Classes Lesson 3.1 MD18

In the following step, the XML data source is analyzed to infer the required data structures of the Web service.

Web Service Interface

On this Page:

The Data Source

Below, the XML File **catalog.xml** containing product data is displayed. The Web service will read from this data source and send the filtered data to the caller.

```
 <catalog>
 <product>
 <title>Pirates of the Caribbean - The Curse of the Black Pearl</title>
 <category>DVD</category>
 <manufacturer>Buena Vista Home Video</manufacturer>
 <priceUSD>17.99</priceUSD>
 <seller>amazon.com</seller>
 <type>web</type>
 <link>http://www.amazon.com/exec/obidos/tg/detail/-/B00005JM5E
/qid=1075478699//ref=pd_ka_1/104-2088971-5001520?v=glance&amp;s=dvd&amp;
n=507846</link>
 </product>
 <product>
 <title>Finding Nemo</title>
 <category>DVD</category>
 <manufacturer>Walt Disney Home Video</manufacturer>
 <priceUSD>17.99</priceUSD>
 <seller>amazon.com</seller>
 <type>web</type>
 <link>http://www.amazon.com/exec/obidos/tg/detail/-/B00005JM02
/qid=1075478699//ref=pd_ka_2/104-2088971-5001520?v=glance&amp;s=dvd&amp;
n=507846</link>
 </product>
 <product>
 <title>Once Upon a Time in Mexico (2003)</title>
 <category>DVD</category>
 <manufacturer>Columbia Tristar Hom</manufacturer>
 <priceUSD>20.27</priceUSD>
 <seller>amazon.com</seller>
 <type>web</type>
 <link>http://www.amazon.com/exec/obidos/tg/detail/-/B0000WN140
/qid=1076055328/sr=1-2/ref=sr_1_2/102-9079676-4243362?v=glance&amp;s=dvd<
/link>
 </product>
 <product>
 <title>The Lord of the Rings - The Two Towers (Platinum Series Special
Extended Edition) (2002)</title>
 <category>DVD</category>
 <manufacturer>New Line Home Video</manufacturer>
 <priceUSD>25.99</priceUSD>
 <seller>amazon.com</seller>
 <type>web</type>
 <link>http://www.amazon.com/exec/obidos/tg/detail/-/B00009TB5G
/ref=pd_ts_d_8/102-9079676-4243362?v=glance&amp;s=dvd&amp;n=404276</link>
 </product>
 <product>
 <title>Lost In Translation (Widescreen Edition) (2003)</title>
 <category>DVD</category>
 <manufacturer>Universal Studios</manufacturer>
 <priceUSD>18.89</priceUSD>
 <seller>amazon.com</seller>
 <type>web</type>
 <link>http://www.amazon.com/exec/obidos/tg/detail/-/B00005JMJ4
/ref=pd_ts_d_2/102-9079676-4243362?v=glance&amp;s=dvd&amp;n=404276</link>
 </product>
 <product>
 <title>Panasonic SC-HT700 5-Disc Progressive-Scan DVD Home Theater System<
/title>
 <category>Audio & Video</category>
 <manufacturer>Panasonic</manufacturer>
 <priceUSD>264.99</priceUSD>
 <seller>amazon.com</seller>
 <type>web</type>
 <link>http://www.amazon.com/exec/obidos/tg/detail/-/B00008XL1I
/qid=1079949505/br=1-1/ref=br_lf_etk_ce_av__1//104-1167198-1926310?
v=glance&amp;s=electronics&amp;n=172593</link>
 </product>
 <product>
 <title>Zenith XBS344 Progressive Scan DVD-VCR Home Theater System (Silver)
</title>
 <category>Audio & Video</category>
 <manufacturer>Zenith</manufacturer>
 <priceUSD>249.99</priceUSD>
 <seller>amazon.com</seller>
 <type>web</type>
 <link>http://www.amazon.com/exec/obidos/tg/detail/-/B0000A2UAT
/qid=1079949505/br=1-13/ref=br_lf_etk_ce_av__13//104-1167198-1926310?
v=glance&amp;s=electronics&amp;n=172593</link>
 </product>
 <product>
 <title>Sony DVD/VHS Home Theater System (HT-V1000DP)</title>
 <category>Audio & Video</category>
 <manufacturer>Sony</manufacturer>
 <priceUSD>499.87</priceUSD>
 <seller>amazon.com</seller>
 <type>web</type>
 <link>http://www.amazon.com/exec/obidos/tg/detail/-/B0000CBCO6
/qid=1079950507/br=1-5/ref=br_lf_etk_ce_av__5//104-1167198-1926310?
v=glance&amp;s=electronics&amp;n=172593</link>
 </product>
 </catalog>
```

Later, you will model a data structure that stores the data of this file.

The XML file is well structured.

- The first line of the XML file is the header, which states what XML version and what character set are used.
- The element **catalog** contains some **product** elements.
- Each product element is containing seven further elements that store product data.

```
 <catalog>
 <product>
 <title>Pirates of the Caribbean - The Curse of the Black Pearl</title>
 <category>DVD</category>
 <manufacturer>Buena Vista Home Video</manufacturer>
 <priceUSD>17.99</priceUSD>
 <seller>amazon.com</seller>
 <type>web</type>
 <link>http://www.amazon.com/exec/obidos/tg/detail/-/B00005JM5E
/qid=1075478699//ref=pd_ka_1/104-2088971-5001520?v=glance&amp;s=dvd&amp;
n=507846</link>
 </product>

 [...]

 </catalog>
```

In this example, the XML structure is built up from XML elements without using attributes. It would also be possible to use attributes within the elements as shown in the example below.

```
<product title="Pirates of the Caribbean - The Curse of the Black
Pearl" category="DVD" manufacturer="Buena Vista Home Video" priceUSD="
17.99" seller="amazon.com" sellerType="web" link="http://www.amazon.com
/exec/obidos/tg/detail/-/B00005JM5E/qid=1075478699//ref=pd_ka_1/104-
2088971-5001520?v=glance&s=dvd&n=507846"/>
```

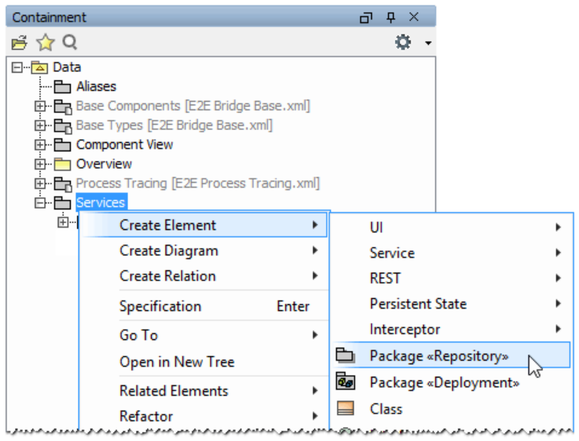
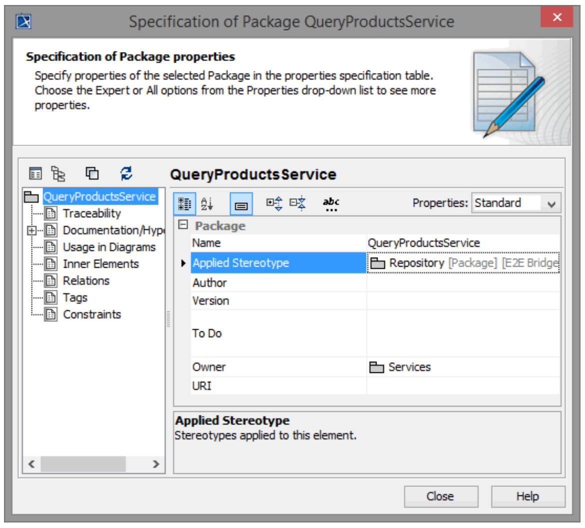
Defining Classes

