Tutorial

ALTOVA®

Altova XMLSpy 2011 Tutorial

Published: 2011

Table of Contents

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:

- Creating an XML Schema. You will be introduced to XML Schemas and the various views available in XMLSpy for viewing and editing XML Schemas.
- Creating an XML document. 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. This involves assigning an XSLT file and carrying out the transformation using XMLSpy's built-in XSLT engines.
- Working with XMLSpy projects, 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\XMLSpy2011\Examples\Tutorial`

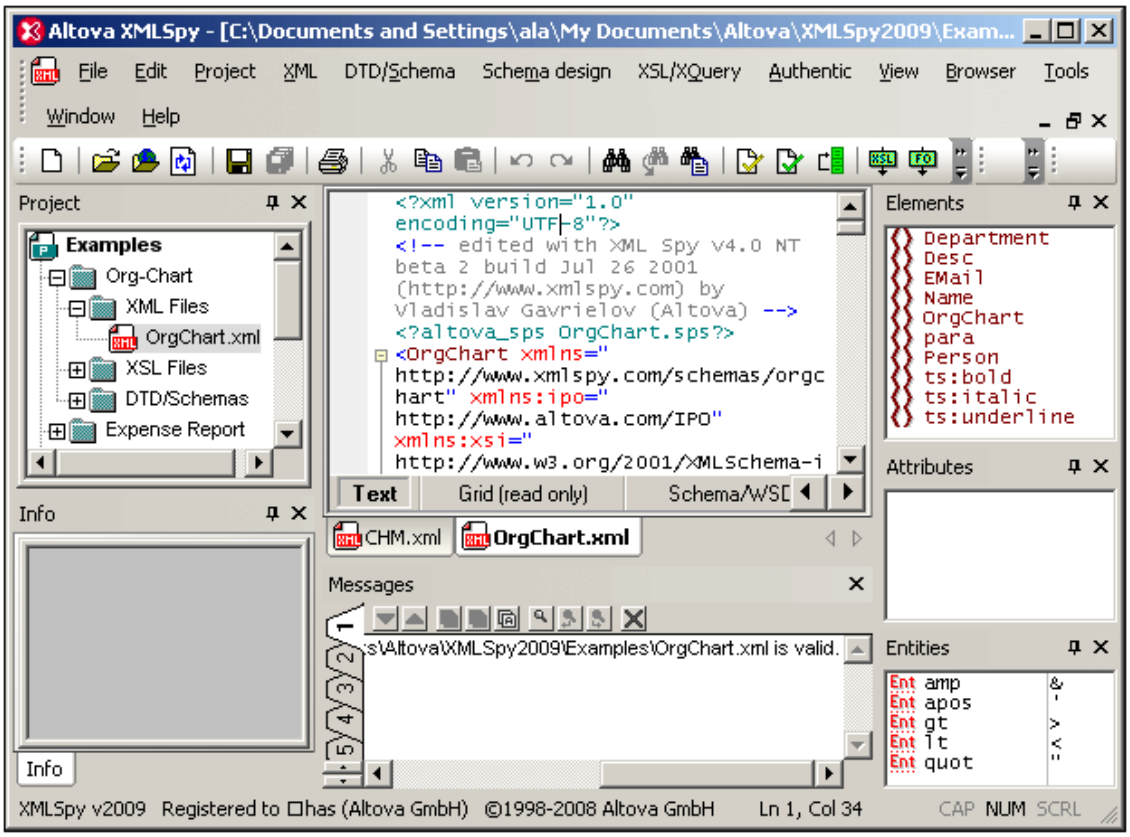
The `Examples` folder contains various XML files for you to experiment with, while the `Tutorial` folder contains all the files used in this tutorial.

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 XMLSpy Interface

The XMLSpy interface is structured into three vertical areas. The central area provides you with multiple views of your XML document. The areas on either side of this central area contain windows that provide information, editing help, and file management features.

- The left area consists of the **Project** and **Info** windows.
- The central area, called the **Main** window, is where you edit and view all types of XML documents. You can switch between different views: Text View, Schema View, Authentic View, and Browser View. In Standard Edition, Grid View and Schema View are read-only views; they are fully functional editing views in the Enterprise and Professional Editions. These views are described in detail in the individual sections about them in the User Manual.
- The right-hand area contains the three **Entry Helper** windows, which enable you to insert or append elements, attributes, and entities. What entries are displayed in the Entry Helper windows depends on the current selection or cursor location in the XML file.



The details of the interface are explained as we go along. Note that the interface changes dynamically according to the document that is active in the Main Window and according to the view selected.

2 XML Schemas

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.

