

# Tutorial



Copyright ©1998-2005 Altova GmbH. All rights reserved. Use of this software is governed by and subject to an Altova software license agreement. XMLSpy, MapForce, StyleVision, SemanticWorks, SchemaAgent, UModel, DiffDog, Authentic, AltovaXML, and ALTOVA are trademarks and/or registered trademarks of Altova GmbH.

**ALTOVA®**

XML, XSL, XHTML, and W3C are trademarks (registered in numerous countries) of the World Wide Web Consortium; marks of the W3C are registered and held by its host institutions, MIT, INRIA, and Keio. UNICODE and the Unicode Logo are trademarks of Unicode, Inc. This software contains material that is ©1994-1998 Dundas Software Ltd., all rights reserved. The Sentry Spelling Checker Engine - ©2000 Wintertree Software Inc., STPort ©1999,2000 Boris Fomitchev, ©1994 Hewlett-Packard Company, ©1996,97 Silicon Graphics Computer Systems, Inc., ©1997 Moscow Center for SPARC Technology. Scintilla Copyright ©1998-2002 by Neil Hodgson <neilh@scintilla.org>. ANTLR Copyright ©1989-2005 by Terence Parr <www.antlr.org>. All other names or trademarks are the property of their respective owners.

## **Altova XMLSpy Tutorial (for Home Edition)**

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Published: 2006

© 2006 Altova GmbH

---

## Table of Contents

<b>1</b>	<b>The interface</b>	<b>2</b>
<b>2</b>	<b>Creating a schema from scratch</b>	<b>3</b>
2.1	Creating a new Schema file .....	4
2.2	Adding elements to a schema .....	7
2.3	Adding elements using drag and drop .....	10
2.4	Completing the basic schema .....	11
<b>3</b>	<b>Making schema components reusable</b>	<b>14</b>
3.1	Creating and extending global components .....	15
3.2	References, attributes and enumerations .....	20
3.3	Navigation shortcuts in schema documents .....	23
<b>4</b>	<b>Creating an XML document</b>	<b>25</b>
4.1	Creating a new XML file .....	26
4.2	Editing in Authentic View .....	28
4.3	Editing in Text View .....	30
4.4	Validating and entering data .....	33
4.5	Manipulating data with Entry Helpers .....	35
<b>5</b>	<b>XSL Transformation</b>	<b>36</b>
5.1	Assigning an XSL file .....	37
5.2	Transforming the XML file .....	38
5.3	Modifying the XSL file .....	39
	<b>Index</b>	<b>41</b>

## XMLSpy Home Edition Tutorial

This tutorial gives a short overview of XML, and takes you through several tasks which provide an overview of how to use XMLSpy to its fullest.

You will learn how to:

- Create a **simple schema** from scratch
- **Generalize** the schema using simple and complex types
- **Validate** the XML document against its schema
- **Transform** the XML document into HTML using XSLT, and view the result in the Browser view

### XMLSpy installation and configuration

This tutorial assumes that you have successfully installed XMLSpy on your computer as a registered user, or you have received a free evaluation key-code for XMLSpy.

The evaluation version of XMLSpy is fully functional but time-limited to 30 days. You can request a regular license from our secure web server or through any one of our resellers.

### Tutorial example files

The tutorial files are available in the `..\Examples\Tutorial` folder.

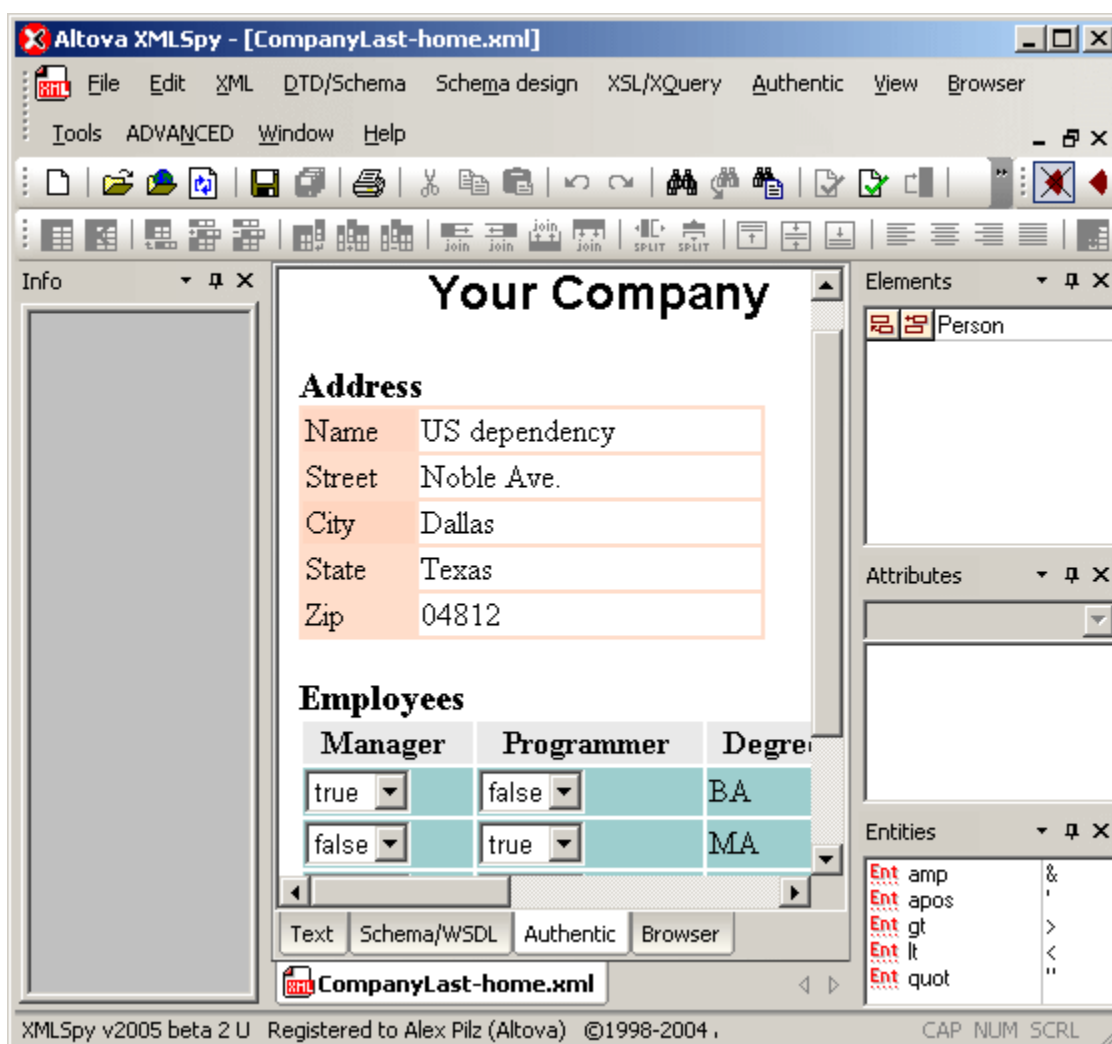
The **Examples folder** contains various XML files for you to experiment with, while the **Tutorial** folder contains all the files used in this tutorial.

The **Template folder** contains all the XML template files that are used whenever you select the menu option **File | New**. These files supply the necessary data (namespaces and XML declarations) for you to start working with the respective XML document immediately.

## 1 The interface

XMLSpy provides several windows that show various aspects of your XML document:

- The left area consists of the **Info** window.
- The central area, called **Main** window, is where you edit and view all types of XML documents.  
You can choose from different views: Text view, Schema/WSDL view, Authentic View or Browser view.
- The right area contains the three **Entry helper** windows which allow you to insert or append: elements, attributes, and entities.



## 2 Creating a schema from scratch

A Schema describes what one or more XML documents can look like, and defines:

- The elements the document contains, and the order in which they appear
- The element content, and element attributes if any

The purpose of a schema is to allow machine validation of document structure. Instead of using the syntax of XML 1.0 DTD declarations, schema definitions use XML element syntax. A correct XML schema definition is, therefore, a well-formed XML document.

### Goal of this section:

The goal of this section is to **create a simple schema** describing a company and its employees. The company is to consist of an **address** and an unlimited number of **persons**.

This will be achieved by:

- **Adding** elements to the schema
- Defining element **sequences**
- Adding **sub-elements** to an element (child elements)
- Creating elements using **drag and drop**
- Making an element **optional**
- Defining an element **facet**

Functions (and their icons) in this section:



**File | New**, creates a new XML instance file.



**Schema design | Display diagram** displays the content model of the selected global component in the top part of the main window. To display the content model of a component, click the "Display diagram" icon located to the left of each component in the "Display all globals" view of the Schema overview. The "Display diagram" function toggles with the "Display all globals" function.



**Schema design | Display all globals** displays all global components of the schema in the top part of the main window. The "Display all globals" function toggles with the "Display diagram" function.

**TAB** Takes you to the next field and automatically opens a drop-down list if one exists.

**CTRL + Drag&Drop**, enables you to copy existing elements.

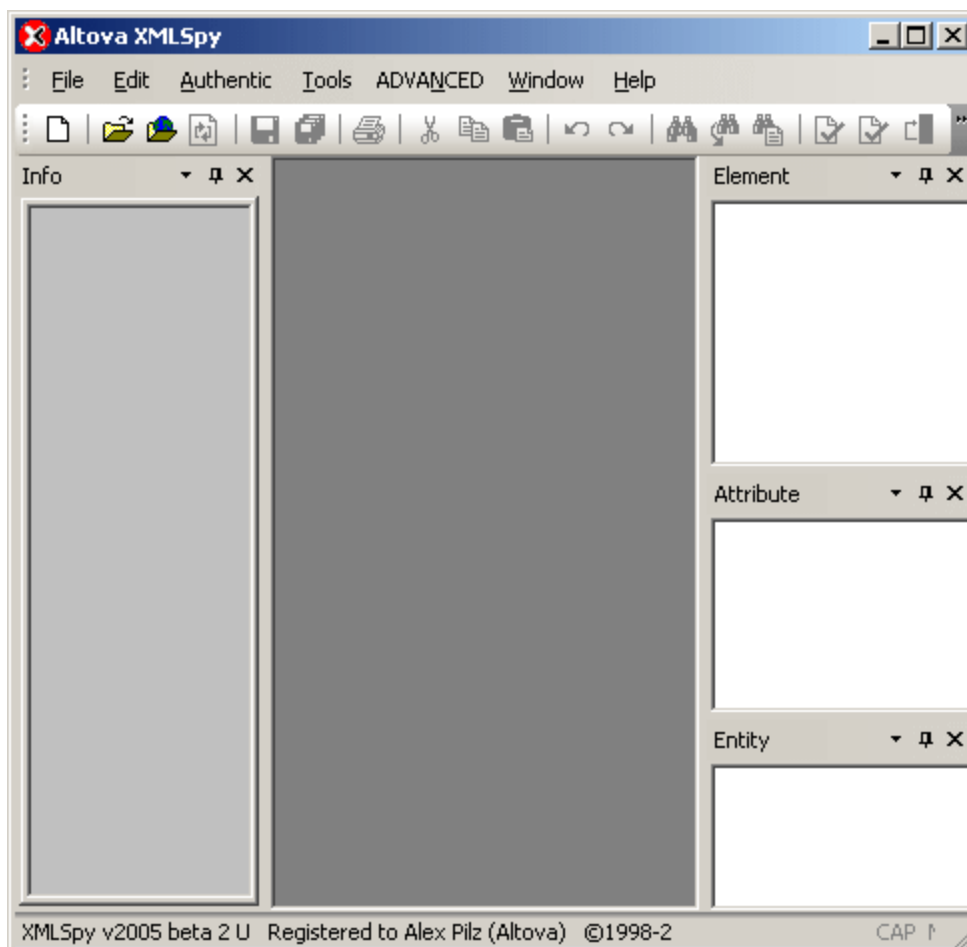


Append icon, allows you to append an element to the schema.