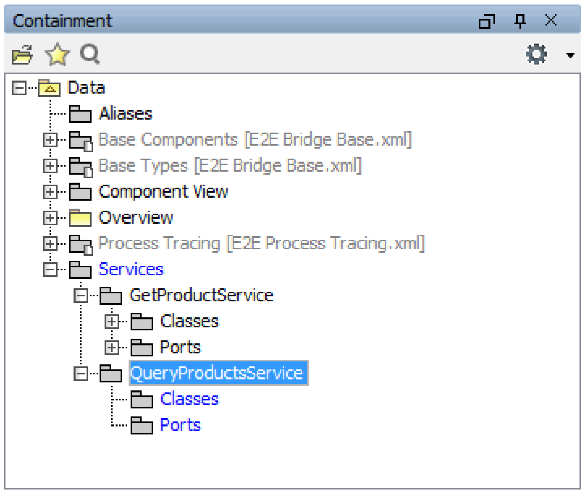
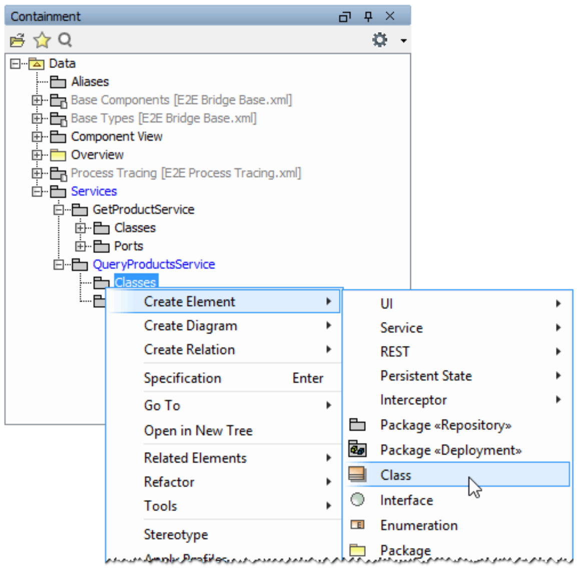
Now, you are going to model a class diagram that describes the data source file. The class diagram contains classes that represent the input and output structure of the service.

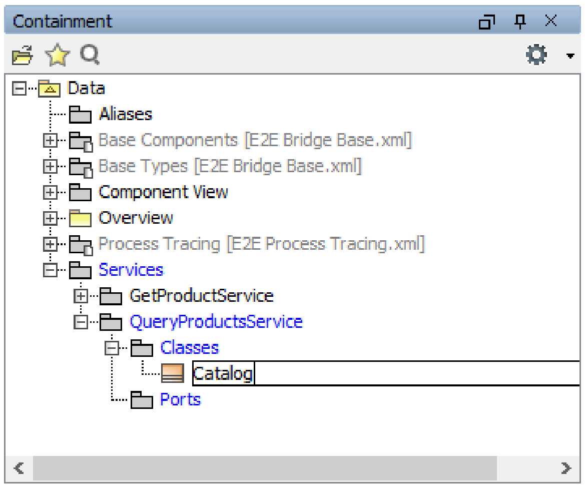
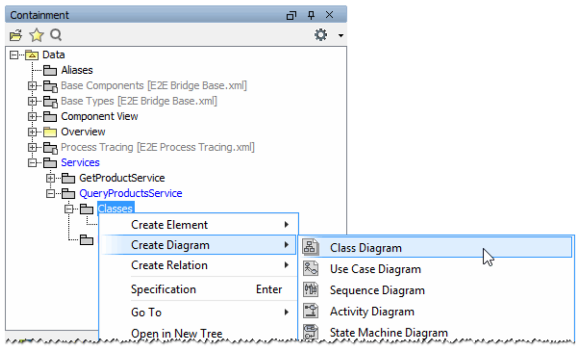
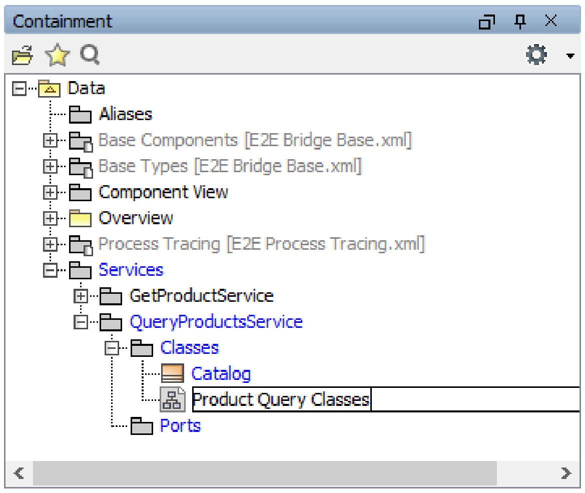
Output

You are going to start with the output classes according to the structure of the XML file.

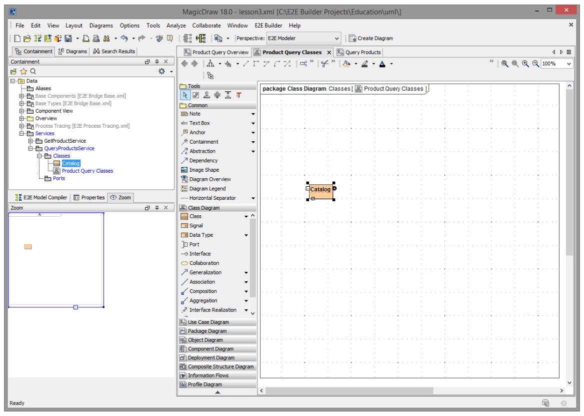
First, create a new service structure for the extended Web service.

 Containment Data Aliases Base Components [E2E Bridge Base.xml] Base Types [E2E Bridge Base.xml] Component View Overview Process Tracing [E2E Process Tracing.xml] Services Create Element Create Diagram Create Relation Specification Enter Go To Open in New Tree Related Elements Refactor UI Service REST Persistent State Interceptor Package «Repository» Package «Deployment» Class	Create a new package in the package **Data / Services** by choosing **Package <<Respository>>** from the context menu. Assign the name **QueryProductsService**.
 Specification of Package QueryProductsService **Specification of Package properties** Specify properties of the selected Package in the properties specification table. Choose the Expert or All options from the Properties drop-down list to see more properties. QueryProductsService Traceability Documentation/Hyp Usage in Diagrams Inner Elements Relations Tags Constraints Properties: Standard Package Name: QueryProductsService Applied Stereotype: Repository [Package] [E2E Bridge Author: Version: To Do: Owner: Services URI: **Applied Stereotype** Stereotypes applied to this element. 	To the newly created package **QueryProductsService**, the stereotype <<Repository>> has been applied automatically.

 Containment Data Aliases Base Components [E2E Bridge Base.xml] Base Types [E2E Bridge Base.xml] Component View Overview Process Tracing [E2E Process Tracing.xml] Services GetProductService Classes Ports QueryProductsService Classes Ports	In package **Query ProductsService**, create the new packages **Classes** and **Ports** with stereotype <<Repository>> using the same menu option.
 Containment Data Aliases Base Components [E2E Bridge Base.xml] Base Types [E2E Bridge Base.xml] Component View Overview Process Tracing [E2E Process Tracing.xml] Services GetProductService Classes Ports QueryProductsService Classes Create Element Create Diagram Create Relation Specification Enter Go To Open in New Tree Related Elements Refactor Tools Stereotype Apply Profiles UI Service REST Persistent State Interceptor Package «Repository» Package «Deployment» Class Interface Enumeration Package	Create a new class in the package **Data / Services / QueryProductsService / Classes** that, as a query result, will store a list of products.

 Containment Data Aliases Base Components [E2E Bridge Base.xml] Base Types [E2E Bridge Base.xml] Component View Overview Process Tracing [E2E Process Tracing.xml] Services GetProductService QueryProductsService Classes Catalog Ports	To find an appropriate name, look at the file structure of the XML file that was introduced at the beginning of this chapter. The first element in the file after the header is called **ca talog**. You find the opening tag of **cata log** at the beginning of the XML structure, and the closing tag at the end of it. Choose the same name for the class and, as class names should begin with a capital letter, type **Catalog**. An instance of this class will serve as output object in order to return the query result.
 Containment Data Aliases Base Components [E2E Bridge Base.xml] Base Types [E2E Bridge Base.xml] Component View Overview Process Tracing [E2E Process Tracing.xml] Services GetProductService QueryProductsService Classes Create Element Create Diagram Create Relation Specification Enter Go To Open in New Tree Class Diagram Use Case Diagram Sequence Diagram Activity Diagram State Machine Diagram	Next, create a new class diagram in the package **Class es** to visualize the data structures.
 Containment Data Aliases Base Components [E2E Bridge Base.xml] Base Types [E2E Bridge Base.xml] Component View Overview Process Tracing [E2E Process Tracing.xml] Services GetProductService QueryProductsService Classes Catalog Product Query Classes Ports	Assign the name **P roduct Query Classes**.