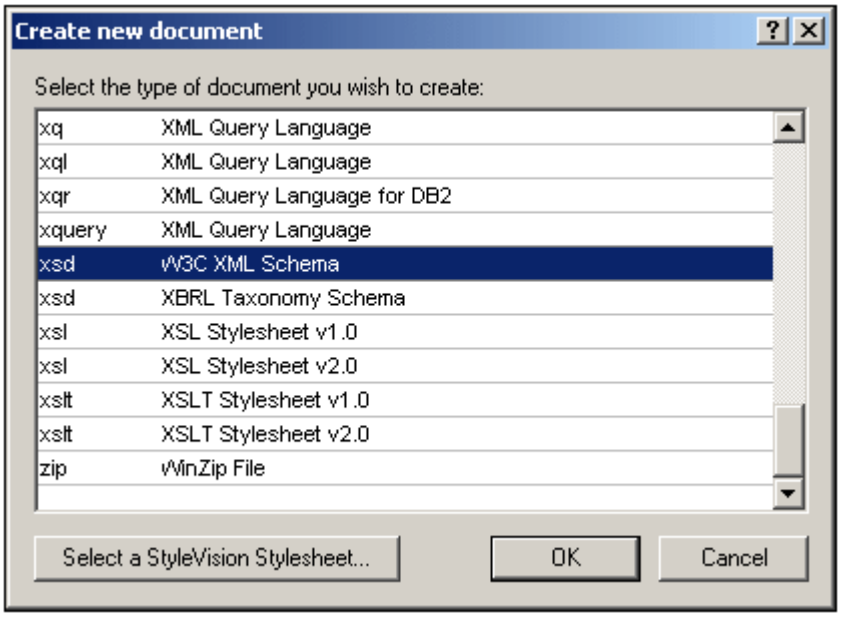
Schema editing views in XMLSpy

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 Schema 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. In the Standard Edition, XML Schema documents can be viewed in Text View, Schema View and Grid View, but can be edited only in Text View. Editing in Schema View and Grid View is available in the Enterprise and Professional editions.

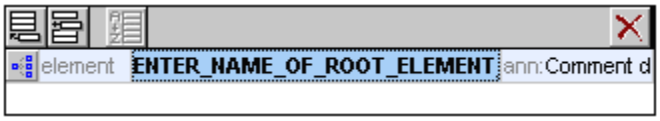
Creating a new XML Schema document

To create a new XML Schema file in XMLSpy, you must first start XMLSpy and then create a new XML Schema (. `xsd`) document. Create the document as follows:

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



2. In the dialog, select the `xsd` 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 (*screenshot below*). In Standard Edition, you cannot edit XML Schema documents in Schema View, so you must switch to Text View to edit the document.



3. The schema you will use for the rest of this tutorial is `AddressLast.xsd`, which is located in the `C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2011\\Examples\Tutorial` folder: Open this file, and explore it in Text View and Schema View. Note that in Standard Edition you cannot edit this document in Schema View. Schema View is a drag-and-drop editing view in the Enterprise and Professional editions, in which you can edit an overview of the schema's global components, and then edit each global component in a separate view (that component's content model view).

The XML file you create in the next part of the tutorial will be based on the `AddressLast.xsd` schema, so make sure that you do not modify the `AddressLast.xsd` schema that is supplied withy our installation.

3 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 Text View using intelligent entry helpers.
- Validate the XML document.

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.

F7. Checks for well-formedness.

F8. Validates the XML document against the associated DTD or Schema.

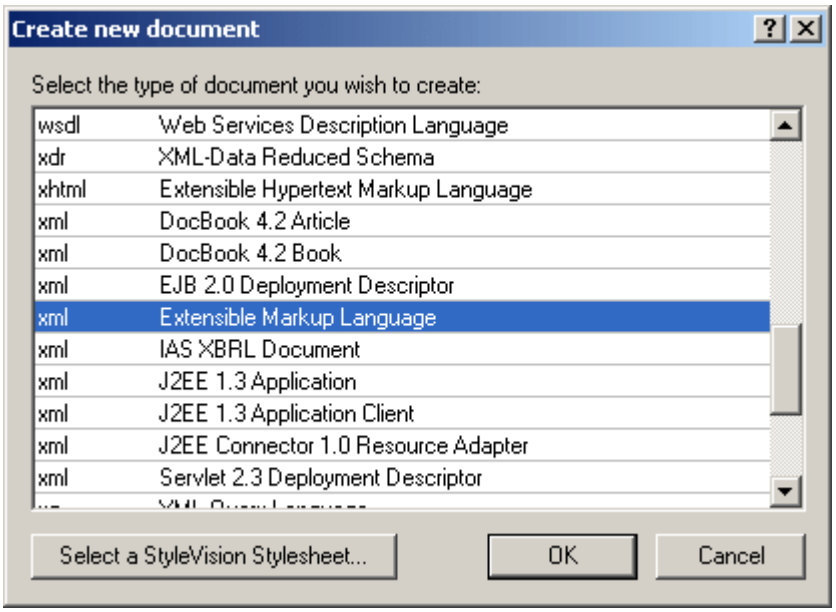
Opens the associated DTD or XML Schema file.