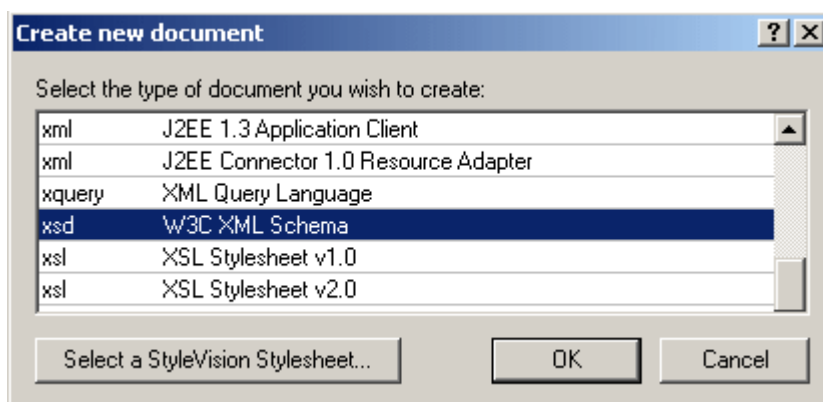
## 2.1 Creating a new Schema file

### To create a new schema file:

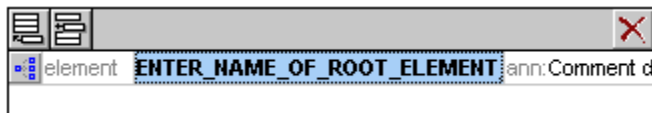
1. Start XMLSpy by double-clicking on the XMLSpy icon. You are presented with an empty environment. There are no XML documents in the main window.



2. Select the menu option **File | New** and select the **W3C XML Schema** entry from the dialog and confirm with **OK**.

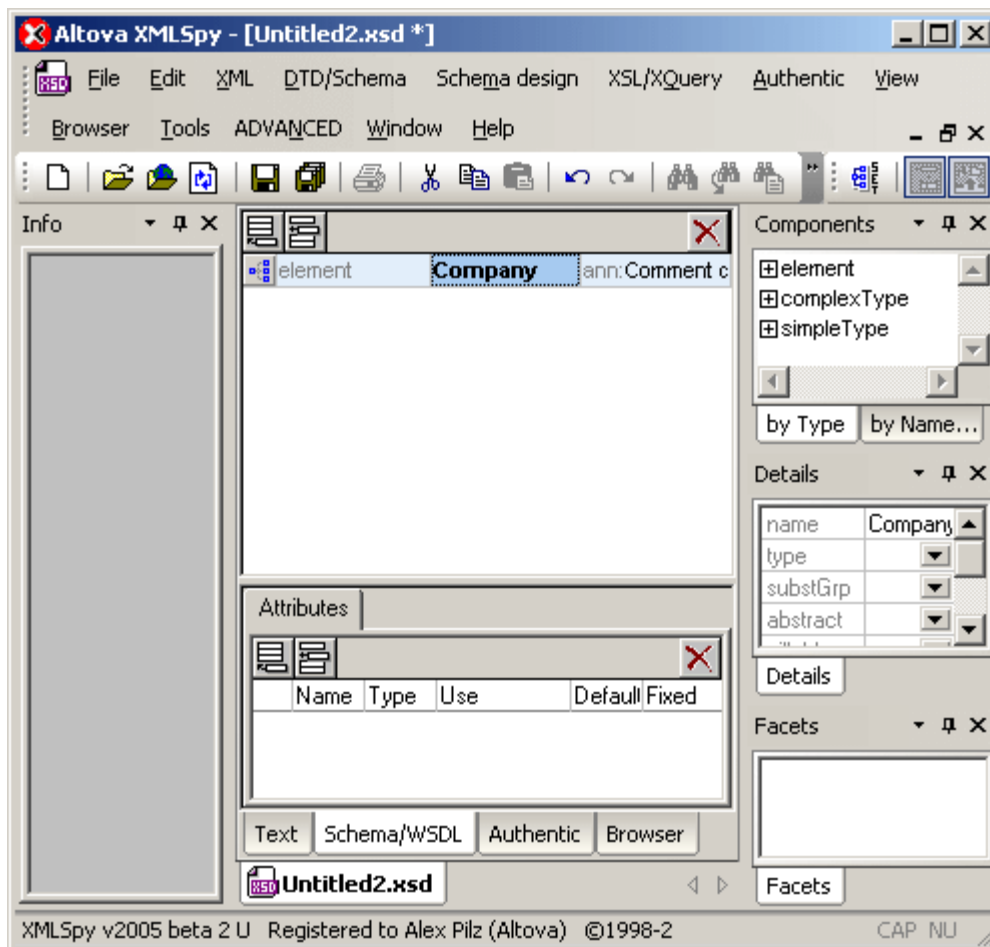


An empty schema file appears in the main window. You are prompted to enter the name of the root element.



3. Click in the highlighted field and enter `Company`. Confirm with **Enter**. `Company` is now the "root" element of this schema and is automatically a "global element" as well.

This view is the **Schema overview** and displays the **global components** in the top window and the **attributes** of the currently selected component, in the lower one.



The **Components** entry helper displays **Company** when you click the Expand icon of the **element** component. The entries in these components can be used to navigate your schema by double-clicking on them.

4. Click the menu option **File | Save as**, and name your schema (**AddressFirst** for example).

#### Defining your own namespace:

1. Select the menu option **Schema Design | Schema settings**.
2. Click the Target namespace radio button, and enter **http://my-company.com/namespace**.



**Please note:**

The namespace defined above will also be used in two related files:

- the **XML** file you will create later in this tutorial and
- in the **XSLT stylesheet** that will be used to transform the XML to HTML.

The namespace must be **identical** in all three files (Schema, XML, and XSL). So if you enter any other namespace than that given above, please ensure that the corresponding namespace entries in the XML and XSLT files match.

Schema settings

Default Element form:  Qualified  Unqualified

Default Attribute form:  Qualified  Unqualified

Block default:

Final default:

Version:

No target namespace

Target namespace:


Prefix	Namespace
	http://my-company.com/namespace
xs	http://www.w3.org/2001/XMLSchema

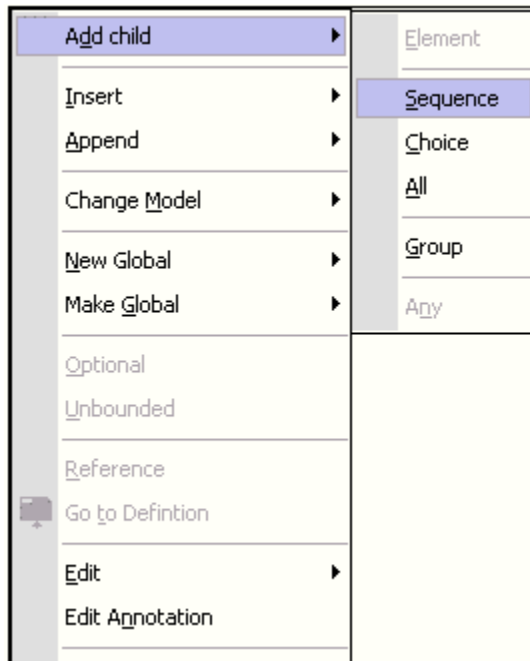
OK Cancel

3. Confirm with **OK**.

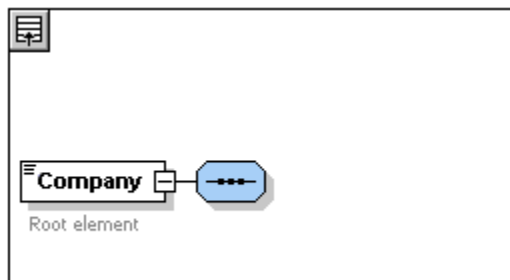
## 2.2 Adding elements to a schema

### To add elements to a schema:

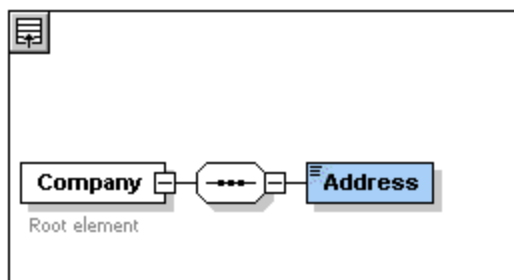
1. Click the component icon  next to the `Company` element in the main window to display the content model (or double-click on the `Company` entry in the Component Navigator).  
The text below the `Company` element is annotation text. Double-click the text if you want to edit it. (shortened to "Root element" here.)
2. Right-click the `Company` element to open the context menu, and select **Add Child | Sequence**.



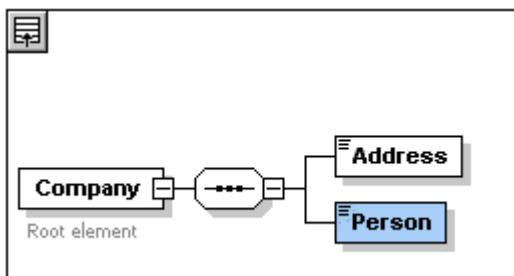
This inserts the **Sequence compositor**, and defines that the following elements must appear in the same sequence (in the XML document).



3. Right-click the Sequence compositor and select **Add Child | Element**.
4. Enter `Address` as the name of the element, and confirm with **Enter**.

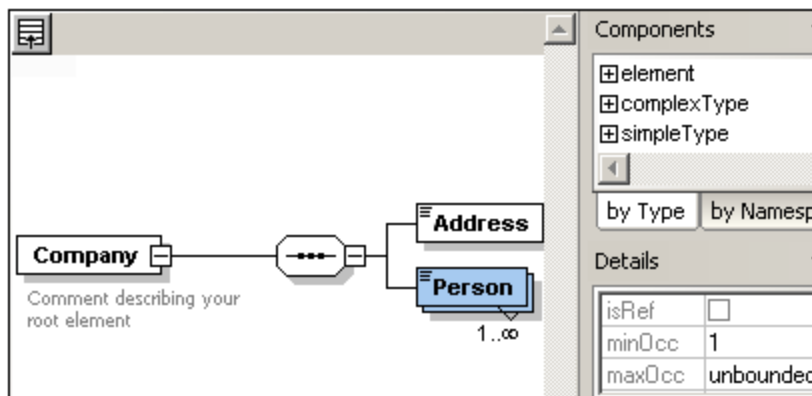


- Right-click the Sequence compositor again, select **Add Child | Element**, and enter `Person` as the name of the element.



We have now defined a schema which allows for one address and one person per company. As this is too restrictive, we want to make sure that we can include as many persons per company as necessary.

- Right-click the Person element, and select **Unbounded** from the context menu. The Person element changes at this point, showing the range in which it can occur, in this case 1 to unbounded.



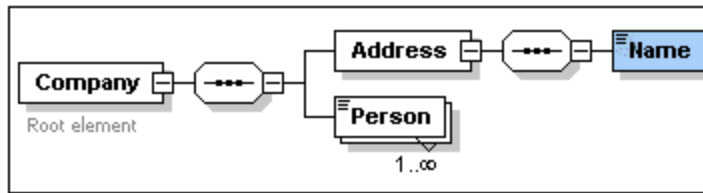
**Please note:**

You can also edit the **minOcc** and **maxOcc** fields in the Details entry helper directly.

We will now add the sub-elements which define the address structure.

**To add sub-elements to an element:**

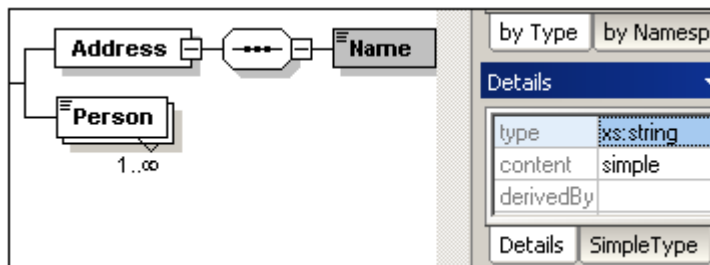
- Right-click the **Address** element to open the context menu, and select **Add Child | Sequence**.
- Right-click the **Sequence** compositor, and select **Add Child | Element**. Enter `Name` as the element name.



