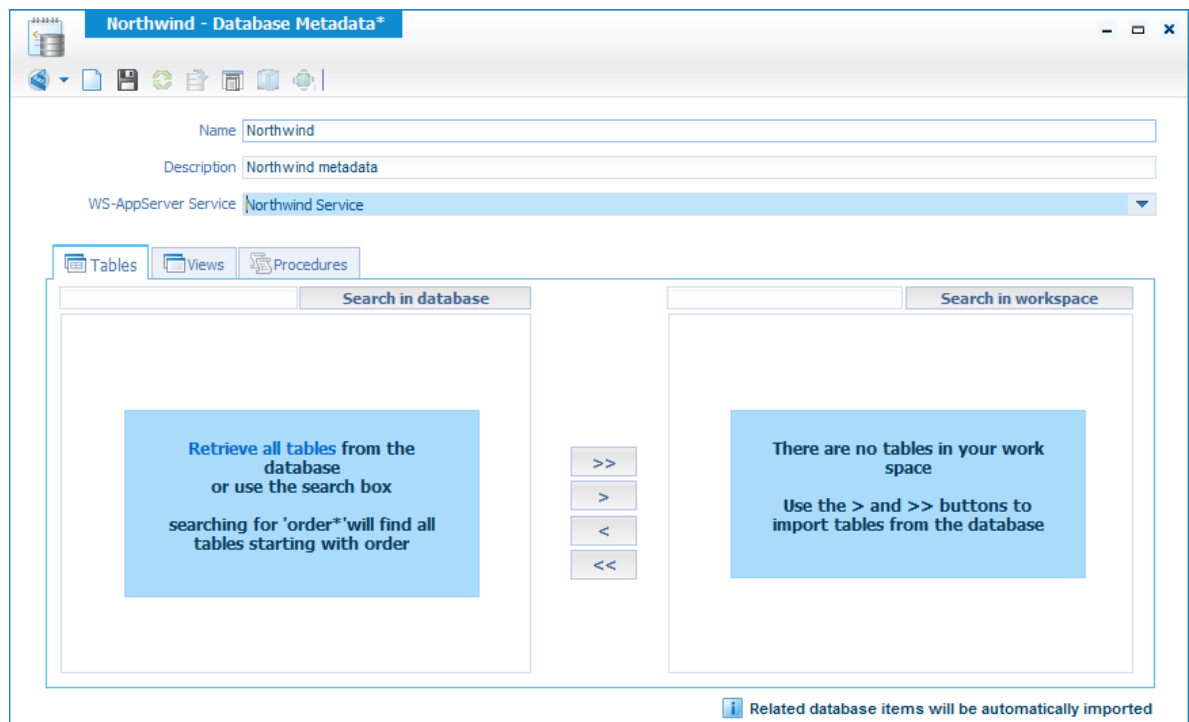
3. In the folder *Databases* add a new *Database Metadata* document.



4. Provide the following values:

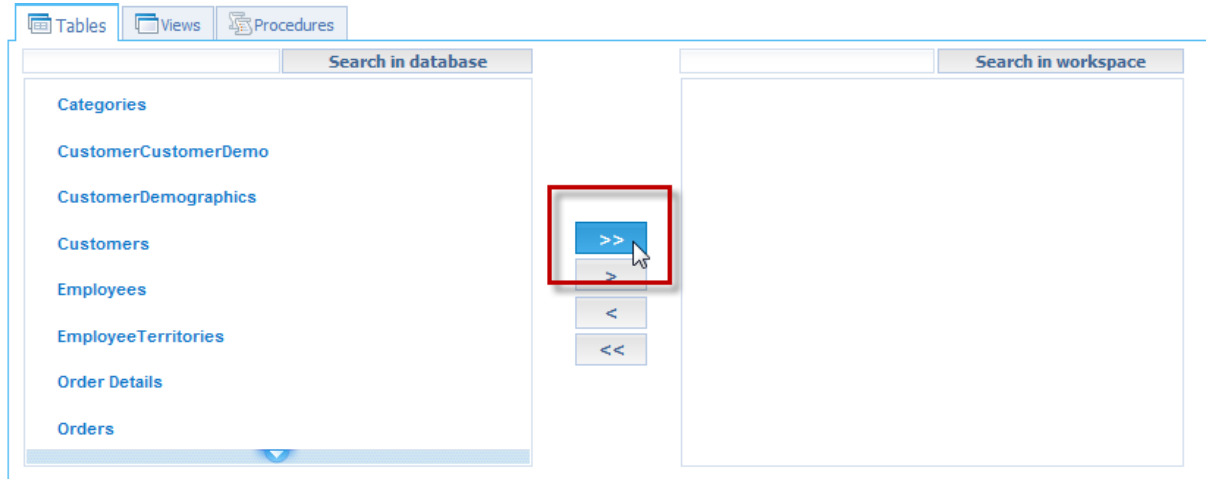
Field	Value
Name	Northwind
Description	Northwind metadata
WS-AppServer Service	Northwind Service

5. Make sure the *Tables* tab is selected:



6. Click the hyperlink *Retrieve all tables*.

7. Click the **Add all database items to the selection button**:



8. Click **Save**.
9. In the *Workspace Documents*, expand *Northwind* in the *Metadata → Databases* folder.

Which information can be found here?

10. Open the *Orders* table.
11. Explore the information displayed and close.

3.6.3 Creating Design Folder Structure

In this part you will configure the design time folder structure for WS-AppServer development.

