

This tutorial is designed to give you a quick hands-on introduction to the FrameSLT Node Wizard. For a more detailed explanation of the Node Wizard, please see your *User Guide*.

Before using this tutorial, note the following:

- The tutorial is designed for use with the sample file `FSLT_Node_Wizard_Sample_1.fm`. To help prevent confusion, all other files should be closed.
- All the XPath expressions used in the tutorial have been preset in the default XPath favorites that came with FrameSLT. To retrieve the XPath favorites, click **Get** in the **XPath favorites** area of the Node Wizard.
- XPath expressions are case-sensitive, and a single character error in an XPath expression will cause it to fail. The expressions in this tutorial are guaranteed to work if entered correctly... take your time and make sure you enter them exactly as shown.

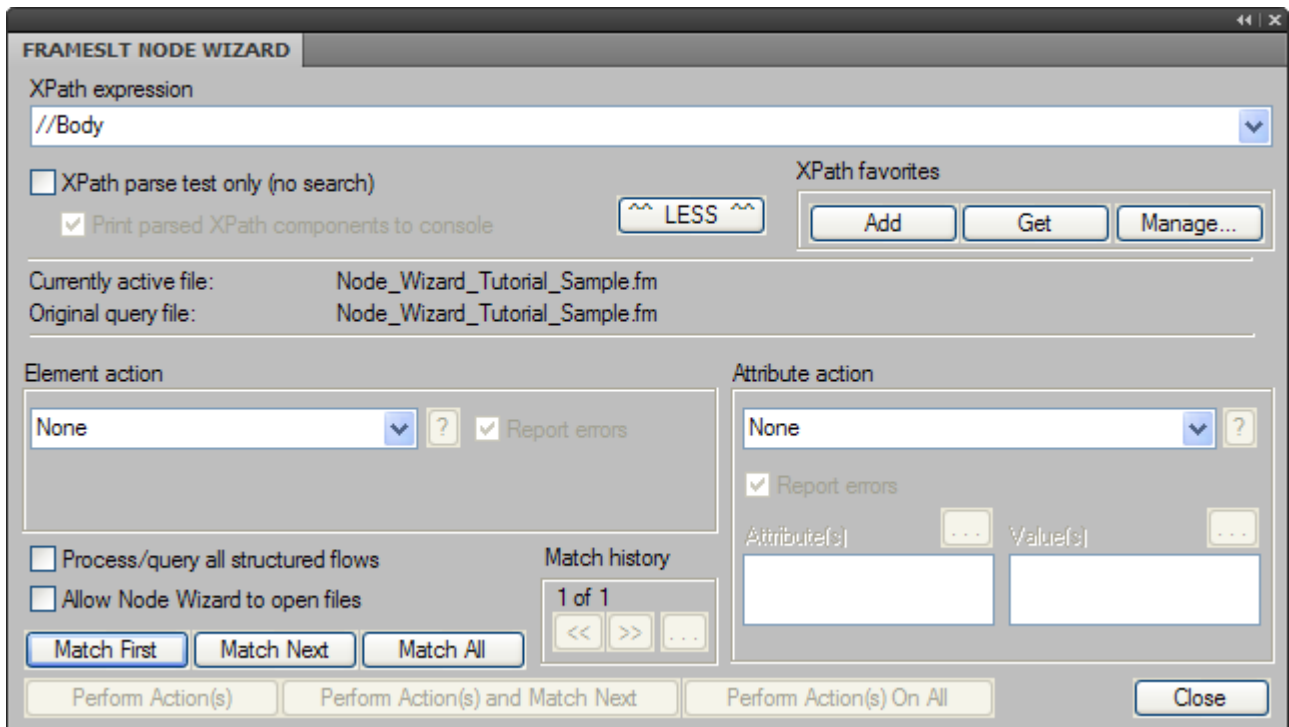
## Part 1 - Simple searching

In this part, you will use some XPath expressions to query the sample document, `Node_Wizard_Tutorial_Sample.fm`.

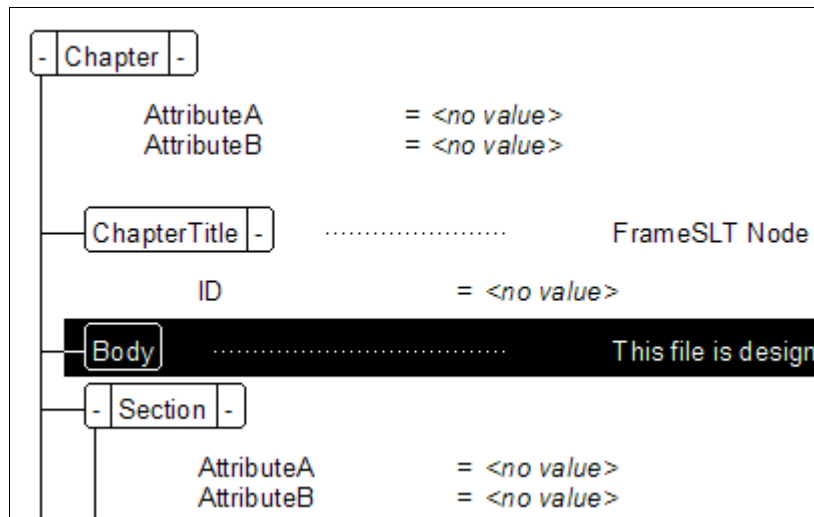
1. Open `Node_Wizard_Tutorial_Sample.fm` and close all other open files.
2. Open the **Structure View** window.
3. Select **FrameSLT > Node Wizard** to open the Node Wizard.
4. Make sure the **Process/query all structured flows** checkbox is **unchecked**.
5. In the **XPath** box, enter the following expression:

```
//Body
```

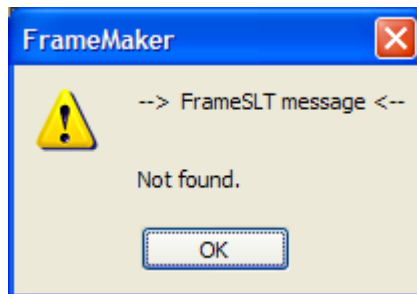
This XPath means literally, "Match all `Body` elements in the document that are the highest-level element (HLE) or descendants of the HLE." Or in other words, "Match all `Body` elements in the document."