### Defining element parameters:

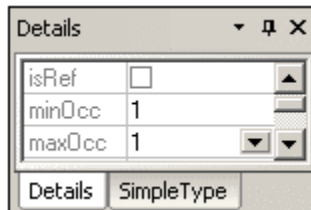
At this point we want to define that the Name element is to occur only once, and contain only textual data.

1. Click the Name element, if not currently selected.
2. Click on the **type** combo box of the middle entry helper, and select the entry **xs:string** from the drop down list.



This entry helper is called **Details** in the Schema/WSDL view, and provides information on the currently selected element. **All data can be edited directly in the Details window!**

Both **minOcc** and **maxOcc** fields contain 1, showing that there is only one occurrence of this element (this is the default setting when creating a new element).

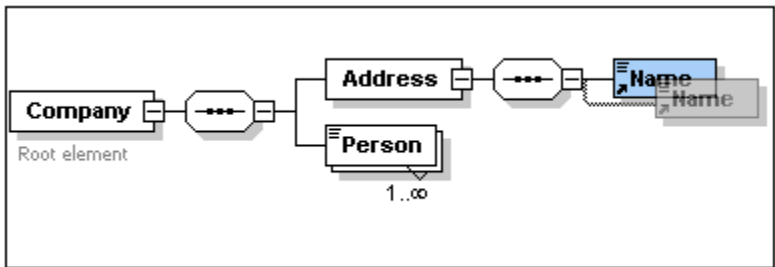


## 2.3 Adding elements using drag and drop

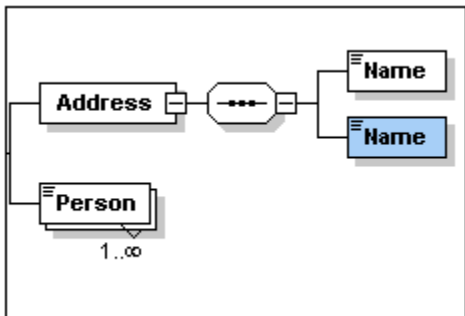
### To add elements using drag and drop:

There is a quicker method of adding new elements to a schema, which avoids multiple menu commands:

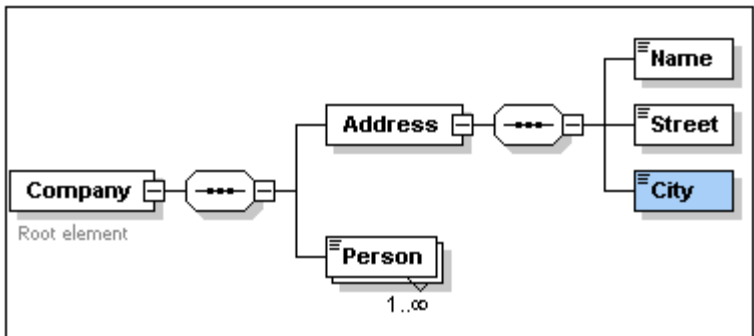
1. Click the Name element, hold down the **CTRL** key, and **drag** "slightly" with the mouse. A small "plus" icon appears as well as a copy of the element, showing that you are about to copy the element.



2. Release the mouse button to create the new element. If the new element appears somewhere else, just drag it near to the Name element and drop it there. This method creates an element of the same type, with the same settings as the one copied.



3. Type `Street` to change the element name.
4. Use the same method to create a third element, "City". The content model should now look like this:



## 2.4 Completing the basic schema

At this point we want to add those sub elements to the Person element that make up the personal data. All these elements will be **simple types** (with **simple content** models).

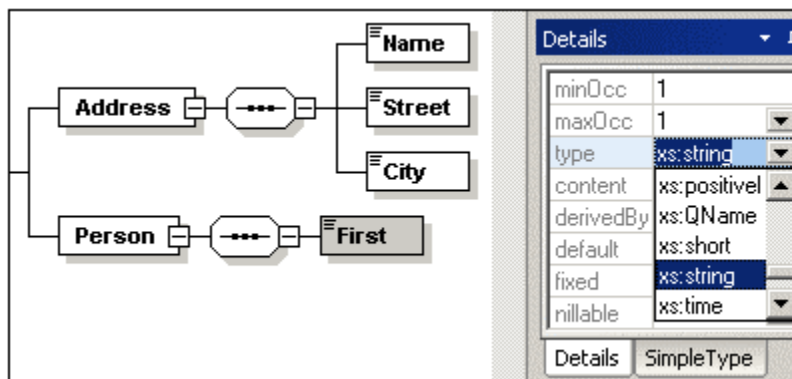
Person sub-elements: First, Last, Title, PhoneExt, and Email.

### Requirements:

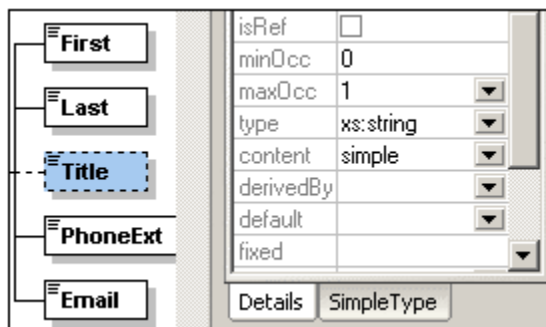
Title element: should be **optional**

PhoneExt: should be an **integer** and limited to **2 digits**

1. Right-click the Person element to open the context menu, and select **Add Child | Sequence**. This inserts the Sequence compositor.
2. Right-click the **Sequence** compositor, and select **Add Child | Element**.
3. Enter `First` as the name of the element and confirm with **Enter**. Go to the **Details** window and click on the down-arrow in the **type** row.



4. Select the **xs:string** entry from the drop down list.
5. Use the drag and drop method to create **four more elements**, and name them: Last, Title, PhoneExt, and Email respectively.

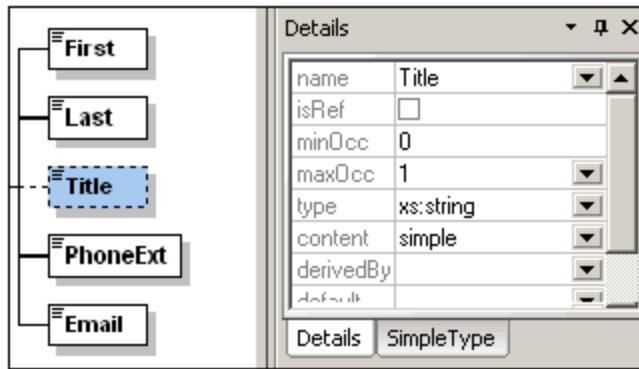


### Please note:

You can select multiple elements by holding down the CTRL key, and clicking each one.

### To make an element optional:

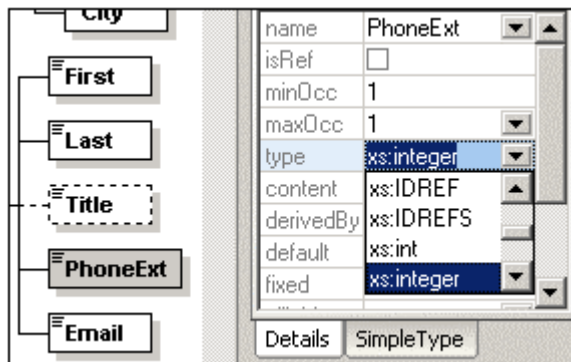
1. Right-click the **Title** element, and select **Optional** from the context menu. The solid element frame changes to a dashed one; this is the visual display that an element is optional.



The **Details** fields have also been updated **minOcc=0** and **maxOcc=1**.

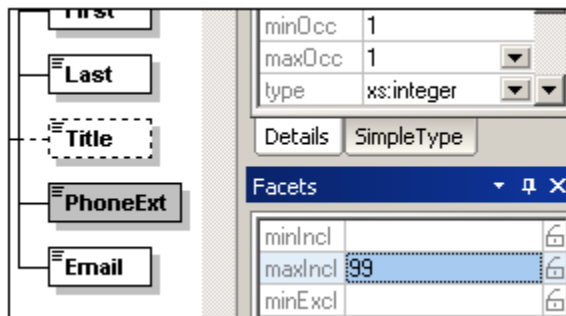
**To limit the content of an element (Facets):**

1. Click the **type** field of the **PhoneExt** element, and select (or enter) the **xs:integer** entry from the drop down list.



The items in the Facets tab (in the lowest entry helper) change at this point.

2. Double-click in the "**maxIncl**" field of the Facets tab (in the lowest entry helper) and enter **99**, confirm with **Enter**.



This defines that all phone extensions up to and including 99 are valid.

3. Select the menu option **File | Save** to save the changes to the schema.

**Please note:**

- Selecting a predefined simple type "text" (i.e. xs:string, xs:date etc.) for an element automatically changes the content model to: content = simple, in the Details entry helper.
- Adding a compositor to an element (selection, choice or all), automatically changes the content model to: content = complex, in the Details entry helper.
- This schema is available as **AddressFirst.xsd** in the **..\Tutorial** folder.



### 3 Making schema components reusable

**Goal of this section:**

To create generic **schema** components which can be reused by other elements.

This will be achieved by:

- Creating a global **AddressType** component, which will be the basis for specific country addresses (a complex type)
- Creating two specific address templates for UK, and US Adresses by **extending** the global address element (extend the complex type)
- Creating a global US-State element, by **restriction** (simpleType)
- Creating a global person element by **reference**
- Defining person **attributes** that supply information about the persons position in the company
- **Limiting** the **attribute contents** to a predefined set of attribute values (enumeration)

Functions (and their icons) in this section:



**Schema design | Display all globals**, takes you back to the schema overview.



Append icon, allows you to append an element or attribute to a schema.



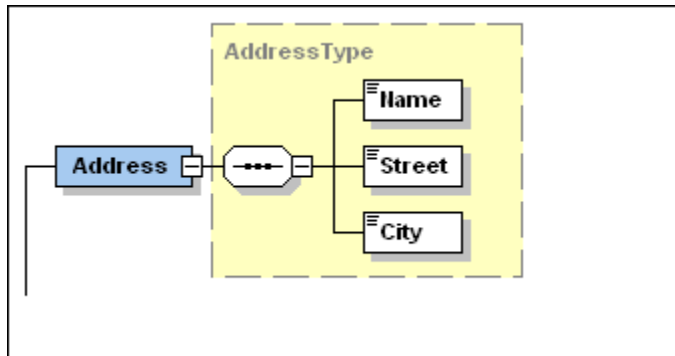
**Schema design | Display diagram**, the component icon displays the content model of the active global component in the schema overview.


### 3.1 Creating and extending global components

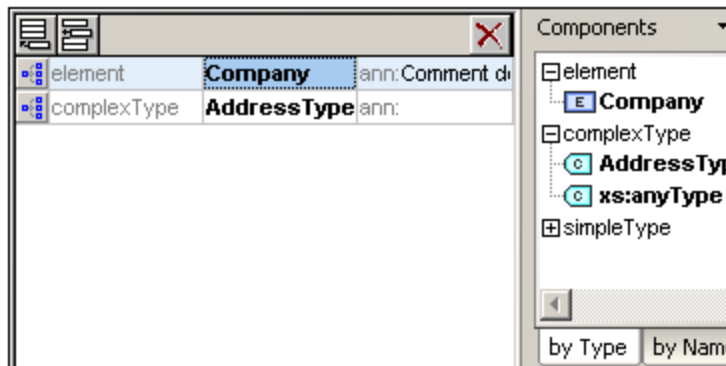
Having defined an element, you may then realize that you want to reuse it somewhere else in your schema. In XMLSpy this is achieved by creating a **global component**.

**To create a global component:**

1. Right-click the `Address` element, and select **Make Global | Complex type**. The `Address` elements appear in a yellow box.

