Altova XMLSpy 2025 Enterprise Edition

Tutorial

Altova XMLSpy 2025 Enterprise Edition Tutorial

Table of Contents

1 XMLSpy Tutorial

This tutorial provides an overview of XML and takes you through a number of key XML tasks. In the process you will learn how to use some of XMLSpy's most powerful features.

The tutorial is divided into the following parts:

- XMLSpy Interface [6], which helps you to familiarize yourself with the applications's graphical user interface (GUI).
- Creating an XML Schema [16]. You will learn how to create an XML Schema in XMLSpy's intuitive Schema View, how to create complex content models using drag-and-drop mechanisms, and how to configure Schema View.
- Using Schema View features [31] to create complex and simple types, global element references, and attribute enumerations.
- Learning how to navigate schemas [44] in Schema View, and how to generate documentation of schemas [44].
- Creating an XML document [51]. You will learn how to assign a schema for an XML document, edit an XML document in Grid View and Text View, and validate XML documents using XMLSpy's built-in validator.
- Transforming an XML file using an XSLT stylesheet [72]. This involves assigning an XSLT file and carrying out the transformation using XMLSpy's built-in XSLT engines.
- Working with XMLSpy projects [76], which enable you to easily organize your XML documents.

Installation and configuration

This tutorial assumes that you have successfully installed XMLSpy on your computer and received a free evaluation key-code, or are a registered user. The evaluation version of XMLSpy is fully functional but limited to a 30-day period. 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 application folder:

`C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2025\Examples\Tutorial`

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 in the application folder (typically in `C:\Program Files\Altova`) 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.

1.1   XMLSpy Interface

In this section of the tutorial, you will start XMLSpy and get to know the interface.

Starting XMLSpy

To start XMLSpy, double-click the XMLSpy icon on your desktop or use the **Start | All Programs** menu to access the XMLSpy program. XMLSpy is started with no documents open in the interface. Open XMLSpy now.

Overview of the interface

The default view of the XMLSpy interface is structured into three vertical areas (*figure below*). These three areas contain, from left to right: (i) the Project and Info windows; (ii) the Main and Output windows; and (iii) the Entry Helper windows. Look at the Project window. It will contain the Examples project, which is opened by default when you start XMLSpy for the first time.

Given below are key points that will help you to understand the layout of the interface and the functions of its various components. The sub-sections of this first part of the tutorial will help you get familiar with the interface.
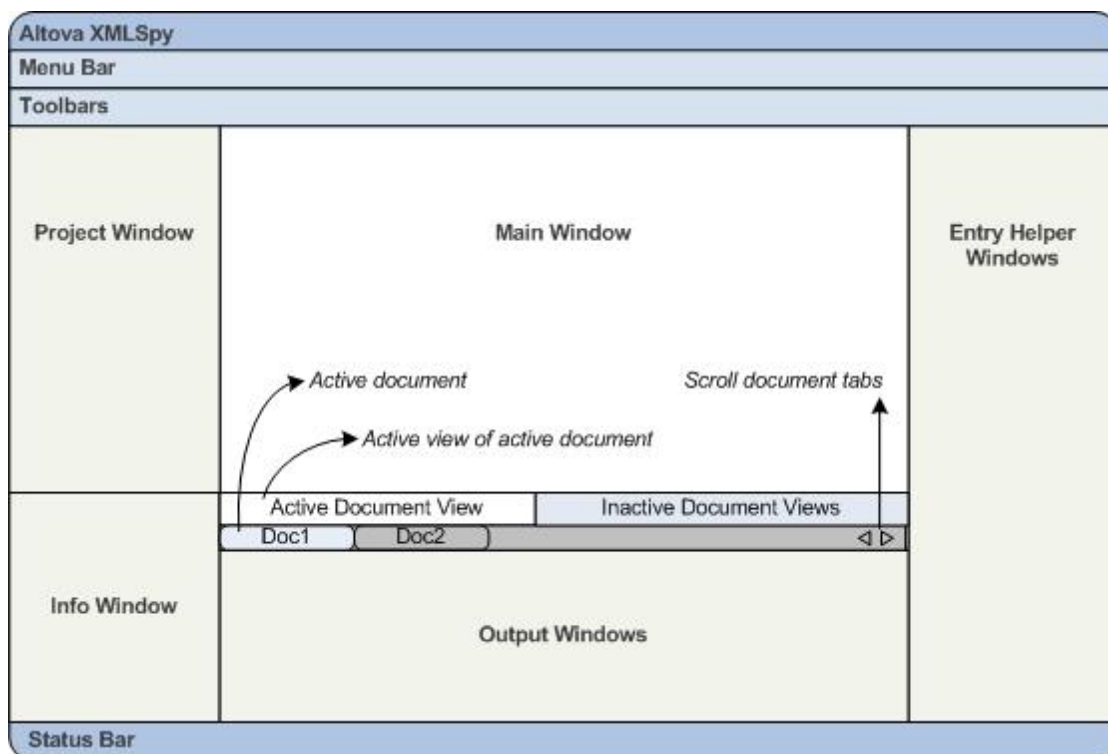
Document bar in the Main window: When multiple documents are open, each document is displayed in a tab in the document bar of the Main window (*see figure*). Clicking a tab makes that document the active document. You can scroll document tabs by clicking the arrows on the right hand side of the document bar. Open two or more files (for example, from the Examples project), and check how the tabs work.

Document editing views: The active document can be viewed in one of multiple applicable editing views. For example:

- An XML (`.xml`) document can be viewed in Text View, Grid View, Authentic View, and Browser View, but cannot be viewed in other views, such as Schema View.
- An XML Schema (`.xsd`) document, on the other hand can be viewed in Text View, Grid View, Schema View, and Browser View, but not in Authentic View.

The following views are available: Text View, Grid View, Schema View, Authentic View, Archive View, and Browser View.

Entry helpers: The entry helper windows change according to the kind of the active document (for example, XML or XSD or CSS or WSDL) and according to the currently active document view (for example, Text View or Schema View). The entry helpers enable you to quickly and correctly edit the active document by providing context-sensitive editing support.

1.1.1   The Views

In this part of the tutorial you will learn: (i) to switch between document editing views, and (ii) to change the default editing view of a particular document type.

Switching between document views

When you open a document it will open in the view that has been set as the default view for that type of document. Open a document as follows:

1. Click the command **File | Open**.
2. Browse for the file **AddressFirst.xsd**, which is located in the `C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2025\Examples\Tutorial` folder, select it, and click **Open**. The file opens in Schema View.
3. Switch among the various views by clicking the view tabs at the bottom of the Main window (Text View, Grid View, etc). You will be able to view the XML Schema document in Text View, Grid View, Schema View, and Browser View.
4. You can also change views by selecting the view you want from the options in the **View** menu. Try switching the view of the **AddressFirst.xsd** document using the **View** menu commands.
5. Close the document (via **File | Close**).

Changing the default view of a document type

All documents with the **.xsd** extension will open by default in Schema View. You can change the default opening view of any type of document in the Options dialog. Let us do this for `.xsd` documents now.

1. Click the command **Tools | Options** and go to the *File Types* section (*screenshot below*).
2. In the *File Types* pane, scroll down to **.xsd** and select it (*highlighted in screenshot*).
3. In the *Default View* pane, select Text View.
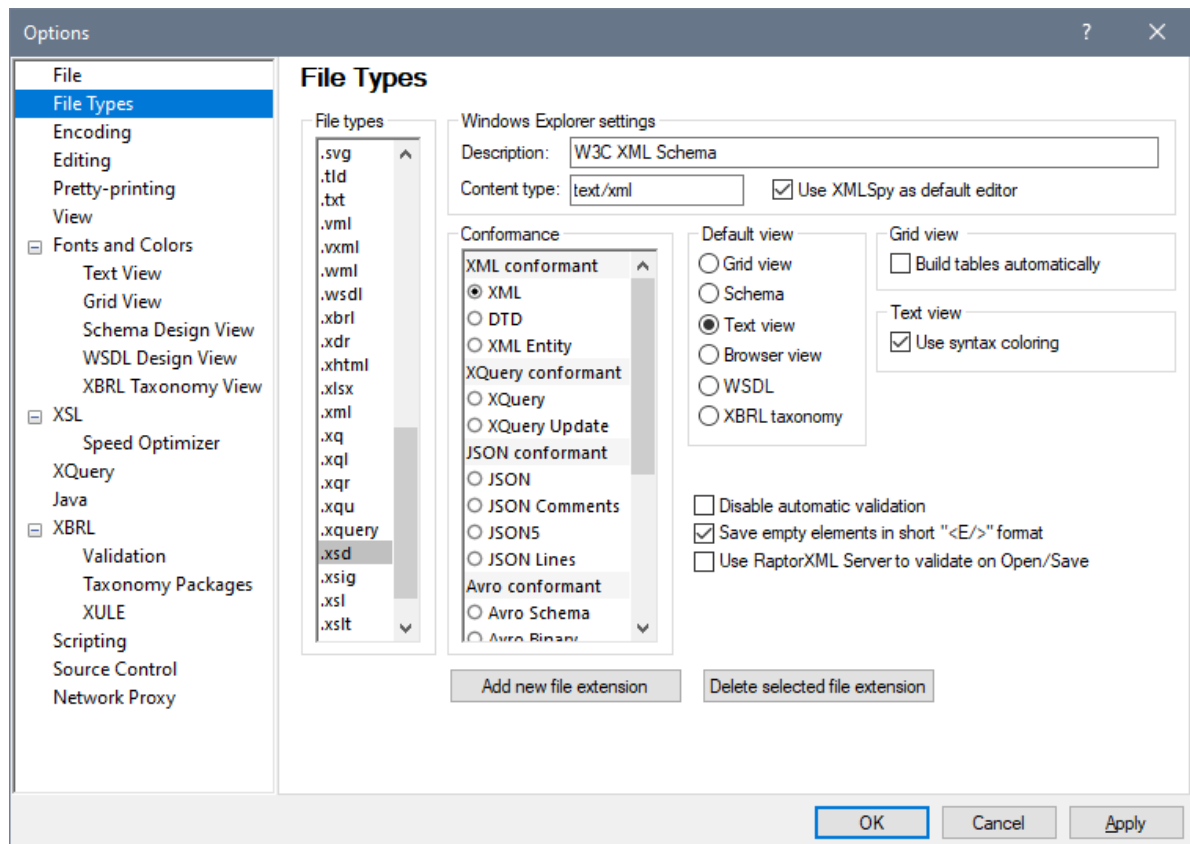
4. Click **OK**.
5. Click the **File | Open** command, and open the file `AddressFirst.xsd`. The file opens in Text View.
6. Switch to Schema View to see the file in this view, then close the file (**File | Close**).
7. Go back to the Options dialog (**Tools | Options**), and, in the *File Types* section, change the default view of `.xsd` files back to Schema View.

Note: In the *File Types* section of the Options dialog (*screenshot above*), you can change the default view of any of the listed file extensions. A new file extension can be added to the list via the **Add New File Extension** button.

1.1.2   The Windows

By default, the various windows are located around the Main window (*see screenshot below*) and are organized into the following window groups:

· Project window
· Info window
· Entry helpers (various, depending on the type of document currently active)
· Output windows: Messages, XPath, XSL Outline, Find in Files, Find in Schemas

```
┌─────────────────────────────────────────────────────────────────┐
│ Altova XMLSpy                                                      │
├─────────────────────────────────────────────────────────────────┤
│ Menu Bar                                                          │
├─────────────────────────────────────────────────────────────────┤
│ Toolbars                                                         │
├──────────────┬──────────────────────────────┬───────────────────┤
│              │                              │                   │
│              │                              │                   │
│ Project      │      Main Window             │  Entry Helper     │
│ Window       │                              │  Windows          │
│              │                              │                   │
│              │   → Active document          │ Scroll document tabs │
│              │   → Active view of active document │             │
│              ├──────────────────────────────┤                   │
│              │ Active Document View │ Inactive Document Views │   │
│              │ [Doc1] [Doc2]                        ◁ ▷ │       │
│ Info Window  │                              │                   │
│              │      Output Windows          │                   │
│              │                              │                   │
├──────────────┴──────────────────────────────┴───────────────────┤
│ Status Bar                                                       │
└─────────────────────────────────────────────────────────────────┘
```
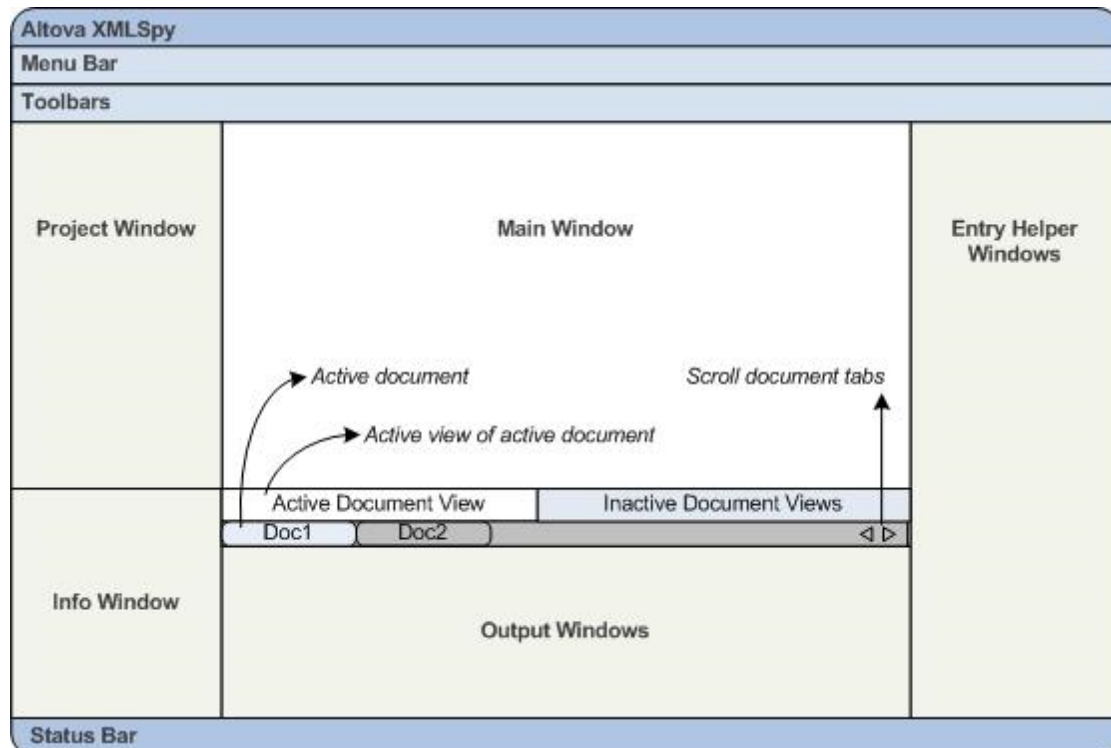
In this section, you will learn how to turn on and off the display of window groups and how to move windows around the screen. Being able to manage the display of windows well will be useful when you need more space within the interface.

Switching the display of window groups on and off

Window groups (Project Window, Info Window, Entry Helpers, Output Windows) can be displayed or hidden by toggling them on and off via the commands in the **Window** menu. A displayed window group can also be hidden by right-clicking its title bar and selecting the command **Hide**. A hidden window can only be displayed via the **Window** menu.

Open any XML file in the `C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2025\Examples\Tutorial` folder, and practise using these basic commands till you are familiar with the way the commands work. For more information about displaying and hiding window groups, see the section, XMLSpy Interface.

Saved and backup status

By default, XMLSpy backs up unsaved documents at intervals of five seconds. Each file's tab at the bottom of the Main Window provides information via indicator symbols about the file's saved/unsaved and its backup status. You should be aware of the meanings of these indicators since you will come across them constantly during your work. See the Automatic Backup of Files section for information about these indicators.

Moving windows around the screen

An individual window can either float free of the interface or be docked within it. A window can also be docked as a tab within a window group (*window groups are explained above*). For example, the screenshot below

---

shows the Components entry helper in Schema View, which has three tabbed windows: the Globals window, Namespaces window, and Identity Constraints window.
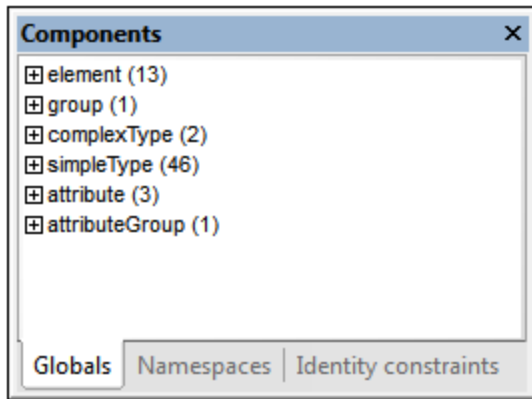
A window can be made to float or dock using one of the following methods in any view:

- Double-click the title bar of the window. If docked, the window will now float. If floating, the window will now dock in the last position in which it was docked.
- Right-click the title bar of a window and choose the required command (**Floating** or **Docking**).
- Drag the window (using its title bar as a handle) out of its docked position so that it floats. Drag a floating window (by its title bar) to the location where it is to be docked. Two sets of blue arrows appear. The outer set of four arrows enables docking relative to the application window (along the top, right, bottom, or left edge of the GUI). The inner set of arrows enables docking relative to the window over which the cursor is currently placed. Dropping a dragged window on the button in the center of the inner set of arrows (or on the title bar of a window) docks the dragged window as a tabbed window within the window in which it is dropped.

To float a tabbed window, double-click its tab. To drag a tabbed window out of a group of tabbed windows, drag its tab.

To practise moving windows around open any XML Schema file from the `C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2025\Examples\Tutorial` folder, and, while in Schema View, try the methods described above till you are able to move windows around the interface comfortably.

1.1.3   Menus and Toolbars

In this section of the tutorial, you will quickly learn about the main features of the menus and toolbars of XMLSpy.

Menus

There are two menu bars: (i) a default menu that is displayed when no document is open, and (ii) the full XMLSpy application menu, which is displayed as soon as a document is open. Do the following:

1. Close all open documents with the menu command **File | Close All**. You will see the default menu.

2. Open the `AddressFirst.xsd` file by clicking its name from the list of most recently opened files located at the bottom of the **File** menu. When the file opens in Schema View, the menu will change to the full XMLSpy application menu.

The menus are organized primarily according to function, and a command in a menu is enabled only when it can be executed at the cursor point or for a selection in the current view of the active document. Do the following to understand the factors that determine whether a menu command is enabled or disabled:

1. Click the **Schema Design** menu. Notice that the **Save Diagram**, **Configure View**, and **Zoom** commands are disabled (*screenshot below*).
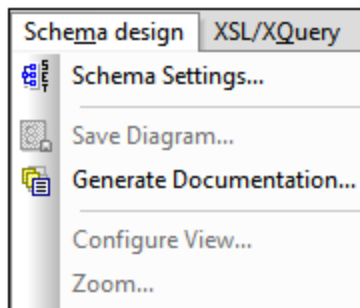

2. Click in a blank space outside the menu to make the menu disappear. Then click the **Display Diagram** icon located to the left of the element component. This takes you to the Content Model View of Schema View (the second of Schema View's two views; the first is Schema Overview). If you check the Schema Design menu now, you will see that the **Save Diagram**, **Configure View**, and **Zoom** commands have been enabled. They are enabled only in the Content Model View of Schema View, not in the Schema Overview of Schema View, nor in any other view. Note also that only XML Schema files can be opened in Schema View.
3. An XML Schema file is also an XML file, so it is displayed as an XML file in Text View and Grid View, and all menu commands that apply to XML files will be enabled in these views. Compare commands in the **Edit** menu (whether they are enabled or not) in Schema View and Text View.
4. Next compare commands in the **XML | Insert** menu (enabled or disabled) in Text View and Grid View. The commands in this menu are enabled only in Grid View.

For descriptions of all the menu commands, see the User Reference section of the user documentation.

Toolbars

The display of toolbars varies according to the current view. The application's default settings provide the correct toolbars for each view and will be different for each view. However, you can customize toolbars in the *Toolbars* tab of the Customize dialog (**Tools | Customize | Toolbars**, *screenshot below*).
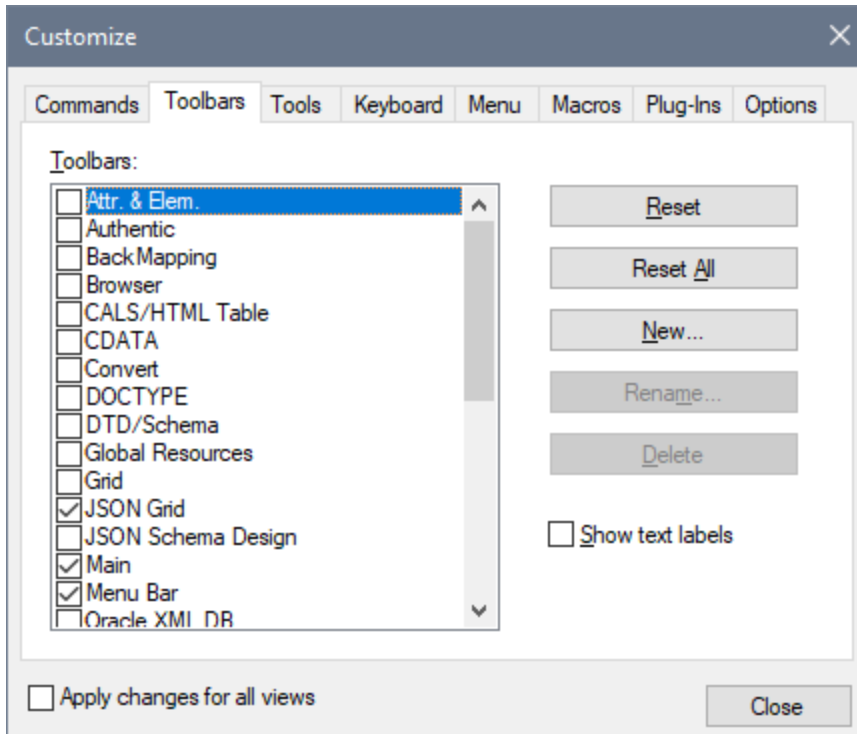
Now practise moving toolbars around the GUI. Click the handle of a toolbar and drag the toolbar to any desired location in the GUI. (The toolbar handle is indicated by the dotted vertical line at the left of each toolbar; *see screenshot below*.)