Drag and drop the class **Catalog** onto the class diagram in the diagram pane.


MagicDraw 18.0 - lesson3.xml [C:\E2E Builder Projects\Education\uml\]
package Class Diagram Classes [Product Query Classes]
Catalog

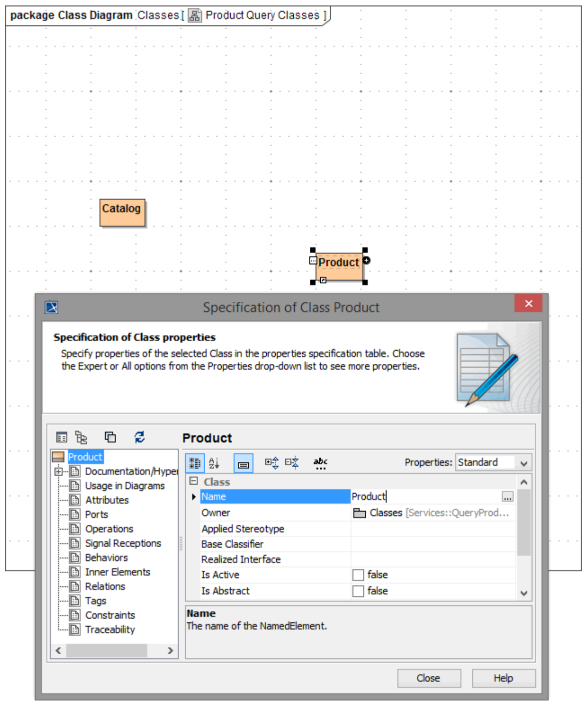
Now, create a class to store the data of a single product. Select a **Class** icon from the diagram toolbar and place it on the diagram pane. Double-click the class symbol to open the class specification dialog.

To find an appropriate name, look at the file structure of the XML file again.

```
<product>
 <title>Pirates of the Caribbean - The Curse of the Black Pearl</title>
 <category>DVD</category>
 <manufacturer>Buena Vista Home Video</manufacturer>
 <priceUSD>17.99</priceUSD>
 <seller>amazon.com</seller>
 <type>web</type>
 <link>http://www.amazon.com/exec/obidos
 /tg/detail//B00005JM5E/qid=1075478699/-
 /ref=pd_ka_1/104-2088971
 -5001520?v=glance&s=dvd&
 n=507846</link>
 </product>
```

The element in the file that represents a single product, and is repeated eight times, is called **product**. You find the opening tag of **product** at the beginning of each XML structure, and the closing tag at the end of it.

Choose the same name for the class (**Product**).

	Add the name **Product** in the Specification dialog and click **close**.

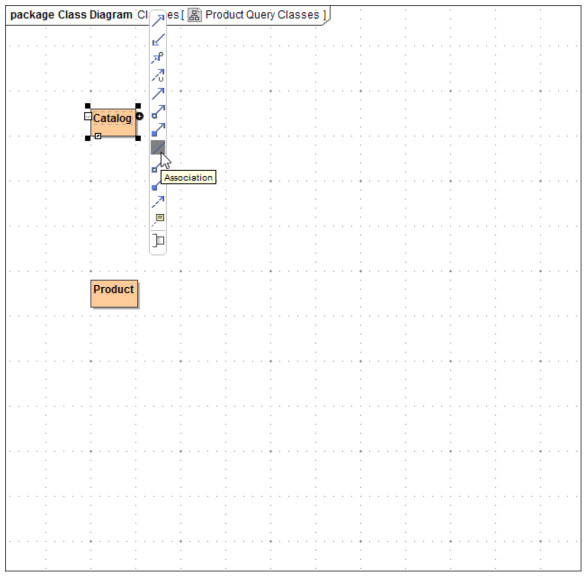
Specification of Class properties
Specify properties of the selected Class in the properties specification table. Choose the Expert or All options from the Properties drop-down list to see more properties.

Properties: Standard

Class

Name	Product
Owner	Classes [Services::QueryProd…
Applied Stereotype	
Base Classifier	
Realized Interface	
Is Active	☐ false
Is Abstract	☐ false

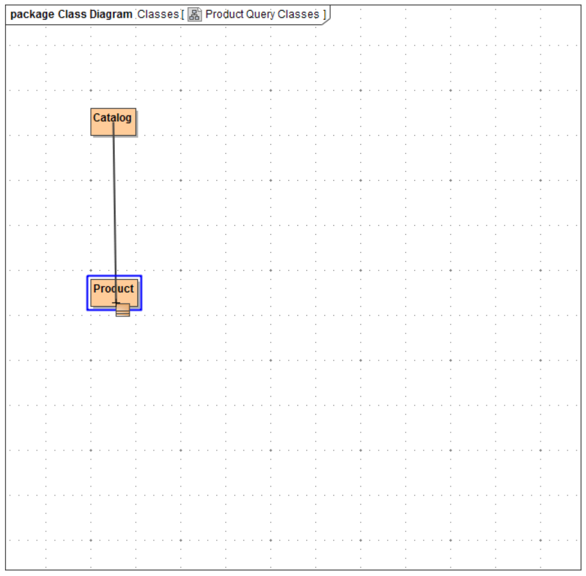
Name
The name of the NamedElement.

 package Class Diagram Cl es [Product Query Classes] Catalog Association Product	In order to map the XML structure to the data structure in the class diagram later on, you need to associate the class **Product** to the class **Catalog**.

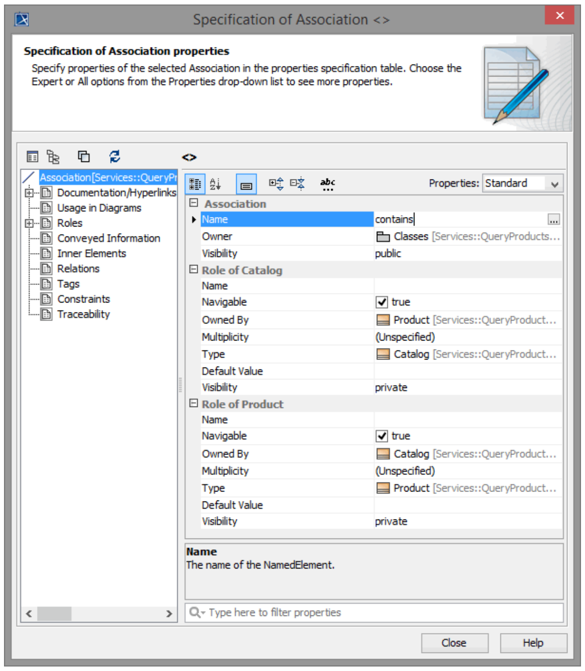
XML data is mapped to UML classes using an E2E Action Language operation. As a rule, **XML elements** are mapped to **association ends** in a class diagram.

An association is a relationship between instances of two classes. At least one of the association ends must have a name. They play a similar role as attribute names. Actually, all associations having an association end name can be represented as an attribute as well.

Click the class **Catalog** and select the **Association** icon from the smart manipulation toolbar. If the association symbol is not visible on the toolbar, click the black arrow on the bottom of the toolbar to expand it to full length.

 package Class Diagram Classes [Product Query Classes] Catalog Product	Draw the association from the class **Catalog** to the class **Product**.

As in the XML file there are stored several products, all **Product** elements have to be stored in an array. The array can be defined by configuring the association between the classes **Product** and **Catalog**. An association can be named, and the ends of an association (also called **roles**) can be adorned with role names, ownership indicators, multiplicity, visibility, and other properties.



Specification of Association <>

Specification of Association properties

Specify properties of the selected Association in the properties specification table. Choose the Expert or All options from the Properties drop-down list to see more properties.

Association[Services::QueryPr
- Documentation/Hyperlinks
- Usage in Diagrams
- Roles
- Conveyed Information
- Inner Elements
- Relations
- Tags
- Constraints
- Traceability

<>

Properties: Standard

Association	
Name	contains
Owner	Classes [Services::QueryProducts...
Visibility	public
Role of Catalog	
Name	
Navigable	✓ true
Owned By	Product [Services::QueryProduct...
Multiplicity	(Unspecified)
Type	Catalog [Services::QueryProduct...
Default Value	
Visibility	private
Role of Product	
Name	
Navigable	✓ true
Owned By	Catalog [Services::QueryProduct...
Multiplicity	(Unspecified)
Type	Product [Services::QueryProduct...
Default Value	
Visibility	private

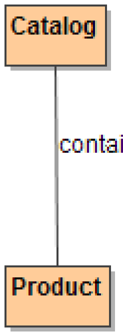
Name
The name of the NamedElement.