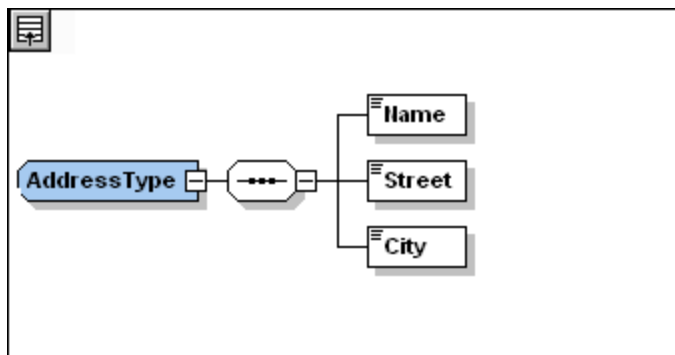


2. Click the Display all Globals icon . The schema overview now displays two global components: the `Company` element and the `complexType` "AddressType".



Click the **Element** and **complexType** entries in the Components entry helper, to see the respective schema constructs.

3. Click on the `AddressType` component icon  to see the content model.





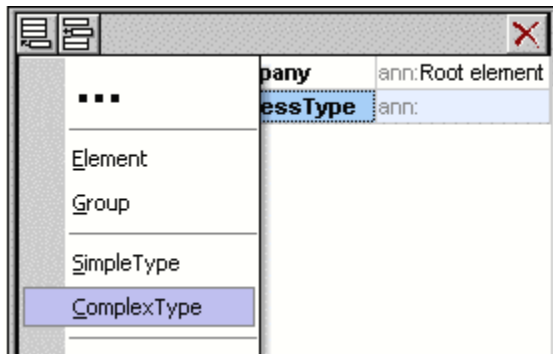
4. Click the Display all Globals icon  to return to the schema overview.

### Extending a "complex type" definition

We now want to use the global AddressType component to create two kinds of country-specific addresses. For this purpose we will define a **new** complex type **based** on the AddressType component.

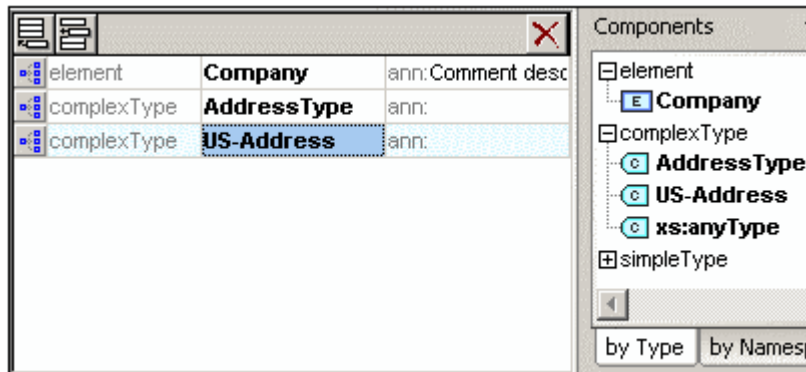
#### To extend a "complex type" definition:


1. Switch to the schema overview, if not already visible (Display all globals )
2. Click the Append icon  at the top left of the component window.
3. Select **ComplexType** from the context menu.

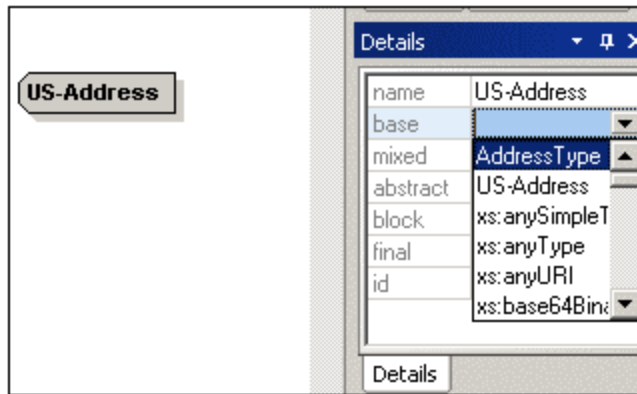


A new line appears in the component list, and the cursor is set for you to enter the component name.

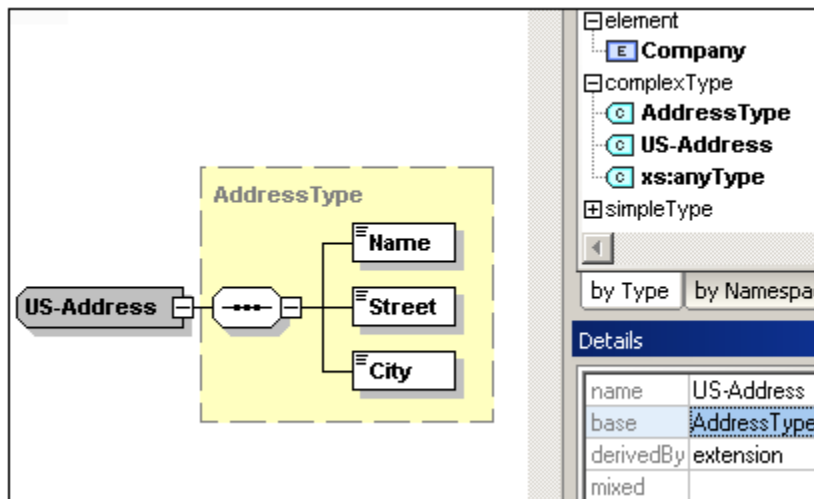
4. Enter **US-Address** and confirm with **Enter**. (If you enter US-Address with a blank space instead of a hyphen character "-", the element name will appear in **red**, signalling an invalid character.)



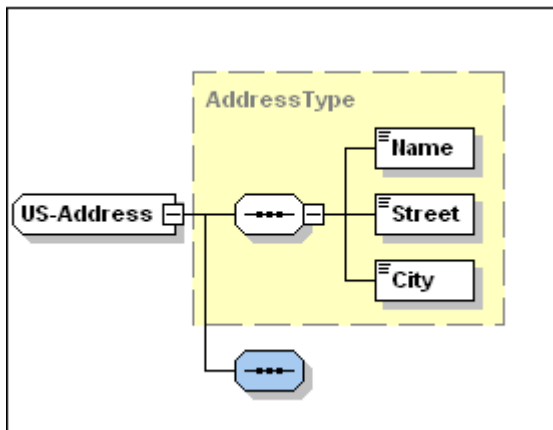
5. Click the **US-Address** component icon  to see the content model.
6. Click the **base** combo box in the Details entry helper, and select the **AddressType** entry.



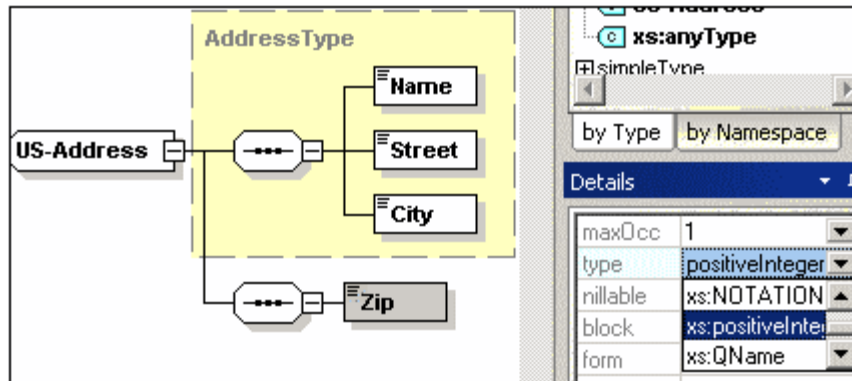
The content model view changes immediately and displays the previously defined generic address.



- Right-click the **US-Address** element, and select **Add Child | Sequence**. A new sequence compositor is displayed **outside** of the AddressType box. This is a visual indication that this is an **extension** to the element.





- Right-click the new **sequence** compositor, and select **Add Child | Element**.
- Name the element `zip`, and confirm with **Enter**.
- Select (or enter) `xs:positiveInteger` from the "type" field combo box, and confirm with **Enter**.



### Creating reusable "simple type" elements


Simple type elements can also be made generic. In this case we want to make the **State** element reusable, so that an abbreviated version could also be included in address labels at a later time (GA for Georgia, for example).

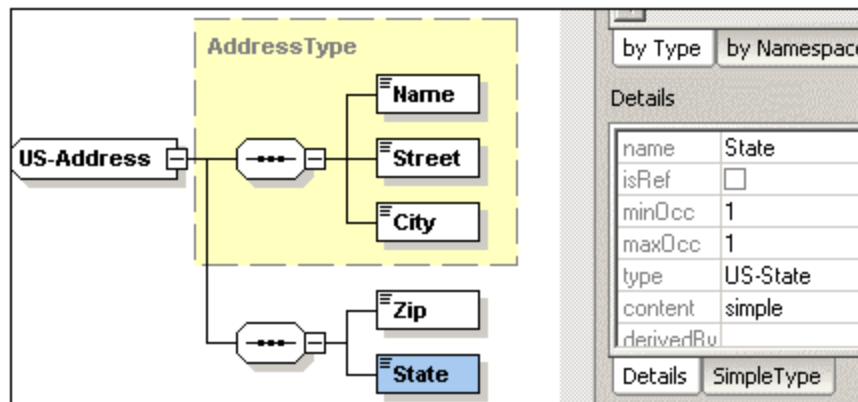
#### To create reusable "simple type" elements:

1. Switch to Schema overview (click the Display all Globals icon .
2. Click the Append icon , select **SimpleType**, and enter `US-State` as the element name. Confirm with **Enter**.
3. Select **xs:string** in the **restr** value field of the Details entry helper. This completes the definition. This element can now be used in the US-Address definition.

Name	Type	Use	Default	Fixed
Company	ann:Comment descri			
AddressType	ann:			
US-Address	ann:			
US-State	ann:			

Name	Type	Use	Default	Fixed
name	US-State			
final				
id				
restr	xs:string			

4. Click the US-Address component icon , then right-click the lower sequence compositor and select **Add Child | Element**.
5. Enter `State` for the element name and move the pointer to the Details entry helper.
6. Select (or enter) **US-State** from the "type" combo box. Confirm with **Enter**.

**Please note:**

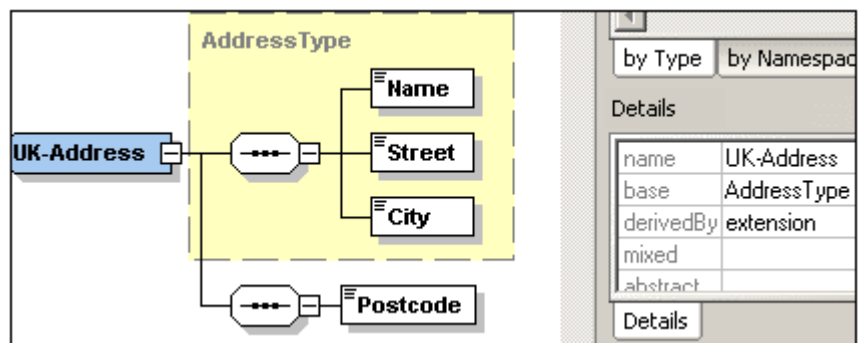
Global simple types can only be created from the schema overview.

**Creating the second Address type**

Using the method described above, define the global complex type **"UK-Address"**.

1. Create the global **complex type** "UK-Address", with the **base**="AddressType"
2. Add a new Postcode element to the content model of UK-Address.

Your UK-Address content model should then look like this:

**Please note:**



Global definitions (global elements, complex types, etc.) can be moved or copied to other schemas visible in the schema overview using drag and drop. You can, of course, reposition definitions in the currently open schema.


Right-clicking a definition opens the context menu in which you can select the standard cut, copy, paste commands, to achieve the same thing. The drag and drop method also applies to attributes visible in the **Attributes** tab.

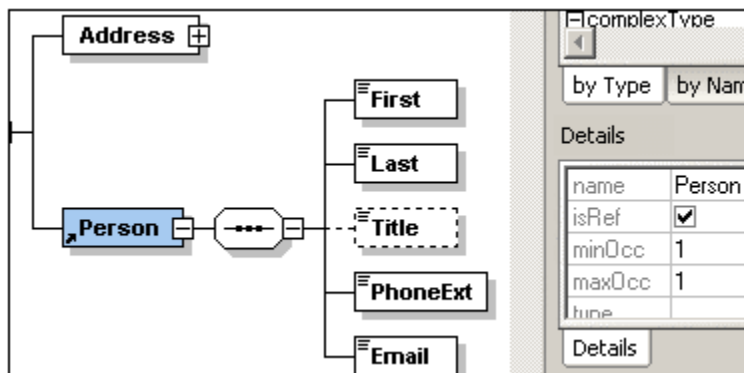
## 3.2 References, attributes and enumerations


To finish the schema definition we will make the `Person` element global, define specific element attributes, and limit the attribute selection.