3.1 Creating a New XML File

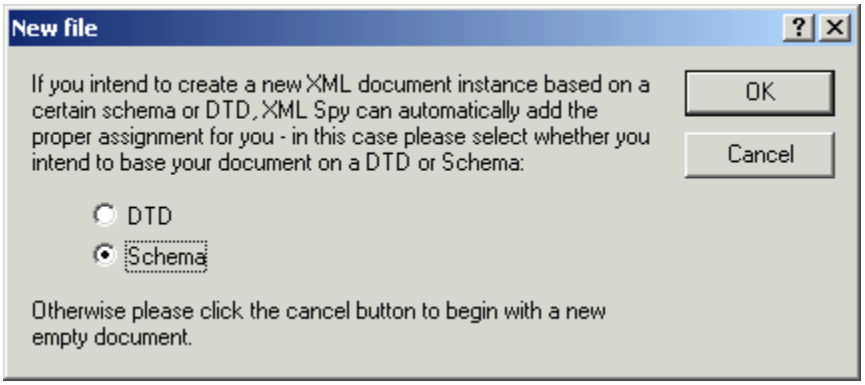
When you create a new XML file in XMLSpy, you are given the option of basing it on a schema (DTD or XML Schema) or not. In this section you will create a new file that is based on the `AddressLast.xsd` schema you created earlier in the tutorial.

To create the new XML file:

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

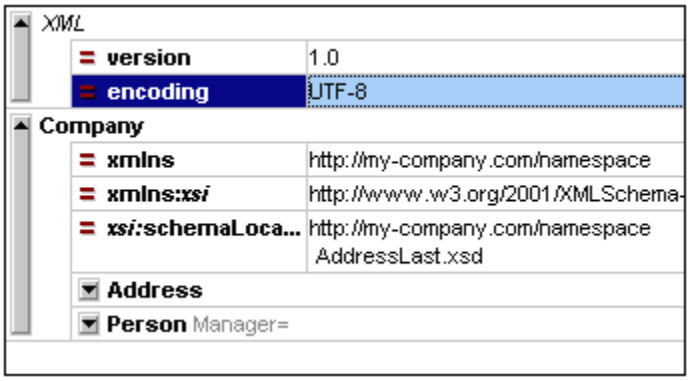


2. Select the `Extensible Markup Language` entry (or generic XML document entry) from the dialog, and confirm with **OK**. A prompt appears, asking if you want to base the XML document on a DTD or Schema.

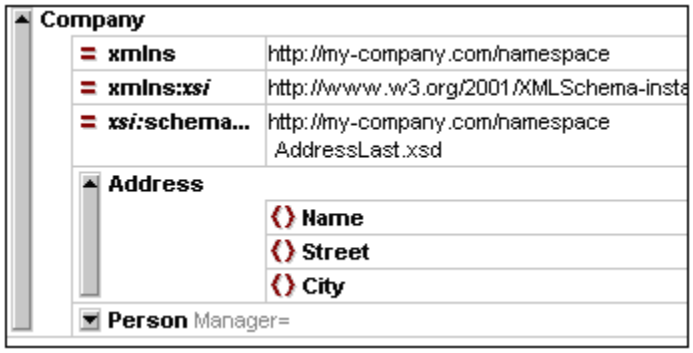


3. Click the Schema radio button, and confirm with **OK**. A further dialog appears, asking you to select the schema file your XML document is to be based on.
4. Use the Browse or Window buttons to find the schema file. The Window button lists all files open in XMLSpy and projects. Select `AddressLast.xsd` (see Tutorial introduction for location), and confirm with **OK**. An XML document containing the main elements defined by the schema opens in the main window. Notice the structure of the document in Text View.
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:

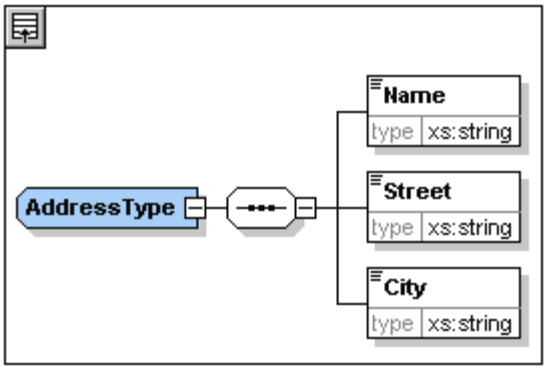


7. Click on the icon next to `Address`, to view the child elements of `Address`. Your document should look like this:

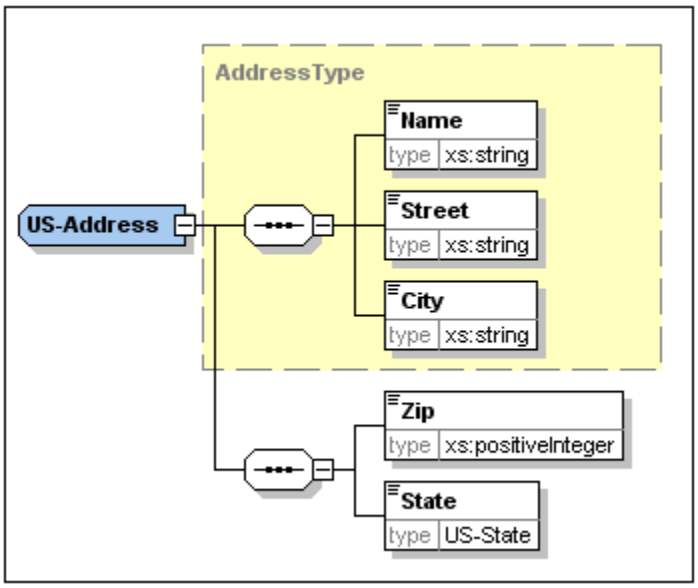


3.2 Specifying the Type of an Element

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 Grid 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.