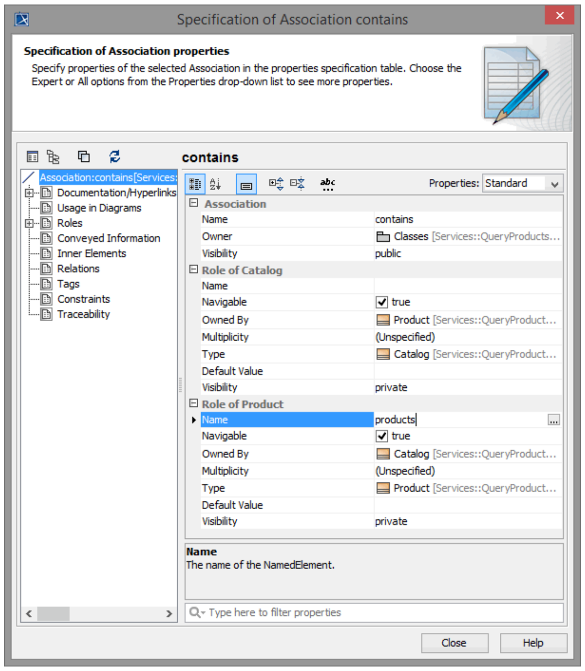
Type here to filter properties

Close Help

Double-click the association to open the **A ssociation** specification dialog.

Enter **contains** as name of the association, as a **C atalog contains Product** s. This increases readability. The small arrow floating next to the association line indicates the reading direction.





Specification of Association contains

Specification of Association properties

Specify properties of the selected Association in the properties specification table. Choose the Expert or All options from the Properties drop-down list to see more properties.

Association:contains[Services:
- Documentation/Hyperlinks
- Usage in Diagrams
- Roles
- Conveyed Information
- Inner Elements
- Relations
- Tags
- Constraints
- Traceability

contains

Properties: Standard

Association	
Name	contains
Owner	Classes [Services::QueryProducts...
Visibility	public
Role of Catalog	
Name	
Navigable	✓ true
Owned By	Product [Services::QueryProduct...
Multiplicity	(Unspecified)
Type	Catalog [Services::QueryProduct...
Default Value	
Visibility	private
Role of Product	
Name	products
Navigable	✓ true
Owned By	Catalog [Services::QueryProduct...
Multiplicity	(Unspecified)
Type	Product [Services::QueryProduct...
Default Value	
Visibility	private

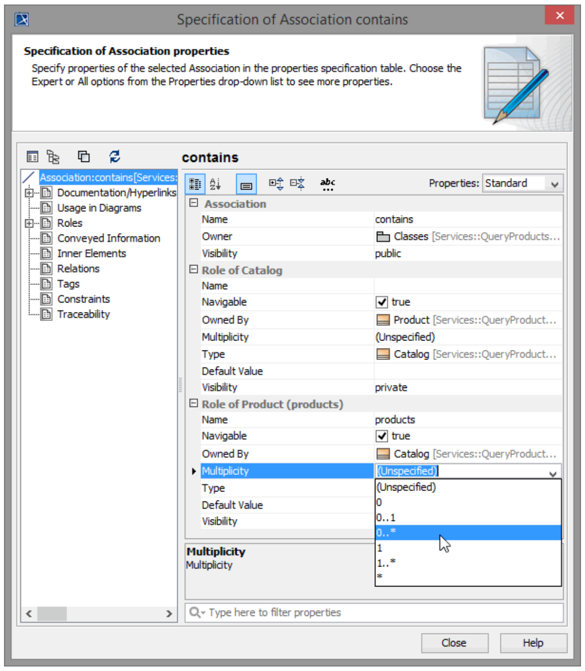
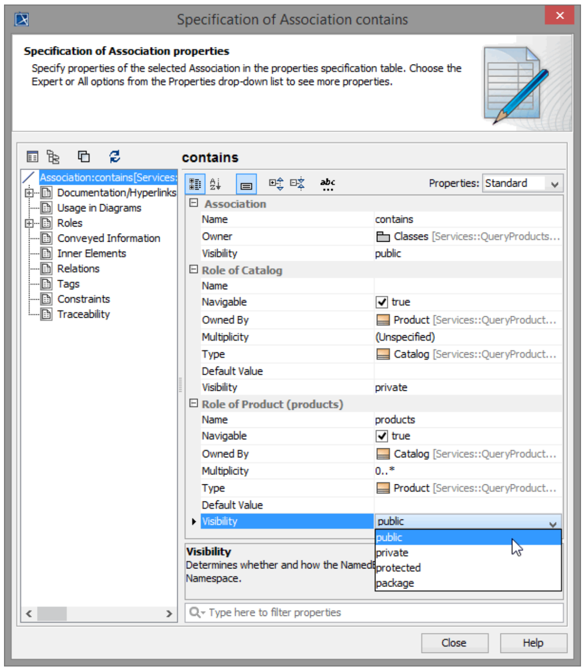
Name
The name of the NamedElement.

Type here to filter properties

Close Help

Enter the name **products** in the section **Role of Product** (which represents the association end belonging to class **Catalog**).

The roles (or association ends) are regarded as attributes of the corresponding class. Assigning them names, they can be accessed within **E2E Action Script** (as you will see later on).
In the next step, you will define **products** as an array containing zero to infinite **Product**s.

	By choosing **0..*** in the multiplicity field, you define that the new attribute **products** is an array. This array can have zero to infinite elements of type **Product** as a **Catalog contains** zero to infinite **Products**.
	The array **products** must be defined as **public**. Public attributes can be read and modified in activities outside the class context.

Specification of Association contains

Specification of Association properties

Specify properties of the selected Association in the properties specification table. Choose the Expert or All options from the Properties drop-down list to see more properties.

contains

- Association:contains[Services:
  - Documentation/Hyperlinks
  - Usage in Diagrams
  - Roles
  - Conveyed Information
  - Inner Elements
  - Relations
  - Tags
  - Constraints
  - Traceability

Properties: Standard

Association	
Name	contains
Owner	Classes [Services::QueryProducts...
Visibility	public
Role of Catalog	
Name	
Navigable	✓ true
Owned By	Product [Services::QueryProduct...
Multiplicity	(Unspecified)
Type	Catalog [Services::QueryProduct...
Default Value	
Visibility	private
Role of Product (products)	
Name	products
Navigable	✓ true
Owned By	Catalog [Services::QueryProduct...
Multiplicity	(Unspecified)
Type	
Default Value	
Visibility	

(Unspecified)
0
0..1
0..*
1
1..*
*

Multiplicity
Multiplicity

Type here to filter properties

Close Help

Specification of Association contains

Specification of Association properties

Specify properties of the selected Association in the properties specification table. Choose the Expert or All options from the Properties drop-down list to see more properties.

contains

- Association:contains[Services:
  - Documentation/Hyperlinks
  - Usage in Diagrams
  - Roles
  - Conveyed Information
  - Inner Elements
  - Relations
  - Tags
  - Constraints
  - Traceability