Try dragging a toolbar to the following locations: (i) another line in the toolbar area; (ii) left or right of another toolbar; (iii) the middle of the Main window; (iv) docked to the left or right side of the application window (for this to happen, the grab handle must be placed above the left or right border of the application window).

After you have finished, close the file **AddressFirst.xsd**.

1.1.4   **Text View Settings**

In this section, you will learn how to set up a "pretty-printed" document and how to use bookmarks while editing. When a document is pretty-printed it is displayed in Text View so that each lower level in the XML hierarchy is indented deeper than the previous level (*see screenshot below*). Bookmarks enable you to mark document positions that you wish to return to quickly.

```
 1    <?xml version="1.0" encoding="UTF-8"?>
 2    <Company>
 3        <Address xsi:type="US-Address">
 4            <Name>US dependency</Name>
 5            <Street>Noble Ave.</Street>
 6            <City>Dallas</City>
 7            <Zip>04812</Zip>
 8            <State>Texas</State>
 9        </Address>
10        <Person Manager="true" Degree="BA" Programmer="false"
11            <First>Fred</First>
12            <Last>Smith</Last>
13            <PhoneExt>22</PhoneExt>
14            <Email>Smith@work.com</Email>
15        </Person>
16    </Company>
```

Pretty-printing

Pretty-printing involves two steps: (i) setting up pretty-printing, (ii) applying pretty-printing.

1. Open the file `CompanyFirst.xml`, which is in the `C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2025\Examples\Tutorial` folder.
2. Switch to Text View if Text View is not the default starting view of XML documents.
3. Select the menu command **View | Text View Settings** to open the Text View Settings dialog (*screenshot below*).
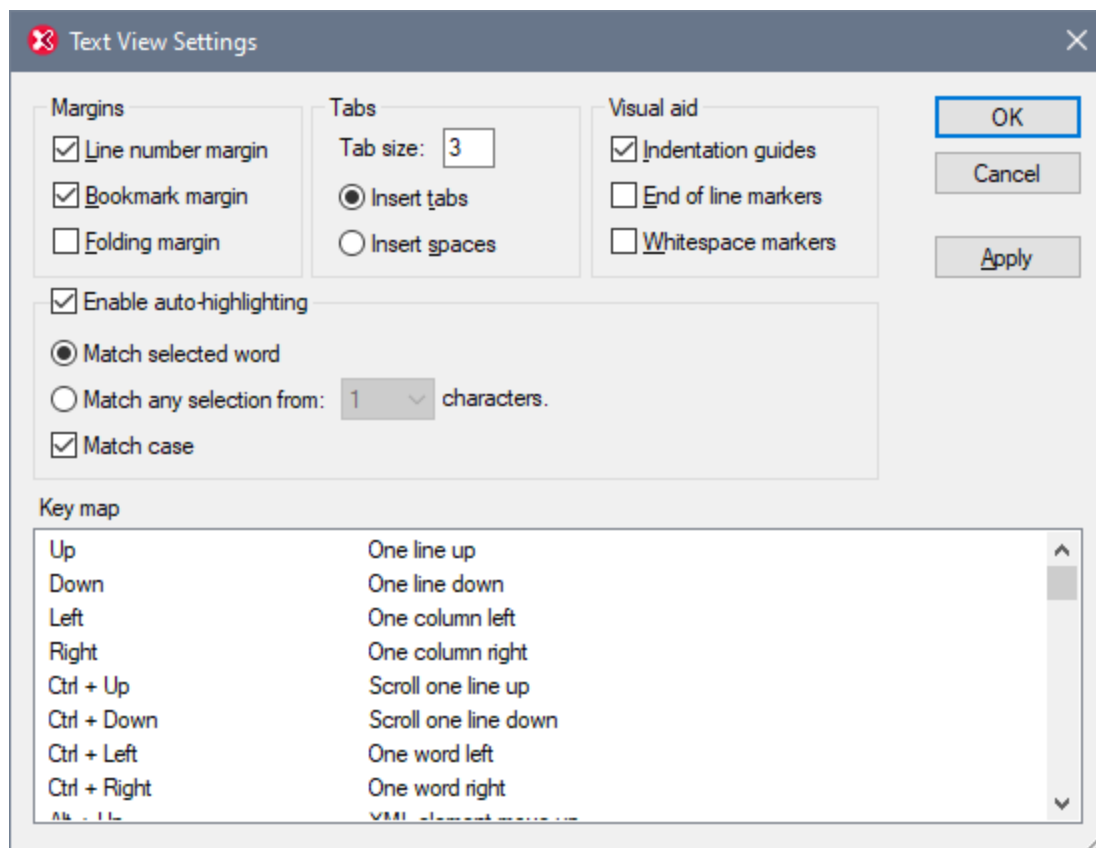
4. In the Tabs pane, decrease the Tab size to 3. Leave the default selection of the *Insert Tabs* radio button as it is. This will cause the pretty-printing indent to be a tab (rather than spaces) and each tab will have a width of three spaces. Click **OK** when done.
5. Click the menu command **Edit | Pretty-Print**. This applies pretty printing. The document display will be reset with the new tab values.
6. Open the Text View Settings dialog again (**View | Text View Settings**) and, in the *Visual Aid* pane, switch on the end-of-line markers.
7. In Text View, go to the end of any line and delete the end-of-line marker so that the next line jumps up a line.
8. Switch to Grid View and back again to Text View.
9. Click the menu command **Edit | Pretty-Print**. The document will be pretty-printed, and the the end-of-line marker you deleted will be reinstated.
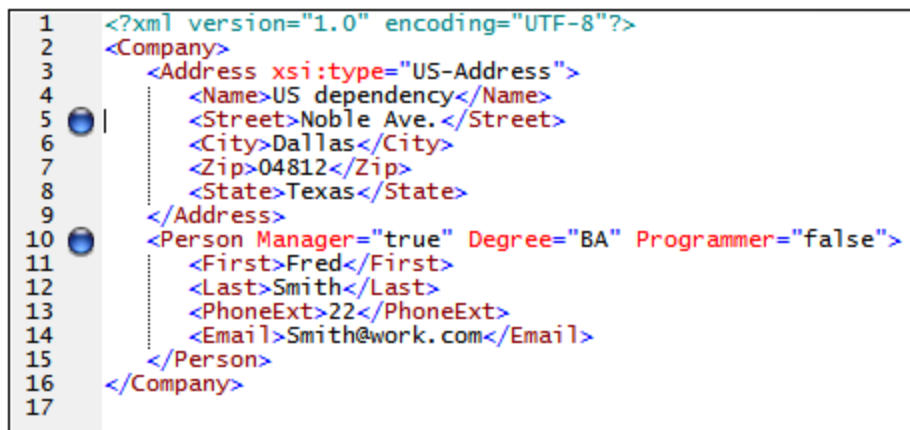
Note: If, in the Pretty-printing section of the Options dialog (**Tools | Options | Pretty-printing**), you uncheck the *Use Indentations* check box and pretty-print, then all lines will begin without any indentation.

Bookmarking

Bookmarks are placed in a bookmark margin on the left of lines you wish to mark. You can then quickly move up and down through the bookmarks in your document.

1. In the Text View Settings dialog (**View | Text View Settings**, *screenshot above*) ensure that the Bookmarks Margin option in the *Margins* pane is selected. Click **OK** when done.

2. In Text View of the file `CompanyFirst.xml`, place the cursor anywhere in a line you wish to bookmark, then select the menu command **Edit | Insert/Remove Bookmark**. The line will be bookmarked and indicated by a blue bookmark in the bookmark margin (*see screenshot below*).
3. Create a bookmark on another line in the same way as in Step 2.

```
1     <?xml version="1.0" encoding="UTF-8"?>
2     <Company>
3         <Address xsi:type="US-Address">
4             <Name>US dependency</Name>
5  ⬤|        <Street>Noble Ave.</Street>
6             <City>Dallas</City>
7             <Zip>04812</Zip>
8             <State>Texas</State>
9         </Address>
10 ⬤    <Person Manager="true" Degree="BA" Programmer="false">
11            <First>Fred</First>
12            <Last>Smith</Last>
13            <PhoneExt>22</PhoneExt>
14            <Email>Smith@work.com</Email>
15        </Person>
16    </Company>
17
```

4. Press **F2** (or the command **Edit | Go to Next Bookmark**) to go down the document to the next bookmark. Press **Shift+F2** (or the command **Edit | Go to Previous Bookmark**) to go up the document to the previous bookmark. Repeat either or both commands as many times as you like.
5. Place the cursor in one of the bookmarked lines and select the menu command **Edit | Insert/Remove Bookmark**. The bookmark is removed.
6. Save and close the file. No bookmark information is saved with the file. Reopen the file to check this.

1.2   XML Schemas: Basics

An XML Schema describes the structure of an XML document. An XML document can be validated against an XML Schema to check whether it conforms to the requirements specified in the schema. If it does, it is said to be **valid**; otherwise it is **invalid**. XML Schemas enable document designers to specify the allowed structure and content of an XML document and to check whether an XML document is valid.

The structure and syntax of an XML Schema document is complex, and being an XML document itself, an XML Schema must be valid according to the rules of the XML specification. In XMLSpy, Schema View enables you to easily build valid XML Schemas by using graphical drag-and-drop techniques. The XML Schema document you construct is also editable in Text View and Grid View, but is much easier to create and modify in Schema View.

Objective

In this section of the tutorial, you will learn how to edit XML Schemas in Schema View. Specifically, you will learn how to do the following:

- Create a new schema file
- Define namespaces for the schema
- Define a basic content model
- Add elements to the content model using context menus and drag-and-drop
- Configure the Content Model View

After you have completed creating the basic schema, you can go to the next section of the tutorial [31], which teaches you how to work with the more advanced features of XML Schema in XMLSpy. This advanced section is followed by a section about schema navigation and documentation [44] in XMLSpy.
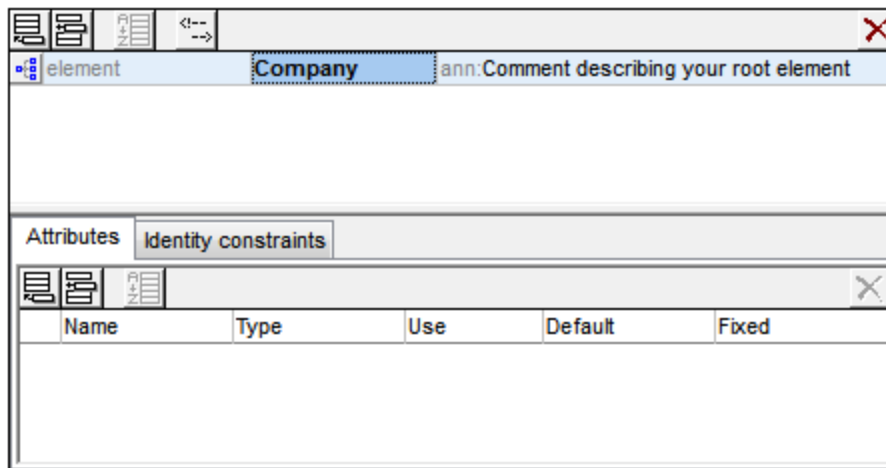
Commands used in this section

In this section of the tutorial, you will use Schema View exclusively. The following commands are used:

▣	Display Diagram (or Display Content Model View). This icon is located to the left of all global components in Schema Overview. Clicking the icon causes the content model of the associated global component to be displayed.

1.2.1   Creating a New XML Schema File

To create a new XML Schema file:

1. Select the menu option **File | New**. The Create new document dialog opens.

2. In the dialog, select the *XSD (XML Schema v1.0)* entry (the document description and the list in the window might vary from that in the screenshot) and confirm with **OK**. An empty schema file appears in the Main Window in Schema View.

3. In the Schema Design toolbar click the **XSD 1.0** mode button (*see screenshot below*) so that Schema View is in XSD 1.0 editing mode.

4. You are prompted to enter the name of the root element.

5. Double-click in the highlighted field and enter `Company`. Confirm with **Enter**. `Company` is now the root element of this schema and is created as a global element. The view you see in the Main Window *(screenshot below)* is called the Schema Overview. It provides an overview of the schema by displaying a list of all the global components in the top pane of the Main Window; the bottom pane displays the attributes and identity constraints of the selected global component. (You can view and edit the content model of individual global components by clicking the Display Diagram icon to the left of that global component.)

6. In the Annotations field (**ann**) of the `Company` element, enter the description of the element, in this case, *Root element*.
7. Click the menu option **File | Save**, and save your XML Schema with any name you like (**AddressFirst.xsd**, for example).

The colored-circle symbols in the file's tab indicate the file's backup status. See Automatic Backup of Files for a description of these indicators.

1.2.2   Defining Namespaces

XML namespaces are an important issue in XML Schemas and XML documents. An XML Schema document must reference the XML Schema namespace and, optionally, it can define a target namespace for the XML document instance. As the schema designer, you must decide how to define both these namespaces (essentially, with what prefixes.)

In the XML Schema you are creating, you will define a target namespace for XML document instances. (The required reference to the XML Schema namespace is created automatically by XMLSpy when you create a new XML Schema document.)

To create a target namespace:

1. Select the menu option **Schema Design | Schema Settings**. This opens the Schema Settings dialog (*screenshot below*).
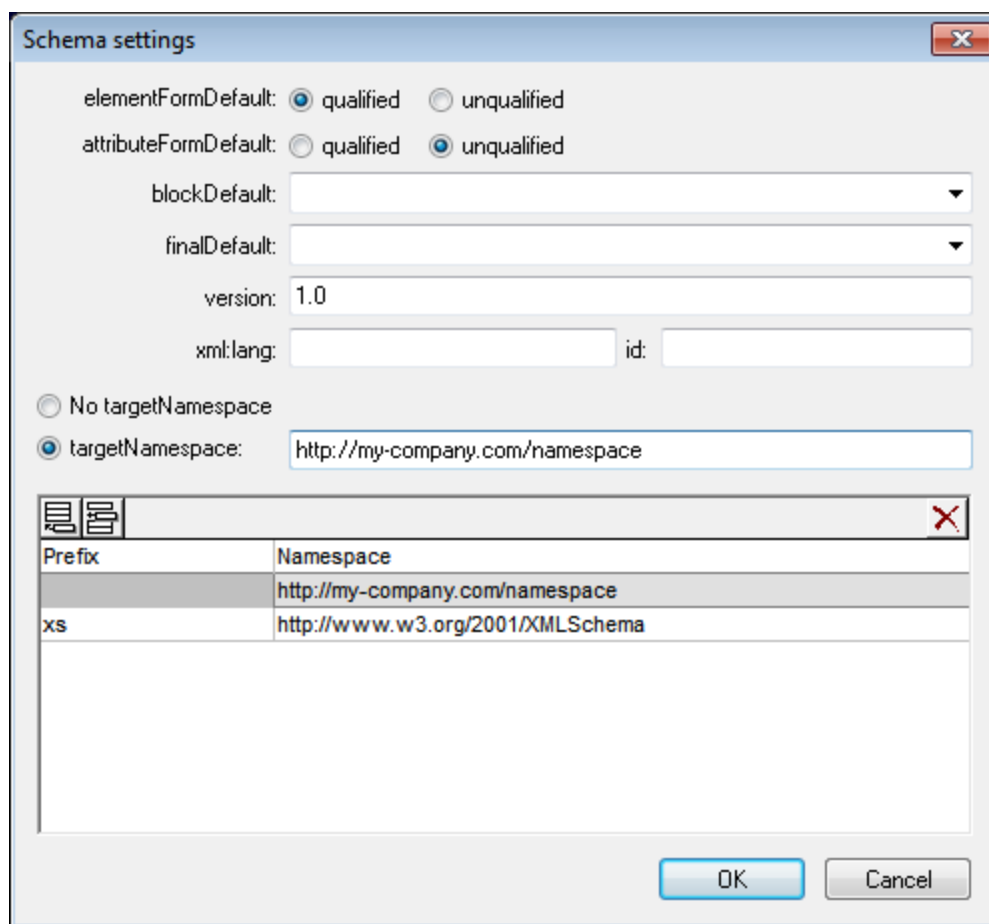
2. Click the *Target Namespace* radio button, and enter `http://my-company.com/namespace`. In XMLSpy, the namespace you enter as the target namespace is created as the default namespace of the XML Schema document and displayed in the list of namespaces in the bottom pane of the dialog.
3. Confirm with the **OK** button.

Note the following:

*   The XML Schema namespace is automatically created by XMLSpy and given a prefix of `xs:`.
*   When the XML document instance is created, it must have the target namespace defined in the XML Schema for the XML document to be valid.

1.2.3   Defining a Content Model

In Schema Overview, you have already created a global element called `Company`. This element is to contain one `Address` element and an unlimited number of `Person` elements. This then is the Company element's content model. Global components that may have content models are: elements, complexTypes, and element groups. In XMLSpy, the content model of a global component is displayed in the Content Model View of Schema View.

To view and edit the content model of a global component, click the Display Diagram icon located to the left of the global component.
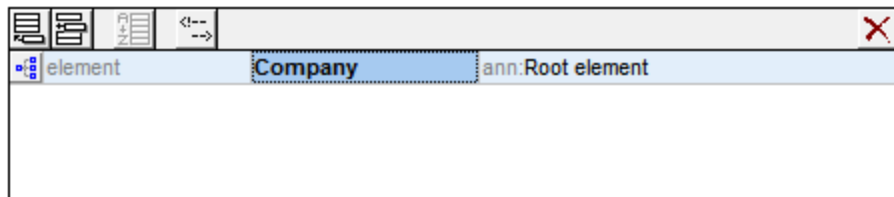
In this section, you will create the content model of the `Company` element.

Creating a basic content model

To create the content model of the `Company` element:

1. In Schema Overview, click the Display Diagram icon of the `Company` element. This displays the content model of the `Company` element (*screenshot below*), which is currently empty. Alternatively, you can double-click the `Company` entry in the Components entry helper to display its content model.
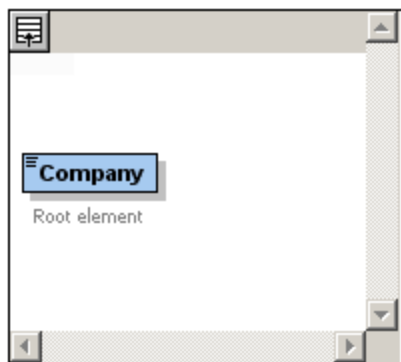
2. A content model consists of **compositors** and **components**. The compositors specify the relationship between two components. At this point of the `Company` content model, you must add a child compositor to the `Company` element in order to add a child element. To add a compositor, right-click the `Company` element. From the context menu that appears, select **Add Child | Sequence**. (Sequence, Choice, and All are the three compositors that can be used in a content model.)
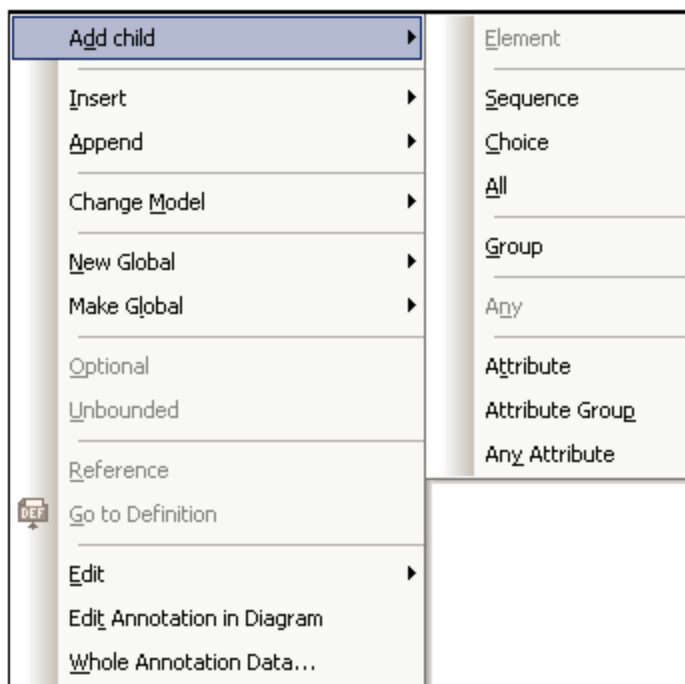
This inserts the Sequence compositor, which defines that the components that follow must appear in the specified sequence.
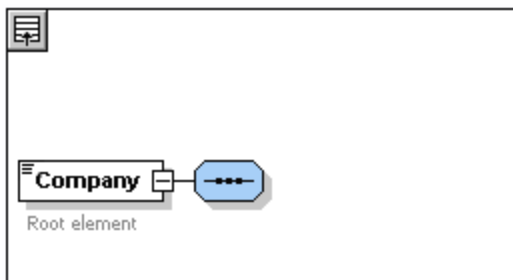
3. Right-click the Sequence compositor and select **Add Child | Element**. An unnamed element component is added.

4. Enter **Address** as the name of the element, and confirm with **Enter**.
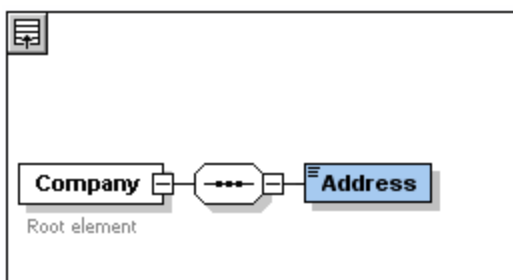
5. Right-click the Sequence compositor again, select **Add Child | Element**. Name the newly created element component `Person`.
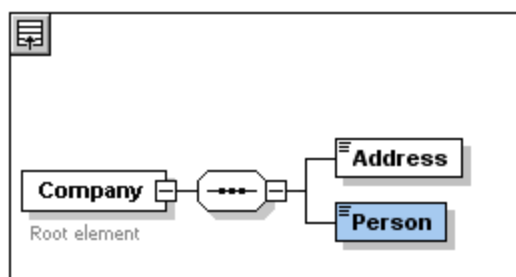
You have so far defined a schema which allows for one address and one person per company. We need to increase the number of **Person** elements.