WS-AppServer package

1. Right click the *My Application Project* and select *New → Folder*.
2. Provide the name: **WS-AppServer**.
3. Add a subfolder **com**.
4. Add a subfolder **companyX**. (X is your student number).
5. Add a subfolder **myapplication**.
6. Add a subfolder named **northwind**.
7. Add a subfolder named **javasources**.

Application files

1. Right click the *My Application Project* and select *New → Folder*.
2. Provide the name: **Cordys Install dir**.

3.6.4 Creating Deployment Structure

In this part you will configure the deployment structure for WS-AppServer deployment.

WS-AppServer package

When you deploy this application, a cmx file is created based on the WS-AppServer package contents that you have created. The name for this file will contain both the folder path created and the name of your WS-AppServer package. In this case, this will result in: com.companyX.myapplication.northwind.<<packagename>>.cmx

1. Right click the *WS-AppServer* folder and select *Set Start Point of Qualified Name*.

Application files

Application files are files like java class files (class.jar files), database scripts etc. which do not need to be accessible by the end users. These files need to be accessible by for example service containers, system administrators and have to be created in the Cordys installation directory. A directory path structure is used to make them unique.

1. Right click the *Cordys Install Dir* folder and select *Set Start Point of Qualified Name*.
2. In the *Cordys Install dir* folder add a subfolder **com**.
3. Add a subfolder **companyX**. (X is your student number)
4. Add a subfolder **myapplication**.

The folder structure should be similar to the following structure:



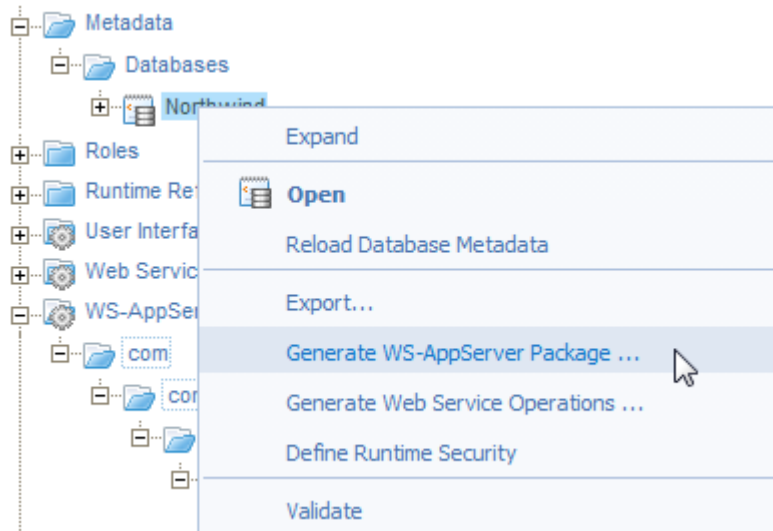
3.6.5 Generate the WS-AppServer Package

In this part you will generate a WS-AppServer package based on the Northwind tables. Related components like the java sources, web service operations will be generated as well.