Properties: Standard

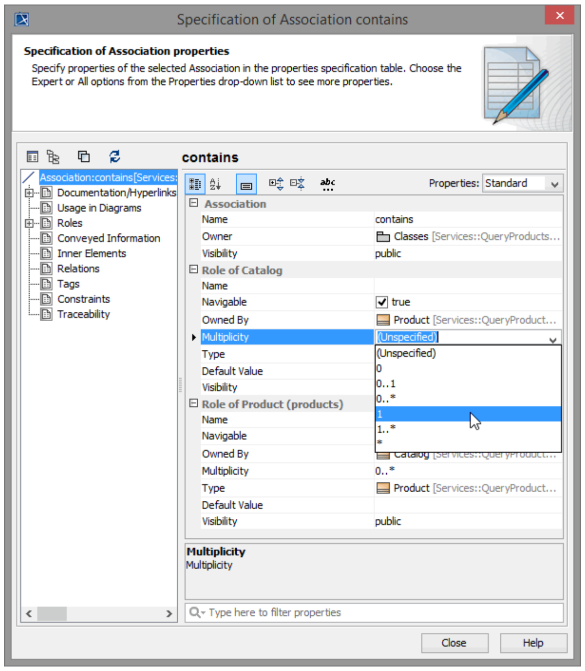
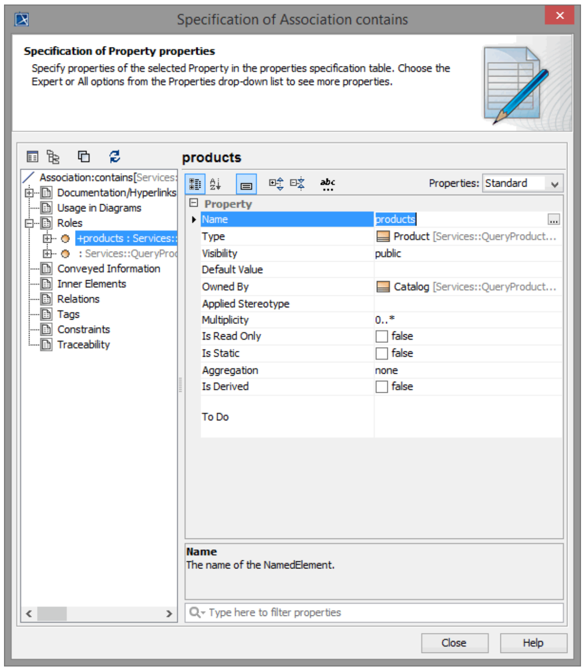
Association	
Name	contains
Owner	Classes [Services::QueryProducts...
Visibility	public
Role of Catalog	
Name	
Navigable	✓ true
Owned By	Product [Services::QueryProduct...
Multiplicity	(Unspecified)
Type	Catalog [Services::QueryProduct...
Default Value	
Visibility	private
Role of Product (products)	
Name	products
Navigable	✓ true
Owned By	Catalog [Services::QueryProduct...
Multiplicity	0..*
Type	Product [Services::QueryProduct...
Default Value	
Visibility	public

public
private
protected
package

Visibility
Determines whether and how the NamedE Namespace.

Type here to filter properties

Close Help

	In the section **Role of Catalog** (which belongs to class **Product**) specify multiplicity **1** as exactly one instance of **Catalog** can have an array containing **0..*** **Products**.
	In the association properties tree on the left, expand the node **Roles** and select role **products**.

Specification of Association contains

Specification of Association properties

Specify properties of the selected Association in the properties specification table. Choose the Expert or All options from the Properties drop-down list to see more properties.

contains

Association:contains[Services:
Documentation/Hyperlinks
Usage in Diagrams
Roles
Conveyed Information
Inner Elements
Relations
Tags
Constraints
Traceability

Properties: Standard

Association	
Name	contains
Owner	Classes [Services::QueryProducts...
Visibility	public
Role of Catalog	
Name	
Navigable	✓ true
Owned By	Product [Services::QueryProduct...
Multiplicity	(Unspecified)
Type	
Default Value	
Visibility	
Role of Product (products)	
Name	
Navigable	
Owned By	Catalog [Services::QueryProduct...
Multiplicity	0..*
Type	Product [Services::QueryProduct...
Default Value	
Visibility	public

(Unspecified)
0
0..1
0..*
1
1..*
*

Multiplicity
Multiplicity

Type here to filter properties

Close Help

Specification of Association contains

Specification of Property properties

Specify properties of the selected Property in the properties specification table. Choose the Expert or All options from the Properties drop-down list to see more properties.

products

Association:contains[Services:
Documentation/Hyperlinks
Usage in Diagrams
Roles
+products : Services::
: Services::QueryPro
Conveyed Information
Inner Elements
Relations
Tags
Constraints
Traceability

Properties: Standard

Property	
Name	products
Type	Product [Services::QueryProduct...
Visibility	public
Default Value	
Owned By	Catalog [Services::QueryProduct...
Applied Stereotype	
Multiplicity	0..*
Is Read Only	false
Is Static	false
Aggregation	none
Is Derived	false
To Do	

Name
The name of the NamedElement.

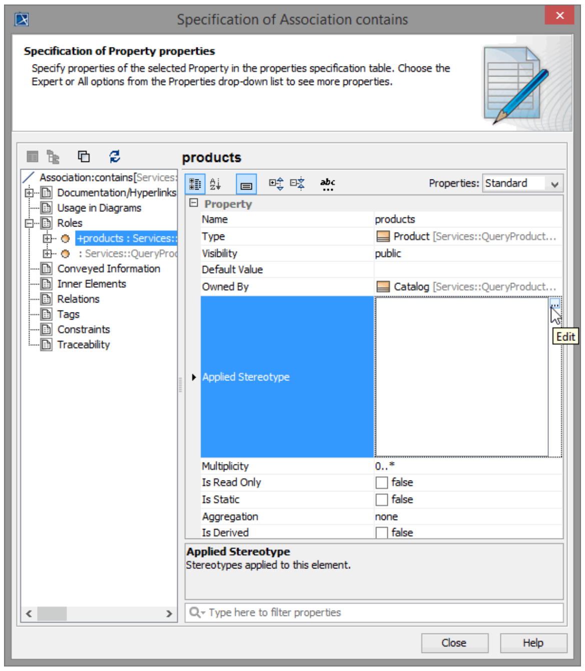
Type here to filter properties

Close Help

As mentioned before, you will map the XML structure to the class structure, after reading the XML file containing the product catalog.

For a better readability of the model, you named the association end **products**, instead of product in singular. But actually, the name of the corresponding association end has to match the XML element's name to perform the mapping. On account of this, you have to specify the external name of **products**.

This is done by assigning the stereotype <<E2EAttribute>> to the association end **products**.



Specification of Association contains

Specification of Property properties

Specify properties of the selected Property in the properties specification table. Choose the Expert or All options from the Properties drop-down list to see more properties.

- Association:contains[Services:
  - Documentation/Hyperlinks
  - Usage in Diagrams
  - Roles
    - +products : Services::
    - : Services::QueryProc
  - Conveyed Information
  - Inner Elements
  - Relations
  - Tags
  - Constraints
  - Traceability

products

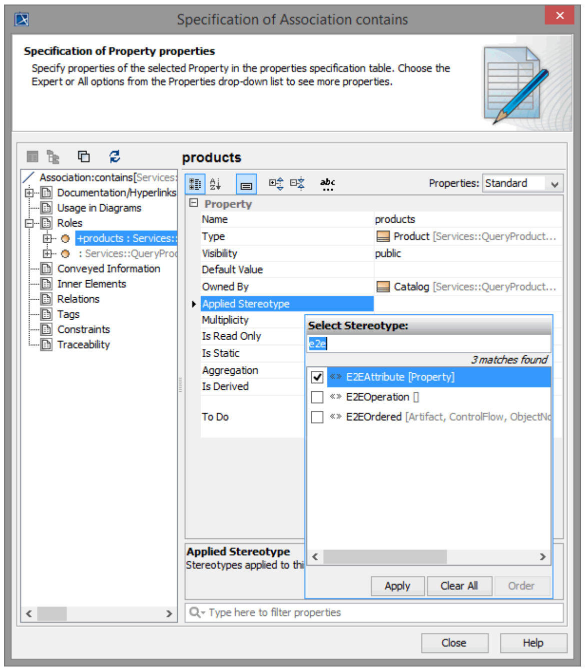
Properties: Standard

Property	
Name	products
Type	Product [Services::QueryProduct…
Visibility	public
Default Value	
Owned By	Catalog [Services::QueryProduct…
Applied Stereotype	
Multiplicity	0..*
Is Read Only	false
Is Static	false
Aggregation	none
Is Derived	false

Applied Stereotype
Stereotypes applied to this element.

Type here to filter properties

Click into the field **Applied Stereotype**, and then on the small edit button to open a list of suggested stereotypes.



Specification of Association contains

Specification of Property properties

Specify properties of the selected Property in the properties specification table. Choose the Expert or All options from the Properties drop-down list to see more properties.