6.   Right-click the **Person** element, and select **Unbounded** from the context menu. The **Person** element in the diagram now shows the number of allowed occurrences: 1 to infinity.
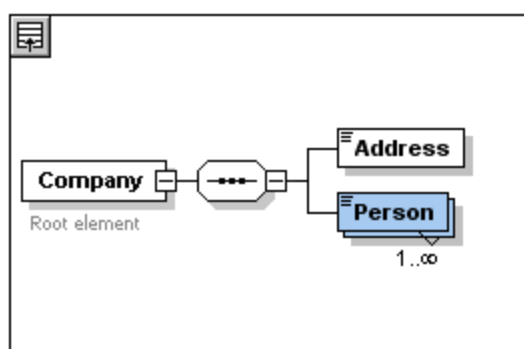
Alternatively, in the Details Entry Helper, you can edit the **minOcc** and **maxOcc** fields to specify the allowed number of occurrences, in this case 1 and unbounded, respectively.

Adding additional levels to the content model structure

The basic content model you have created so far contains one level: a child level for the **company** element which contains the **Address** and **Person** elements. Now we will define the content of the **Address** element so it contains **Name**, **Street**, and **City** elements. This is a second level. Again we need to add a child compositor to the **Address** element, and then the element components themselves.

Do this as follows:

1.   Right-click the **Address** element to open the context menu, and select **Add Child | Sequence**. This adds the Sequence compositor.
2.   Right-click the Sequence compositor, and select **Add Child | Element**. Name the newly created element component **Name**.
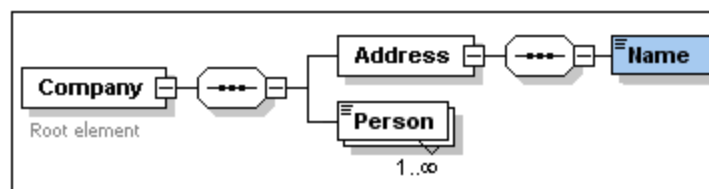
Complex types, simple types, and XML Schema data types

Till this point, we have not explicitly defined any element type. Click the **Text** tab to display the Text View of your schema (*listing below*). You will notice that whenever a Sequence compositor was inserted, the `xs:sequence` element was inserted within the **xs:complexType** element. In short, the **Company** and **Address** elements, because they contain child elements, are complex types. A complex type element is one which contains attributes or elements.

```
<xs:element name="Company">
   <xs:annotation>
     <xs:documentation>Root element</xs:documentation>
   </xs:annotation>
   <xs:complexType>
     <xs:sequence>
       <xs:element name="Address">
         <xs:complexType>
           <xs:sequence>
             <xs:element name="Name"/>
           </xs:sequence>
         </xs:complexType>
       </xs:element>
       <xs:element name="Person"/>
     </xs:sequence>
   </xs:complexType>
</xs:element>
```

Simple type elements, on the other hand, contain only text and have no attributes. Text can be strings, dates, numbers, etc. We want the **Name** child of **Address** to contain only text. It is a simple type, the text content of which we want to restrict to a string. We can do this using the XML Schema data type **xs:string**.

To define the **Name** element to be of this datatype:

1. Click the **Schema** tab to return to Schema View.
2. Click the **Name** element to select it.
3. In the Details Entry Helper, from the dropdown menu of the `type` combo box, select the **xs:string** entry.
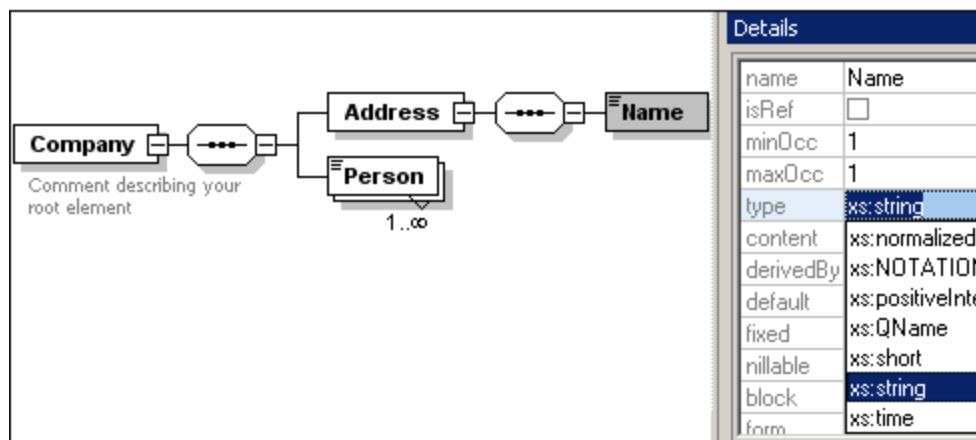
Note that both **minOcc** and **maxOcc** have a value of 1, showing that this element occurs only once.

The text representation of the `Name` element is as follows:

```
<xs:element name="Name" type="xs:string"/>
```

Note: A simple type element can have any one of several XML Schema data types. In all these cases, the icon indicating text-content appears in the element box.

1.2.4   Adding Elements with Drag-and-Drop

You have added elements using the context menu that appears when you right-click an element or compositor. You can also create elements using drag-and-drop, which is quicker than using menu commands. In this section, you will add more elements to the definition of the `Address` element using drag-and-drop, thus completing this definition.

To complete the definition of the Address element using drag-and-drop:

1. Click the **Name** element of the **Address** element, hold down the **Ctrl** key, and drag the element box with the mouse. A small "plus" icon appears in the element box, indicating that you are about to copy the element. A copy of the element together with a connector line also appears, showing where the element will be created.
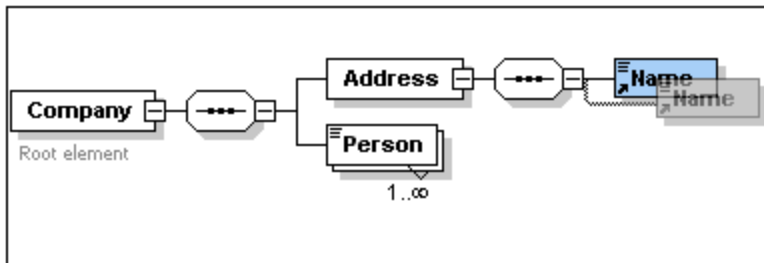
2. Release the mouse button to create the new element in the **Address** sequence. If the new element appears at an incorrect location, drag it to a location below the **Name** element.
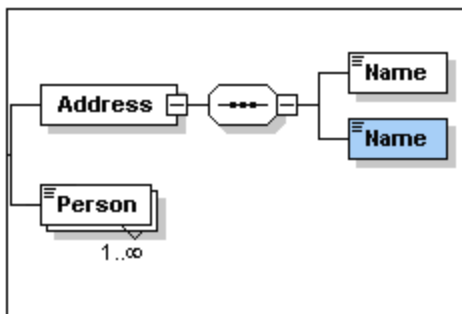
3. Double-click in the element box, and type in **street** to change the element name.
4. Use the same method to create a third element called **city**. The content model should now look like this:
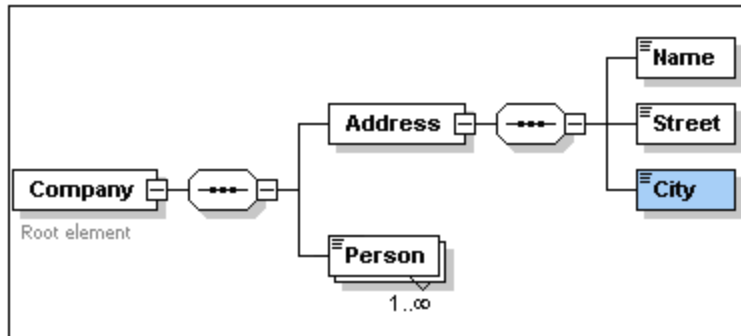
The `Address` element has a sequence of a `Name`, a `Street`, and a `City` element, in that order.

1.2.5   Configuring the Content Model View

This is a good time to configure the Content Model View. We want to configure the view so that an element's type is displayed in the element's box. Do this as follows:

1. Select the Content Model View of any component (by clicking its Content Model View icon).
2. When in Content Model View, the menu command **Schema Design | Configure View** is enabled. Select the command to display the Schema Display Configuration dialog (*screenshot below*).
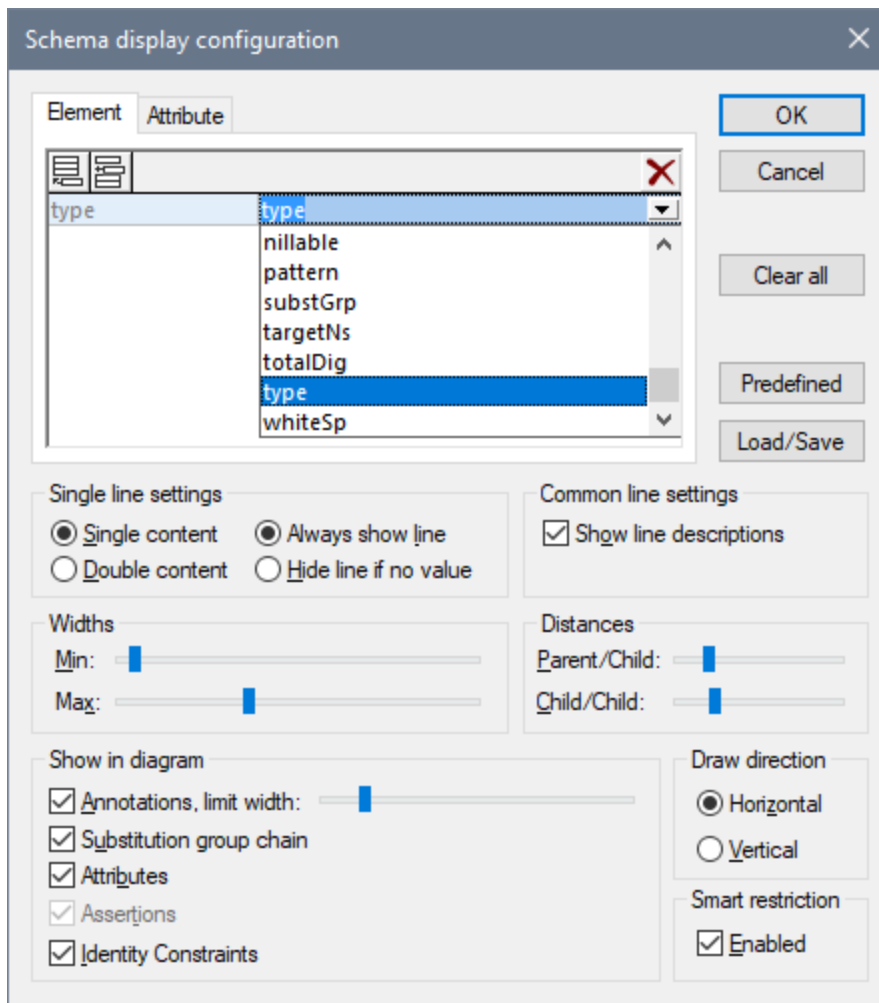
3.   In the *Element* tab (*see screenshot above*), click the **Append** icon, and select *Type (see screenshot)* to add this property descriptor line to element boxes.

4.   In the *Single Line Settings* pane, select *Hide line if no value*. This hides the description of the datatype in the element box if the element does not have a datatype (for example, if the element is a complex type). In the screenshot below, notice that the type descriptor line appears for the Name, Street, and City elements, which are simple types of type xs:string, but not for the complex type elements. This is because the Hide Line If No Value toggle is selected.
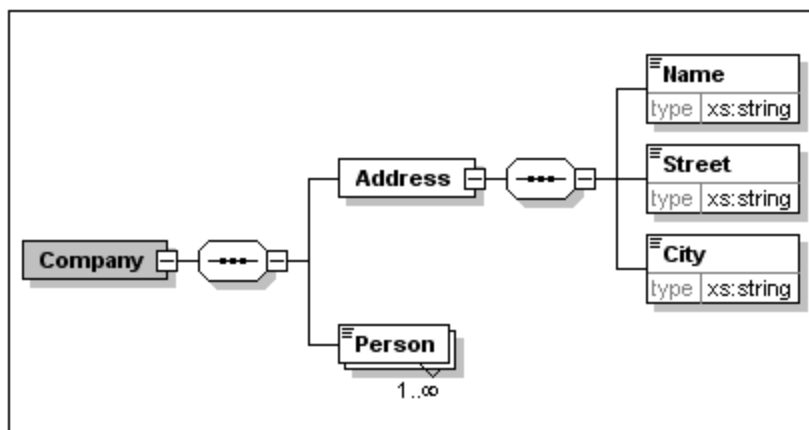
5.   In the *Single Line Settings* pane, select the *Always Show Line* radio button.
6.   Click **OK** to confirm the changes.
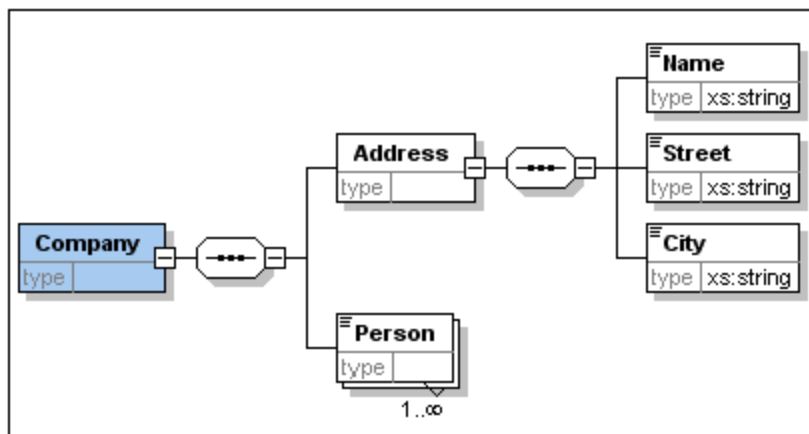
Notice that the descriptor line for the data type is always shown—even in element boxes of complex types, where they appear without any value.

Note the following:

*   The property descriptor lines are editable, so values you enter in them become part of the element definition.
*   The settings you define in the Schema display configuration dialog apply to the schema documentation output as well as the printer output.

1.2.6   Completing the Basic Schema

You have defined the content of the `Address` element. Now you need to define the content of the `Person` element, which must contain the following simpleType child elements: `First`, `Last`, `Title`, `PhoneExt`, and `Email`. All these elements must be mandatory, except `Title`, and they must occur in the order just specified. All should be of type `xs:string` except `PhoneExt`, which must be of type `xs:integer` and limited to two digits.

To create the content model for `Person`, do the following:

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 press the **Tab** key. This automatically places the cursor in the type field.
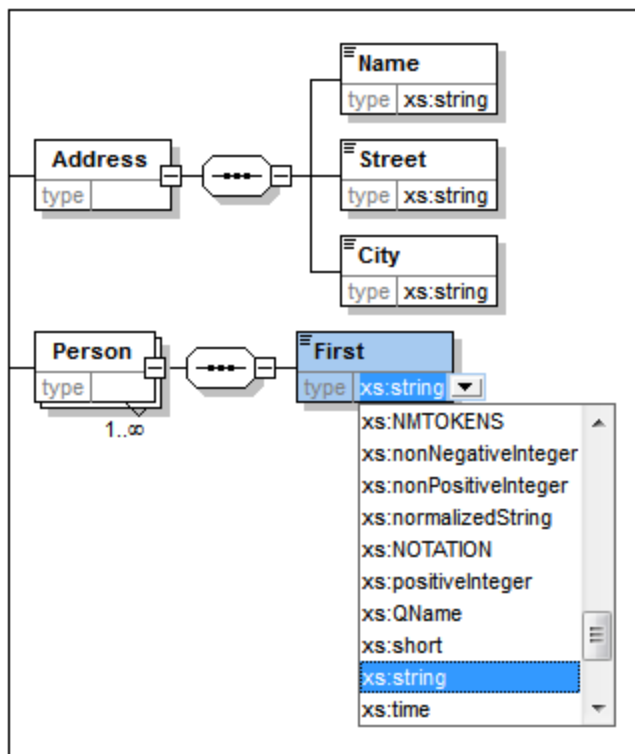

4. Select the `xs:string` entry from the dropdown list or enter it into the *Type* field.
5. Use the drag-and-drop method to create four more elements. Name them `Last`, `Title`, `PhoneExt`, and `Email`, respectively.
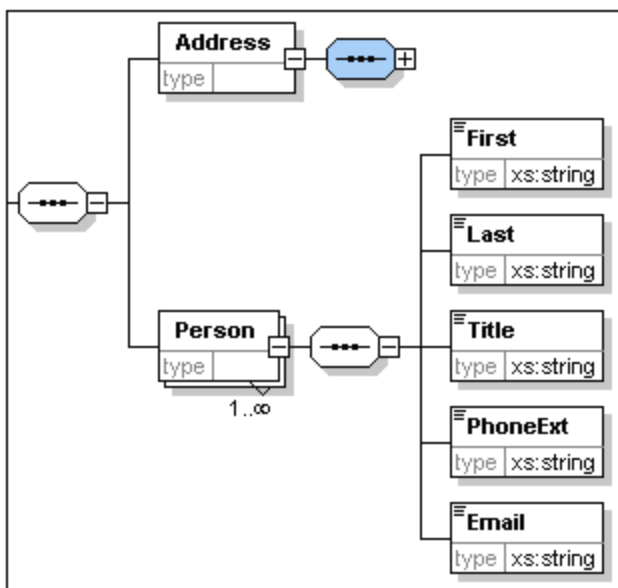
Note: You can select multiple elements by holding down the **Ctrl** key and clicking each of the required elements. This makes it possible to, e.g., copy several elements at once.

Making an element optional

Right-click the `Title` element and select **Optional** from the context menu. The frame of the element box changes from solid to dashed; this is a visual indication that an element is optional.
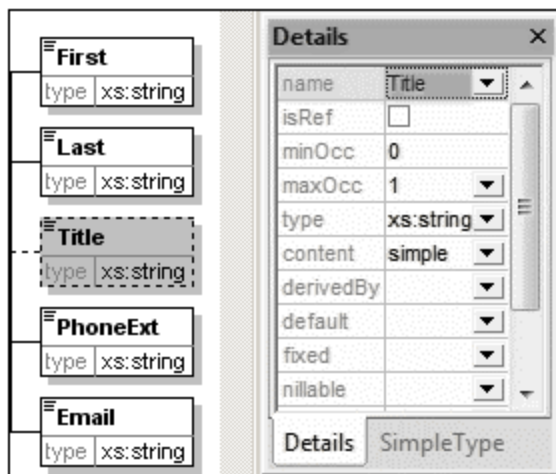

In the Details Entry Helper, you will see that `minOcc=0` and `maxOcc=1`, indicating that the element is optional. Alternatively to using the context menu to make an element optional, you can set `minOcc=0` in order to make the element optional.

Limiting the content of an element

To define the `PhoneExt` element to be of type `xs:integer` and have a maximum of two digits, do the following:

1. Double-click in the type field of the `PhoneExt` element, and select (or enter) the `xs:integer` entry from the dropdown list (*see screenshot below*).
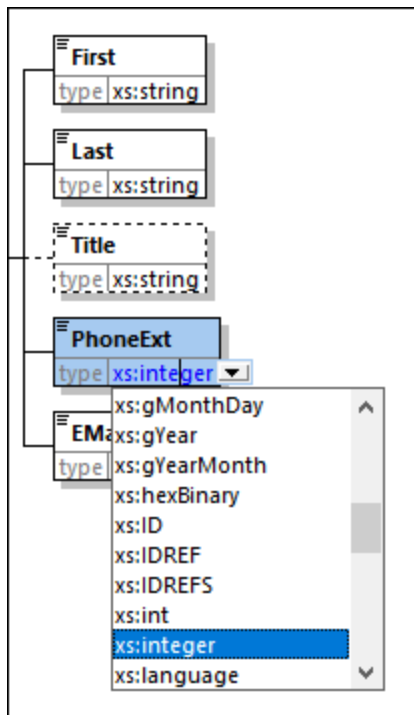
2. The items in the Facets entry helper change at this point. In the Facets entry helper, double-click in the `maxIncl` field and enter `99`. Confirm with **Enter**. This specifies that phone extensions can have values from `0` to `99`.
3. Select **File | Save** to save the schema changes.

Note the following

- Selecting an XML Schema datatype that is a simple type (for example, `xs:string` or `xs:date`) automatically changes the content model to simple in the Details entry helper (`content = simple`).
- Adding a compositor to an element (sequence, choice, or all), automatically changes the content model to complex in the Details entry helper (`content = complex`).
- The schema described above is available as `AddressFirst.xsd` in the Tutorial folder of your XMLSpy application folder.

1.3　XML Schemas: Advanced

Now that you have created a basic schema, we can move forward to a few advanced aspects of schema development.

Objective

In this section, you will learn how to:

- Work with complex types and simple types [31], which can then be used as the types of schema elements.
- Create global elements [39] and reference them from other elements.
- Create attributes [41] and their properties, including enumerated values.

You will start this section with the basic `AddressFirst.xsd` schema you created in the first part of this tutorial.

Commands used in this section

In this section of the tutorial, you will use Schema View exclusively. The following commands are used:

▪	Display Diagram (or Display Content Model View). This icon is located to the left of all global components in Schema Overview. Clicking the icon causes the content model of the associated global component to be displayed.
▣	Display All Globals. This icon is located at the top left-hand corner of the Content Model View. Clicking the icon switches the view to Schema Overview, which displays all global components.
▤	Append. The Append icon is located at the top left-hand corner of the Schema Overview. Clicking the icon enables you to add a global component.

1.3.1　Working with Complex Types and Simple Types

Having defined the content model of an element, you may decide you want to reuse it elsewhere in your schema. This might happen, for example, if you want to define a content model for addresses in the US and the UK. Some components of the two address formats are common, for example, the street and city components. Other components, however, are different. A sensible strategy, therefore, would be to reuse, in each address content model (US and UK), the common components, and complete each content model with the components that are specific to it (such as ZIP in the US and postal code in the UK). In order to do this, we can create the common components as a global complex type (or, alternatively, each common component as a global element), and reuse the global complex type (or the global elements) in the content model of each address type.