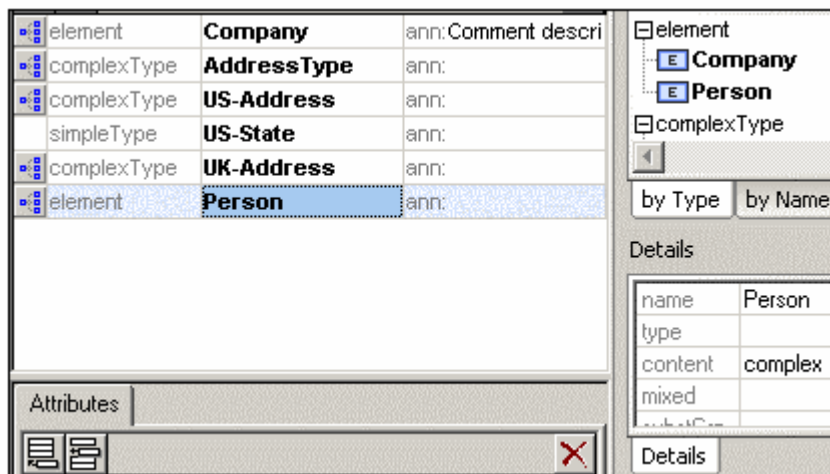
### To create a reference:

1. Switch to Schema overview (Display all Globals .
2. Click on the component icon of the `Company` element .
3. Right-click the `Person` element, and select **Make Global | Element**.

A small "link" icon  appears in the `Person` element, showing that this element now references the globally declared `Person` element. The `isRef` check box in the **Details** entry helper is now checked.



4. Click the Display all Globals icon  to return to the schema overview. The `Person` element is now also visible in the component list, as well as in the **Elements** entry helper.




The screenshot shows the XMLSpy Schema Overview with the 'Person' element selected in the component list. The 'Details' panel for the 'Person' element is open, showing the following table:

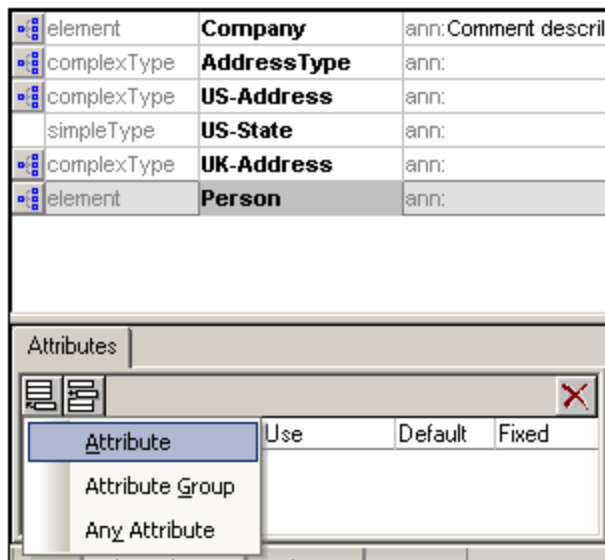
Details	
name	Person
type	
content	complex
mixed	
substCom	

### Please note:

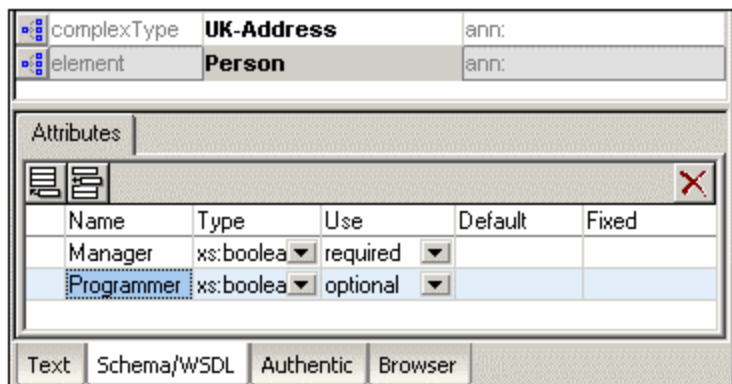
**Global declarations** do not describe where an element is to be used in an XML document, they only describe what it contains. Global definitions have to be referenced from within a complex type or another element to determine their position in the XML document.

**To define Element attributes:**

1. Click the **Person** element to make it active.
2. Click the **Append** icon  in the top left of the **Attributes** tab (the lower window of the schema overview), and select the **Attribute** entry.




3. Enter **Manager** as the attribute name in **Name** field.
4. Use the **Type** combo box to select **xs:boolean**.
5. Use the **Use** combo box to select **required**.

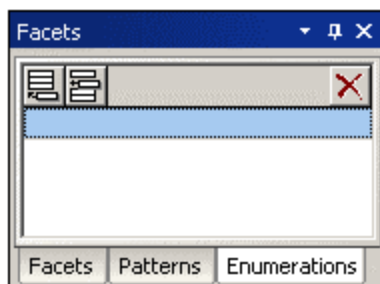



6. Use the same method to:  
Add a **Programmer** attribute in the **Name** field (type="xs:boolean"), and set its **Use** to optional.

**To limit the contents of an attribute (Enumerations):**

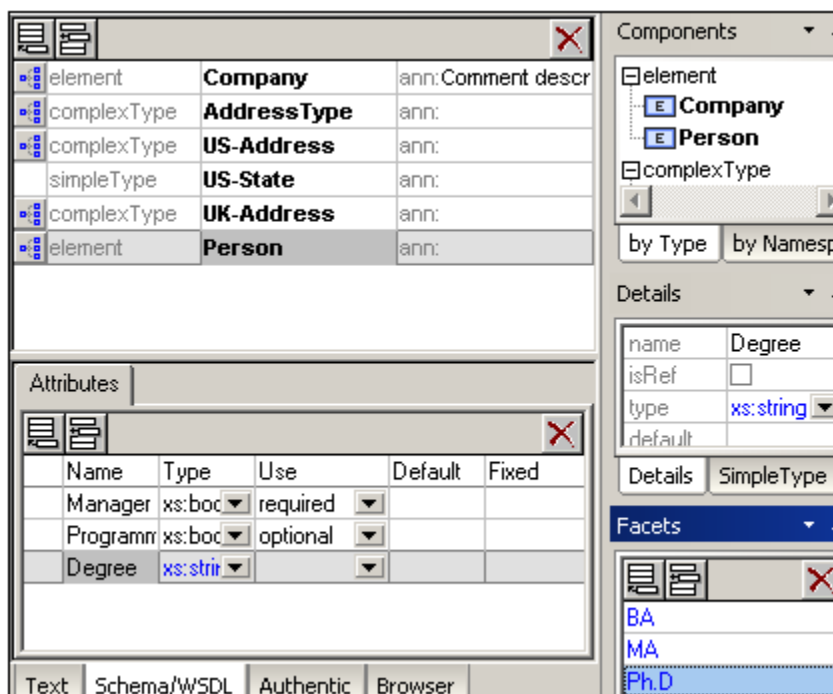
1. Click the **Append** icon  in the top left of the **Attributes** window, and select the **Attribute** entry.
2. Enter **Degree** as the attribute name, and select **xs:string** as the attribute type.
3. Click the **Enumerations** tab of the Facets entry helper.





4. Click the Append icon  of the **Enumerations** tab and enter BA. Confirm with **Enter**.
5. Use the same method to add two more items to the enumerations list (MA and Ph.D).

The finished schema should look like this:



6. Select the menu command **File | Save As**, and save the file as **AddressLast.xsd**.

**Please note:**

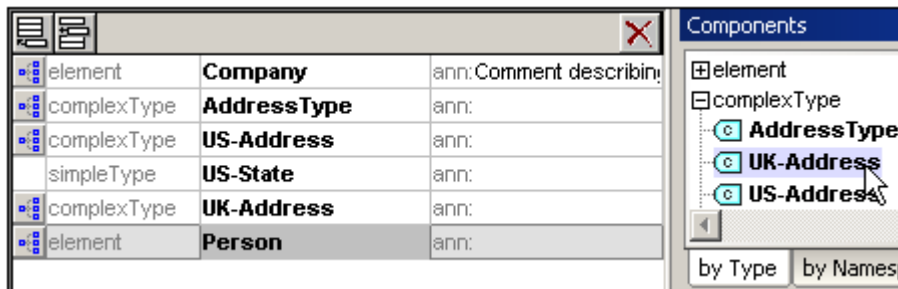
This schema is available as **AddressLast.xsd** in the Tutorial folder.

### 3.3 Navigation shortcuts in schema documents

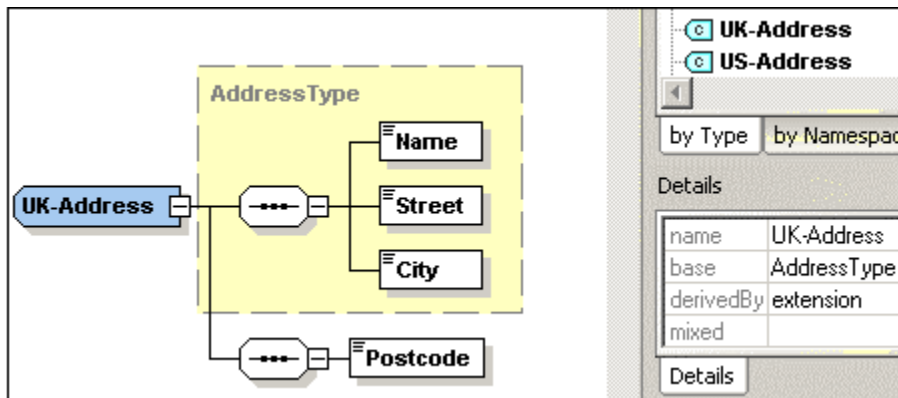
This section is designed to show you how you can navigate the Schema view efficiently.

#### Displaying the content model of any element:

- Select the **element type** you want to see by double-clicking it in the list in the **Elements** Entry Helper. For example, to see the content model of UK-Address, expand the complexType list in the **Elements** Entry Helper, and **double-click** the element name.



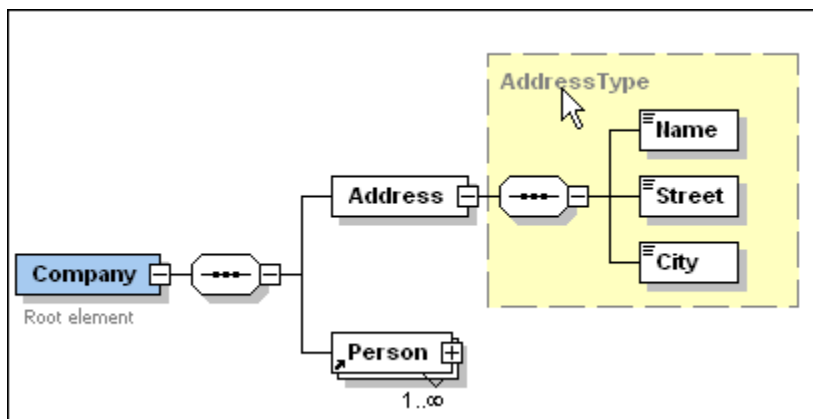
The content model of the UK-Address element is displayed. The specific settings are shown in the **Details** tab.



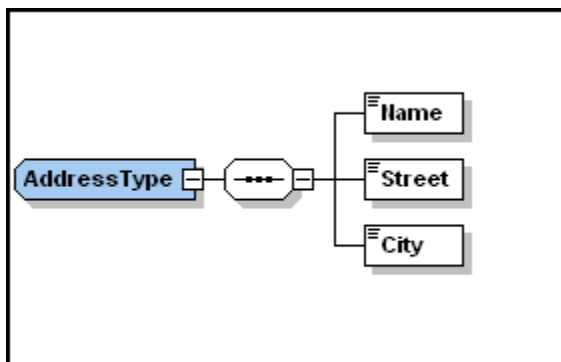
#### Go to "ElementType" definition:

For example, while viewing the `Company` content model:

- Double-click the *AddressType* text in the yellow box to go to the AddressType definition.