- Association:contains[Services:
  - Documentation/Hyperlinks
  - Usage in Diagrams
  - Roles
    - +products : Services::
    - : Services::QueryProc
  - Conveyed Information
  - Inner Elements
  - Relations
  - Tags
  - Constraints
  - Traceability

products

Properties: Standard

Property	
Name	products
Type	Product [Services::QueryProduct…
Visibility	public
Default Value	
Owned By	Catalog [Services::QueryProduct…
Applied Stereotype	
Multiplicity	
Is Read Only	
Is Static	
Aggregation	
Is Derived	
To Do	

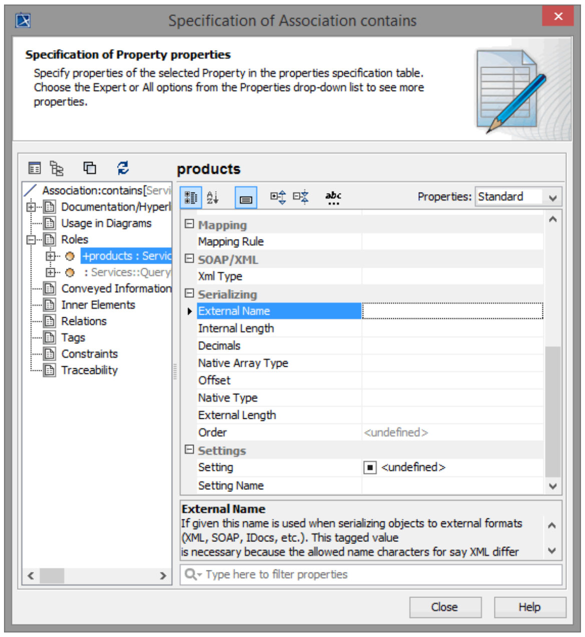
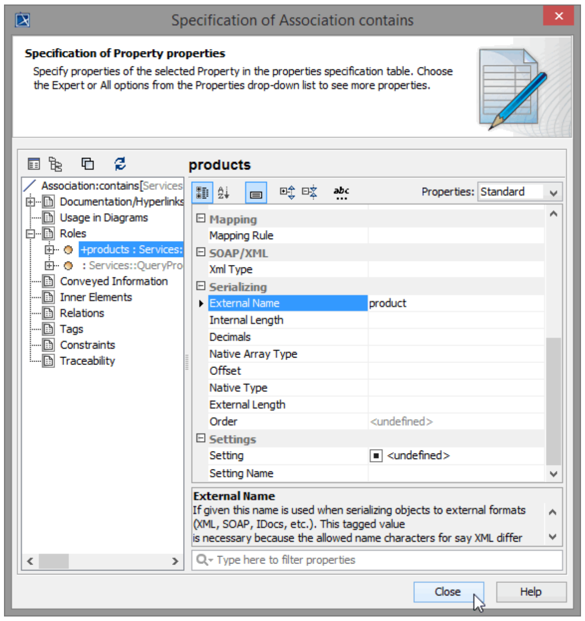
Select Stereotype:
e2e
3 matches found
- [✓] E2EAttribute [Property]
- [ ] E2EOperation []
- [ ] E2EOrdered [Artifact, ControlFlow, ObjectNo

Applied Stereotype
Stereotypes applied to thi

Type here to filter properties

Filter the list by entering **e2e** into the text field and select the stereotype <<E2E Attribute>>.

Click **Apply** to assign the stereotype.

	Assigning the stereotype <<E2E Attribute>> led to the properties list being appended by several additional properties. They are tagged values belonging to the stereotype. If necessary, scroll down to the end of the properties list to see the additional properties.
	Specify the external name of **products** as **product**, the corresponding name from the XML file. Click **Close**.

Specification of Association contains

Specification of Property properties

Specify properties of the selected Property in the properties specification table. Choose the Expert or All options from the Properties drop-down list to see more properties.

products

Properties: Standard

Mapping	
Mapping Rule	
SOAP/XML	
Xml Type	
Serializing	
External Name	
Internal Length	
Decimals	
Native Array Type	
Offset	
Native Type	
External Length	
Order	<undefined>
Settings	
Setting	<undefined>
Setting Name	

External Name
If given this name is used when serializing objects to external formats (XML, SOAP, IDocs, etc.). This tagged value is necessary because the allowed name characters for say XML differ

Specification of Association contains

Specification of Property properties

Specify properties of the selected Property in the properties specification table. Choose the Expert or All options from the Properties drop-down list to see more properties.

products

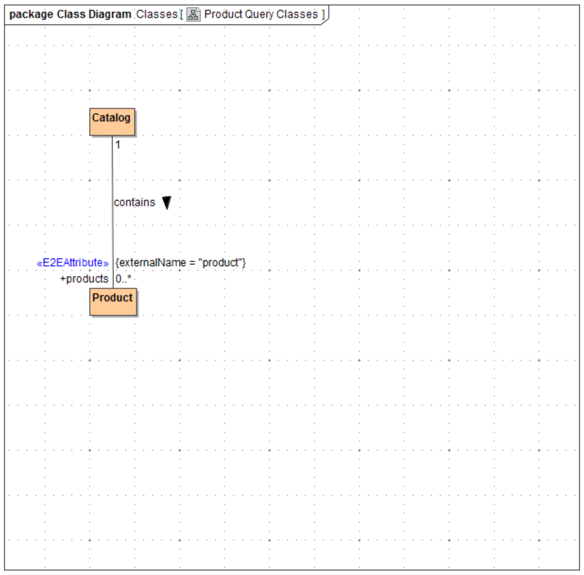
Properties: Standard

Mapping	
Mapping Rule	
SOAP/XML	
Xml Type	
Serializing	
External Name	product
Internal Length	
Decimals	
Native Array Type	
Offset	
Native Type	
External Length	
Order	<undefined>
Settings	
Setting	<undefined>
Setting Name	

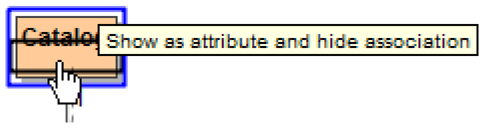
External Name
If given this name is used when serializing objects to external formats (XML, SOAP, IDocs, etc.). This tagged value is necessary because the allowed name characters for say XML differ

Now, the class diagram should look as shown below.

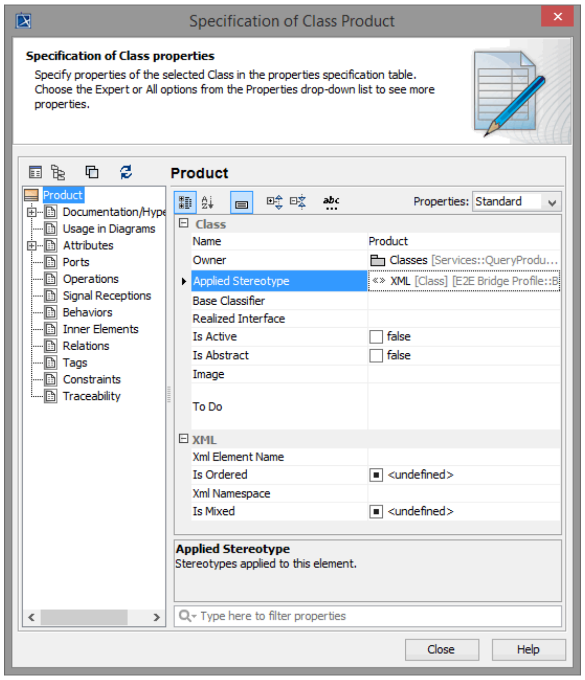
One **Catalog contains** zero to infinite **products**. The external name of **products** is **product**. Association end **products** constitutes an attribute of class **Catalog**. It is an array that contains elements of type **Product**.


package Class Diagram Classes [Product Query Classes]
Catalog
1
contains
«E2EAttribute»
{externalName = "product"}
+products
0..*
Product

Comparing the association to the XML structure, you will notice the analogy when saying that instances of class **Catalog** contain **Product**s. The data of all **product** XML elements will be mapped to the association end **products** (**products** is an array attribute of class **Catalog**).

	Class attributes and association ends are different notations for the same issue. You could drag and drop the association end **pr oducts** to class **catalog** to change the notation.

In the next steps, you will map each XML element that is included in an XML element **product** to new attributes of the class **Product**. For instance, the XML element **manufacturer** will be mapped to attribute **manufacturer** of class **Product**.

	Double-click class **Product** to open the Specification Dialog and apply the stereotype <<X ML>>.

Specification of Class Product

Specification of Class properties
Specify properties of the selected Class in the properties specification table.
Choose the Expert or All options from the Properties drop-down list to see more properties.