In this section, you will work with global complex types. A complex type is a type definition for elements that contain other elements and/or attributes. You will first create a complex type at the global level and then import it into a content model and extend it. You will learn about global elements later in this tutorial.

Creating a global complex type

The basic `Address` element that we defined (containing `Name`, `Street`, and `City` elements) can be reused in various address formats. So let us create this element definition as a complex type, which can be reused.
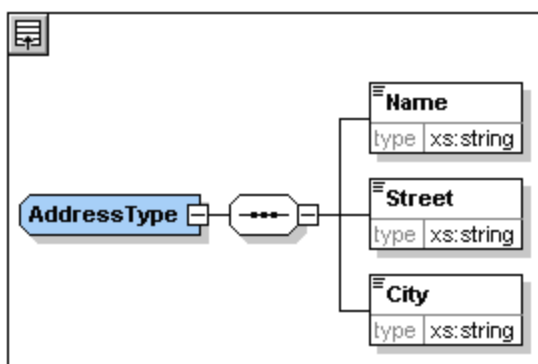
To create a global complex type:

1. In the Content Model View, right-click the `Address` element.
2. In the context menu that now appears, select **Make Global | Complex type**. A global complex type called `AddressType` is created, and the `Address` element in the `Company` content model is assigned this type. The content of the `Address` element is the content model of `AddressType`, which is displayed in a yellow box. Notice that the datatype of the `Address` element is now `AddressType`.
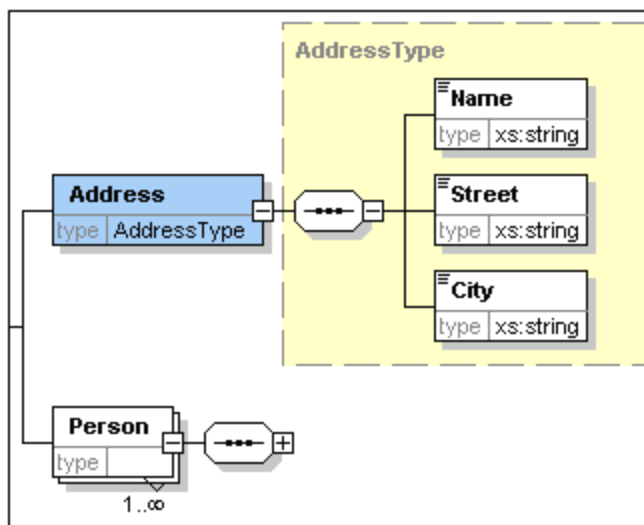
3. Click the Display All Globals [icon] icon. This takes you to the Schema Overview, in which you can view all the global components of the schema.
4. Click the *Expand* icons for the `element` and `complexType` entries in the Components entry helper so that their respective schema constructs are displayed. The Schema Overview now displays two global components: the `Company` element and the complex type `AddressType`. The Components Entry Helper also displays the `AddressType` complex type.
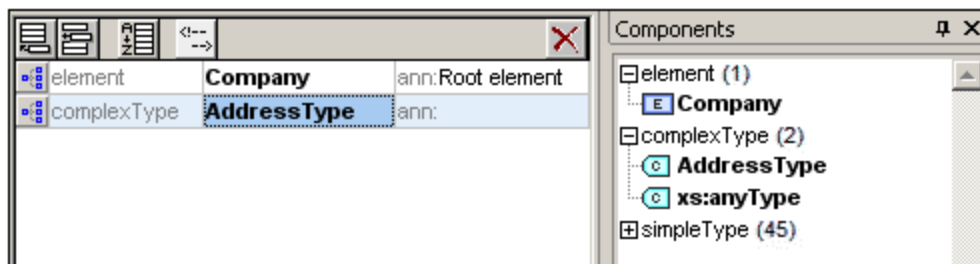
5. Click on the Content Model View icon [icon] of `AddressType` to see its content model (*screenshot below*). Notice the shape of the complex type container.

6. 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 basic **AddressType** component, and then extend that definition.

Do this as follows:

1. Switch to Schema Overview. (If you are in Content Model View, click the Display All Globals icon .)
2. Click the **Append** icon at the top left of the component window. The following menu opens:
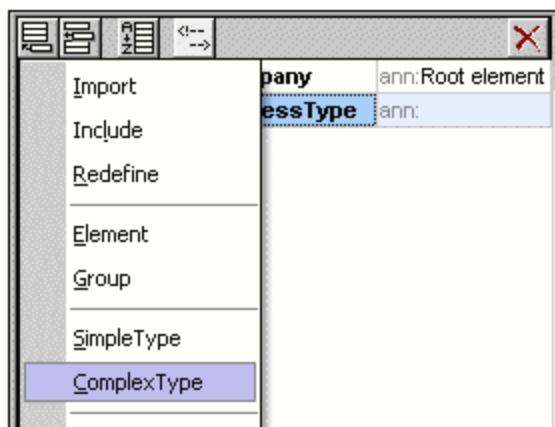
3. Select **ComplexType** from the menu. A new line appears in the component list, and the cursor is set for you to enter the component name.
4. Enter **US-Address** and confirm with **Enter**. (If you forget to enter the hyphen character **"-"** and enter a space, the element name will appear in red, signalling an invalid character.)
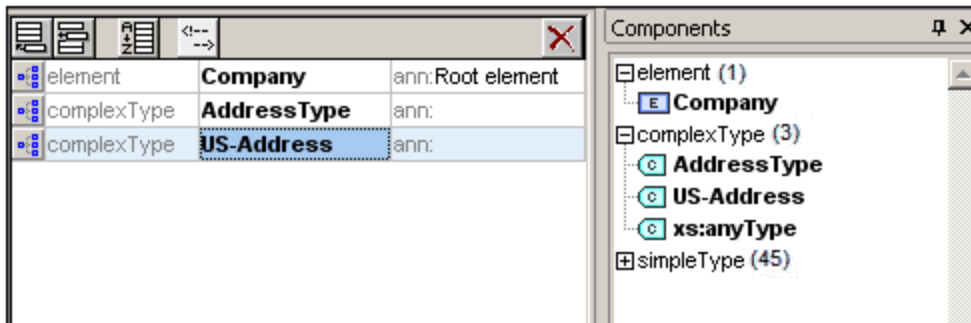
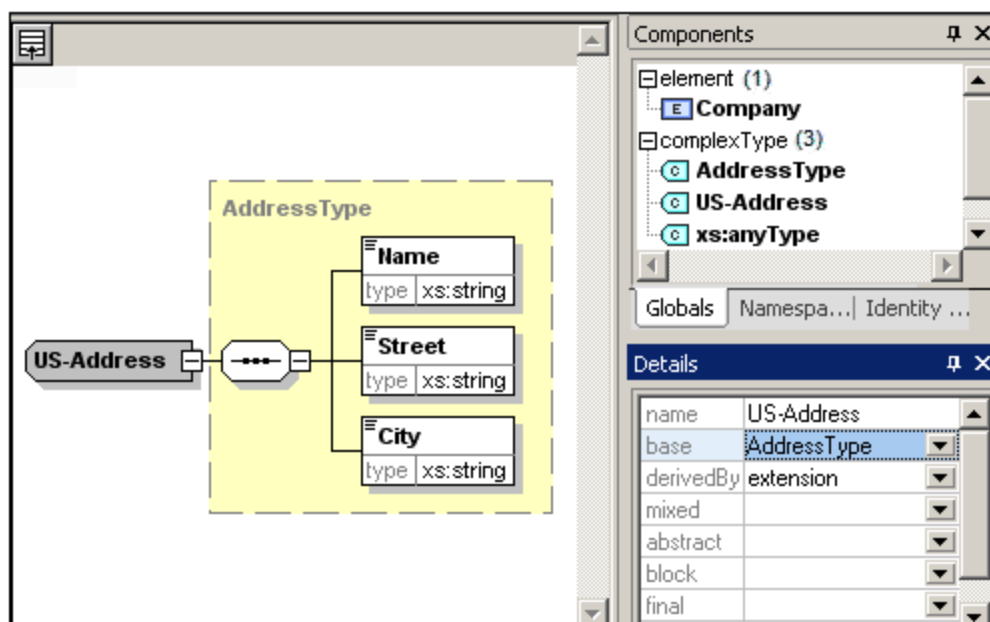5. Click the Content Model View icon of `US-Address` to see the content model of the new complex type. The content model is empty (*see screenshot below*).
6. In the Details entry helper, click the `base` combo box and select the `AddressType` entry.
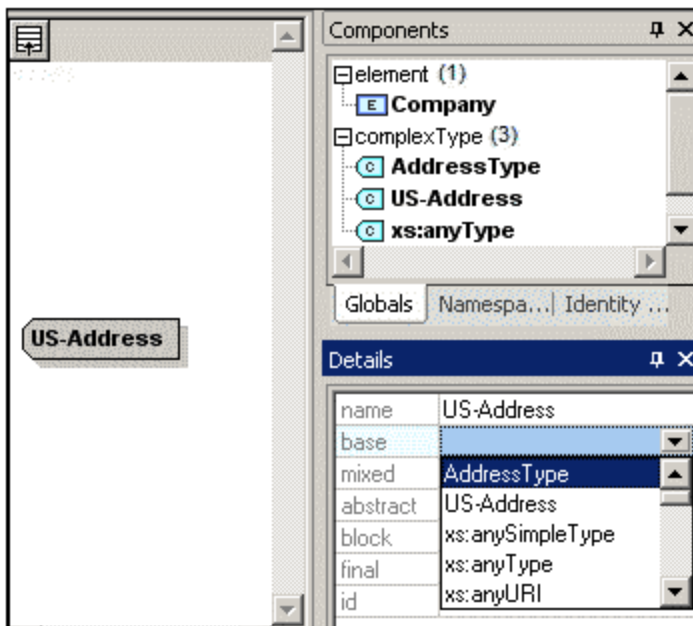
The Content Model View now displays the `AddressType` content model as the content model of US-Address (*screenshot below*).

7. Now we can extend the content model of the `US-Address` complex type to take a ZIP Code element.
    To do this, right-click the `US-Address` component, and, from the context menu that appears, select
    Add Child | Sequence. A new sequence compositor is displayed outside the `AddressType` box
    (*screenshot below*). This is a visual indication that this is an extension to the element.
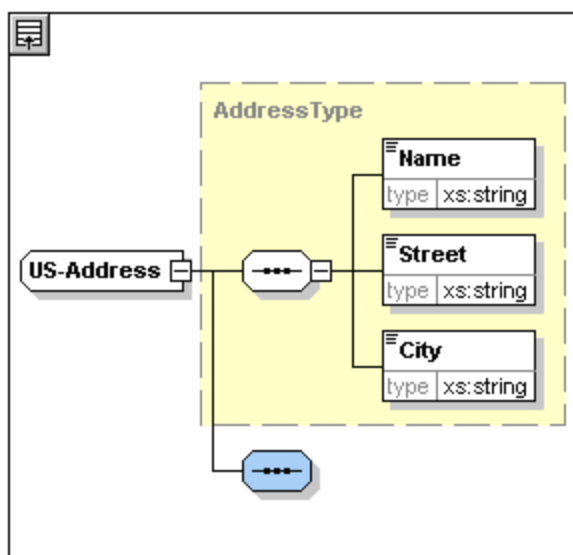
8. Right-click the new sequence compositor and select **Add Child | Element**.
9. Name the newly created element `zip`, and then press the **Tab** key. This places the cursor in the value
    field of the type descriptor line.
10. Select `xs:positiveInteger` from the dropdown menu that appears, and confirm with **Enter**.
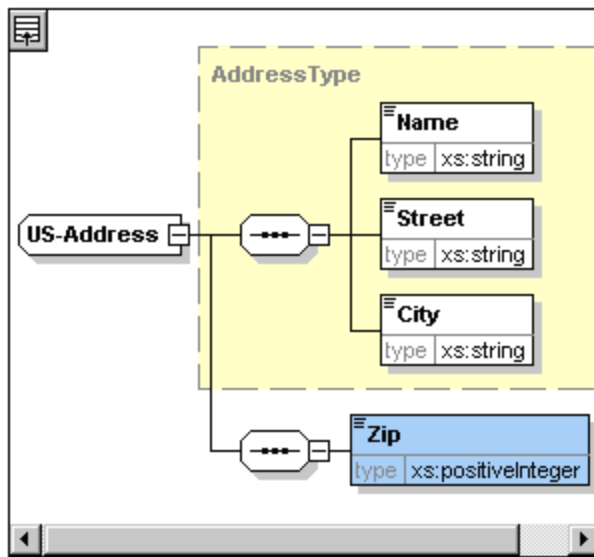
You now have a complex type called `US-Address`, which is based on the complex type `AddressType` and extends it to contain a ZIP code.

Global simple types

Just as the complex type `US-Address` is based on the complex type `AddressType`, an element can also be based on a simple type. The advantage is the same as for global complex types: the simple type can be reused. In order to reuse a simple type, the simple type must be defined globally. In this tutorial, you will define a content model for US states as a simple type. This simple type will be used as the basis for another element.

Creating a global simple type

Creating a global simple type consists of appending a new simple type to the list of global components, naming it, and defining its datatype.

To create a global simple type:

1. Switch to Schema Overview. (If you are in Content Model View, click the Display All Globals icon .)
2. Click the **Append** icon, and in the context menu that appears, select **SimpleType**.
3. Enter `US-state` as the name of the newly created simpleType.
4. Press **Enter** to confirm. The simple type `US-state` is created and appears in the list of simple types in the Components Entry Helper (Click the expand icon of the simpleType entry to see it).
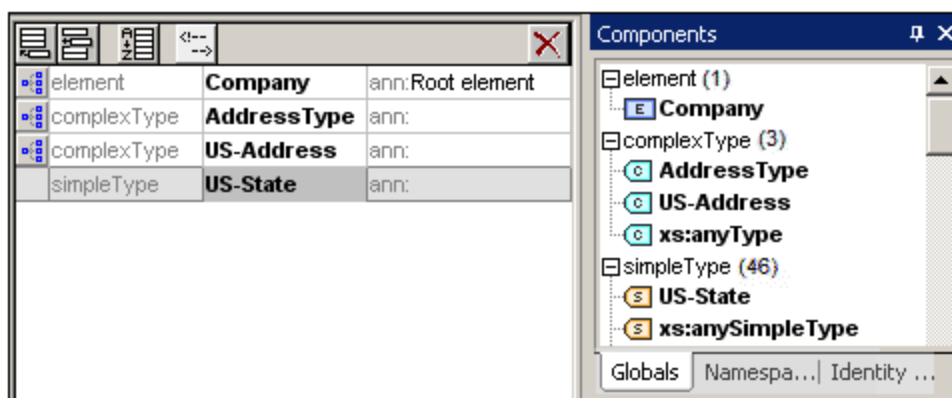
5. In the Details Entry Helper (*screenshot below*), place the cursor in the value field of **restr** and enter **xs:string**, or select **xs:string** from the dropdown menu in the **restr** value field.
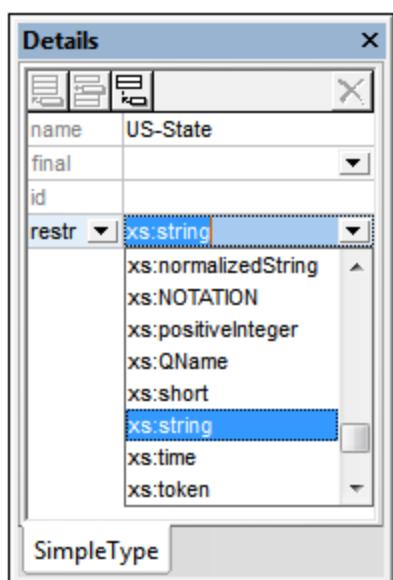
This creates a simple type called **US-State**, which is of datatype **xs:string**. This global component can now be used in the content model of **US-Address**.

Using a global simple type in a content model

A global simple type can be used in a content model to define the type of a component. We will use **US-State** to define an element called **state** in the content model of **US-Address**.

Do the following:

1. In Schema Overview, click the Component Model View icon of **US-Address**.
2. Right-click the lower sequence compositor and select **Add Child | Element**.
3. Enter **state** for the element name.
4. Press the **Tab** key to place the cursor in the value field of the type descriptor line.
5. From the drop-down menu of this combo box, select **US-State**.
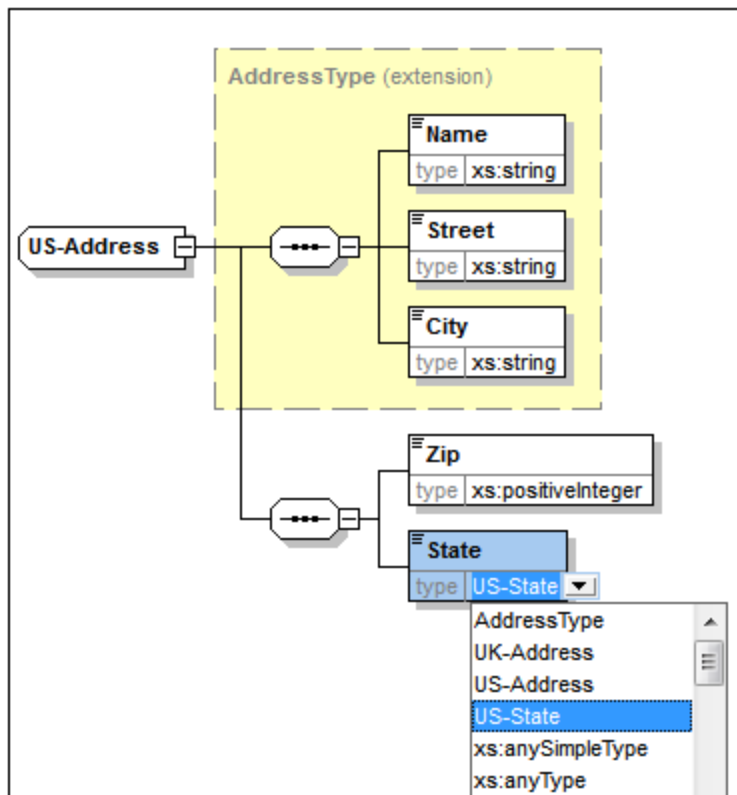
The State element is now based on the **US-State** simple type.

Creating a second complex type based on AddressType

We will now create a global complex type to hold UK addresses. The complex type is based on **AddressType**, and is extended to match the UK address format.

Do the following:

1. In Schema Overview, create a global complex type called **UK-Address**, and base it on **AddressType** (**base=AddressType**).
2. In the Content Model View of **UK-Address**, add a **Postcode** element and give it a type of **xs:string**. Your **UK-Address** content model should look like this:
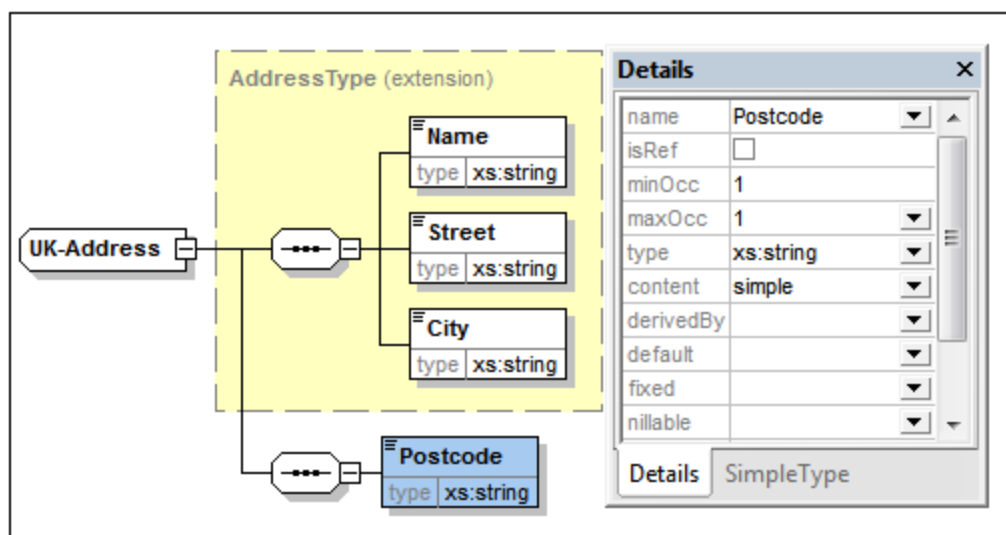
Note: In this section you created global simple and complex types, which you then used in content model definitions. The advantage of global types is that they can be reused in multiple definitions.

1.3.2   Referencing Global Elements

In this section, we will convert the locally defined `Person` element to a global element and reference that global element from within the `Company` element.