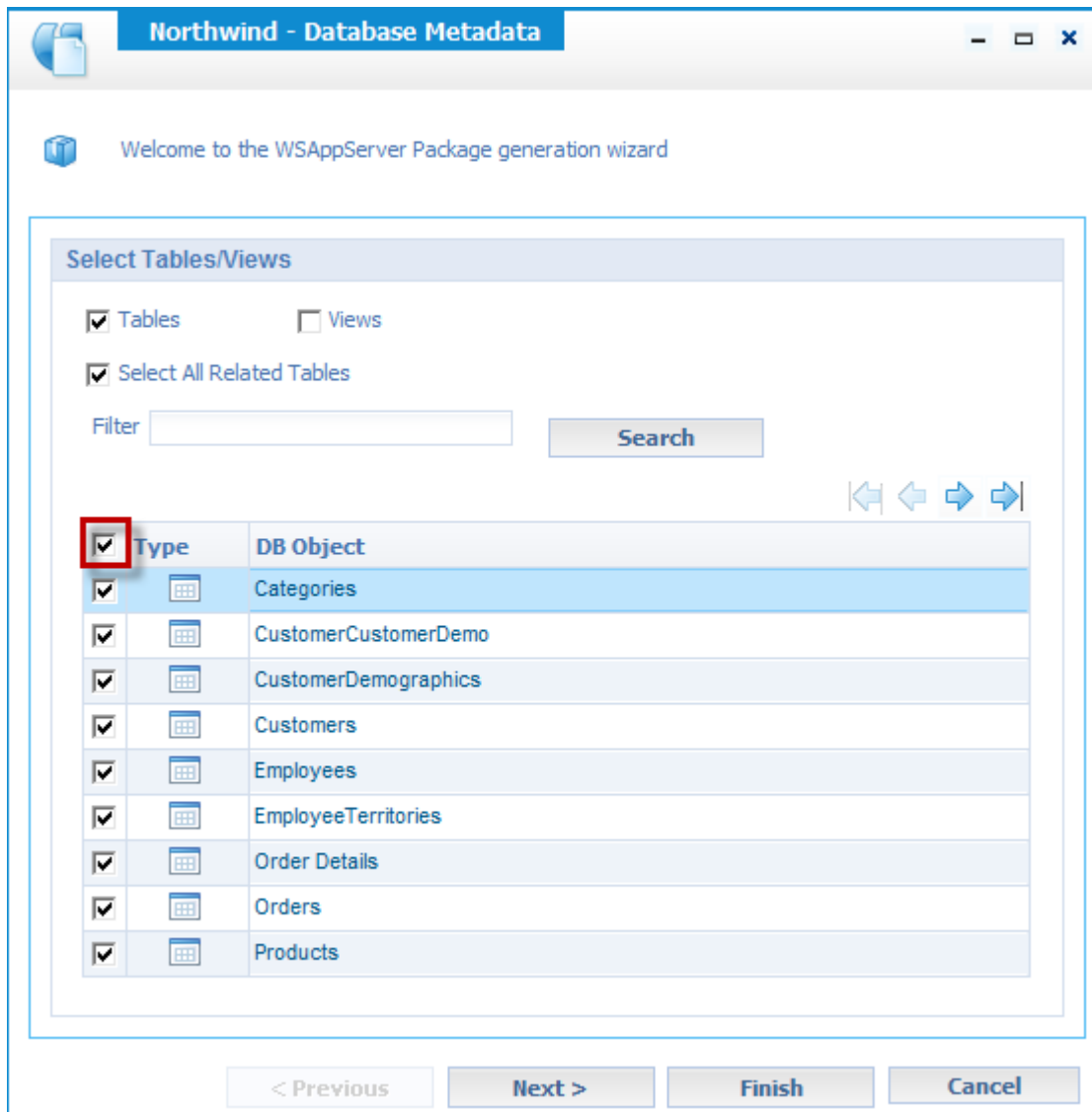
Create the WS-AppServer package

1. Navigate to: *Metadata* → *Databases* → *Northwind*.

2. Right click *Northwind* and select *Generate WS-AppServer Package...*



3. Select all tables of the *Northwind* database:



4. Click **Next**.

5. Provide the following values:

Field	Value
Package Name	Northwind
Java Package Name	com.companyX.myapplication.northwind
Namespace	http://schemas.companyX.com/myapplication/northwind
Package Folder	Existing
Folder	My Application project/WS-AppServer/com/companyX/myapplication/northwind
Enable web service generation for: checked	all get operations (get and getAll) all relational operations (getObjectsFor...) all navigational operations (getNext and getPrevious)

Northwind - Database Metadata

Welcome to the WS-AppServer Package Generation Wizard

WS-AppServer Package

Package Name
Northwind

Java Package Name
com.company1.myapplication.northwind

Namespace
http://schemas.company1.com/myapplication/northwind

Package Folder
☐ New
 ☒ Existing

Select a Folder
northwind

Enable Web service Generation for:

☒ all get operations (get and getAll)
☒ all relational operations (getObjectsFor...)
☒ all navigational operations (getNext and getPrevious)

This will set the flag 'SOAP' on the operations of a WS-AppServer class.

< Previous Next > Finish Cancel

NOTE

In this training you select to generate all web service operations, in a real project you probably do not require the long list of all default web services and would manually selected the required ones.

6. Click **Next**.

7. Provide the following details for the java generation:

Field	Value
Generate Java Code	Checked
Java Generation details	New
Java Sources Folder	My Application project/WS-AppServer/com/companyX/myapplication/northwind/javasources
Java Archive Folder	My Application project/Cordys Install dir/com/companyX/myapplication
Java Archive Name	Northwind

Northwind - Database Metadata

Welcome to the WS-AppServer Package Generation Wizard

☒ Generate Java Code

Java Generation Details

☒ New ☐ Generate into existing Java Archive Source

Java Sources Folder:

Java Archive Folder:

Java Archive Name:

< Previous Next > Finish Cancel

8. Click Finish.

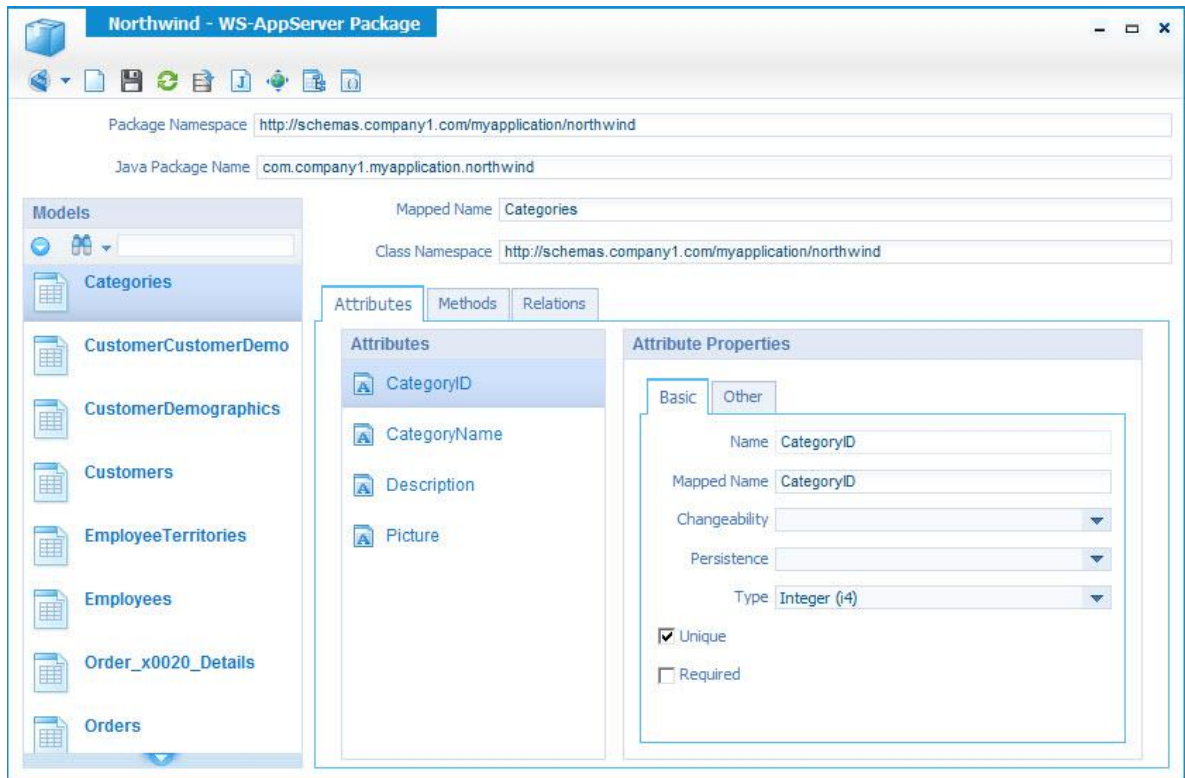
Look in the (sub)folders WS-AppServer and Cordys Install dir, which content is created?