Do this as follows. In the XML document, on the `Address` element, enter an `xsi:type` attribute with a value of `US-Address` (*screenshot below*).


```
<Address xsi:type="US-Address">
  <Name>US dependency</Name>
  <Street>Noble Ave.</Street>
  <City>Dallas</City>
  <Zip>04812</Zip>
  <State>Texas</State>
</Address>
```

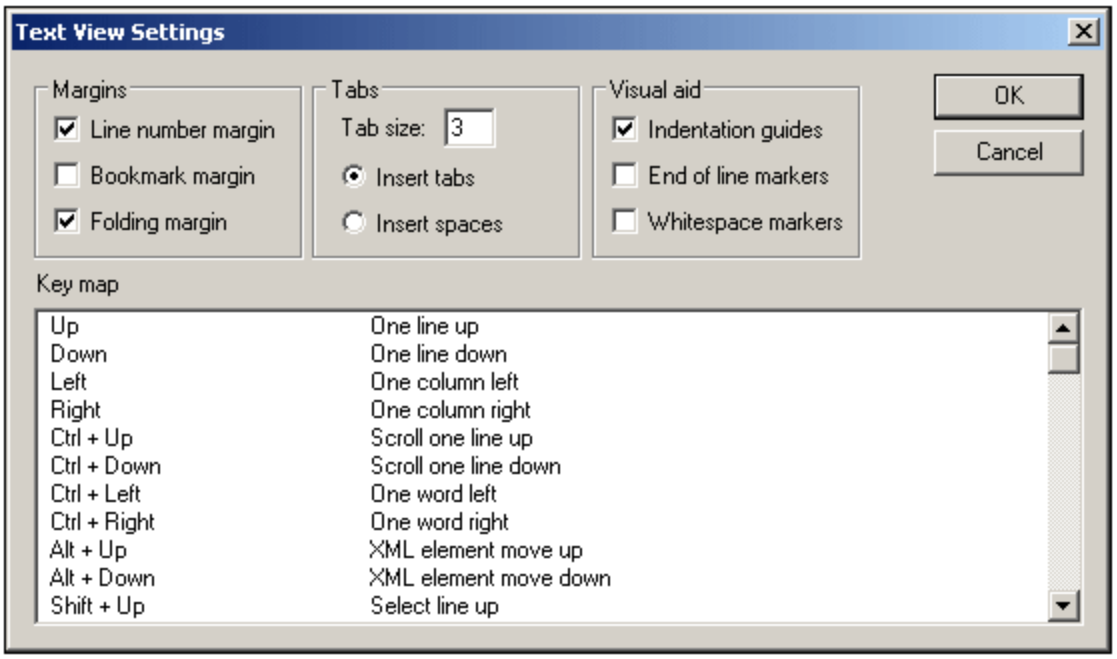
You can now enter data for the `Address` element. Enter the values shown in the screenshot above. Then delete the `Person` element (it will be added in the next section of the tutorial).

Please 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.

3.3 Entering Data in Text View

Text View is ideal for editing the actual data and markup of XML files because of its DTD/XML Schema-related intelligent editing features.

Structural layout features
In addition, Text View has a number of viewing and structural editing features that make editing large sections of text easy. These features can be switched on and off in the Text View Settings dialog (**View | Text View Settings**, *screenshot below*).



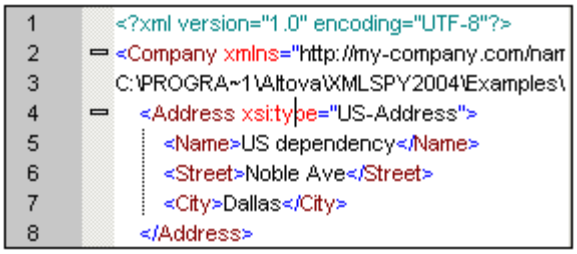
The following margins in Text View can be switched on and off:

- Line number margins
- Bookmark margins, in which individual lines can be highlighted with a marker
- Source folding margins, which contain icons to expand and collapse the display of elements

Additionally, visual aids such as indentation guides, end-of-line markers, and whitespace markers can be switched on and off, by checking and unchecking, respectively, their check boxes in the *Visual Aid* pane of the Text View Settings dialog (*see screenshot above*).

The bookmark feature is useful for setting up markers in your document. To insert a bookmark, use the command **Edit | Insert/Remove Bookmark**. Once bookmarks have been inserted you can navigate these bookmarks using commands in the **Edit** menu.

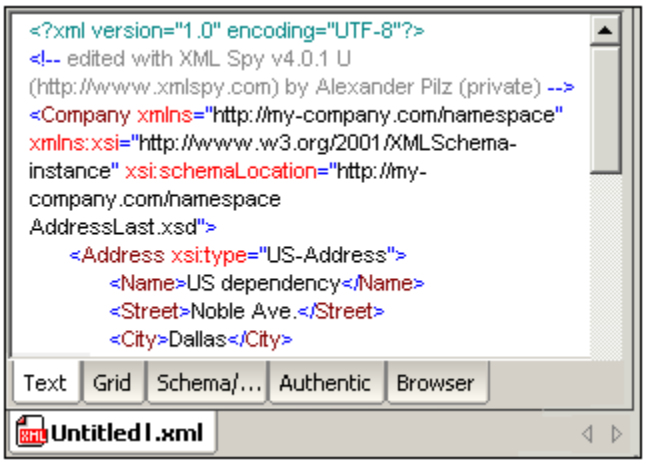
The screenshot below shows the current XML file in Text View with all structural editing features enabled. For the sake of clarity, none of the line numbers, indentation guides, etc, will be shown in Text View in rest of this tutorial. Please see the User Manual for more information on Text View.