1. Click (Display All Globals) to switch to Schema Overview.
2. Click the Display Diagram icon of the `Company` element.
3. Right-click the **Person** element, and select **Make Global | Element**. A small link arrow icon appears in the **Person** element, showing that this element now references the globally declared **Person** element. In the Details Entry Helper, the **isRef** check box is now activated.
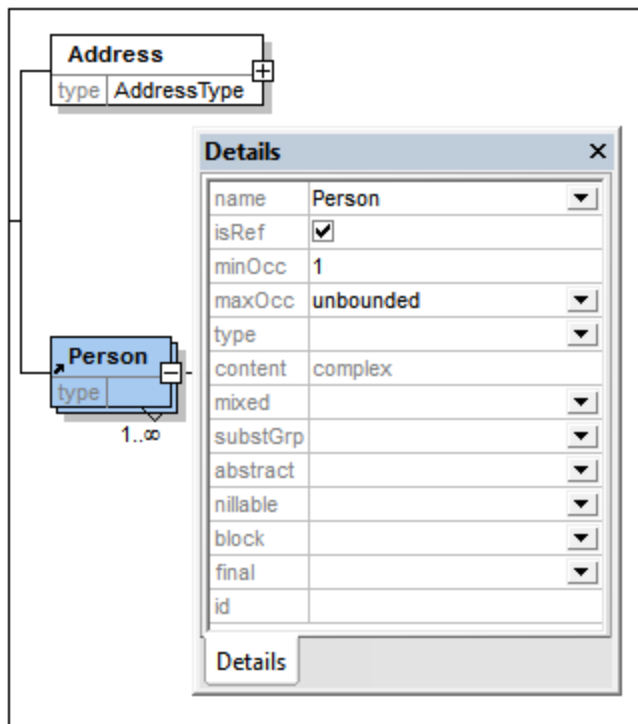
4. Click the Display All Globals icon to return to Schema Overview. The `Person` element is now listed as a global element. It is also listed in the Components Entry Helper.
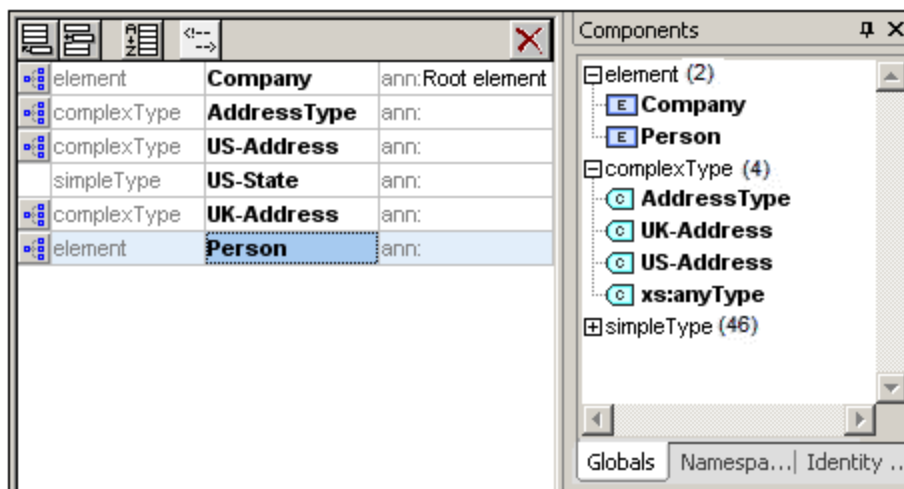
5. In the Components Entry Helper, double-click the `Person` element to see the content model of the global `Person` element.
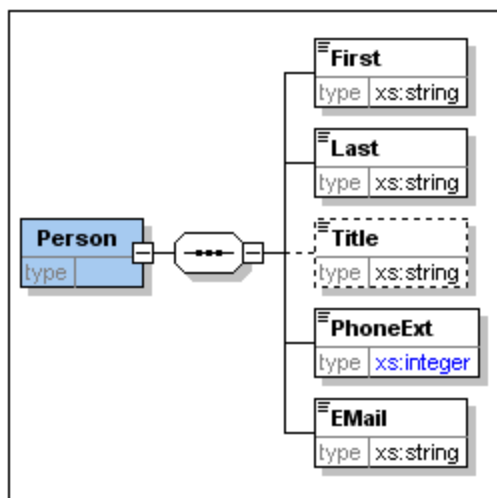
Notice that the global element box does **not** have a link arrow icon. This is because it is the referenced element, not the referencing element. It is the referencing element that has the link arrow icon.

Note the following:

- An element that references a global element must have the same name as the global element it references.
- A global declaration does not describe where a component is to be used in an XML document. It only describes a content model. It is only when a global declaration is referenced from within another component that its location in the XML document is specified.
- A globally declared element can be reused at multiple locations. It differs from a globally declared complex type in that its content model cannot be modified without also modifying the global element itself. If you change the content model of an element that references a global element, then the content model of the global element will also be changed, and, with it, the content model of all other elements that reference that global element.

1.3.3   Attributes and Attribute Enumerations

In this section, you will learn how to create attributes and enumerations for attributes.

Defining element attributes

1. In the Schema Overview, click the `Person` element to make it active.
2. Click the **Append** icon ⊞, in the top left of the Attributes/Identity Constraints tab group (in the lower part of the Schema Overview window), and select the Attribute entry.
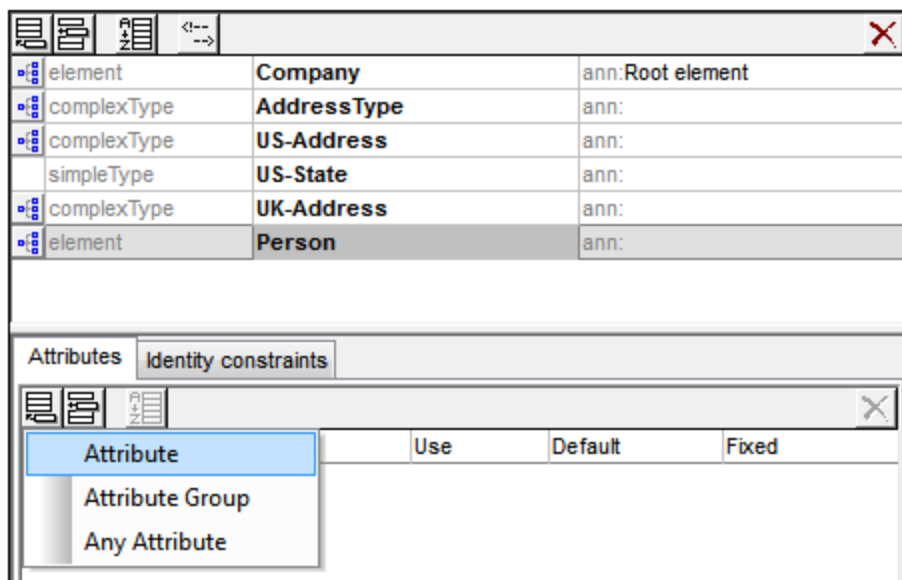
3. Enter `Manager` as the attribute name in the Name field.
4. Use the *Type* combo box to select `xs:boolean`.
5. Use the *Use* combo box to select required.
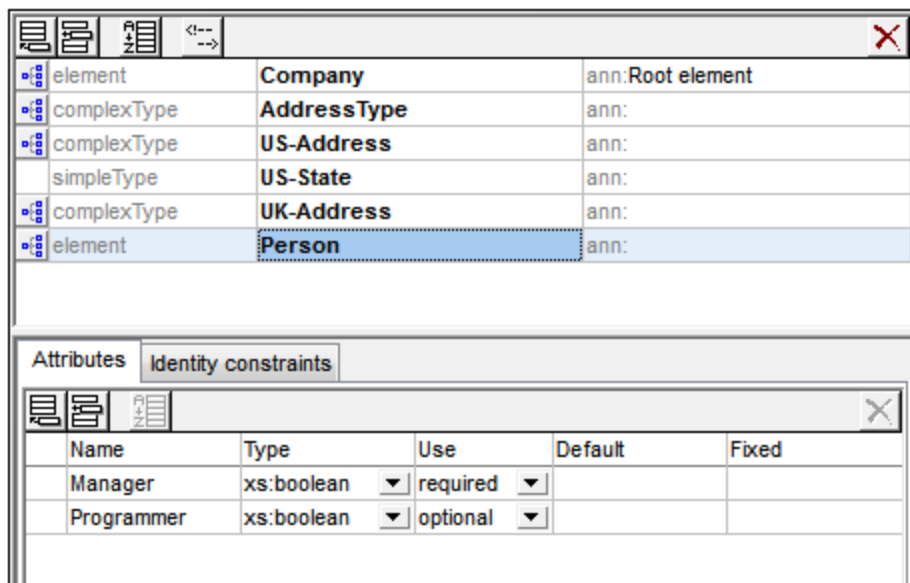
6. Use the same procedure to create a `Programmer` attribute with `Type=xs:boolean` and `Use=optional`.

Defining enumerations for attributes

Enumerations are values allowed for a given attribute. If the value of the attribute in the XML instance document is not one of the enumerations specified in the XML Schema, then the document is invalid. We will create enumerations for the `Degree` attribute of the `Person` element.

Do the following:

1. In the Schema Overview, click the **Person** element to make it active.
2. Click the **Append** icon ⊞ in the top left of the Attributes window, and select the Attribute entry.
3. Enter **Degree** as the attribute name, and select **xs:string** as its type.
4. With the **Degree** attribute selected, in the Facets Entry Helper, click the **Enumerations** tab (*see screenshot*).
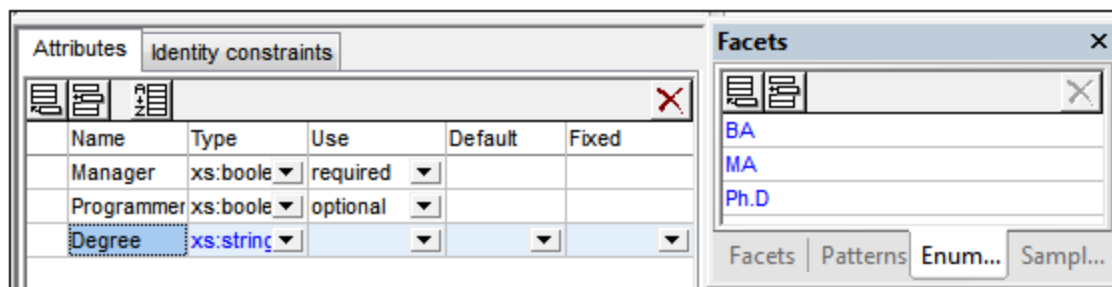
5. In the **Enumerations** tab, click the Append icon ⊟.
6. Enter **BA**, and confirm with **Enter**.
7. Use the same procedure to add two more enumerations: **MA** and **PhD**.
8. Click on the Content Model View icon of **Person**.
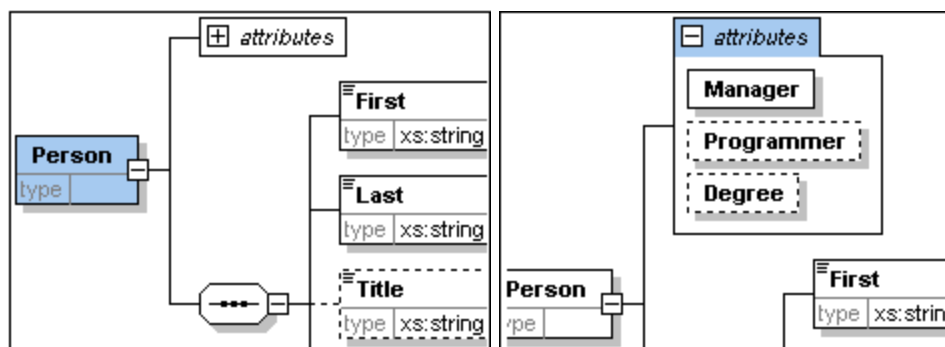
The previously defined attributes are visible in the Content Model View. Clicking the expand icon displays all the attributes defined for that element. This display mode and the Attributes tab can be toggled by selecting the menu option **Schema Design | Configure view**, and checking and unchecking the **Attributes** check box in the **Show in diagram** pane.

9. Click the Display all Globals icon to return to the Schema Overview.

Saving the completed XML Schema

Before saving your schema file, rename the **AddressLast.xsd** file that is delivered with XMLSpy to something else (such as **AddressLast_original.xsd**), so as not to overwrite it. Save the completed schema with any name you like (**File | Save as**). We recommend that you save it with the name **AddressLast.xsd**. This is because the the XML file you will create in the next part of the tutorial will be based on the `AddressLast.xsd` schema.

1.4   XML Schemas: XMLSpy Features

After having completed the XML Schema, we suggest you become familiar with a few [navigation shortcuts](#)[44] and learn about the [schema documentation](#)[46] that you can generate from within XMLSpy. These are described in the subsections of this section.

Commands used in this section

In this section of the tutorial, you will use Schema View exclusively. The following commands are used:

	Display Diagram (or Display Content Model View). This icon is located to the left of all global components in Schema Overview. Clicking the icon causes the content model of the associated global component to be displayed.

1.4.1   Schema Navigation

This section shows you how to navigate Schema View efficiently. We suggest that you try out these navigation mechanisms to become familiar with them.

Displaying the content model of a global component

Global components that can have content models are complex types, elements, and element groups. The Content Model View of these components can be opened in the following ways:

- In Schema Overview, click the **Display Diagram** icon to the left of the component name.
- In either Schema Overview or Content Model View, double-click the element, complex type, or element group in the Components Entry Helper (*screenshot below*). This displays the content model of that component.
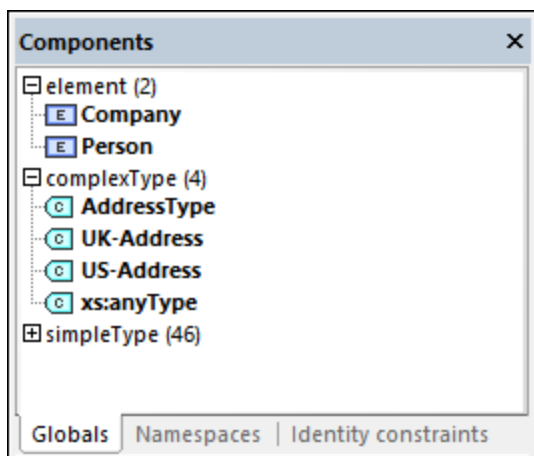
If you double-click any of the other global components (simple type, attribute, attribute group) in the Components Entry Helper, that component will be highlighted in Schema Overview (since such a component would not have a content model).

In the Components Entry Helper, the double-clicking mechanism works in both the Globals and Namespaces tabs.

Going to the definition of a global element from a referencing element

If a content model contains an element that references a global element, you can go directly to the content model of that global element or to any of its contained components by holding down **Ctrl** and double-clicking the required element.
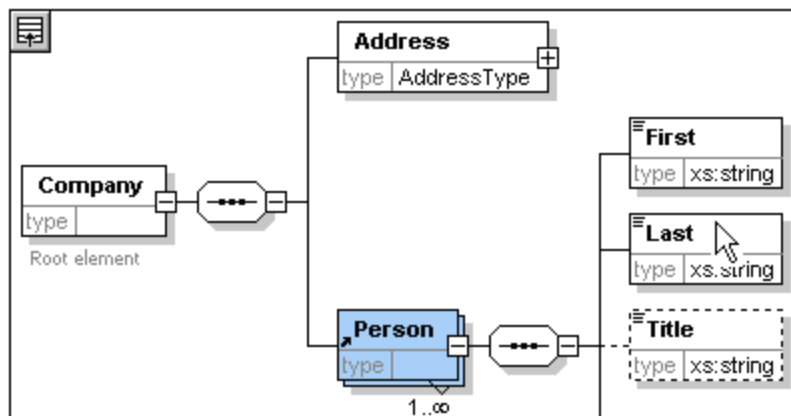
When the `Last` element is highlighted, all its properties are immediately displayed in the relevant entry helpers and information window.

Going to the definition of a complex type

Complex types are often used as the type of some element within a content model. To go directly to the definition of a complex type from within a content model, double-click the **name** of the complex type in the yellow box (*see mouse pointer in screenshot below*).
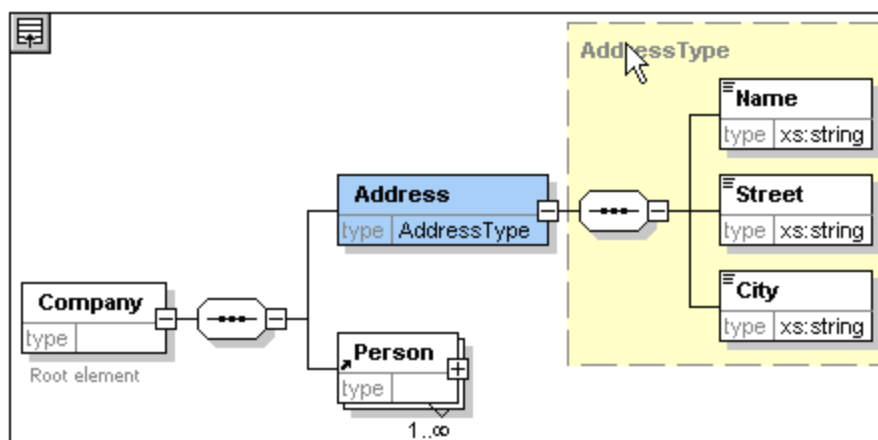
This takes you to the Content Model View of the complex type.
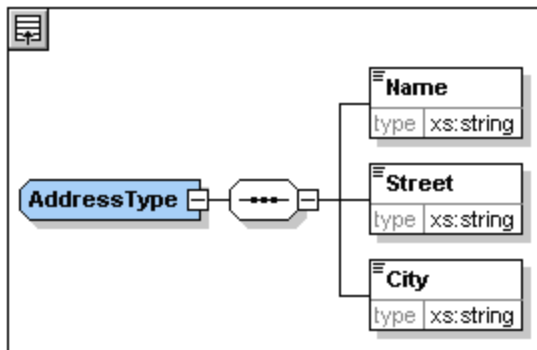
Note: Just as with referenced global elements, you can go directly to an element within the complex type definition by holding down **Ctrl** and double-clicking the required element in the content model that contains the complex type.

1.4.2 Schema Documentation

XMLSpy provides detailed documentation of XML Schemas in HTML and Microsoft Word (MS Word) formats. You can select the components and the level of detail you want documented. Related components are hyperlinked in both HTML and MS Word documents. In order to generate MS Word documentation, you must have MS Word installed on your computer (or network).

In this section, we will generate documentation for the `AddressLast.xsd` XML Schema.

Do the following:

1. Select the menu option **Schema design | Generate documentation**. This opens the Schema Documentation dialog.
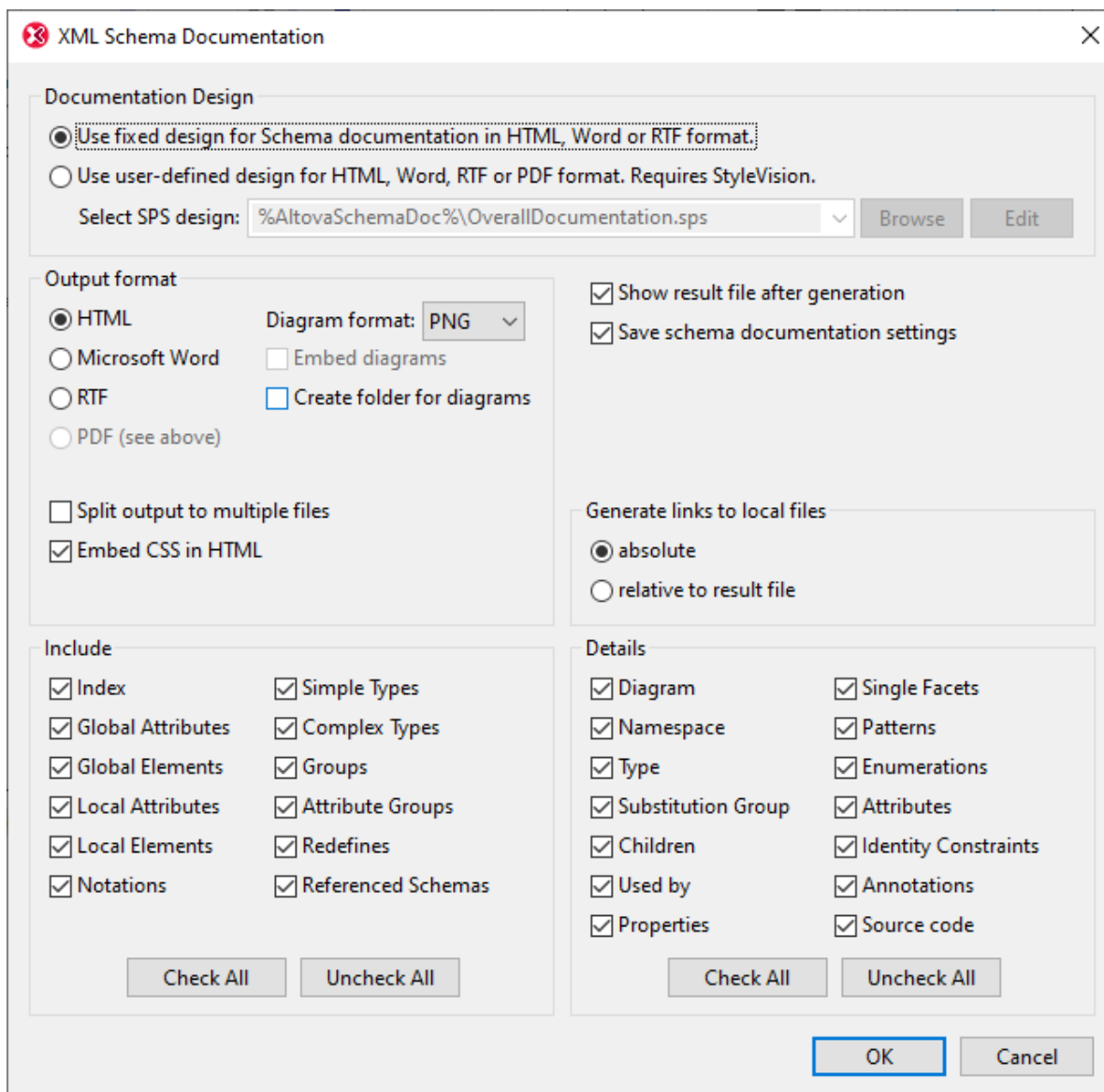
2. For the Output Format option, select HTML, and click **OK**.
3. In the Save As dialog, select the location where the file is to be saved and give the file a suitable name (say `AddressLast.html`). Then click the **Save** button.

The HTML document appears in the Browser View of XMLSpy. Click on a link to go to the corresponding linked component.