Generate Web Service Interface

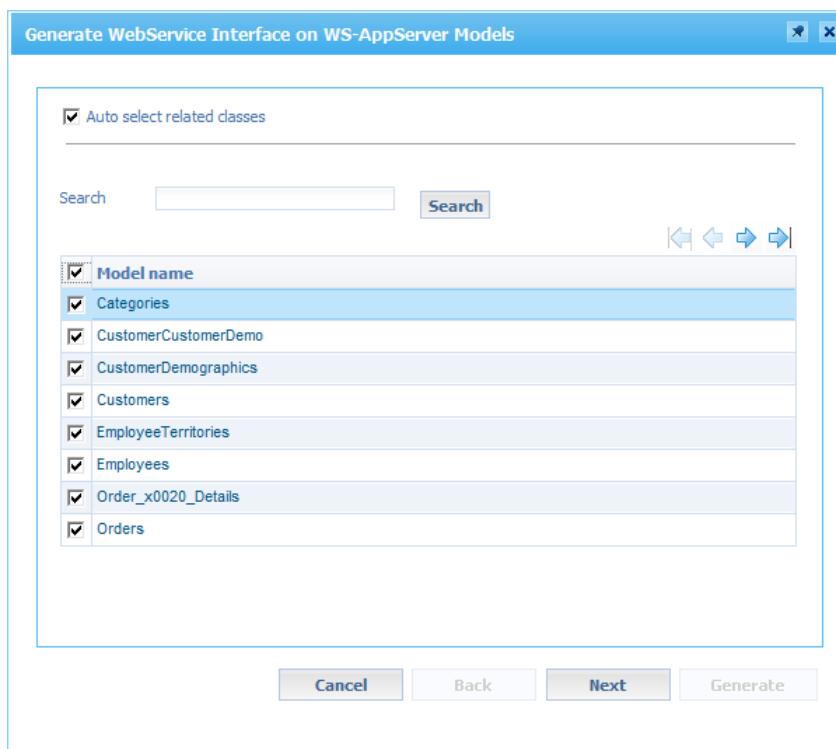
In this part you will generate the web service operations from the generated WS-AppServer package contents.

1. Navigate to: *WS-AppServer* → *com* → *companyX* → *myapplication* → *northwind*.

2. Open the *Northwind* WS-AppServer package ( Northwind).



3. Click **Generate Web Service Interface** () in the toolbar.
4. Select All *Model* names.



5. Click **Next**.

6. Provide the following values:

Field	Value
Web Service Interface	New
Web Service Name	Northwind
Web Service interface Name	Basic
Folder Name	My Application project/Web Services

7. Click **Generate**.

What can be generated from the WS-AppServer package as well?

-
8. Close the *Northwind - WS-AppServer Package*.
 9. Navigate to: *Web Services* → *Northwind*
 10. Explore its contents.

3.6.6 Publishing and Testing WS-AppServer Package

In this part you will publish the WS-AppServer package and related components to the organization and test the generated web service Operations.

Publish to Organization

To be able to use web service operations from a WS-AppServer package, the following components needs to be published:

- Web service interface
- WS-AppServer package
 - Java archive file

1. Navigate to: *Web Services → Northwind*.
2. Right click *Northwind* and select *Publish to Organization*.
Note that this will also publish the related WS-AppServer package and the java archive file in the Cordys Install dir folder.

Test Web Service Operation

1. Navigate to *Web Services → Northwind → Basic → GetProductsObject*.
2. Right click *GetProductsObject* and select *Test Web Service Operation*.
3. Test the operation (ProductID values range from 1 to 77).

How is it possible that you can use the web service operation without specifying the Northwind.jar file in the Service Container class path?

In the module Developing User Interfaces you will add some static business logic to this package and study the integration with a user interface.