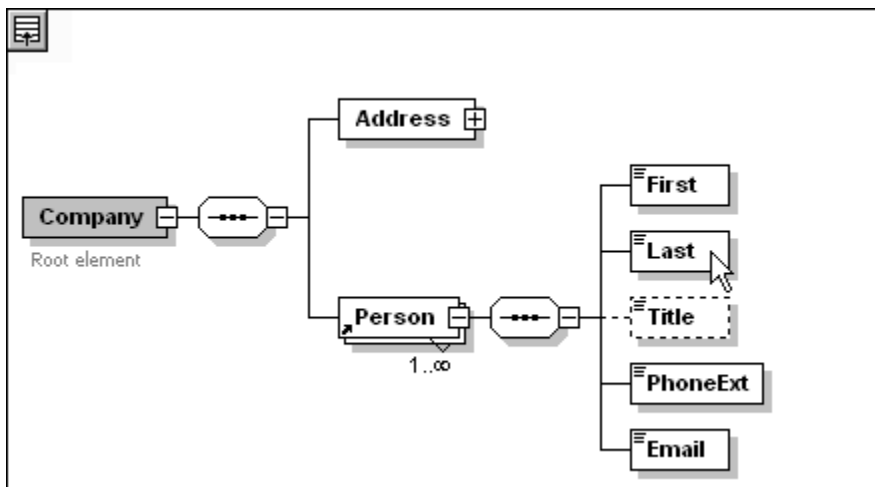
The AddressType definition:



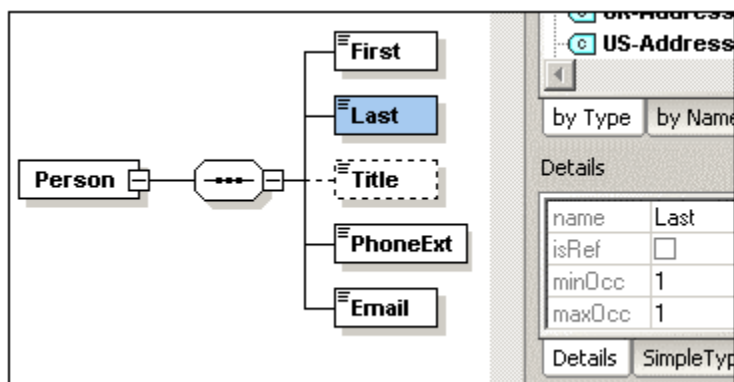
#### Go to element definition:

For example, while viewing the **Company** content model:

- Press and hold down the **CTRL** keyboard key, and
- **Double-click** on any element definition you want to see (here, the element **Last**).



The element **Last**, which is a sub-element of the **Person** element, is displayed. The specific settings are shown in the **Details** tab.



## 4 Creating an XML document

### Goal of this section:

To create a new XML document and use the various XMLSpy views and intelligent editing capabilities to rapidly enter and validate data.

This will be achieved by:

- Creating a new XML document based on the **AddressLast-Home** schema
- Adding elements using **intelligent entry helpers** in Authentic View
- **Validating** the XML document

Functions (and their icons) in this section:



**File | New**, creates a new XML file.



**View | Text View**, switches to Text view.



**View | Authentic View**, switches to Authentic View.



Checks for well-formedness. Shortcut: **F7**.



Validates the XML file against the associated DTD or Schema. Shortcut: **F8**.

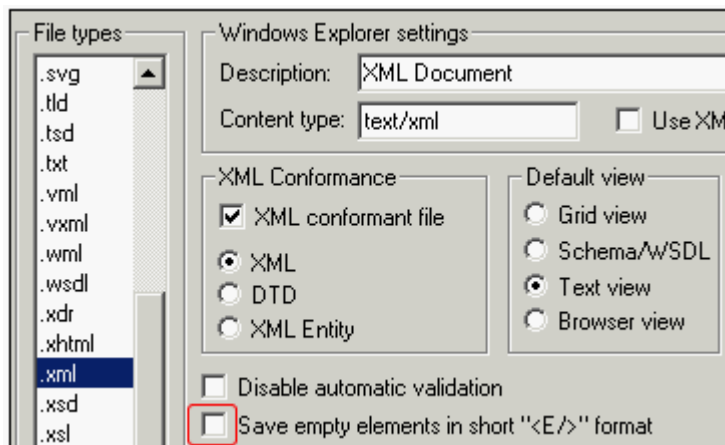
### Please note:

The XML file used in the following section uses a slightly simplified schema file (**AddressLast-home.xsd**) to the one you created in the schema section of this tutorial.

## 4.1 Creating a new XML file

Before we create an XML instance file based on a schema, let's make sure the short format method of creating/saving empty elements is inactive.

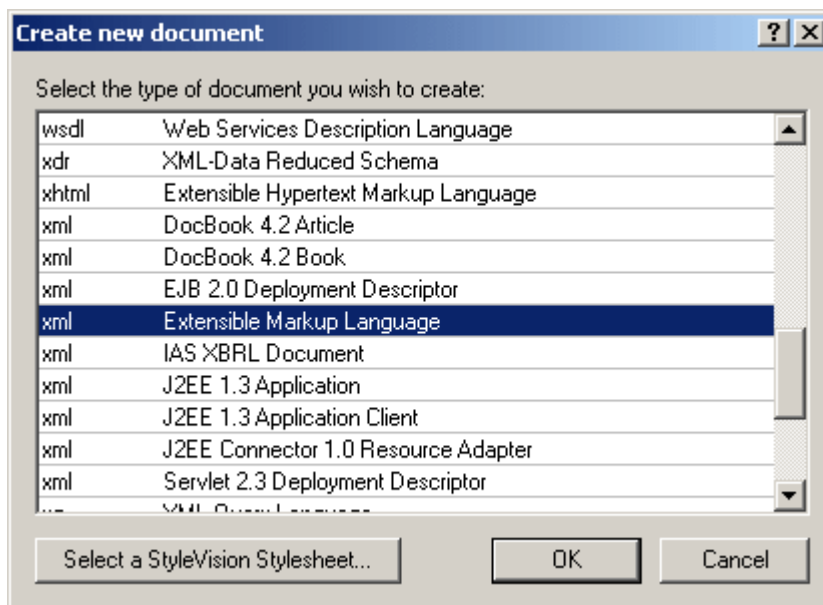
1. Select the menu option **Tools | Options**, and then click the **File types** tab.
2. Click the **.xml** entry in the list box, and deactivate the Save empty elements in short "<E/>" format check box.



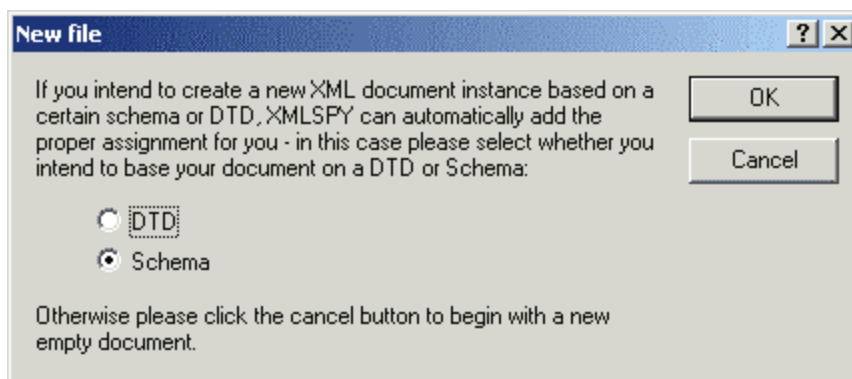
3. Click **OK** to confirm.

### To create a new XML document:

1. Select the menu option **File | New**, and select the **Extensible Markup Language** entry in the dialog. Confirm with **OK**.

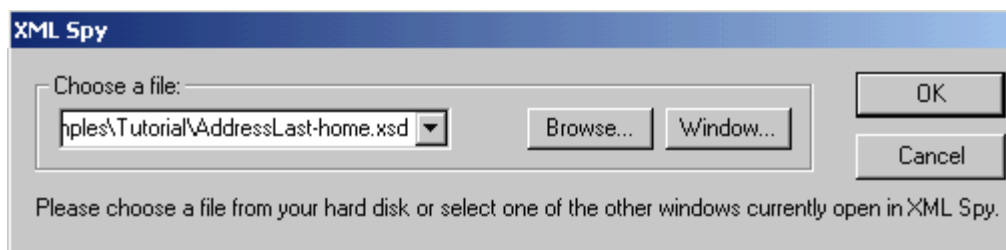


2. A prompt appears, asking if you want to base the XML document on a DTD or Schema. Click the **Schema** radio button, and confirm with **OK**.



A further dialog appears, prompting you to select the schema file your XML document is to be based on.

3. Use the **Browse** or **Window** buttons to find the schema file, in our case the **AddressLast-home** schema, and confirm with **OK**.



An XML document containing the main elements defined by the schema opens in the main window.

**Please note:**

XMLSpy tries to find the root element of a schema automatically. The Select the Root Element dialog box is opened if it is unclear which is the root element. You can then select the root element manually.

```
<?xml version="1.0" encoding="UTF-8"?>
<Company xmlns="http://my-company.com/namespace" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="C:\PROGRAMS\Altova\XMLSPY\2004\Examples\Tutorial\AddressLast-home.xsd">
  <Address>
    <Name></Name>
    <Street></Street>
    <City></City>
    <State></State>
    <Zip></Zip>
  </Address>
  <Person Manager="">
    <First></First>
    <Last></Last>
    <PhoneExt></PhoneExt>
    <Email></Email>
  </Person>
</Company>
```

4. Click on any element to deselect the data.

## 4.2 Editing in Authentic View

**Authentic View** enables you to create and edit an XML document that is based on a **StyleVision Power Stylesheet (.sps or SPS file) created in StyleVision**. It is a flexible and easy-to-use interface that features WYSIWYG capabilities and familiar data-entry devices. This enables even users unfamiliar with XML to easily create and edit an XML document.

The StyleVision Power Stylesheet to which an XML document is linked is created in StyleVision by the person who designs the document—and not by you, the user of Authentic View.

StyleVision Power Stylesheets (SPS files) for commonly used schemas are available in the **..\Template\Examples** folder. You can open a new XML document in Authentic View by selecting an Authentic template SPS file in the **File | New** dialog. Alternatively, you can open a new XML document by browsing for the required StyleVision Power Stylesheet. Note that the StyleVision Power Stylesheet (SPS file) is created in StyleVision—not in Authentic View.

### Assigning a StyleVision Power Stylesheet to an XML document

1. Select the menu option **Authentic | Assign a StyleVision stylesheet** and click **OK** when the prompt appears, to reparse the XML text.
2. Select the **AddressLast-home.sps** file from the **..\Examples\Tutorial** folder, and click **OK**.  
The assignment is added to the XML file.
3. Click the **Authentic** tab to switch to the Authentic View.

### Your Company

**Address**

Name	<input type="text"/>
Street	<input type="text"/>
City	<input type="text"/>
State	<input type="text"/>
Zip	<input type="text"/>

**Employees**

Manager	Programmer	Degree	First	Last	Phone
<input type="text"/>	<a href="#">add Programmer</a>	<a href="#">add Degree</a>			

**Entering (and deleting) data**

1. Double-click in the **Name** value field (or use the arrow keys) and enter `US dependency`, then press the **TAB** key to move into the next field.
2. Use the same method to enter **Street** and **City** names (e.g., Noble Ave. and Dallas, etc.), and following on, enter the State and Zip code.

**Address**

Name	US dependency
Street	Noble Ave.
City	Dallas
State	Texas
Zip	04812

3. Click in the **First** field of the Employees table and select the menu option **Authentic | Delete row** (we will add it again in a few moments in the Text view!).