Schema **AddressLast.xsd**

schema location: C:\Users\alU\Documents\Altova\XMLSpy2013
                    \Examples\Tutorial\AddressLast.xsd
attributeFormDefault: **unqualified**
elementFormDefault: **qualified**
targetNamespace: **http://my-company.com/namespace**

Elements Complex types Simple types
Company AddressType   US-State
Person   UK-Address
          US-Address

element **Company**

| diagram	
	(diagram: Company → sequence → Address [type AddressType], Person [type], 1..∞. Company labelled "Root element")
namespace	**http://my-company.com/namespace**
properties	content **complex**
children	**Address Person**
annotation	documentation **Root element**

source	```xml
	<xs:element name="Company">
	  <xs:annotation>
	    <xs:documentation>Root element</xs:documentation>
	  </xs:annotation>
	  <xs:complexType>
	    <xs:sequence>
	      <xs:element name="Address" type="AddressType"/>
	      <xs:element ref="Person" maxOccurs="unbounded"/>
	    </xs:sequence>
	  </xs:complexType>
	</xs:element>
	``

The diagram above shows the **first page** of the schema documentation in HTML form. If components from other schemas have been included, then those schemas are also documented.

complexType **US-Address**	
diagram	
namespace	http://my-company.com/namespace
type	extension of **AddressType**
properties	name US-Address base AddressType
children	**Name** **Street** **City** **Zip** **State**

source	`<xs:complexType name="US-Address">`
	` <xs:complexContent>`
	`  <xs:extension base="AddressType">`
	`   <xs:sequence>`
	`    <xs:element name="Zip" type="xs:positiveInteger"/>`
	`    <xs:element name="State" type="US-State"/>`
	`   </xs:sequence>`
	`  </xs:extension>`
	` </xs:complexContent>`
	`</xs:complexType>`
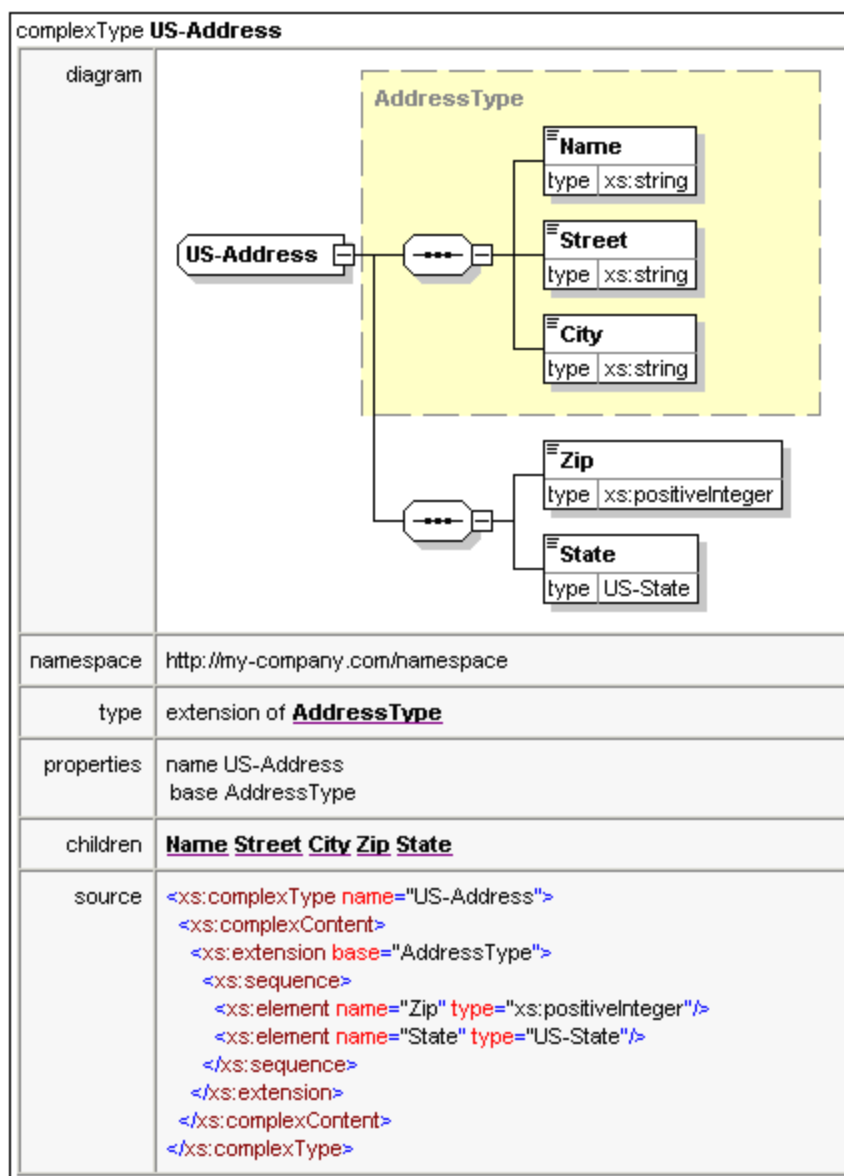
The diagram above shows how complex types are documented.

element **US-Address/Zip**	
diagram	**Zip** type | xs:positiveInteger
namespace	http://my-company.com/namespace
type	**xs:positiveInteger**
properties	name Zip isRef 0 content simple
source	`<xs:element name="Zip" type="xs:positiveInteger"/>`

element **US-Address/State**	
diagram	**State** type | US-State
namespace	http://my-company.com/namespace
type	**US-State**
properties	name State isRef 0 content simple
source	`<xs:element name="State" type="US-State"/>`

simpleType **US-State**	
namespace	http://my-company.com/namespace
type	**xs:string**
properties	name US-State
used by	element **US-Address/State**

source	`<xs:simpleType name="US-State">`
	  `<xs:restriction base="xs:string"/>`
	`</xs:simpleType>`

The diagram above shows how elements and simple types are documented.

You can now try out the MS Word output option. The Word document will open in MS Word. To use hyperlinks in the MS Word document, hold down **Ctrl** while clicking the link.

1.5   XML Documents

In this section you will learn how to create and work with XML documents in XMLSpy. You will also learn how to use the various intelligent editing features of XMLSpy.

Objective

The objectives of this section are to learn how to do the following:

- Create a new XML document based on the `AddressLast.xsd` schema.
- Specify the type of an element so as to make an extended content model for that element available to the element during validation.
- Insert elements and attributes and enter content for them in Grid View and Text View using intelligent entry helpers.
- Copy XML data from XMLSpy to Microsoft Excel; add new data in MS Excel; and copy the modified data from MS Excel back to XMLSpy. This functionality is available in the Table Display of Grid View.
- Sort XML elements using the sort functionality of Table Display.
- Validate the XML document.
- Modify the schema to allow for three-digit phone extensions.

Commands used in this section

In this section of the tutorial, you will mostly use the Grid View and Text View, and in one section Schema View. The following commands are used:

🗋	**File	New**. Creates a new type of XML file.
📄	**View	Text View**. Switches to Text View.
🗐	**View	Grid View**. Switches to Enhanced Grid View.
🎞	**XML	Display as Table**. Displays multiple occurrences of a single element type at a single hierarchic level as a table. This view of the element is called its Table Display. The icon is used to switch between the Table Display and regular Grid View.
🗹	**F7**. Checks for well-formedness.
🗹	**F8**. Validates the XML document against the associated DTD or Schema.
🔧	Opens the associated DTD or XML Schema file.

1.5.1   Creating a New XML File

When you create a new XML file in XMLSpy, you are given the option of basing it on a schema (DTD or XML Schema) or not. In this section you will create a new file that is based on the **AddressLast.xsd** schema you created earlier in the tutorial.

To create the new XML file:

1. Select the menu option **File | New**. The *Create new document* dialog opens.
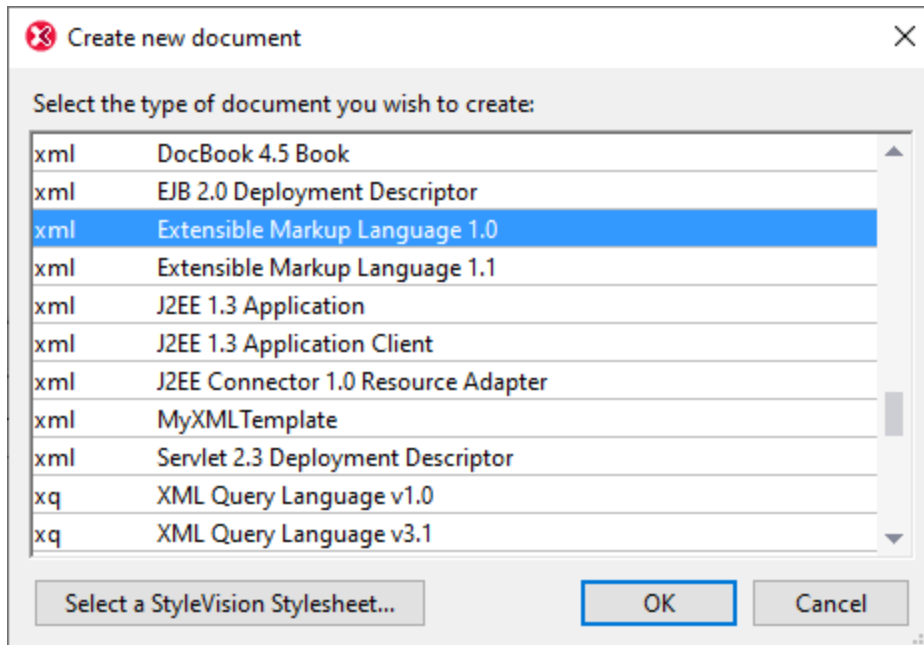
2. Select *Extensible Markup Language 1.0* and confirm with **OK**. The Choose Schema or DTD dialog appears.
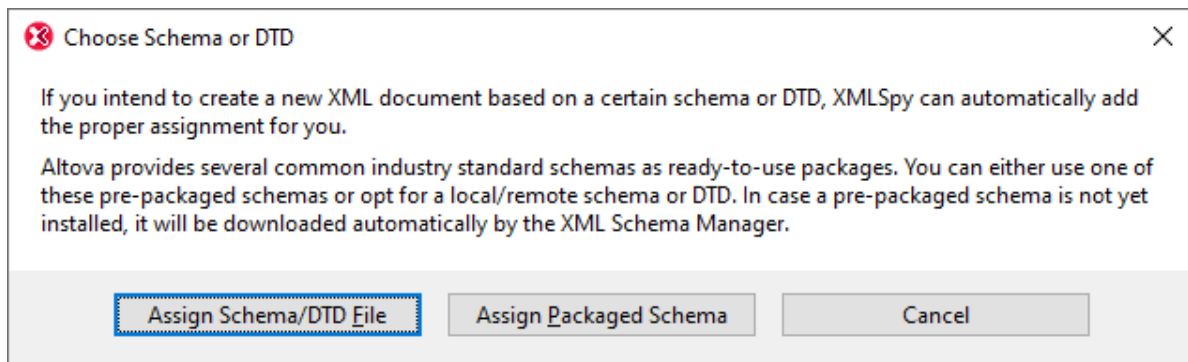
3. Click **Assign Schema/DTD File**.
4. In the dialog that appears, use either the **Browse** button or **Window** button to find the schema file. (The **Window** button lists all files currently open in XMLSpy.) Select `AddressLast.xsd` (see Tutorial introduction [5] for location), and confirm with **OK**. An XML document containing the main elements defined by the schema opens in the main window.
5. Click the Grid tab to select Grid View.
6. In Grid View, notice the structure of the document. Click on any element to reduce selection to that element. Your document should look something like this:

7. Click on the ▼ icon next to **Address**, to view the child elements of **Address**. Your document should look like this:

8. Select the menu option **File | Save** and save it in the **Tutorial** folder. Give your XML document a suitable name (for example **CompanyFirst.xml**). Note that the finished tutorial file **CompanyFirst.xml** is in the **Tutorial** folder, so you may need to rename it before you give that name to the file you have created.

1.5.2   Specifying the Type of an Element

The child elements of **Address** are those defined for the global complex type **AddressType** (the content model of which is defined in the XML Schema **AddressLast.xsd** shown in the Schema View screenshot below).
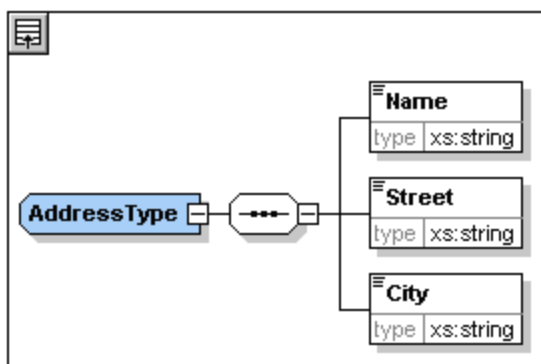
We would, however, like to use a specific US or UK address type rather than the generic address type. You will recall that, in the `AddressLast.xsd` schema, we created global complex types for `US-Address` and `UK-Address` by extending the `AddressType` complex type. The content model of `US-Address` is shown below.
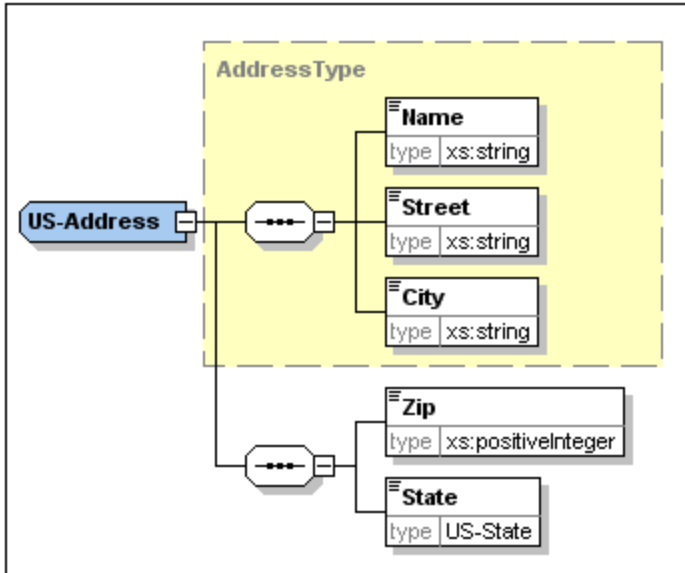
In the XML document, in order to specify that the `Address` element must conform to one of the extended `Address` types (`US-Address` or `UK-Address`) rather than the generic `AddressType`, we must specify the required extended complex type as an attribute of the `Address` element.

We add this attribute of the `Address` element as follows:

1. In the XML document, right-click the `Name` element, and select **Insert Before** from the context menu (*see screenshot below*).
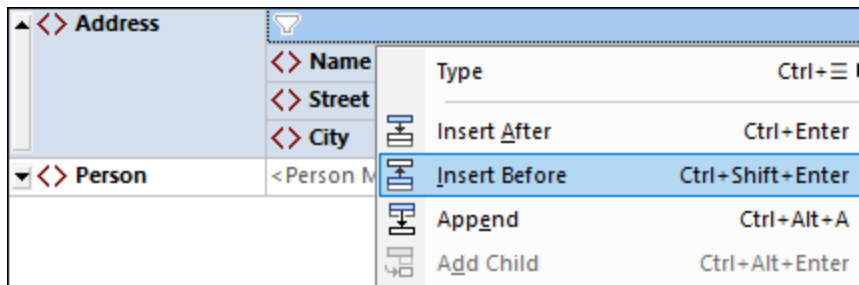
2. A new element node named `new_elem` is added above the `Name` element (*see screenshot below*). Click the element type to the left of the node's name and, in the menu that appears (*screenshot below*), select the *Attribute* node type. The node type will be changed to the *Attribute* node type; however, the name is still `new_elem`.
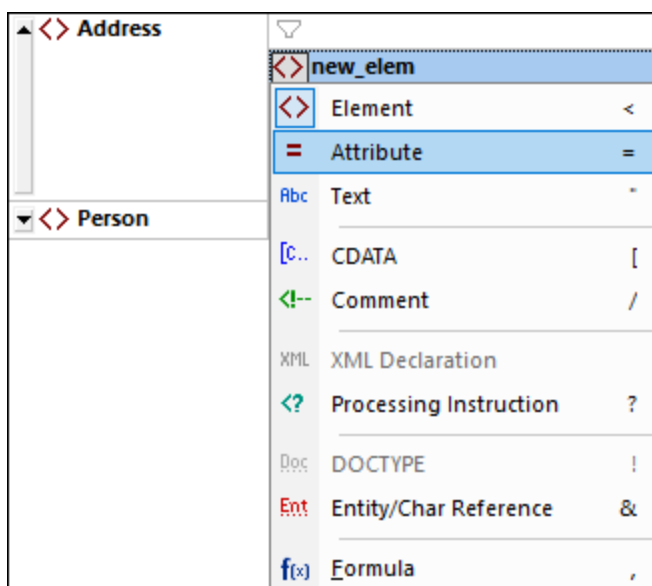
3. Double-click the node name and, in the entry helper popup that appears, select `xsi:type`.
4. Press the **Tab** key to move to the attribute's value field. A popup menus containing the available `xsi:type` values is displayed (*screenshot below*). These values are the complex types that have been defined for the `Address` element in the schema.
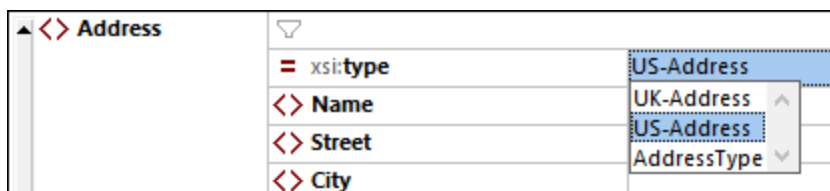
5. Select `US-Address` as the value of the `xsi:type` attribute.

Note: The `xsi:` prefix allows you to use special XML Schema related commands in your XML document instance. Notice that the the namespace for the `xsi:` prefix was automatically added to the document element when you assigned a schema to your XML file. In the above case, you have specified a type for the `Address` element. See the [XML Schema specification](#) for more information.

1.5.3   Entering Data in Grid View

You can now enter data into your XML document. Do the following:

1. Double-click in the `Name` value field (or use the arrow keys) and enter *US dependency*. Confirm with **Enter**.

2. Use the same method to enter a `street` and `city` name (for example, *Noble Ave* and *Dallas*).
3. Click the `Person` element and press **Delete** to delete the `Person` element. (We will add it back in the next section of the tutorial.) After you do this, the entire `Address` element is highlighted.
4. Click on any child element of the `Address` element to deselect all the child elements of `Address` except the selected element. Your XML document should look like this:

1.5.4   Entering Data in Text View

Text View presents the actual data and markup of XML files in an easy-to-follow structural layout, as well as schema-related intelligent editing features.

Document layout

The document layout of Text View is defined in two locations:

Pretty-printing options
These settings are in the Pretty-printing section of the Options dialog (*screenshot below*). When you set an option, its effect can be immediately seen in the preview pane at bottom. Set up the pretty-printing options as you like. While you are editing in Text View, you might find that the document's layout becomes unstructured, especially after you copy-paste blocks of text. Whenever you want to obtain a neat and hierarchical layout, simply click the **Edit | Pretty Print** command.

Pretty-printing

Pretty-print is used when the command is executed in Text View or when switching or saving a modified document from all other views.

☐ Automatically pretty-print a document when opened in Text View.

☑ Use indentation determined by the tab configuration of Text View =>   [Text View Settings...]

| XML | JSON | YAML |

Empty elements: ⦿ Self-closing   ○ Self-closing with space   ○ End tag

Inline attributes: ⦿ Always   ○ Up to [5] attributes   ○ Never

Attribute values: ☐ Spacing   ☐ Single-quotes preferred   ☑ Keep quotes, if possible

Significant whitespace: ○ Preserve   ⦿ Collapse

Allow use of xml:space: [Always ⌄]

Preserve whitespace: [pre, code, script   ] [...]

```
<root attribute="value">
——><empty/>
——><element a1="value" a2='value' a3='"string"' a4="'string'"/>
——><significantWhiteSpace/>
——><pre>
···<span>whitespace</span> <b>preserve</b>
</pre>
——><p xml:space="preserve">  <x/>
·</p>
</root>
```

Text View settings
The Text View Settings dialog (*screenshot below*) not only provides additional layout options but also switches on/off useful Text View features such as line numbering and folding margins. Access the Text View Settings dialog with the **View | Text View Settings** command.
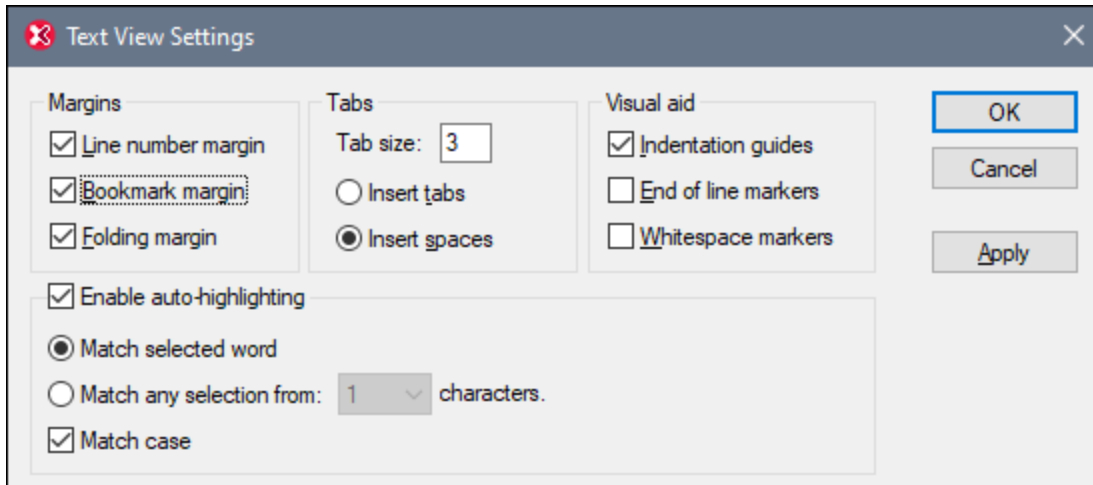
The screenshot below shows the current XML file in Text View with features switched on according to the settings in the dialog above.

On the left are the three margins: (i) the line number margin, (ii) the bookmark margin (containing two blue bookmarks), and (iii) the source folding margin (which allows you to expand and collapse the display of XML elements). Indentation guides are the light gray vertical lines that show the indentation of tags at the same hierarchical level. Additionally visual aids are end-of-line markers and whitespace markers, which can be switched on or off in the *Visual Aid* pane (*see screenshot above*).

Note: The Text View-related pretty-printing and bookmark features were covered in the earlier Text View Settings [12] section of this tutorial.

Editing in Text View

In this section, you will enter and edit data in Text View in order to become familiar with the features of Text View.