Product

Product
- Documentation/Hype
- Usage in Diagrams
- Attributes
- Ports
- Operations
- Signal Receptions
- Behaviors
- Inner Elements
- Relations
- Tags
- Constraints
- Traceability

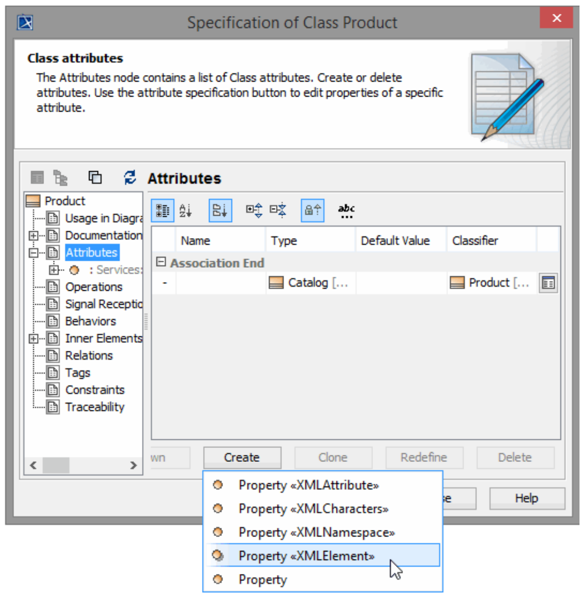
Properties: Standard

Class	
Name	Product
Owner	Classes [Services::QueryProdu…
Applied Stereotype	XML [Class] [E2E Bridge Profile::B
Base Classifier	
Realized Interface	
Is Active	☐ false
Is Abstract	☐ false
Image	
To Do	
XML	
Xml Element Name	
Is Ordered	▪ <undefined>
Xml Namespace	
Is Mixed	▪ <undefined>

Applied Stereotype
Stereotypes applied to this element.

Type here to filter properties

Close Help



Class attributes

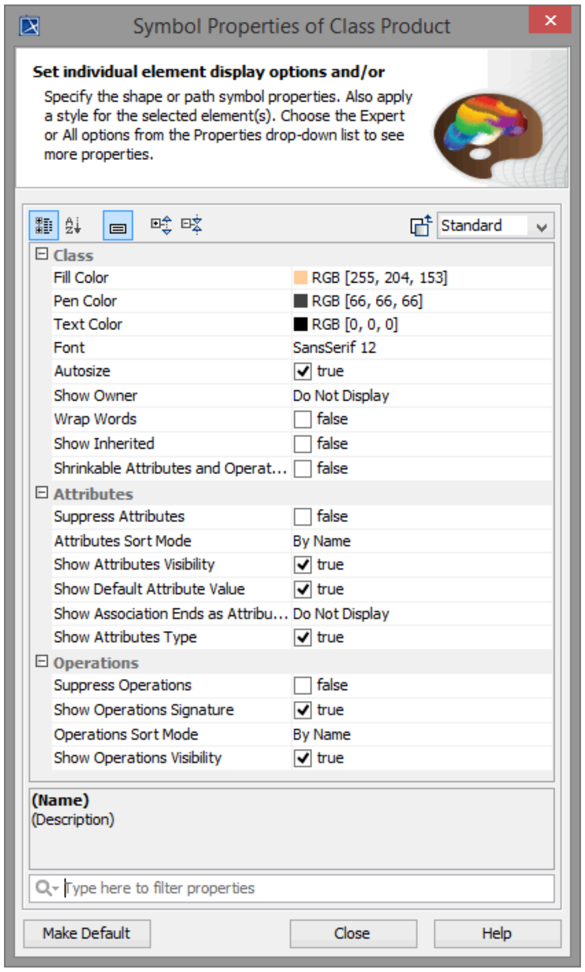
The Attributes node contains a list of Class attributes. Create or delete attributes. Use the attribute specification button to edit properties of a specific attribute.

Name	Type	Default Value	Classifier
Association End			
-	Catalog [...		Product [...

Switch to the **Attri butes** section and add the attributes listed in the table below using **Creat e > Property <<XMLElement >>**.

Attribute Name	Attribute Type	Visibility	Stereotype
title	String	public	<<XMLElement>>
category	String	public	<<XMLElement>>
manufacturer	String	public	<<XMLElement>>
priceUSD	Float	public	<<XMLElement>>
seller	String	public	<<XMLElement>>
type	String	public	<<XMLElement>>
link	String	public	<<XMLElement>>

You use stereotypes like <<XMLElement>> to control how XML data is mapped to UML classes. For instance, using this stereotype for class attribute **manufacturer**, the attribute does not need to be defined on an association end. You will find more detailed information about data mapping between XML documents and UML classes in the xUML Services Reference Guide.



Symbol Properties of Class Product

Set individual element display options and/or

Specify the shape or path symbol properties. Also apply a style for the selected element(s). Choose the Expert or All options from the Properties drop-down list to see more properties.

Standard

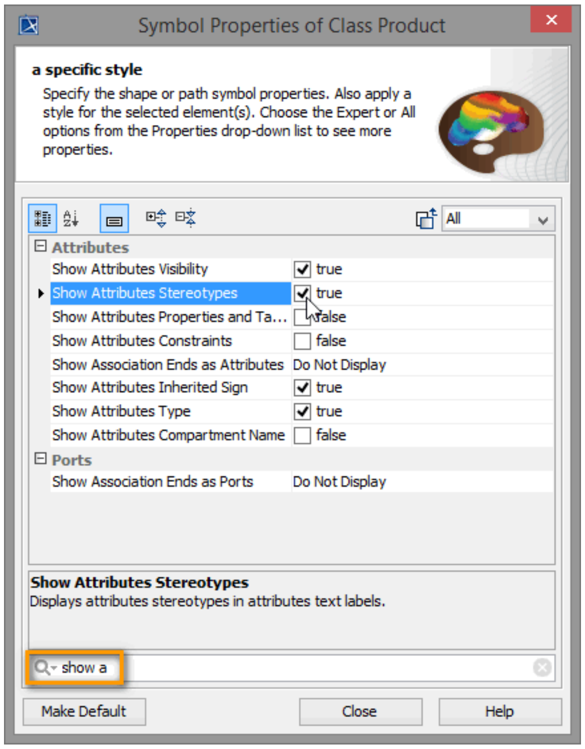
Class	
Fill Color	RGB [255, 204, 153]
Pen Color	RGB [66, 66, 66]
Text Color	RGB [0, 0, 0]
Font	SansSerif 12
Autosize	✓ true
Show Owner	Do Not Display
Wrap Words	false
Show Inherited	false
Shrinkable Attributes and Operat…	false
Attributes	
Suppress Attributes	false
Attributes Sort Mode	By Name
Show Attributes Visibility	✓ true
Show Default Attribute Value	✓ true
Show Association Ends as Attribu…	Do Not Display
Show Attributes Type	✓ true
Operations	
Suppress Operations	false
Show Operations Signature	✓ true
Operations Sort Mode	By Name
Show Operations Visibility	✓ true

(Name)
(Description)

Type here to filter properties

Make Default | Close | Help

To make the stereotype <<XMLElement>> visible on the diagram pane, right-click class **Product** and select **Symbol Properties** from the context menu.



Symbol Properties of Class Product

a specific style

Specify the shape or path symbol properties. Also apply a style for the selected element(s). Choose the Expert or All options from the Properties drop-down list to see more properties.

All

Attributes	
Show Attributes Visibility	✓ true
Show Attributes Stereotypes	✓ true
Show Attributes Properties and Ta…	false
Show Attributes Constraints	false
Show Association Ends as Attributes	Do Not Display
Show Attributes Inherited Sign	✓ true
Show Attributes Type	✓ true
Show Attributes Compartment Name	false
Ports	
Show Association Ends as Ports	Do Not Display

Show Attributes Stereotypes
Displays attributes stereotypes in attributes text labels.

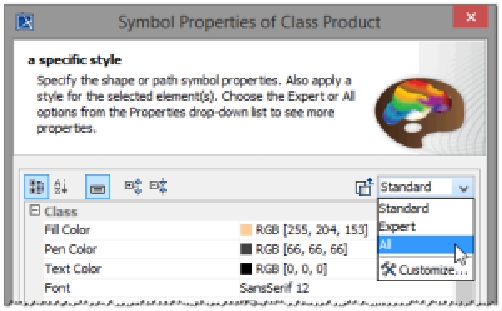
show a

Make Default | Close | Help

In the search field start typing **show a** to filter the list. Set the parameter **Show Attributes Stereotypes** to **true** .