## Your Company

**Address**

Name	US dependency
Street	Noble Ave.
City	Dallas
State	Texas
Zip	04812

[add Person](#)

**Please note:**

The **add Person** placeholder that you now see below the address table is a feature of Authentic View. Clicking the placeholder text would automatically add the Person table in this view.



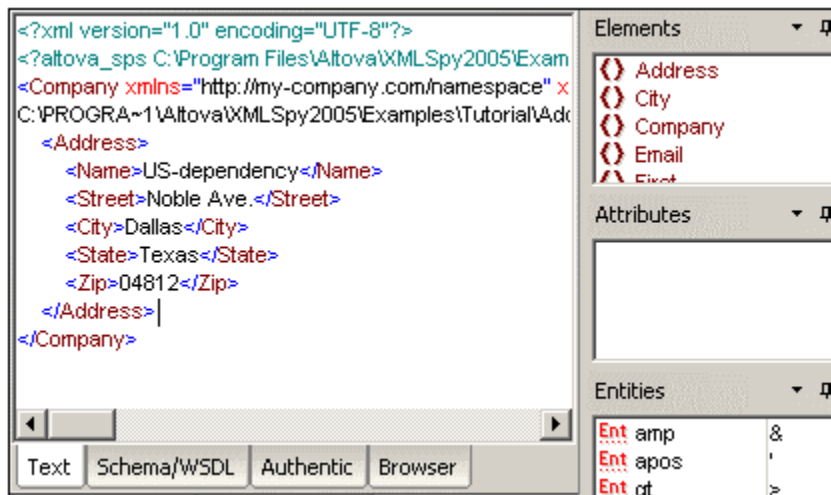
## 4.3 Editing in Text View

### XMLSpy Text view

When it comes down to low-level work, the Text view of XMLSpy is suitable for editing any type of XML files in textual or source code form, and provides **intelligent editing** capabilities if you are working with an XML document based on a DTD or XML Schema.

#### Viewing and entering data in the Text view:

1. Select the menu item **View | Text view**, or click the **Text** tab.  
You now see the XML document in its raw text form (with syntax coloring).



2. Place the text cursor after the **</Address>** end tag, and press **Enter** to add a new line.
3. Enter the "less than" angle bracket **<** at this position.




4. A drop-down list appears; select the **Person** entry.  
The element name "Person" as well as the attribute "Manager", are inserted. The attribute value drop-down list is also automatically opened.

```

<Address>
  <Name>US dependency</Name>
  <Street>Noble Ave.</Street>
  <City>Dallas</City>
  <State>Texas</State>
  <Zip>04812</Zip>
</Address>
<Person Manager="
</Company>

```




5. Enter the letter "t" and confirm with **Enter**.

```

</Address>
<Person Manager="t"
</Company>

```



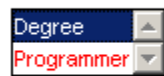
This selects the attribute "true" and inserts it at the cursor position.

6. Move the cursor to the end of the line (**End** key), and press the space bar. This opens the drop-down list again. There are now fewer entries available in the list; "Manager" is grayed out in the Attribute entry helper.

```

</Address>
<Person Manager="true"
</Company>

```




Attributes	
<input checked="" type="checkbox"/>	Programmer
<input checked="" type="checkbox"/>	Degree
<input checked="" type="checkbox"/>	xsitype
<input type="checkbox"/>	Manager

7. Select "Degree" with the Down arrow key, and press **Enter**. This opens a further drop-down list from which you can select one of the predefined enumerations (**BA**, **MA** or **Ph.D**).

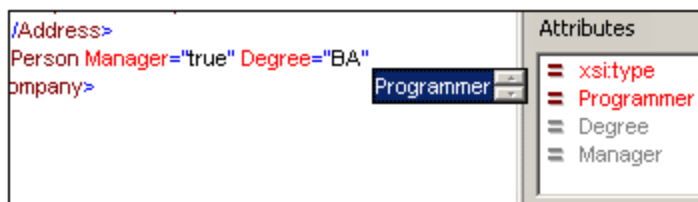
```

<Address>
  <Name>US dependency</Name>
  <Street>Noble Ave.</Street>
  <City>Dallas</City>
  <State>Texas</State>
  <Zip>04812</Zip>
</Address>
<Person Manager="true" Degree="
</Company>

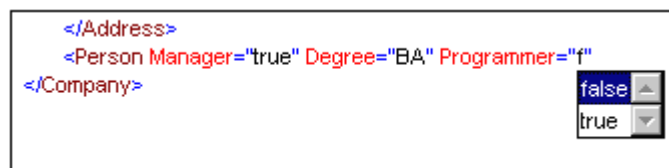
```



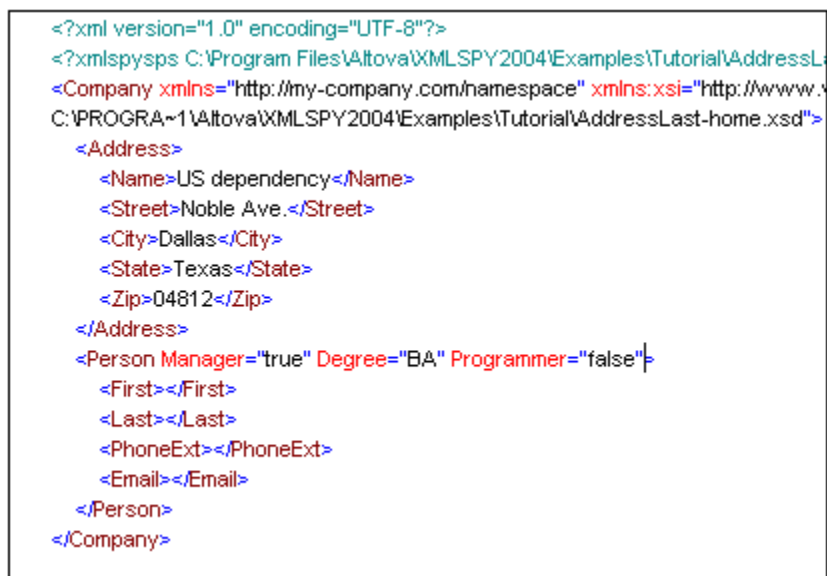
8. Select **BA** using the Down arrow key (confirm with **OK**), move the cursor to the end of the line (**End** key), and press the space bar. **Manager** and **Degree** are now grayed out in the **Attributes** entry helper.



9. Select **Programmer** with the Down arrow key, and press **Enter**.



10. Enter the letter **"f"** and press **Enter**.  
 11. Move the cursor to the end of the line (**End** key), and enter the "greater than" angle bracket **>**.



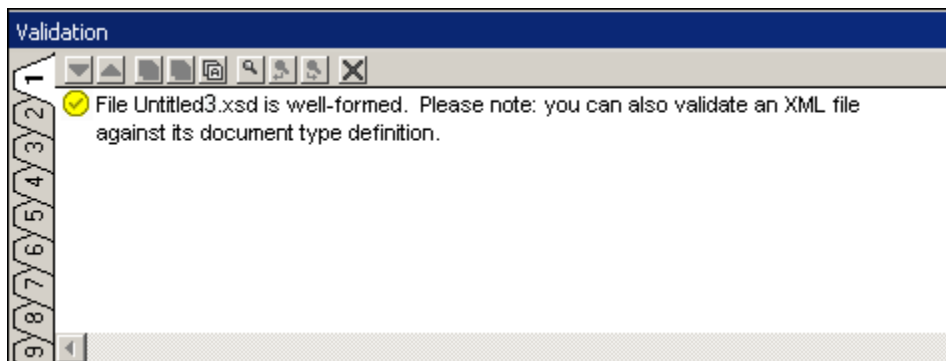
XMLSpy automatically inserts all the **Person** element tags. Each element is supplied with start and end tags.

## 4.4 Validating and entering data

At this point let's check if the document is well-formed and valid. There might still be work to do.

**To check for well-formedness:** 

1. Select the menu option **XML | Check well-formedness** or press **F7**.  
A message appears in the Validation window stating that the document is well formed.



Being well-formed means that the XML document **syntax** is correct (i.e., there is a root element, each start tag has a corresponding end tag, all elements are nested correctly, etc.).

This check does not check against a schema file (or any other external file). Element sequence or element content are not checked either. Well-formedness can only be checked in Text view.

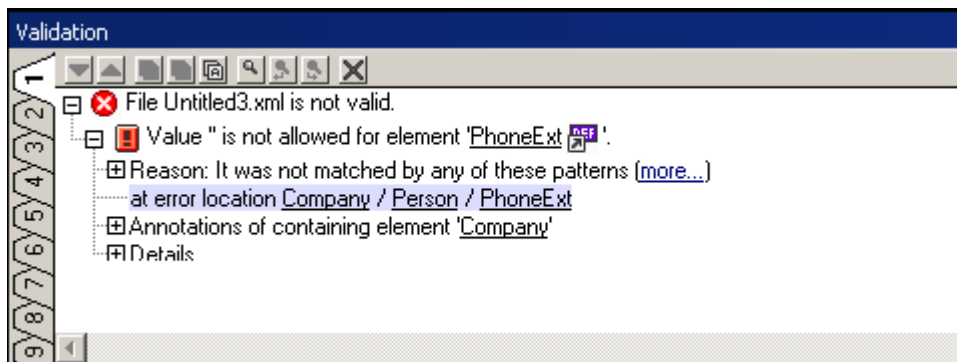
**To check for validity:** 

The **validity** of an XML document can be checked both in Text and Authentic View.

1. Click the **Authentic** tab to switch into Authentic View.
2. Select the menu option **XML | Validate** or press **F8**.  
An **error message** appears in the Validation window saying that the file is not valid, and why (see *screenshot*).

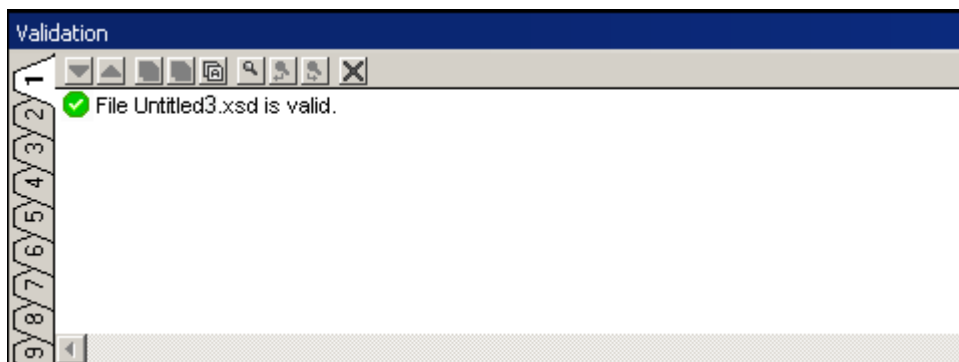
The error message describes what is wrong with our XML document. It says that the 'PhoneExt' element has an empty string as value. This, of course, is not allowed, because the schema says that this element is required.

**Please Note:** You can click on the links in the error message to jump to the spot in the XML file where the error was found.



3. Fill in the rest of the table fields (e.g., Alfred, Aldrich, PhoneExt=33).

4. Press **F8** again to check if the document is now valid.  
The Validation window displays a message saying that the file is valid. The XML document is now valid against its schema.



Being valid means that the XML document adheres to the assigned schema, i.e., the elements and the sequence they appear in is correct, as well as the element "contents" and their attributes.

5. Select the menu option **File | Save As...** and name the XML document (e.g., **CompanyLast-Home.xml**)

**Please note:**

An XML document does not have to be valid in order to save it. Saving an invalid document causes a prompt to appear which then allows you to select **Save anyway**. The document is then saved in its current state.

Authentic View provides **real-time** content **validation** against the referenced schema. Whenever invalid data is entered, it is automatically highlighted in red. Press **F8** for more information on why the data is invalid.

Positioning the text cursor over the **Degree** field for a few seconds displays a tooltip describing the type of data that must be entered here. The tooltip text is defined in StyleVision.

## 4.5 Manipulating data with Entry Helpers


At this point we want to enter more employees in the XML document.