Note: Since the *Validate on Edit* feature is switched on by default, any validation error created during editing will be immediately flagged, with the error message/s being displayed in the Messages Window. Ignore these errors and messages for now. If you do not want background validation, you can switch off *Validate on Edit* in the Validation settings of the Options dialog. In the event you do this, note that you can always validate your document at any time (described in the next section [61] of this tutorial).

Do the following:

1. Select the menu item **View | Text View**, or click on the **Text** tab. You now see the XML document in its text form, with syntax coloring.
2. Place the text cursor after the end tag of the `Address` element, and press **Enter** to add a new line.
3. Enter the less-than angular bracket **<** at this position. A dropdown list of all elements allowed at that point (according to the schema) is displayed. Since only the `Person` element is allowed at this point, it will be the only element displayed in the list.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy 2021 -->
<Company xmlns="http://my-company.com/namespace" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://my-company.com/namespace AddressLast.xsd">
    <Address xsi:type="US-Address">
        <Name>US dependency</Name>
        <Street>Noble Ave.</Street>
        <City>Dallas</City>
    </Address>
    <
    Person

</Company>
```

4. Select the `Person` entry. The `Person` element, as well as its attribute `Manager`, are inserted, with the cursor inside the value-field of the `Manager` attribute.
5. From the dropdown list that pops up for the `Manager` attribute, select `true`.

```
        </Address>
        <Person Manager="">
</Company>
                         false
                         true
```
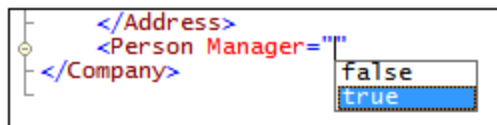
6. Move the cursor to the end of the line (using the **End** key if you like), and press the space bar. This opens a dropdown list containing a list of attributes allowed at that point. Also, in the Attributes Entry Helper, the available attributes are listed in red. The `Manager` attribute is grayed out because it has already been used.

```
        </Address>                             Attributes
        <Person Manager="true"
</Company>                          Degree        Degree
                                    Programmer    Programmer
                                    xsi:type      xsi:type
                                                  Manager
```
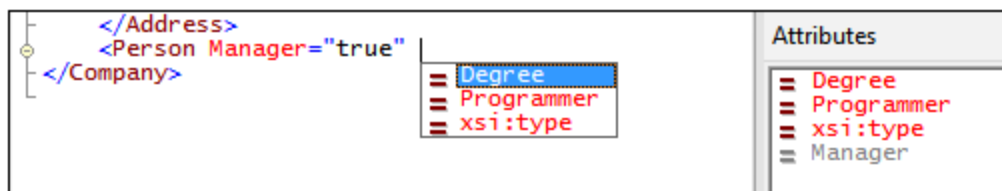
7. Select `Degree` with the Down arrow key, and press **Enter**. This opens another list box, from which you can select one of the predefined enumerations (`BA`, `MA`, or `PhD`). (Enumerations are values that are allowed by the XML Schema.)

```
        </Address>
        <Person Manager="true" Degree="">
</Company>
                                        BA
                                        MA
                                        Ph.D
```
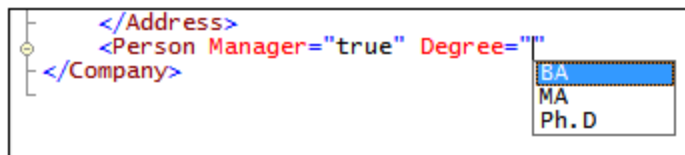
8.   Select **BA** with the Down arrow key and confirm with **Enter**. Then move the cursor to the end of the line (with the **End** key), and press the space bar. **Manager** and **Degree** are now grayed out in the Attributes Entry Helper.

```
        </Address>
        <Person Manager="true" Degree="BA"              Attributes
    </Company>                                          ≡ Programmer
                                              ≡ Programmer     ≡ xsi:type
                                              ≡ xsi:type       ≡ Degree
                                                               ≡ Manager
```
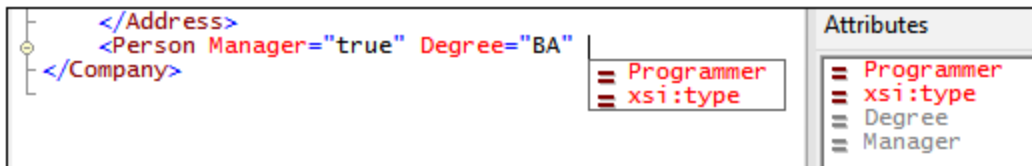
9.   Select **Programmer** with the Down arrow key and press **Enter**.

```
        </Address>
        <Person Manager="true" Degree="BA" Programmer="f|"
    </Company>                                          false
                                                        true
```
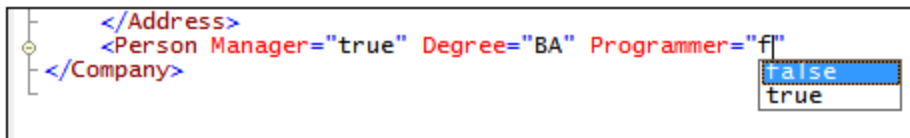
10. Enter the letter **"f"** and press **Enter**.
11. Move the cursor to the end of the line (with the **End** key), and enter the greater-than angular bracket **>**. XMLSpy automatically inserts all the required child elements of **Person**. (Note that the optional **Title** element is not inserted.) Each element has start and end tags but no content.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy 2021 -->
<Company xmlns="http://my-company.com/namespace" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://my-company.com/namespace AddressLast.xsd">
    <Address xsi:type="US-Address">
        <Name>US dependency</Name>
        <Street>Noble Ave.</Street>
        <City>Dallas</City>
    </Address>
    <Person Manager="true" Degree="BA" Programmer="false">
        <First></First>
        <Last></Last>
        <PhoneExt></PhoneExt>
        <Email></Email>
    </Person>
</Company>
```

You could now enter the **Person** data in Text View, but let's move to Grid View to see the flexibility of moving between views when editing a document.

Switching to Grid View

To switch to Grid View, select the menu item **View | Grid View,** or click the **Grid** tab. See how the newly added child nodes of Person are displayed.

Now let us validate the document and correct any errors that the validation finds.

1.5.5   Validating the Document

XMLSpy provides two important checks of the XML document:

- A well-formedness check
- A validation check

|	
☑	Check Well-Formedness
☑	Validate XML

Since the *Validate on Edit* feature is switched on by default, any validation error created during editing will be immediately flagged, with the error message/s being displayed in the Messages Window. If you do not want background validation, you can switch off *Validate on Edit* in the Validation settings of the Options dialog. In the event you do this, note that you can always carry out well-formed checks and validation checks at any time by invoking the respective command in the XML menu. This part of the tutorial shows you how to carry out these checks.

Checking well-formedness

An XML document is well-formed if starting tags match closing tags, elements are nested correctly, and there are no misplaced or missing characters (such as an entity without its semi-colon delimiter). You can do a well-formedness check in any editing view. Check your document as follows:

1. Select Text View.
2. Select the menu option **XML | Check Well-Formedness** or press the **F7** key. (Alternatively, you can click the command's icon in the toolbar.) A message appears in the Messages window at the bottom of the Main Window saying the document is well-formed.
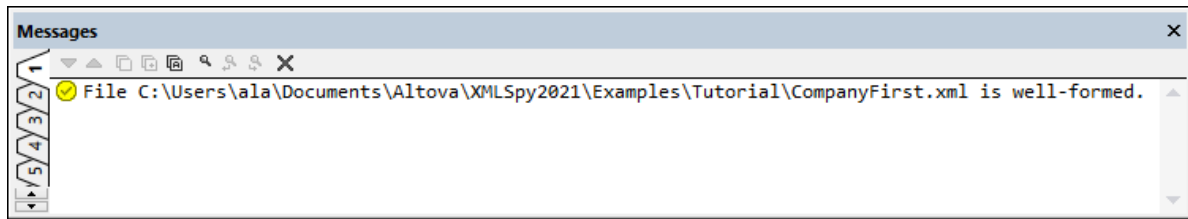
Notice that the output of the Messages window has nine tabs, with the action's result always being displayed in the active tab. So you could check well-formedness in Tab1, and switch to Tab2 for a validation check. If you do not switch tabs, the new result overwrites the previous result in the active tab.

Note: This check does not check the the XML document for conformance with the schema. Schema conformance is evaluated in the validity check.

Checking validity

An XML document is valid according to a schema if it conforms to the document structure and document content specified in that schema. You can do a validity check in any editing view. Validate your document as follows:

1.   Select Grid View.
2.   Select the menu option **XML | Validate** or press the **F8** key. (Alternatively, you can click the command's icon in the toolbar.) An error message appears in the Messages window saying the file is not valid. Mandatory elements are expected after the `city` element in `Address`. If you check your schema, you will see that the `US-Address` complex type (which you have set this `Address` element to be via its `xsi:type` attribute) has a content model in which the `city` element must be followed by a `Zip` element and a `state` element.

Fixing the invalid document

The point at which the document becomes invalid is highlighted in red, together with an error flag and a smart fix. The invalid element in this case is the `Address` element. If you click the smart fix icon, you will see the popup: *Add missing element <Zip> with sample content.* If you check the schema, you will find that the `Address/City` element must be followed by the mandatory element `zip`. To double-check this, select the `City` element and look at the Siblings entry helper. You will notice that the `zip` element is prefixed with an exclamation mark, which indicates that the element is mandatory in the current context.
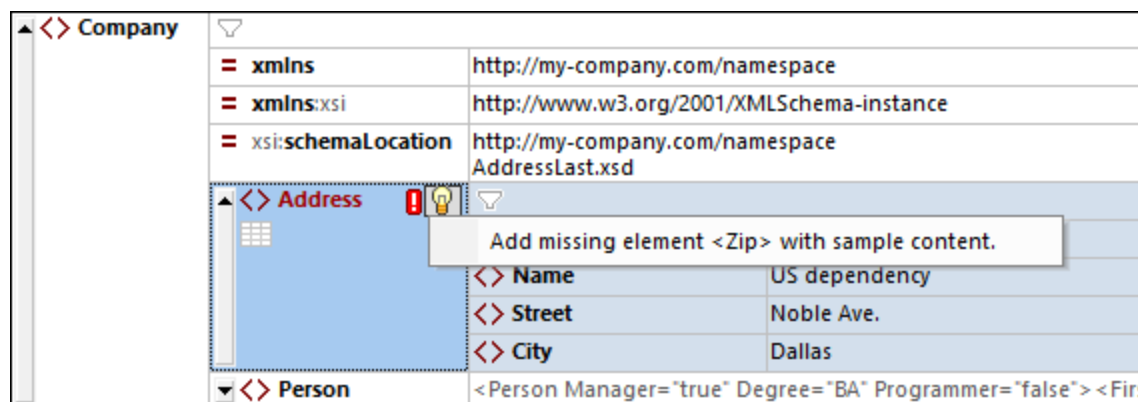
Now click the smart fix (*see screenshot above*). The `zip` element will be added and will contain sample content that makes the element valid. Enter the correct `zip` code (say `04812` for Dallas). Look at the Siblings entry helper again. It now shows that the `state` element is mandatory (it is prefixed with an exclamation mark). If you select the `state` element, the entry helper options available for it become enabled (*see screenshot below*). These are the actions to insert the `state` element after the element currently selected in the Main Window (which is `city`) or to append `state` after all the sibling elements of `city`.
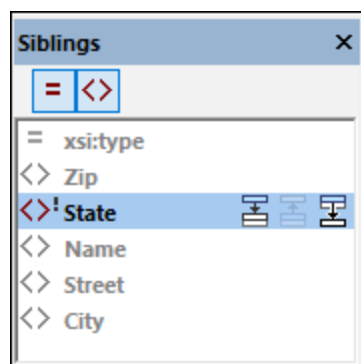
Since, in this case, both actions have the same effect, select either of the two actions. A `state` element is added after `city`. Double-click inside the contents field of `state` and enter the state's name, `Texas` (*screenshot below*). Notice that the Siblings entry helper now contains only grayed-out elements, indicating that there are no more mandatory elements to add.
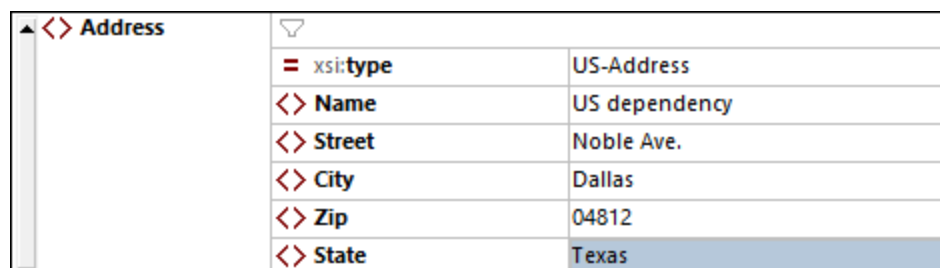
Completing the document and revalidating

Let us now complete the document (by entering the remaining data of the first `Person` element) before revalidating.

Do the following:

1. Click the value field of the element `First`, and enter a first name (say `Fred`). Then press **Enter**.

▲ <> Person	▽	
	= **Manager**	true
	= **Degree**	BA
	= **Programmer**	false
	<> **First**	Fred
	<> **Last**	
	<> **PhoneExt**	
	<> **Email**	

2. In the same way enter data for all the child elements of `Person`, that is, for `Last`, `PhoneExt`, and `Email`. You can use the Tab key to move forward through the cells. Note that the value of `PhoneExt` must be an integer with a maximum value of `99` (since this is the range of allowed `PhoneExt` values you defined in your schema). Your XML document should then look something like this in Grid View:

▲ <> Company	▽	
	= **xmlns**	http://my-company.com/namespace
	= **xmlns**:xsi	http://www.w3.org/2001/XMLSchema-instance
	= **xsi:schemaLocation**	http://my-company.com/namespace AddressLast.xsd
	▲ <> **Address**	▽
		= xsi:**type**	US-Address
		<> **Name**	US dependency
		<> **Street**	Noble Ave.
		<> **City**	Dallas
	▲ <> **Person**	▽
		= **Manager**	true
		= **Degree**	BA
		= **Programmer**	false
		<> **First**	Fred
		<> **Last**	Smith
		<> **PhoneExt**	22
		<> **Email**	Smith@work.com

3. Click 🗹 again to check if the document is valid. A message appears in the Messages window stating that the file is valid. The XML document is now valid against its schema.
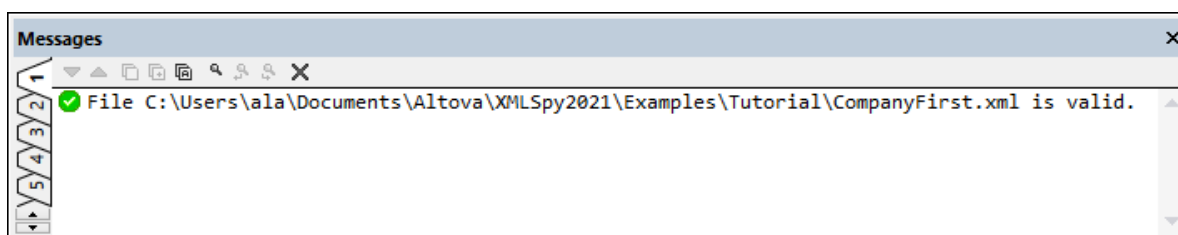
4.   Save the file with **File | Save**.

Note: An XML document does not have to be valid in order to save it. Saving an invalid document causes a prompt to appear warning you that you are about to save an invalid document. You can select **Save anyway**, if you wish to save the document in its current invalid state.

1.5.6   Adding Elements and Attributes

At this point, there is only one `Person` element in the document.

To add a new `Person` element, do the following:

1.   Click the gray sidebar to the left of the `Address` element to collapse the `Address` element. This clears up some space in the view.
2.   Select the entire `Person` element by clicking on or below the name of the `Person` element in Grid View. Notice that the `Person` element is now available in the Siblings entry helper.
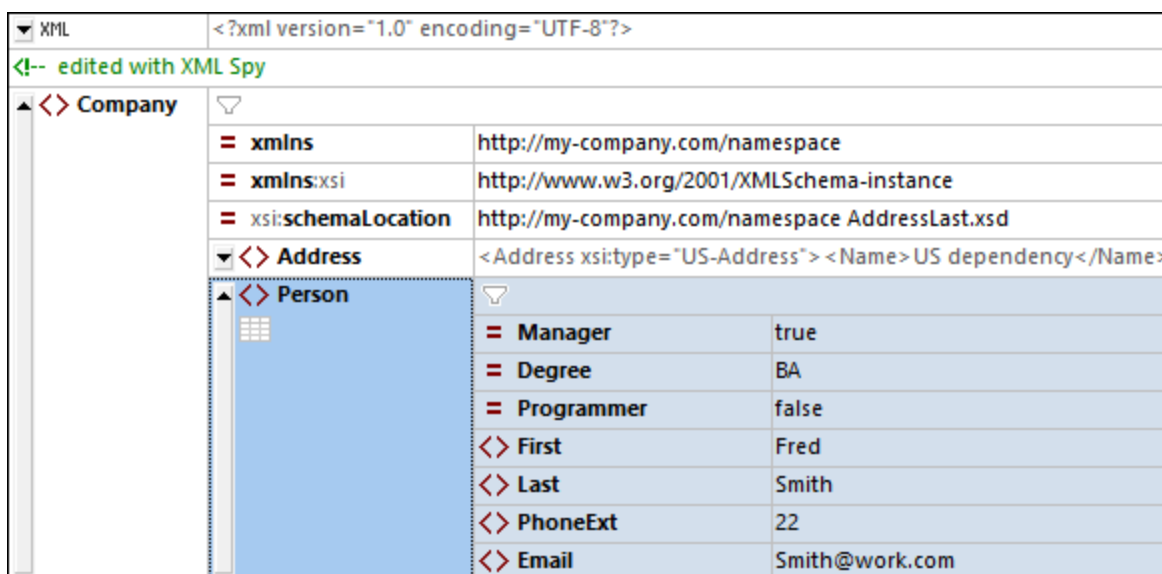
3.   Select the `Person` element in the Siblings entry helper and click either **Insert After** or **Append**. A new `Person` element is appended (*screenshot below*).

4. When the `Person` element is selected, you will see, in the Children entry helper, this element's available child attributes and elements. Double-click attributes and elements to add the same child nodes as for the first `Person` element. When the focus in the Main Window changes from the `Person` element to an added child element, you can add additional children of the `Person` element in one of two ways: (i) Switch focus to the `Person` element (by selecting it) and adding a new child from the Children entry helper; (ii) With the focus on the added child element, add a sibling child element from the Siblings entry helper. In both entry helpers, child nodes of `Person` that have already been added will be grayed out.

You could enter content for the child nodes of the `Person` element in normal Grid View, but let's switch to the Table Display of Grid View since it is more suited to editing a structure with multiple occurrences, such as `Person`.
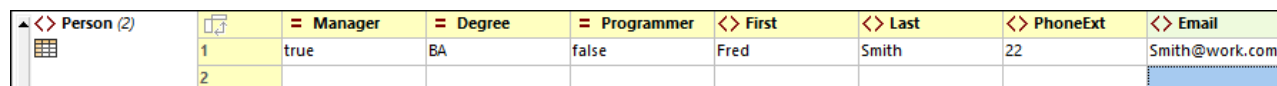
1.5.7   Editing in Table Display

Grid View contains a special view called Table Display, which is convenient for editing elements that have multiple occurrences. For example, the `Person` element has multiple occurrences (*see screenshot below*), so it can be displayed as a table. To display such an element as a table, click the **Table Display** icon of the first occurrence of the element. For example, in the screenshot below, the **Table Display** icon of the `Person` elements is circled in green. (Alternatively, select the menu command **XML | Display as Table** or the command's toolbar icon in the Grid View toolbar.)

When you click the Table Display icon, the `Person` element will be displayed as a table. In Table Display, the child nodes of the element (its attributes and elements) are displayed as columns, while each `Person` element is displayed as a row (*see screenshot below*).

Advantages of Table Display
Table Display provides the following advantages:

- You can drag-and-drop a column header to reposition entire columns relative to each other. In the actual XML document, this translates to a change of the the relative position of child nodes of all element occurrences (that correspond to the rows of the table).
- Tables—and, correspondingly, the element occurrences they represent—can be sorted (in ascending or descending order) according to the contents of any column. Use the menu command **XML | Ascending Sort** or **Descending Sort** for this.
- Additional rows (that is, element occurrences) can be appended or inserted quickly using commands in the **XML** menu. The advantage is that not only is a new element added but all its children that are represented by the columns of the table.
- You can copy-and-paste *structured data* to and from third party products, such as Microsoft Excel.
- The intelligent editing features of XMLSpy are available in Table Display also.

Displaying an element with multiple occurrences as a table

To display the **Person** element type as a table, do te following:

1.   Click the **Table Display** icon of the first occurrence of the **Person** element as described above.
2.   Select the menu option **View | Optimal Widths** or the **Optimal Widths** icon in the Grid View toolbar.

Note: Table Display can be toggled on/off for all elements that have multiple occurrences. However, child elements that were displayed as tables will continue to be displayed as tables.

Entering content in Table Display

To enter content for the second **Person** element, double-click in each of the table cells in the second row, and enter some data. The intelligent editing features are active also within cells of a table, so you can select options from dropdown lists where available (for example, Boolean content and the enumerations of the **Degree** attribute).

	= Manager	= Degree	= Programmer	<> First	<> Last	<> PhoneExt	<> Email
1	true	BA	false	Fred	Smith	22	Smith@work.com
2	false	MA	true	Alfred	Aldrich	33	Aldrich@work.com

Dynamic validation

Note that, as defined in the schema, **PhoneExt** must be an integer from **0** to **99** in order for the file to be valid. You can toggle on XMLSpy's function to validate while editing. When switched on, the file is validated each time the focus switches to a new node. Try out dynamic validation as follows:

1.   Toggle on the menu command **XML | Validate on Edit**.
2.   Enter an invalid **PhoneExt** value (any value greater than **99**), as shown in the screenshot below.
3.   Press the **Tab** key. An error icon and a smart fix icon appear in the **PhoneExt** cell (*see screenshot below*).
4.   Hover over the error icon to see the validation-error message (*screenshot below*).