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.

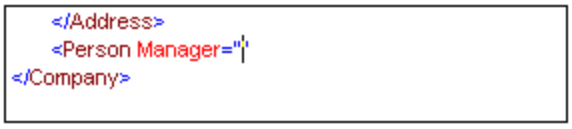
Do the following:

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.

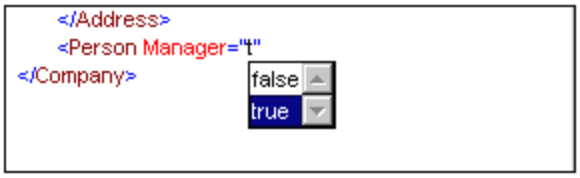


2. Place the text cursor after the end tag of the Address element, and press **Enter** to add a new line.
3. Enter the less-than angular bracket **<** at this position. A dropdown list of all elements allowed at that point (according to the schema) is displayed. Since only the `Person` element is allowed at this point, it will be the only element displayed in the list.

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 for the `Manager` attribute, select `true`.

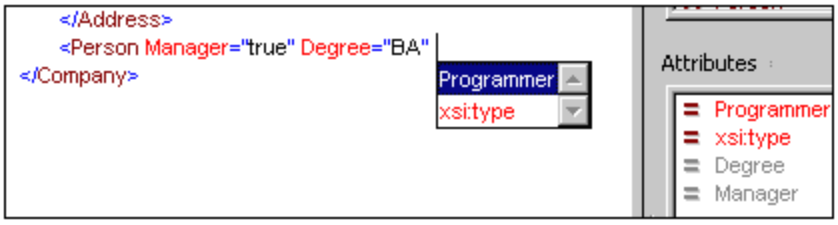


Press **Enter** to insert the value `true` at the cursor position.

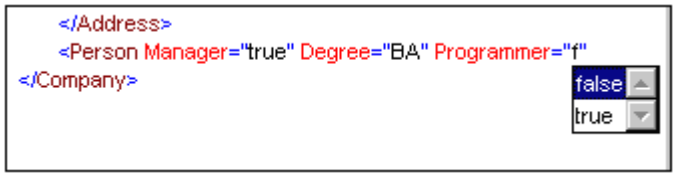
6. Move the cursor to the end of the line (using the **End** key if you like), and press the space bar. This opens a dropdown list, this time containing a list of attributes allowed at that point. Also, in the Attributes Entry Helper, the available attributes are listed in red. The Manager attribute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

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`).

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.