### Inserting elements and attributes (intelligent entry help):

1. Click into one of the table fields.

The screenshot shows a form titled "Your Company" with an "Address" section containing fields for Name (US-dependency), Street (Noble Ave.), and City (Dallas). To the right, an "Elements" entry helper is open, showing a "Person" element with a red append icon. Below it, an "Attributes" helper shows the "Last" attribute.

You will notice that the `Person` element is now visible in the **Elements** entry helper. The entries in the **Element** and **Attributes** entry helpers are dependent on where you click in the Authentic document.

2. Click the append icon  to append a person row to the employees table.

Employees				
Manager	Programmer	Degree	First	Last
true	false	BA	Alfred	Aldrich
	<a href="#">add Programmer</a>	<a href="#">add Degree</a>		

The Manager combo box is visible because the attribute was defined as "required" in the associated schema. The **add...** (element/attribute) placeholders signal that the element or attribute is defined as optional. Click the placeholder to enter data at that point.

3. Fill in the rest of the table data.

Street	Noble Ave.
City	Dallas
State	Texas
Zip	04812

Employees				
Manager	Programmer	Degree	First	Last
true	false	BA	Alfred	Aldrich
false	true	MA	Fred	Smith
false	true	Ph.D	Colin	Colleti

The XML document shown above is available as **CompanyLast-Home.xml** in the `..Examples\Tutorial` folder.

## 5 XSL Transformation

### Objective

To generate an HTML file from the XML file using an XSL stylesheet to **transform** the XML file. Note that a "transformation" does not change the XML file into anything else; instead a new output file is generated. The word "transformation" is a convention.

### Method

The method used to carry out the transformation is as follows:

- Assign a predefined XSL file, `Company.xsl`, file to the XML document.
- Execute the transformation within the XMLSpy interface using one of the two built-in Altova XSLT engines. (See *note below*.)

The following XMLSpy commands are used in this section:



**XSL/XQuery | Assign XSL**, which assigns an XSL file to the active XML document.



**XSL/XQuery | Go to XSL**, opens the XSL file referenced by the active XML document.



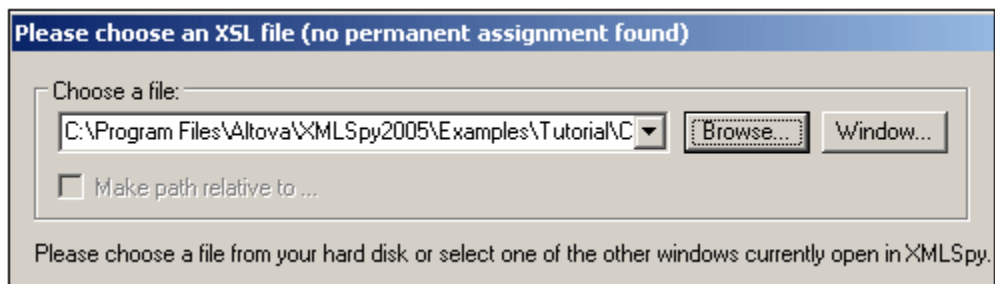
**XSL/XQuery | XSL Transformation (F10)**, or the toolbar icon, transforms the active XML document using the XSL stylesheet assigned to the XML file. If an XSL file has not been assigned then you will be prompted for one when you select this command.

**Please note:** XMLSpy has two built-in XSLT engines, the Altova XSLT 1.0 Engine and Altova XSLT 2.0 Engine. The Altova XSLT 1.0 Engine is used to process XSLT 1.0 stylesheets. The Altova XSLT 2.0 Engine is used to process XSLT 2.0 stylesheets. The correct engine is automatically selected by XMLSpy on the basis of the version attribute in the `xsl:stylesheet` or `xsl:transform` element. In this tutorial transformation, we use XSLT 1.0 stylesheets. The Altova XSLT 1.0 Engine will automatically be selected for transformations with these stylesheets when the **XSL Transformation** command is invoked.

## 5.1 Assigning an XSL file

To assign an XSL file to the `CompanyLast.xml` file:

1. Click the `CompanyLast.xml` tab in the main window so that `CompanyLast.xml` becomes the active document.
2. Click the **Text** tab.
3. Select the menu command **XSL/XQuery | Assign XSL**.
4. Click the **Browse** button, and select the `Company.xsl` file from the Tutorial folder. In the dialog, you can activate the option **Make Path Relative to CompanyLast.xml** if you wish to make the path to the XSL file (in the XML document) relative.




5. Click **OK** to assign the XSL file to the XML document.

An `XML-stylesheet` processing instruction is inserted in the XML document that references the XSL file. If you have activated the **Make Path Relative to CompanyLast.xml** check box, then the path is relative; otherwise absolute (as in the screenshot above).



## 5.2 Transforming the XML file

To transform the XML document using the XSL file you have assigned to it:

1. Ensure that the XML file is the active document.
2. Select the menu option **XSL/XQuery | XSL Transformation (F10)** or click the  icon. This starts the transformation using the XSL stylesheet referenced in the XML document. (Since the `Company.xml` file is an XSLT 1.0 document, the built-in Altova XSLT 1.0 Engine is automatically selected for the transformation.) The output document is displayed in Browser View; it has the name `XSL Output.html`. It shows the Company data in one block down the left, and the Person data in tabular form below.



**Your Company**

**Name:** US dependency  
**Street:** Noble Ave  
**City:** Dallas  
**State:** Texas  
**Zip:** 04812

First	Last	Ext.	E-Mail	Manager	Degree
Alfred	Aldrich	33	Aldrich@work.com	false	MA
Colin	Coletti	444	Coletti@work.com	true	Ph.D
Fred	Smith	22	Smith@work.com	true	BA

Text Browser

CompanyLast.xml AddressLast.xsd XSL Output.html

**Please note:** Should you only see a table header and no table data in the output file, make sure that you have defined the target namespace for your schema as detailed in [Defining your own namespace](#) at the beginning of the tutorial. The namespace must be **identical** in all three files (Schema, XML, and XSL).

## 5.3 Modifying the XSL file

You can change the output by modifying the XSL document. For example, let's change the background color of the table in the HTML output from lime to yellow.

Do the following:

1. Click **File | Open**, and browse for the `Company.xsl` file, which is in the `Examples/Tutorials` folder. This command opens the `Company.xsl` file ( *screenshot below*).

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xmlns:my="http://my-company.com/namespace">

<xsl:template match="/">
  <html>
    <head> <title>Your company</title></head>
    <body>
      <h1><center>Your Company</center></h1>
      <xsl:apply-templates select="//my:Address"/>
      <table border="1" bgcolor="lime">
        <thead align="center">
          <td><strong>First</strong></td>
          <td><strong>Last</strong></td>
          <td><strong>Ext.</strong></td>
```

2. Find the line `<table border="1" bgcolor="lime">`, and change the entry `bgcolor="lime"` to `bgcolor="yellow"`.

```
<h1><center>Your Company</center></h1>
<xsl:apply-templates select="//my:Address"/>
<table border="1" bgcolor="yellow">
  <thead align="center">
    <td><strong>First</strong></td>
    <td><strong>Last</strong></td>
```

3. Select the menu option **File | Save** to save the changes made to the XSL file.
4. Click the `CompanyLast.xml` tab to make the XML file active, and select **XSL/XQuery | XSL Transformation**, or press the **F10** key. A new `XSL Output.html` file appears in the XMLSpy GUI in Browser View. The background color of the table is yellow.

**Your Company**

**Name:** US dependency  
**Street:** Noble Ave  
**City:** Dallas  
**State:** Texas  
**Zip:** 04812

First	Last	Ext.	E-Mail	Manager	Degree
Alfred	Aldrich	33	Aldrich@work.com	false	MA
Colin	Coletti	444	Coletti@work.com	true	Ph.D
Fred	Smith	22	Smith@work.com	true	BA

5. Select the menu option **File | Save**, and save the document as `Company.html`.

# Index

## A

### Accessing,

elementTypes, 26

### Add,

child element, 7

elements with drag and drop, 10

sequence compositor, 7

### Append,

attribute, 20

attribute / element, 35

### Attribute,

adding, 20

append, 20

element, 20

entry helper, 30, 35

enumerations, 20

limiting contents, 20

window, 20

## B

### Base,

element, 15

## C

### Check,

validity, 33

well-formedness, 33

### Child element,

adding, 7

### Com,

Com(plex) tab, 15

### Complex,

type, 15

type extending, 15

### Component,

global, 15

icon, 7

navigator, 4

### Compositor,

sequence, 7

### Create,

new schema, 4

new XML document, 26

## D

### Declarations,

global, 20

### Delete,

element, 26

### Details,

entry helper, 7

tab, 11

### Display all Globals,

icon, 7

### Document,

save invalid doc., 33

### Drag and drop,

add elements with, 10

### DTD / Schema, 26

## E

### Edit,

intelligent editing, 30

### Element,

adding, 7

attributes, 20

base, 15

deleting, 26

display (shortcut), 23

facet, 11

global, 4

mandatory, 33

optional, 11

reference, 20

restricting definition, 15

### ElementTypes,

accessing, 26

### Enhanced Grid,

**Enhanced Grid,**

view, 30

**Entry helper, 2**

attribute, 30, 35

component navigator, 4

Details, 7

facet, 11

insert attribute / element, 35

update, 35

**Enumeration,**

attribute, 20

**Example files,**

tutorial, 1

**Extend,**

complex type definition, 15

## F

**Facet,**

element, 11

entry helper, 11

## G

**Global,**

component, 15

declarations, 20

element, 4

**Go to,**

Element definitions (schema), 23

**Grayed-out,**

attribute / element, 30

## I

**Info,**

window, 2

**Insert,**

attribute / element, 35

**Intelligent editing, 30****Invalid,**

document - save, 33

**isRef,**

reference, 20

## L

**Limit,**

attribute contents, 20

## M

**Main window, 2****Mandatory,**

element, 33

**Marquee,**

drag and drop, 10

**maxIncl,**

facet, 11

**maxOcc,**

details, 7, 11

**minOcc,**

details, 7, 11

## N

**Namespace,**

schema, 4

**Navigation,**

shortcuts in schemas, 23

## O

**Optional,**

element, 11

**Overview, 2**

## P

**Project,**

window, 2

## R

- Reference,**
  - element, 20
- Refresh,**
  - entry helper, 35
- Restriction,**
  - element, 15
- Revalidate, 33**
- Root element,**
  - schema, 4
  - selecting, 26

## S

- Save,**
  - anyway, 33
  - invalid document, 33
- Schema,**
  - adding elements, 7
  - creating new, 4
  - definition, 3
  - namespace, 4
  - root element, 4
  - Schema / DTD, 26
  - settings, 4
  - shortcuts, 23
  - view, 4
- Schema overview, 4**
  - display all globals, 7
- Sequence,**
  - compositor, 7
  - unbounded, 7
- Settings,**
  - schema, 4
- Shortcuts,**
  - in schema documents, 23
- Simple type, 11**
- Syntax check,**
  - well-formedness, 33

## T

- Tab,**
  - Com(plex) element, 15
  - details, 11
- Template folder, 1**
- Text view, 30**
- Tutorial,**
  - example files, 1
  - goals, 1
- Type,**
  - AddressType, 15
  - complex, 15
  - simple, 11

## U

- Unbounded,**
  - element, 7
- Update,**
  - entry helper, 35

## V

- Valid,**
  - invalid document - save, 33
- Validate,**
  - check validity, 33
  - revalidate, 33
  - XML document, 33
- View,**
  - content model (shortcut), 23
  - Enhanced Grid, 30
  - Schema view, 4
  - Text view, 30

## W

- Well-formedness, 33**
- Windows,**

**Windows,**  
overview, 2

## X

**XML,**  
document and icons, 25  
new document, 26  
validate document, 33

**xs,**  
xs:boolean, 20  
xs:date, 11  
xs:integer, 11  
xs:positiveInteger, 15  
xs:string, 7, 11

**xsi,**  
xsi:type, 26

**XSL transformation,**  
see XSLT, 36, 37, 38, 39

**XSLT,**  
modifying in XMLSpy, 39

**XSLT transformation,**  
assigning XSLT file, 37  
in XMLSpy, 38  
tutorial, 36