<> Last	<> PhoneExt	<> Email
Smith	22	Smith@work.com
Aldrich	330 ❗💡	Aldrich@work.com

❗ Value '330' is not allowed for element <PhoneExt> .

5.   Click the smart fix icon and then the smart fix option that pops up. The invalid value will be substituted with a valid value, and the error flag disappears.

Copying XML data to and from spreadsheet applications

When you are in Table Display, you can copy data as Tab-separated text so that it can be interchanged with spreadsheet applications such as MS Excel. To copy data from your XML file, do the following:

1. Click on the `Person` element (*see screenshot below*). This selects the column headers as well as both rows of the table.

▼ ‹› **Address**	‹Address xsi:type="US-Address"› ‹Name›US dependency‹/Name› ‹Street›Noble Ave.‹/Street› ‹City›Dallas‹/City›							
▲ ‹› **Person** *(2)*		= **Manager**	= **Degree**	= **Programmer**	‹› **First**	‹› **Last**	‹› **PhoneExt**	‹› **Email**
⊞	1	true	BA	false	Fred	Smith	22	Smith@work.com
	2	false	MA	true	Alfred	Aldrich	33	Aldrich@work

2. Right-click inside the selection, and, in the context menu that appears, select the command **Copy | Copy as Tab-separated Text**. Alternatively, press **Ctrl+C**.
3. Switch to an Excel worksheet, select the cell A1 and paste (**Ctrl+V**) the XML data. The data will be entered as rows that correspond to the table structure in Table Display (*see screenshot below*).

	A	B	C	D	E	F	G	H
1	Manager	Degree	Programm	First	Last	PhoneExt	Email	
2	TRUE	BA	FALSE	Fred	Smith	22	Smith@work.com	
3	FALSE	MA	TRUE	Alfred	Aldrich	33	Aldrich@work	
4								

4. Enter a new row of data in Excel as shown in the screenshot below. Make sure that you enter a three digit number for the `PhoneExt` element (say, `444`).
5. Mark the table data in Excel, excluding the column headers (*the green frame in the screenshot below*), and copy it with **Ctrl+C**.

	A	B	C	D	E	F	G	H
1	Manager	Degree	Programm	First	Last	PhoneExt	Email	
2	FALSE	MA	TRUE	Alfred	Aldrich	33	Aldrich@work	
3	TRUE	BA	FALSE	Fred	Smith	22	Smith@work.com	
4	TRUE	Ph.D	FALSE	Colin	Coletti	444	Coletti@work.com	
5								

6. In XMLSpy make sure that the **XML | Validate on Edit** command is toggled on.
7. In the Table Display of your XML document in XMLSpy, select the `Manager` cell of the first row and paste the clipboard contents with **Ctrl+V**. Your new table will look something like the screenshot below.

▼ ‹› **Address**	‹Address xsi:type="US-Address"› ‹Name›US dependency‹/Name› ‹Street›Noble Ave.‹/Street› ‹City›Dallas‹/City›							
▲ ‹› **Person** *(3)*		= **Manager**	= **Degree**	= **Programmer**	‹› **First**	‹› **Last**	‹› **PhoneExt**	‹› **Email**
⊞	1	FALSE ❗💡	MA	TRUE ❗💡	Alfred	Aldrich	33	Aldrich@work
	2	TRUE ❗💡	BA	FALSE ❗💡	Fred	Smith	22	Smith@work.com
	3	TRUE ❗💡	Ph.D	FALSE ❗💡	Colin	Coletti	444 ❗💡	Coletti@work.com

8. The validation errors for the Boolean values have been caused by the casing difference between XML and Excel. To fix these, apply the smart fixes of the respective table cells.

Sorting the table on the contents of a column

A table in Table Display can be sorted, in ascending or descending order, on any of its columns. We want to sort the `Person` table on last name. Do this as follows:

1. Select the `Last` column by clicking its header.

▲ ⟨⟩ Person *(3)*	🖾	= Manager	= Degree	= Programmer	⟨⟩ First	⟨⟩ Last	⟨⟩ PhoneExt	⟨⟩ Email
⊞	1	true	BA	false	Fred	Smith	22	Smith@work.com
	2	false	MA	true	Alfred	Aldrich	33	Aldrich@work
	3	true	Ph.D	false	Colin	Coletti	444 ❗💡	Coletti@work.com

2. Select the menu option **XML | Ascending Sort** or click the **Ascending Sort** icon in the Grid View toolbar. The column, and the whole table with it, is now sorted alphabetically. The column remains highlighted.

▲ ⟨⟩ Person *(3)*	🖾	= Manager	= Degree	= Programmer	⟨⟩ First	⟨⟩ Last	⟨⟩ PhoneExt	⟨⟩ Email
⊞	1	false	MA	true	Alfred	Aldrich	33	Aldrich@work
	2	true	Ph.D	false	Colin	Coletti	444 ❗💡	Coletti@work.com
	3	true	BA	false	Fred	Smith	22	Smith@work.com

Since the phone extension `444` is correct but invalid, what we need to do is modify the XML Schema so that this number is valid. We will do this in the next section.

1.5.8   Modifying the Schema

Since we have a phone extension that is outside the range defined in the XML Schema (`0` to `99`), let's extend the range to `999`. Do this as described below.

1. In Grid View, select any of the `PhoneExt` cells (*see screenshot below*).

▲ ⟨⟩ Person *(3)*	🖾	= Manager	= Degree	= Programmer	⟨⟩ First	⟨⟩ Last	⟨⟩ PhoneExt	⟨⟩ Email
⊞	1	false	MA	true	Alfred	Aldrich	33	Aldrich@work
	2	true	Ph.D	false	Colin	Coletti	444 ❗💡	Coletti@work.com
	3	true	BA	false	Fred	Smith	22	Smith@work.com

2. Select the menu option **DTD/Schema | Go to definition** or click the **Go To Definition** icon in the Grid View toolbar. The associated schema, in this case `AddressLast.xsd`, is opened, and the `PhoneExt` definition will be highlighted (*screenshot below*).
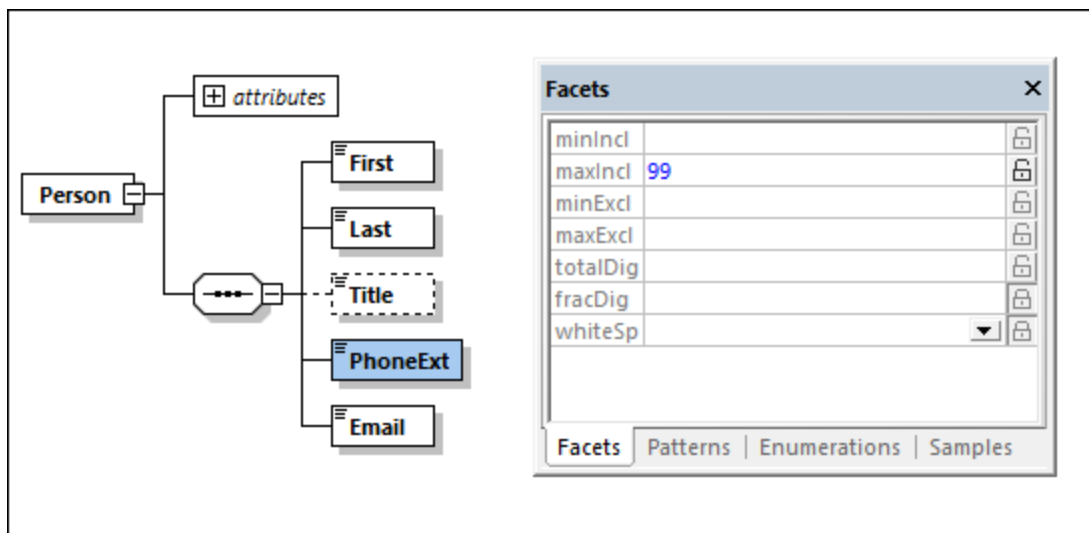
3. The element's `maxIncl` facet is `99` (*see screenshot*). Edit this value to `999`, and then save the schema.
4. Go back to the XML document and validate it. It will be valid.
5. Save your file as `CompanyLast.xml`.

Note: The Tutorial folder of XMLSpy contains a file named `CompanyLast.xml`, which contains the same data as the file you will have saved when you complete this tutorial.

1.6   XSLT Transformations

Objective

To generate an HTML file from the XML file using an XSL stylesheet to transform the XML file. You should 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`, to the XML document.
- Execute the transformation within the XMLSpy interface using one of the three built-in Altova XSLT engines. (*See note below.*)

Commands used in this section

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.

Note: XMLSpy has built-in XSLT engines for XSLT 1.0, 2.0, and 3.0. 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, we use an XSLT 3.0 stylesheet, so the XSLT 3.0 Engine will be selected automatically when the **XSL Transformation** command is invoked.

1.6.1   Assigning an XSLT File

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

1. Click the `CompanyLast.xml` tab in the main window so that `CompanyLast.xml` becomes the active document, and switch to Text View.
2. Select the menu command **XSL/XQuery | Assign XSL**.
3. 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 in the XML document relative.
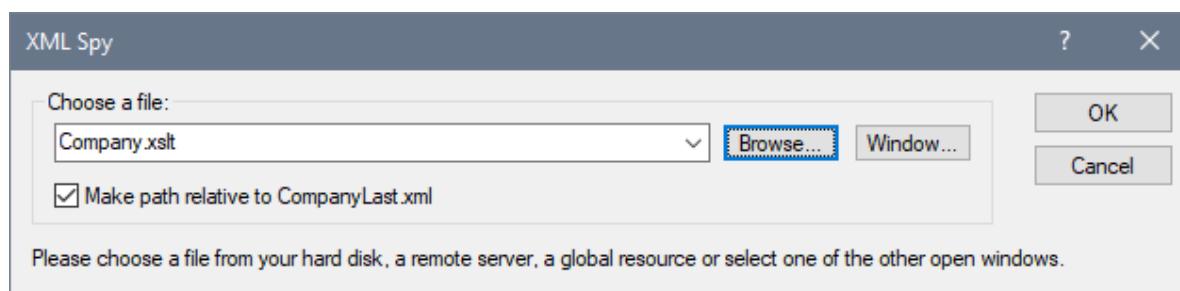
4. Click **OK** to assign the XSL file to the XML document.
5. Switch to Grid View to see the assignment (*screenshot below*). An XML-stylesheet processing instruction is inserted in the XML document that references the XSL file. If you activated the *Make Path Relative to CompanyLast.xml* check box, then the path is relative; otherwise it is absolute (as in the screenshot).

1.6.2   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 command's icon in the toolbar. This starts the transformation using the XSLT stylesheet referenced in the XML document. The output document is displayed in Browser View; it has the name `XSL Output.html`. The HTML document shows the `Company/Address` data in one block on the left, and the `Company/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	Programmer
Alfred	Aldrich	33	Aldrich@work	false	MA	true
Colin	Coletti	444	Coletti@work.com	true	Ph.D	false
Fred	Smith	22	Smith@work.com	true	BA	false

Note: Since the `Company.xslt` file is an XSLT 3.0 document, the built-in Altova XSLT 3.0 Engine is automatically selected for the transformation. If the HTML output file is not generated, ensure that, in the *XSL* section of the Options dialog (**Tools | Options**), the default file extension of the output file has been set to `.html`. This ensures that the browser reads the output document correctly as an HTML file.

1.6.3   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 `#ccccff` to `#99cc99`. Do this as follows:

1. Click the `CompanyLast.xml` tab to make this document the active document.
2. Select the menu option **XSL/XQuery | Go to XSL**. The command opens the `Company.xslt` file that is referenced in the XML document.
3. Find the start tag of the `table` element and then the element's `bgcolor` attribute (*shown highlighted in the screenshot below*). Change the attribute's value from `#ccccff` to `#99cc99`.

```
 6    <xsl:template match="/">
 7        <html>
 8            <head><title>Your company</title></head>
 9            <body>
10                <h1><center>Your Company</center></h1>
11                <xsl:apply-templates select="//my:Address"/>
12                    <table border="1" bgcolor="#ccccff">
13                        <thead align="center">
14                            <td><strong>First</strong></td>
15                            <td><strong>Last</strong></td>
16                            <td><strong>Ext.</strong></td>
17                            <td><strong>E-Mail</strong></td>
18                            <td><strong>Manager</strong></td>
19                            <td><strong>Degree</strong></td>
20                            <td><strong>Programmer</strong></td>
21                        </thead>
22                        <xsl:apply-templates select="//my:Person"/>
23                    </table>
24            </body>
25        </html>
26    </xsl:template>
27
```

4.   Select the menu option **File | Save** to save the change.
5.   Click the `CompanyLast.xml` tab to make the XML file active.
6.   Run the menu command **XSL/XQuery | XSL Transformation**; alternatively, press **F10**. A new `XSL Output.html` file appears in Browser View, with a table that has the new background color (*see screenshot below*).

Your Company

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

First	Last	Ext.	E-Mail	Manager	Degree	Programmer
Alfred	Aldrich	33	Aldrich@work	false	MA	true
Colin	Coletti	444	Coletti@work.com	true	Ph.D	false
Fred	Smith	22	Smith@work.com	true	BA	false
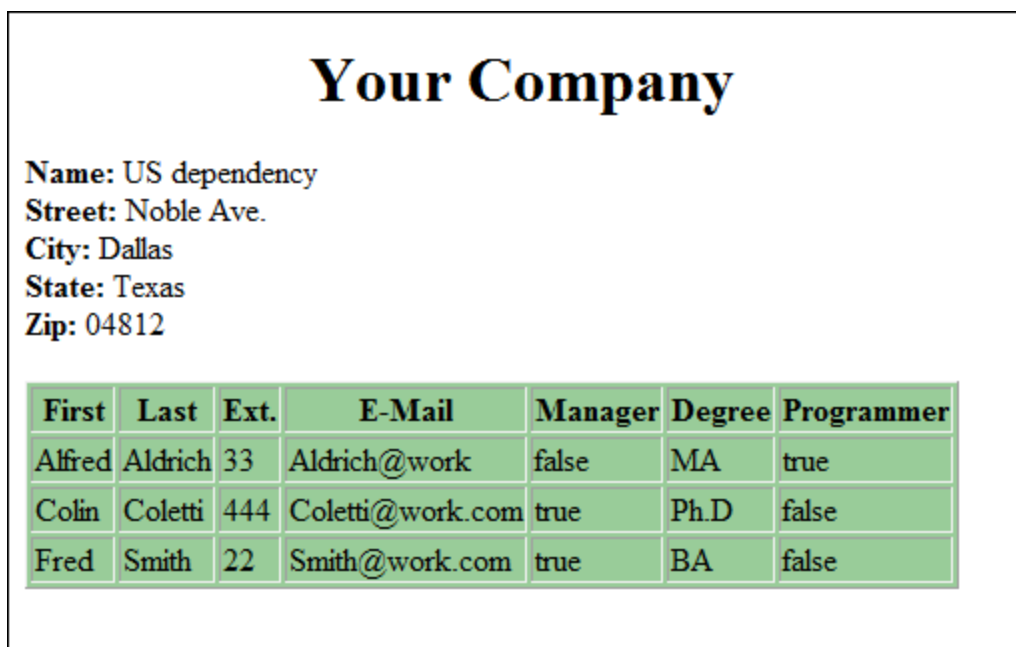
7.   Select the menu option **File | Save**, and save the output document as `Company.html`.

1.7   Project Management

This section introduces you to the project management features of XMLSpy. After first describing the benefits of organizing your XML files into projects, this section then shows you how to organize the files you have just created into a simple project.

1.7.1   Benefits of Projects

The benefits of organizing your XML files into projects are listed below.

- Files can be grouped into folders on some common criterion. For example, you could group XML files and XSD files into separate folders. You can create any hierarchy you like.
- Each folder has certain properties that you can set. For example, a folder of XML files can be assigned a schema for validation. All the files in this project folder can then be validated in a batch against the folder's schema file. If you change the project folder's schema assignment, then you can quickly run a new batch validation. You can set several other useful folder properties, such as an XSLT file for batch transformations with a single XSLT.
- Batch processing can be applied to specific folders or the project as a whole.
- A DTD or XML Schema can be assigned to specific folders, allowing validation of the files in that folder.
- XSLT files can be assigned to specific folders, allowing transformations of the XML files in that folder using the assigned XSLT.
- The destination folders of XSL transformation files can be specified for the folder as a whole.

All the above project settings can be defined using the menu option **Project | Properties**. Project commands are also available in context menus of the project and individual project folders. In the next section, you will create a project using the **Project** menu.

Additionally, the following advanced project features are available:

- XML files can be placed under source control using the menu option **Project | Source control | Add to source control**. (See the Source Control section for more information.)
- External folders on your network as well web folders can be added to projects. This allows all the features of project folders, such as validation and transformations, to be applied to folders that are on your network or on the Internet.

1.7.2   Building a Project

Having come to this point in the tutorial, you will have a number of tutorial-related files open in the Main Window. You can group these files into a tutorial project. First you create a new project and then you add the tutorial files into the appropriate sub-folders of the project.

Create a new project

Create a new project as follows:

1. Select the menu option **Project | New Project**. A new project folder called `New Project` is created in the Project Window (*screenshot below*). The new project contains empty folders for typical categories of XML files.
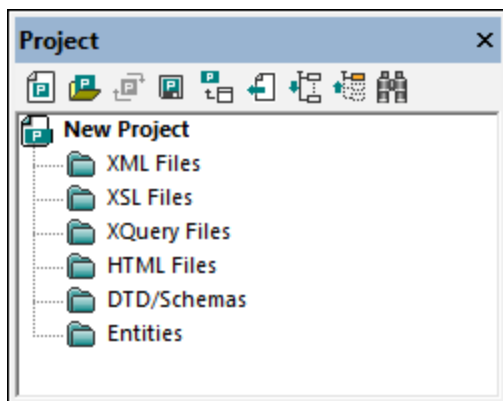
2. Click the `CompanyLast.xml` tab to make the `CompanyLast.xml` file the active file in the Main Window.
3. Select the menu option **Project | Add Active and Related Files** to project. Two files are added to the project: `CompanyLast.xml` and `AddressLast.xsd`. The XML file is added to the *XML* subfolder because it is the active file. The schema file is added to the *DTD/Schemas* folder because a reference to it is contained in the XML file, making it a related file. Note that files referenced with processing instructions, such as XSLT files, do not qualify as related files.
4. Select the menu option **Project | Save Project** and save the project under the name `Tutorial`.

Note: Folders (but not the project) each have a property named *File extensions*. This is a list of file extensions separated by semi-colons (for example, `xml;svg;wml`). This list determines what files are added to which folders when files are added to a project. For example, when active and related files are added to a project, as done above, the *File extensions* properties of the folders determine into which folders the added files will be placed.

Project and folder properties

Properties (such as the schema for validation and the XSLT for transformation) can be set not only on the entire project, but also on individual folders. You can then carry out actions, such as validation and transformation, on the entire project or individual folders. To carry out an action, right-click the project or folder, and select the action you want to carry out from the context menu that appears.

Note the following points:

*   A property that is set on a folder overrides the same property of the project.
*   If a property is set on the project, it is applied to all folders that do not have the same property set.
*   If an action is carried out on a project, it is applied to all applicable file types in all folders of the project. For example, if a validation is carried out on a project, the validation is run on all XML files in all folders of the project. In this case, the schema that has been set for the project is used for all validations, except for XML files that are in folders which have the schema validation property set to some other schema.

Adding files to the project

You can add other files to the project as well. Do this as follows:

1. Click on any open XML file (`.xml` file extension) other than `CompanyLast.xml` to make that XML file the active file. (If no other XML file is open, open one or create a new XML file.)
2. Select the menu option **Project | Add Active File to Project**. The XML file is added to the *XML Files* folder on the basis of its `.xml` file type.
3. In the same way, add an HTML file and XSD file (say, the `Company.html` and `AddressFirst.xsd` files) to the project. These files will be added to the *HTML Files* folder and *DTD/Schemas* folder, respectively.
4. Save the project, either by selecting the menu option **Project | Save Project**, or by selecting any file or folder in the Project Window and clicking the **Save** icon in the toolbar (or **File | Save**).

Note: Alternatively, you can right-click a project folder and select **Add Active File** to add the active file to that specific folder.

Other useful commands

Here are some other commonly used project commands:

- To add a new folder to a project, select **Project | Add Project Folder to Project**, and insert the name of the project folder.
- To delete a folder from a project, right-click the folder and select **Delete** from the context menu.
- To delete a file from a project, select the file and press the **Delete** key.

1.8   That's It

If you have come this far, congratulations and thank you!

We hope that this tutorial has been helpful in introducing you to the basics of XMLSpy and that you will now be able to carry out your XML work using XMLSpy as your editor. If you need more information about specific features, please use the Index or Search functions of this manual. Note that you can also print the PDF version of this tutorial. It is available as `Tutorial.pdf` in your XMLSpy application folder.

Index

A

Attribute, 41

    in schema definitions, 41

    toggle in Content model view, 41

C

Complex type, 31

    extending definition, 31

    in schema definitions, 31

Component definition,

    reusing, 31

Compositor,

    for sequences, 19

Content Model,

    creating a basic model, 19

    toggle attributes, 41

Content Model View, 16

D

Database/Table View,

    how to use, 67

Details Entry Helper, 19

Documentation,

    for schema, 46

E

Element, 27

    making optional, 27

    restricting content, 27

element type,

    specifying in XML document, 53

Enhanced Grid View,

    see Grid View, 55

Entry Helper,

    Details, 19

    in Grid View, 65

Enumeration,

    defining for attributes, 41

G

Global element,

    using in XML Schema, 39

Grid View, 65

    and Table View, 67

    appending elements and attributes, 65

    data-entry in, 55

    using Entry Helpers, 65

I

Identity constraint,

    toggle in Content model view, 41

N

Namespace,

    in schemas, 18

Navigation,

    shortcuts in schema design, 44

New XML document,

    creating, 51

O

Occurrences,

    number of, 19

Optional element,

    making, 27

P

Project management in XMLSpy, 76