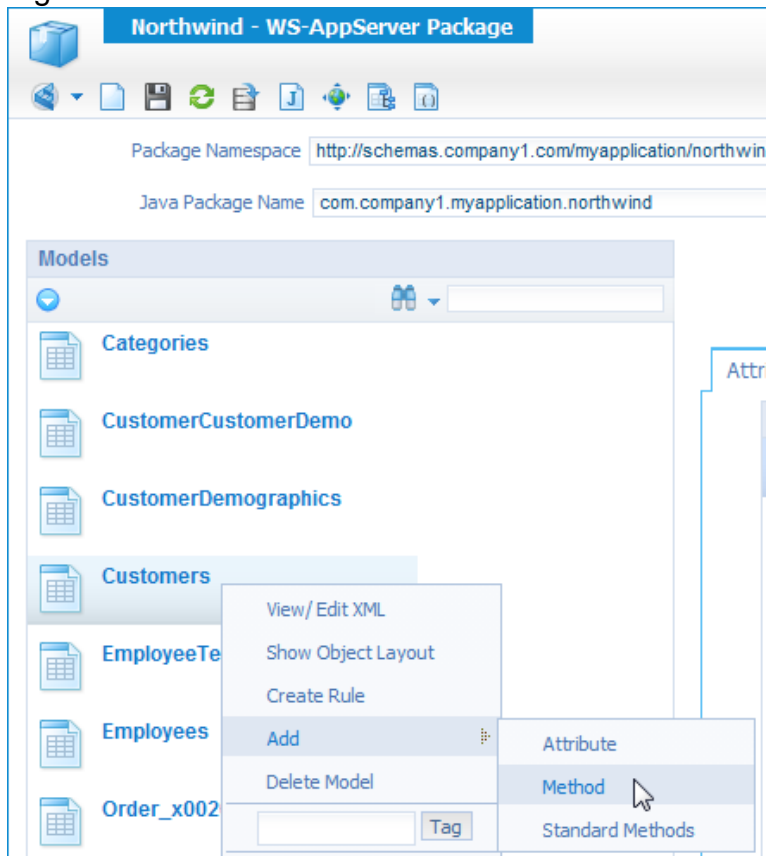
3.7 Adding Web Service Operations Manually

Next to the web service operations that are created automatically, you can also define your own, e.g. when you want to list a set of customers by city or country. All the default web service operations are based on the primary key.

3.7.1 Create the web service operation

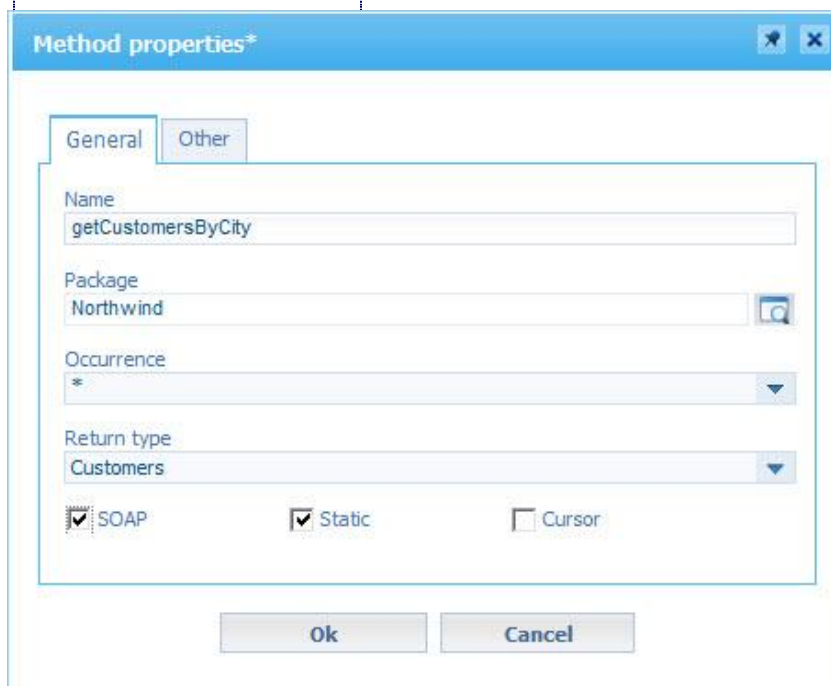
1. Navigate to: *WS-AppServer → com → company → myapplication → northwind* and open the *Northwind* package.

2. Right click the *Customers* class and select **Add → Method:**



3. Provide the following values:

Field	Value
Name	getCustomersByCity
Package	Northwind
Return type	Customers
Occurrence	*
SOAP	Checked
Static	Checked



4. Click **OK**.
5. Select the method you have just created to see its properties:

The screenshot shows the 'Method Properties' dialog box. On the left, under the 'Methods' tab, a list of methods is shown: `getCustomersByCity`, `getCustomersObjects`, `getNextCustomersObjects`, `getCustomersObject`, and `getPreviousCustomersObjects`. The `getCustomersByCity` method is selected. The right pane shows the properties for this method. The 'Name' field contains `getCustomersByCity`. Under 'Return Type Properties', the 'Package' is 'Northwind', 'Return Type' is 'Customers', 'Occurrence' is '*', and 'SOAP Result' is 'tupleset'. The 'Transaction' is 'automatic' and the 'Implementation' is 'default:GetByAttributes'. There are checkboxes for 'SOAP' (checked), 'Cursor' (unchecked), and 'Static' (checked). The 'Parameters' section shows a table with one parameter:

Name	Package	Type	Attribute
City		string	City

NOTE

When you change the implementation the value for Occurrence will also change. Attributes is 1 occurrence, AttributesRange is * occurrence.