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



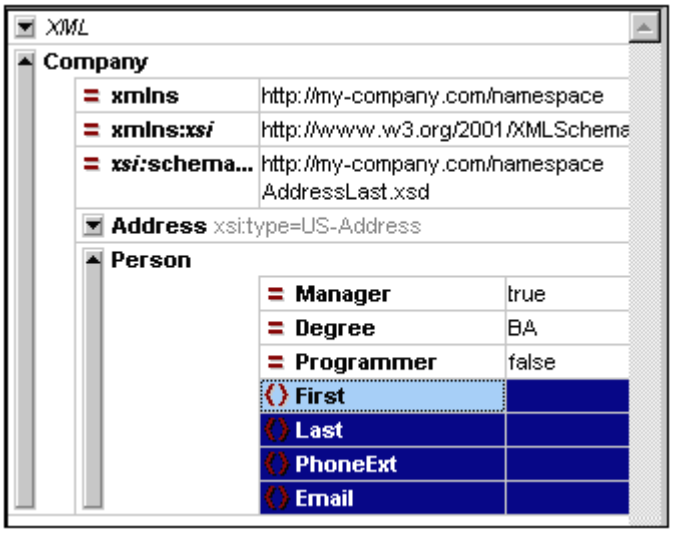
10. Enter the letter `"f"` and press **Enter**.
11. Move the cursor to the end of the line (with the **End** key), and enter the greater-than angular bracket **>**. XMLSpy automatically inserts all the required child elements of `Person`. (Note that the optional `Title` element is not inserted.) Each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


```
<?xml version="1.0" encoding="UTF-8"?>
<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. The newly added child elements of `Person` are highlighted.



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

3.4 Validating the Document

XMLSpy provides two evaluations of the XML document:

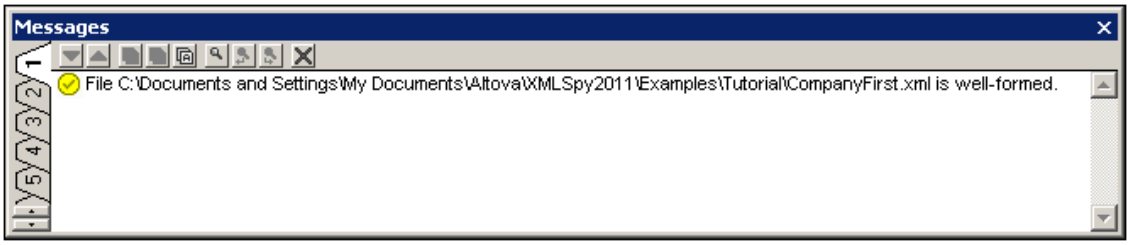
- A well-formedness check
- A validation check

If either of these checks fails, we will have to modify the document appropriately.

Checking well-formedness
An XML document is well-formed if starting tags match closing tags, elements are nested correctly, there are no misplaced or missing characters (such as an entity without its semi-colon delimiter), etc.

You can do a well-formedness check in any editing view. Let us select Text View. To do a well-formedness check, select the menu option **XML | Check well-formedness**, or press the **F7** key, or click  . 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 9 tabs. The validation output is always displayed in the active tab. Therefore, you can check well-formedness in Tab1 for one schema file and keep the result by switching to Tab2 before validating the next schema document (otherwise Tab1 is overwritten with the validation result).



Please note: This check does not check the structure of the XML file 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 structure and content specified in that schema.

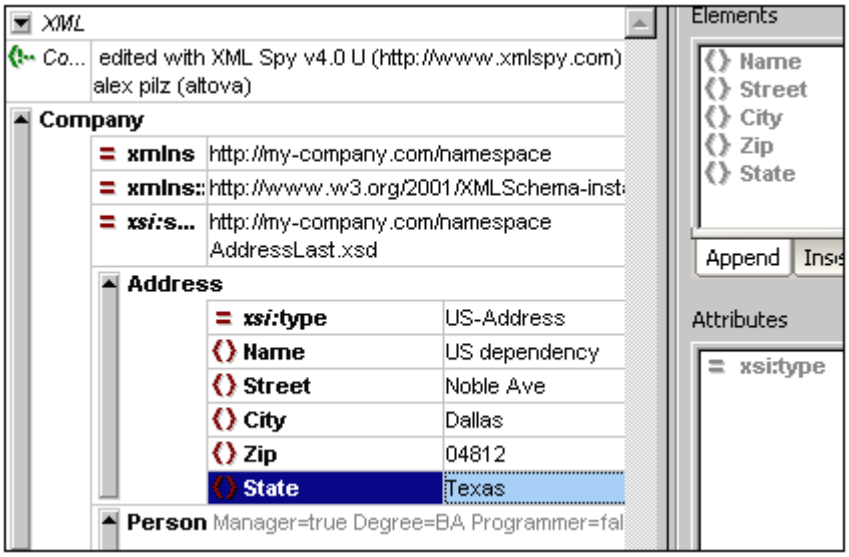
To check the validity of your XML document, first select Text View, then select the menu option **XML | Validate**, or press the **F8** key, or click  . An error message appears in the Messages window saying the file is not valid. Mandatory elements are expected after the `City` element in `Address`. If you check your schema, you will see that the `US-Address` complex type (which you have set this `Address` element to be with 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 this case the `City` element.

Now look at the Elements Entry Helper (at top right). Notice that the `Zip` element is prefixed with an exclamation mark, which indicates that the element is mandatory in the current context.

To fix the validation error:

1. Place the cursor after the `City` element and, in the Elements Entry Helper, double-click the `Zip` element.
2. Ensure the cursor is between the start and end tags of the `Zip` element, and enter the Zip Code of the State (`04812`), then confirm with **Enter**. The Elements Entry Helper now shows that the `State` element is mandatory (it is prefixed with an exclamation mark).
3. Place the cursor after the Zip element, and in the Elements Entry Helper, double-click the `State` element. Then enter the name of the state (`Texas`). Confirm with **Enter**. The Elements Entry Helper now contains only grayed-out elements. This shows that there are no more required child elements of `Address`. Switch to Grid View to view your changes (*screenshot below*).



Completing the document and revalidating

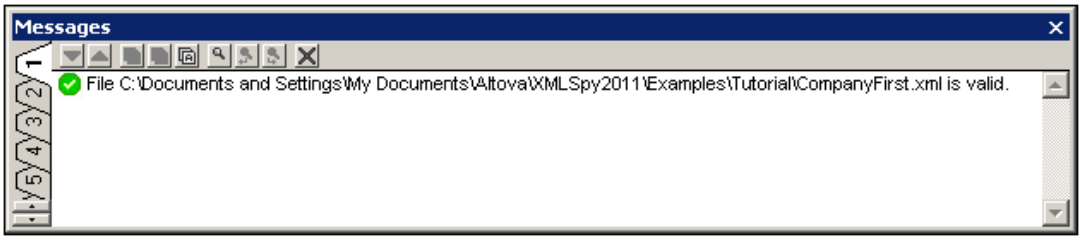
Let us now complete the document (enter data for the `Person` element) before revalidating.

Do the following:

1. In the element `First`, enter a first name (say `Fred`). Then press **Enter**.
2. In the same way enter data for all the child elements of `Person`, that is, for `Last`, `PhoneExt`, and `Email`. 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 Text View:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.0.1 U
(http://www.xmlspy.com) by Alexander Pilz
(private) -->
<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>
    <Zip>04812</Zip>
    <State>Texas</State>
  </Address>
  <Person Manager="true" Degree="BA" Programmer
="false">
    <First>Fred</First>
    <Last>Smith</Last>
    <PhoneExt>22</PhoneExt>
    <Email>Smith@work.com</Email>
  </Person>
</Company>
```

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. Select the menu option **File | Save** and 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.