Click **Close**.

If the option **Show Attributes Stereotypes** is not displayed, change the drop-down list on top of the dialog from **Standard** to **All**.



Symbol Properties of Class Product

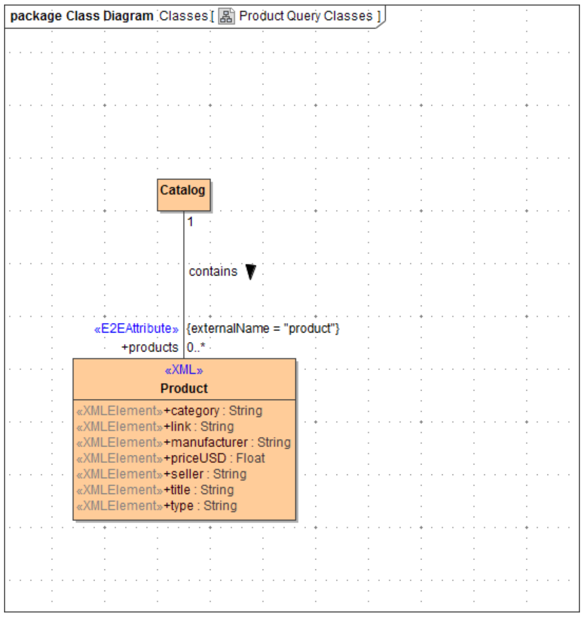
a specific style

Specify the shape or path symbol properties. Also apply a style for the selected element(s). Choose the Expert or All options from the Properties drop-down list to see more properties.

Standard: Standard / Expert / All / Customize...

Class	
Fill Color	RGB [255, 204, 153]
Pen Color	RGB [66, 66, 66]
Text Color	RGB [0, 0, 0]
Font	SansSerif 12

Now, your class diagram should look like the one below.


package Class Diagram Classes[Product Query Classes]
Catalog
1
contains
«E2EAttribute» {externalName = "product"}
+products 0..*
«XML»
Product
«XMLElement»+category : String
«XMLElement»+link : String
«XMLElement»+manufacturer : String
«XMLElement»+priceUSD : Float
«XMLElement»+seller : String
«XMLElement»+title : String
«XMLElement»+type : String

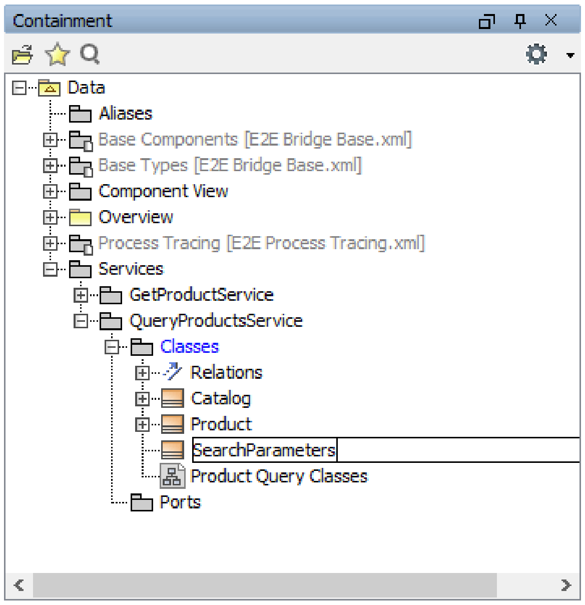
In more complex examples that require to map complex XML data to UML classes and vice versa, you would import an XML schema into the model. The E2E XSD Importer creates a UML model with all classes having all required stereotypes and associations.

You modeled the output data according to the structure of the XML file by creating the classes **Catalog** and **Product**. The XML elements of each product record were modeled as stereotyped attributes of class **Product**.

Save the UML model.

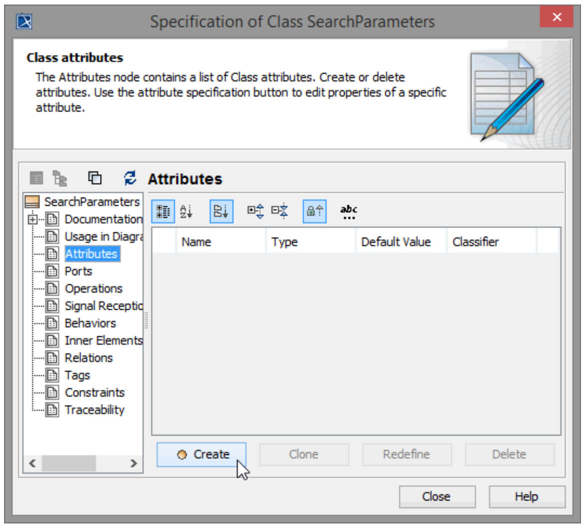
Input

Now, define the structure of the input data.



Create a new class in the package **Data / Services / QueryProductsService / Classes** that will contain the title and the search keywords.

Assign the name **SearchParameters**.



Class attributes

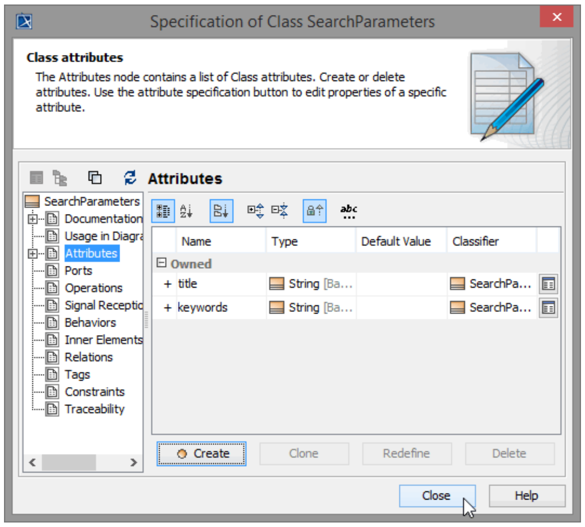
The Attributes node contains a list of Class attributes. Create or delete attributes. Use the attribute specification button to edit properties of a specific attribute.

Double-click the class to open the class specification dialog and change to the tab **Attributes** .

Now, define the attributes of the new class according to the input data.

Add all attributes listed in the table below.

Attribute Name	Attribute Type	Multiplicity	Visibility
title	String		public
keywords	String	0..*	public



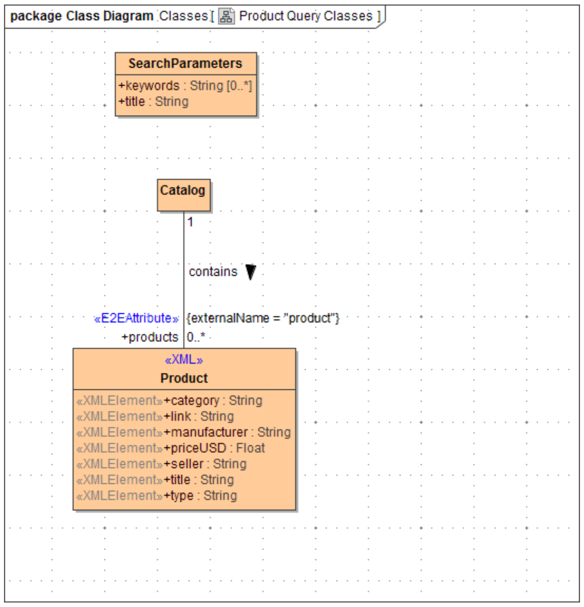
Class attributes

The Attributes node contains a list of Class attributes. Create or delete attributes. Use the attribute specification button to edit properties of a specific attribute.

Name	Type	Default Value	Classifier
Owned			
+ title	String [Ba...		SearchPa...
+ keywords	String [Ba...		SearchPa...

After all attributes have been defined, click **Close** .

Drag and drop the class **SearchParameters** onto the class diagram **Product Query Classes**.


package Class Diagram Classes [Product Query Classes]
SearchParameters
+keywords : String [0..*]
+title : String
Catalog
1
contains
«E2EAttribute»
{externalName = "product"}
+products
0..*
«XML»
Product
«XMLElement»+category : String
«XMLElement»+link : String
«XMLElement»+manufacturer : String
«XMLElement»+priceUSD : Float
«XMLElement»+seller : String
«XMLElement»+title : String
«XMLElement»+type : String

Save the UML model.