6. Click the Insert button (+) to add the parameter.
7. Fill in **City** as the parameter name and choose *City* from the select box for *Linked Attribute*:

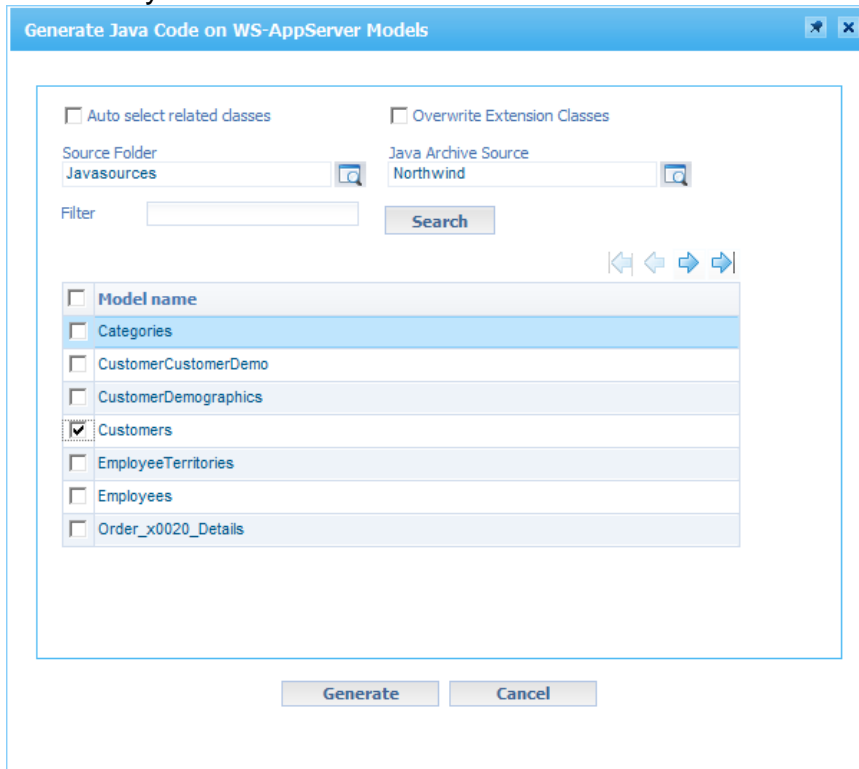
The 'Parameter properties*' dialog box is shown. It has a title bar with a close button. The main area is titled 'Parameter properties'. It contains the following fields:

- Name:** City
- Package:** (empty)
- Linked Attribute:** City
- Type:** string

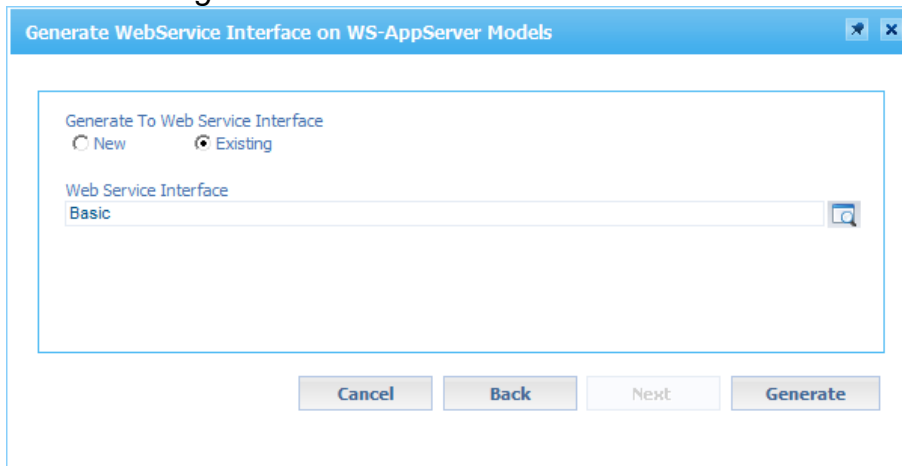
At the bottom, there are 'Ok' and 'Cancel' buttons.

8. Click **OK**.
9. A message will pop up telling you that you need to re-generate the Java code.
10. Select the **Generate Java** (J) button.

11. Uncheck *Auto select related classes*.
12. Check only the *Customers* model and click **Generate**.



13. Select the **Generate Web Service Interface** (🌐) button.
14. Uncheck *Auto select related classes* again and select only the *Customers* model.
15. Click **Next**.
16. Select *Existing* and use the zoom button to select the *Basic* Web Service Interface:



17. Click **Generate**.
18. Close the *Northwind* package.
19. Navigate to **Web Services** → **Northwind** → **Basic** → **GetCustomersByCity**
20. Right click the web service operation and select *Publish to Organization*. This will automatically recompile the Java source files.
21. Test the web service operation by filling in some city names like **Paris**, **Portland**, **Berlin**, **México D.F.**, etc.

3.8 Web Services Security

By default, users cannot use web service operations. You need to grant this permission explicitly. Therefore you must have at least one role in every application that has execute permission for the web service interface(s). When using multiple roles you can fine grain the security by allowing only certain web service interfaces or operations.

1. If closed, open the *Workspace Documents*.

3.8.1 Prerequisites

NOTE

The following steps are only required if you have not yet created the *Purchase Manager* role in the Roles folder.