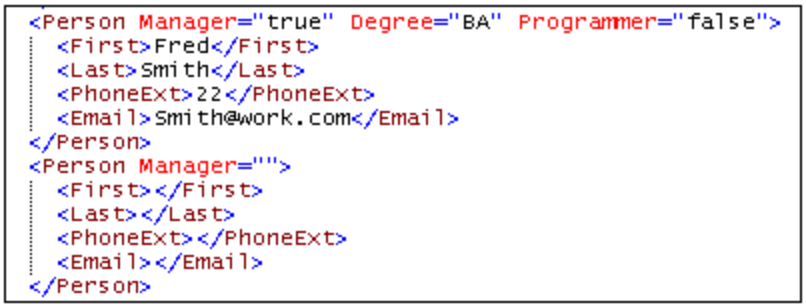
Please note: An XML document does not have to be valid in order to save it. Saving an invalid document causes a prompt to appear warning you that you are about to save an invalid document. You can select **Save anyway**, if you wish to save the document in its current invalid state.

3.5 Adding Elements and Attributes

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

To add a new `Person` element:

1. Place the cursor after the already created `Person` element.
2. Press Enter. This creates a new line, with the cursor positioned at the start of the new line. Notice that the `Person` element is now available in the Elements Entry Helper.
3. Double-click the `Person` element in the Elements Entry Helper. A new `Person` element with all mandatory child elements is appended (*screenshot below*). Notice that the optional `Title` child element of `Person` is not inserted.



4. Place the cursor before the closing angular bracket of the opening tag. Then, in the **Append** tab of the Attributes Entry Helper, double-click the `Programmer` entry. This inserts an empty `Programmer` attribute after the `Manager` attribute. The `Programmer` attribute is now grayed out in the Attributes Entry Helper.

Select the menu option **File | Save As...** and save the file as `CompanyLast.xml`. (Remember to rename the original `CompanyLast.xml` file that is delivered with XMLSpy to something else, like `CompanyLast_orig.xml`).

Please note: The `CompanyLast.xml` file delivered with XMLSpy is in the in the `Tutorial` folder.

4 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 two 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.

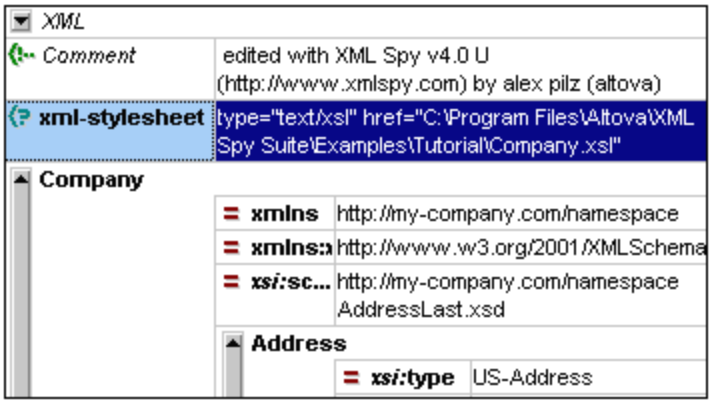
 **XSL/XQuery | XSL Transformation (F10)**, or the toolbar icon , transforms the active XML document using the XSL stylesheet assigned to the XML file. If an XSL file has not been assigned then you will be prompted for one when you select this command.

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

4.1 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 to the XSL file (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*).

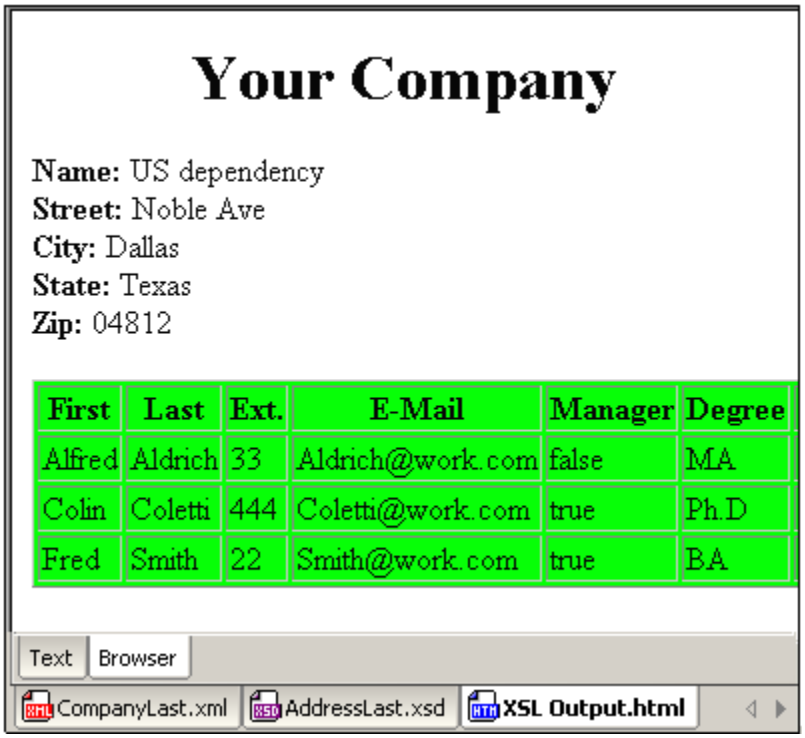


An `XML-stylesheet` processing instruction is inserted in the XML document that references the XSL file. If you activated the Make Path Relative to `CompanyLast.xml` check box, then the path is relative; otherwise absolute (as in the screenshot above).

4.2 Transforming the XML File

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

1. Ensure that the XML file is the active document.
2. Select the menu option **XSL/XQuery | XSL Transformation (F10)** or click the  icon. This starts the transformation using the XSL stylesheet referenced in the XML document. (Since the `Company.xsl` file is an XSLT 1.0 document, the built-in Altova XSLT 1.0 Engine is automatically selected for the transformation.) The output document is displayed in Browser View; it has the name `XSL Output.html`. (If the HTML output file is not generated, ensure that, in the XSL tab of the Options dialog (**Tools | Options**), the default file extension of the output file has been set to `.html`.) The HTML document shows the Company data in one block down the left, and the Person data in tabular form below.



Please note: Should you only see a table header and no table data in the output file, make sure that you have defined the target namespace for your schema. The namespace must be **identical** in all three files (Schema, XML, and XSL).

4.3 Modifying the XSL File

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

Do the following:

1. Click the `CompanyLast.xml` tab to make it the active document, and make sure you are in Grid View.
2. Select the menu option **XSL/XQuery | Go to XSL**.

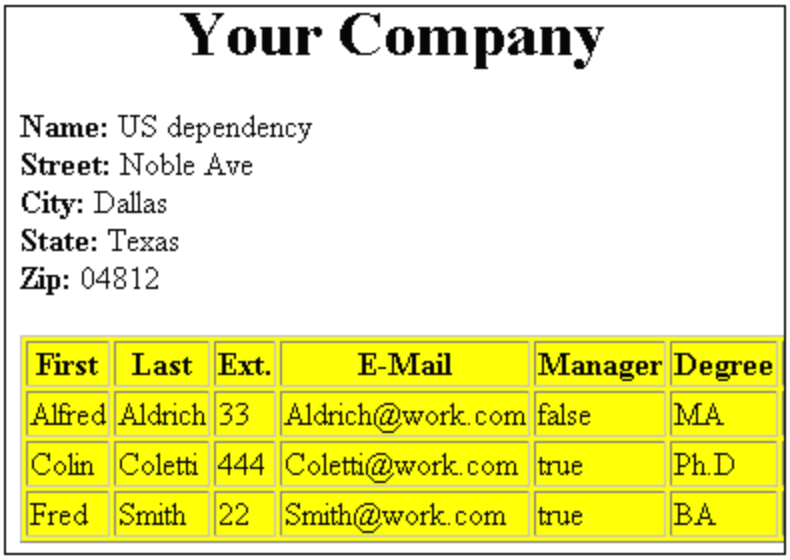
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:xsi ="http://www.w3.org/2000/10/XMLSchema-instance"
xmlns:my="http://my-company.com/namespace">

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

   The command opens the `Company.xsl` file referenced in the XML document.
3. Find the line `<table border="1" bgcolor="lime">`, and change the entry `bgcolor="lime"` to `bgcolor="yellow"`.

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

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



Your Company

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

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

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

5 Project Management

This section introduces you to the project management features of XMLSpy. After learning about the benefits of organizing your XML files into projects, you will organize the files you have just created into a simple project.

5.1 Benefits of Projects

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

- Files and URLs can be grouped into folders by common extension or any other criteria.
- 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 | Project Properties...**. In the next section, you will create a project using the Project menu.

Additionally, the following advanced project features are available:

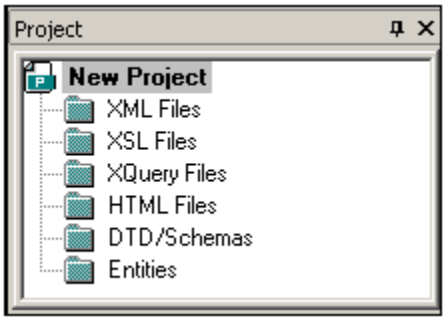
- XML files can be placed under source control using the menu option **Project | Source control | Add to source control...**. (Please see the Source Control section in the online help for more information.)
- Personal, network and web folders can be added to projects, allowing batch validation.

5.2 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.

Creating a basic project
To create a new project:

1. Select the menu option **Project | New Project**. A new project folder called `New Project` is created in the Project Window. The new project contains empty folders for typical categories of XML files in a project (*screenshot below*).



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`. 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`.

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 (with the `.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**).

Please 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.

6 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. If you need more information please use the context-sensitive online help system, or print out the PDF version of the tutorial, which is available as `tutorial.pdf` in your XMLSpy application folder.

Index