1. In *My Application project* navigate to *Roles*.
2. Right click *Roles* and select *New → Other*.
3. Select *Role*.
4. Click **Save**.
5. Provide the following values:

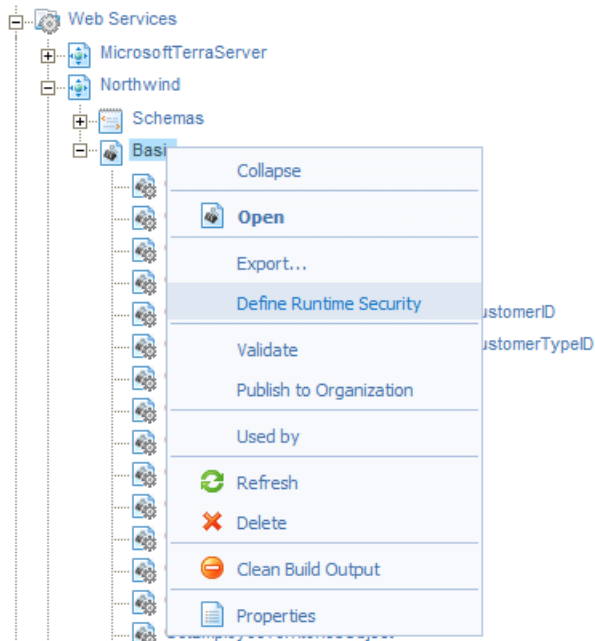
Field	Value
Name	Purchase Manager
Description	Purchase managers
Location	Prefilled with: My Application project/Roles

6. Click **Save**.

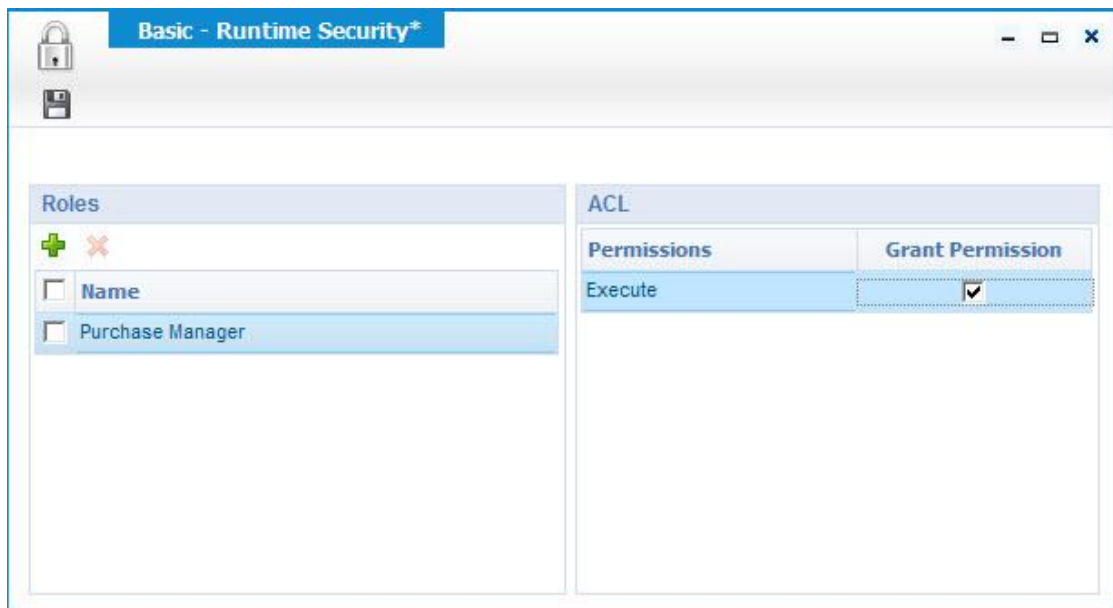
3.8.2 Define Runtime Security

1. Navigate to *Web Services → Northwind → Basic*.

2. Right click *Basic* and select *Define Runtime Security*.



3. Click **Add identity** (+) and select the *Purchase Manager* role.
4. Check *Grant Permission to Execute*:



5. Click **Save**.

After you have assigned this role to users, they are allowed to use all the Web Service Operations in this Web Service interface.

6. Expand the *Basic* Web Service Interface.
7. Right click the *GetEmployeesObject* operation and select *Define Runtime Security*.



By default the permission is inherited from the parent, but you can decline or allow a web service operation itself as well, regardless of the parent web service interface.

8. Close without making changes.


3.9 Make Changes Available to SCM

In this exercise you will make your developed content/changes available to your team members by sending the changes to the SCM application.

You should only make changes available after you have tested these to work correctly. This is to ensure that your team members' work is not affected when they incorporate your changes.

NOTE

This only applies when your workspace is created using an SCM application.

1. If closed, open the Workspace Documents.
2. Click **Make Changes Available to Others** () in the toolbar.
3. Review the modified content.
4. Provide as comment **Developing Web Services**.
5. Click **Make Available**.

4. Optional Exercises

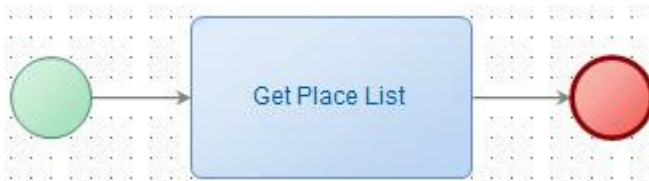
If time permits you can perform the optional exercises to extend your knowledge.

4.1 Exploring Web Service Operation Usage - Optional

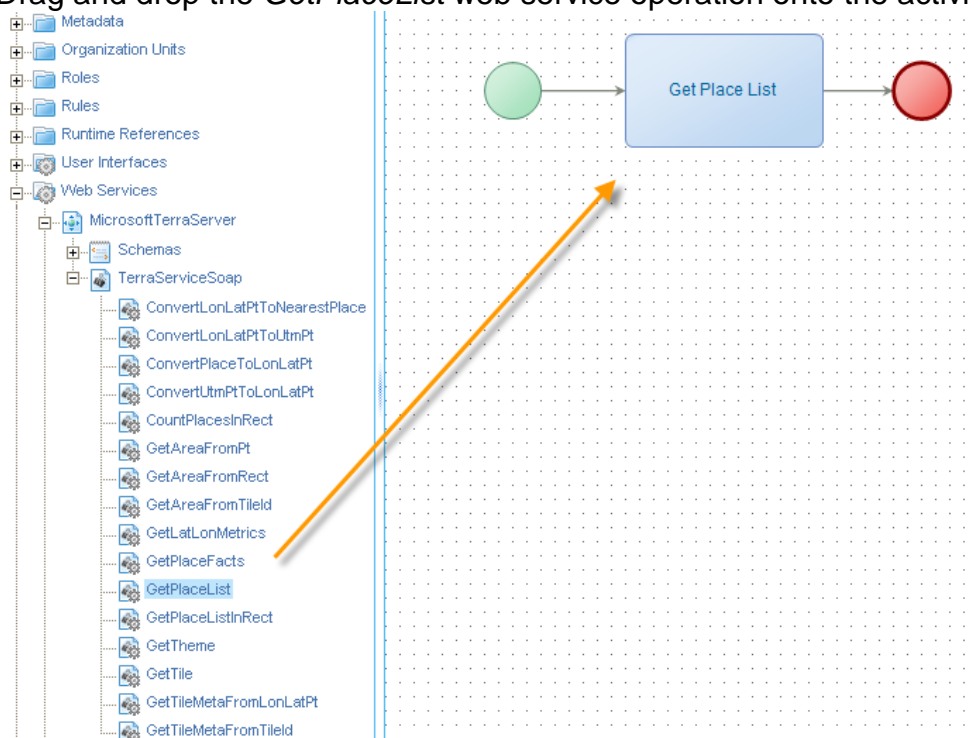
For every document in your project you can see which other documents are using it. For example a form is used in a BPM.

In this exercise you will add the *GetPlaceList* web service operation to the Demo process to study the used by option.

1. In *My Application Project* navigate to:
Business Process Models → *com* → *companyX* → *myapplication*.
2. Open the *Demo* process. (Created in module Application Management).
3. Add the following in the model area:
 - Start event
 - An activity named *Get Place List*
 - End event

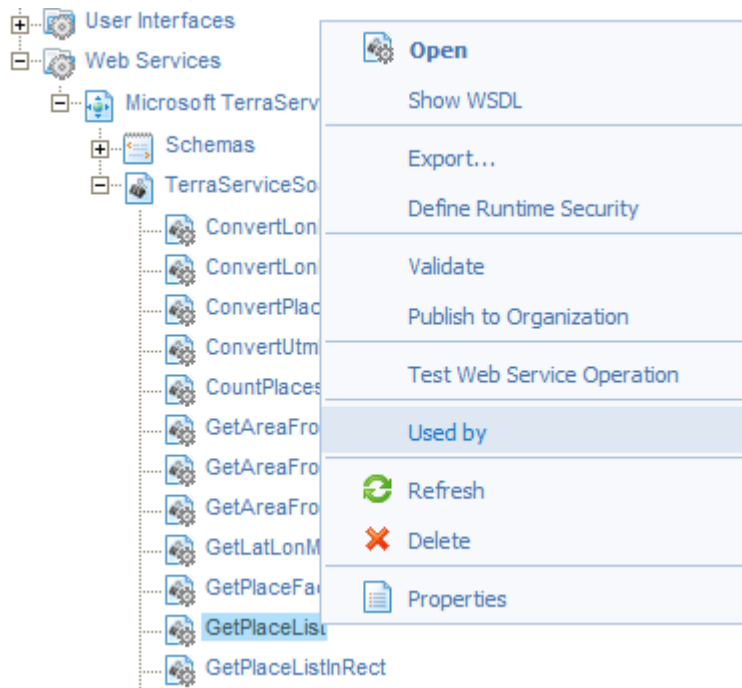


4. Select the *Workspace* tab.
5. Navigate to:
Web Services → *MicrosoftTerraServer* → *TerraServiceSoap*
6. Drag and drop the *GetPlaceList* web service operation onto the activity.



7. Save the Business Process Model.
8. Close the process editor.

9. In the tree, right click the *GetPlaceList* operation and select *Used By*.



10. Select the BPM and click **Open** (📁), to open the Business Process model from the *Used By* window.
11. Close the BPM Editor.
12. Close the *Used By* window.

4.2 Custom WS Operations - GetAllEmployees

1. In your Northwind WS-AppServer package, add a method to fetch all employees.

5. Learning Report

Achievements

- ☐ I know the concept of web services and metadata.
- ☐ I know a number of different approaches to build web services.
- ☐ I know the advantage of using web services with additional business logic.
- ☐ I can generate web services on a database.
- ☐ I can generate web services on business process models.
- ☐ I can generate web services on external services.

Notes