- Click **Match First**, and note how the first `Body` element was found.



- Click **Match Next** repeatedly, and note how each `Body` element is found, in document order.
- Click **Match First** again, and note how the query restarts at the beginning.
- Again, click **Match Next** repeatedly until the last `Body` element is found, indicated by the “Not found” message. XPath queries always have some starting point and proceed sequentially until the last match is made.



- Under **Match History**, click the `<<` and `>>` buttons, noting how you can shuffle backwards and forwards through the history of previous matches. The match history is just that... a simple history of matches from the previous XPath queries. Moving through the history does not involve any usage of the XPath.
- Click **Match All** and note how the query collects all possible matches automatically and stores them in the match history. This button has the same effect as clicking **Match First**, then **Match Next** until all matches are exhausted..

## Part 2 - More searching

- In the **XPath** box, enter the following expression:

```
//Section/Section/Body
```

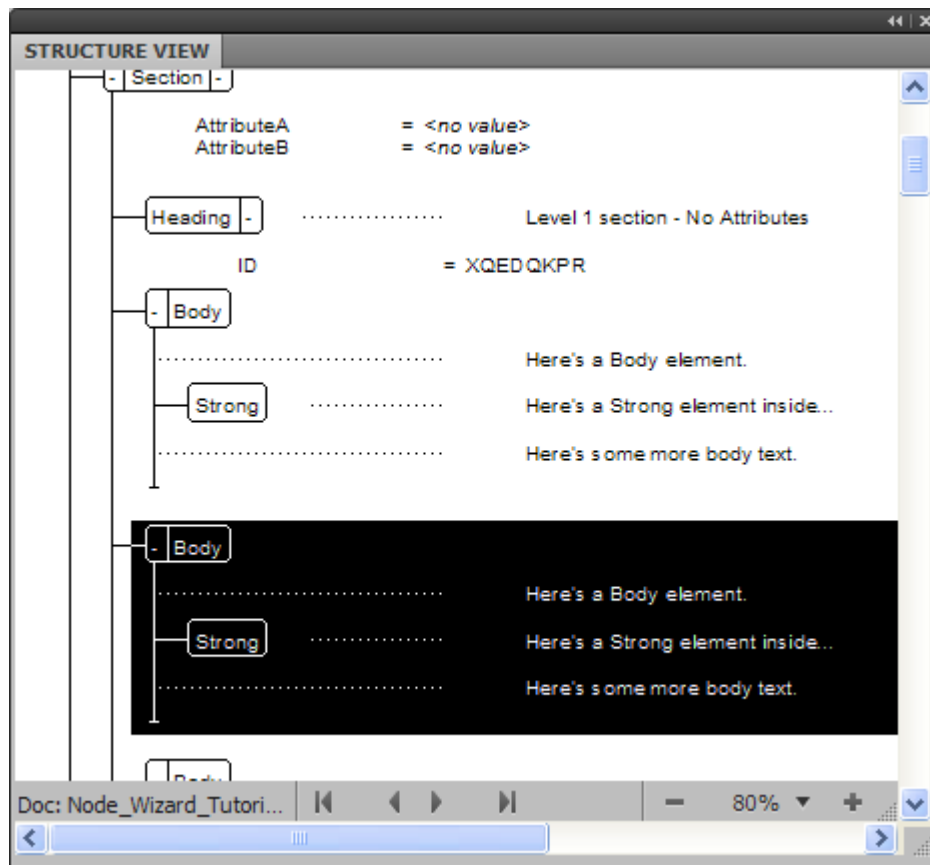
This XPath means literally, “Match all `Body` elements with at least two `Section` ancestors.”

- Click **Match First** and **Match Next**, noting how only those `Body` elements within the “level 2” sections are selected.
- In the **XPath** box, enter the following expression:

```
//Body[2]
```

This XPath means literally, “Match all `Body` elements that are in the second position in their respective branches, relative to any sibling `Body` elements.”

4. Click **Match First** and **Match Next**, noting the results.



5. In the **XPath** box, enter the following expression:

```
//Comment[@Author = "Jazmyn Campbell"]
```

This XPath means literally, “Match all `Comment` elements whose `Author` attribute is set to “Jazmyn Campbell”.

6. Click **Match First** and **Match Next**, noting the results. Randall's comments are not found.

7. In the **XPath** box, enter the following expression:

```
//*[ @AttributeA != "" ]
```

This XPath means literally, “Match any element whose `AttributeA` element is not empty, or in other words, specified with any value.”

8. Click **Match First** and **Match Next**, noting the results.

9. In the **XPath** box, enter the following expression:

```
//*[ . = "Here is a BulletItem" ]
```

This XPath means literally, “Match all elements that contain only and exactly the text ‘Here is a BulletItem’.”

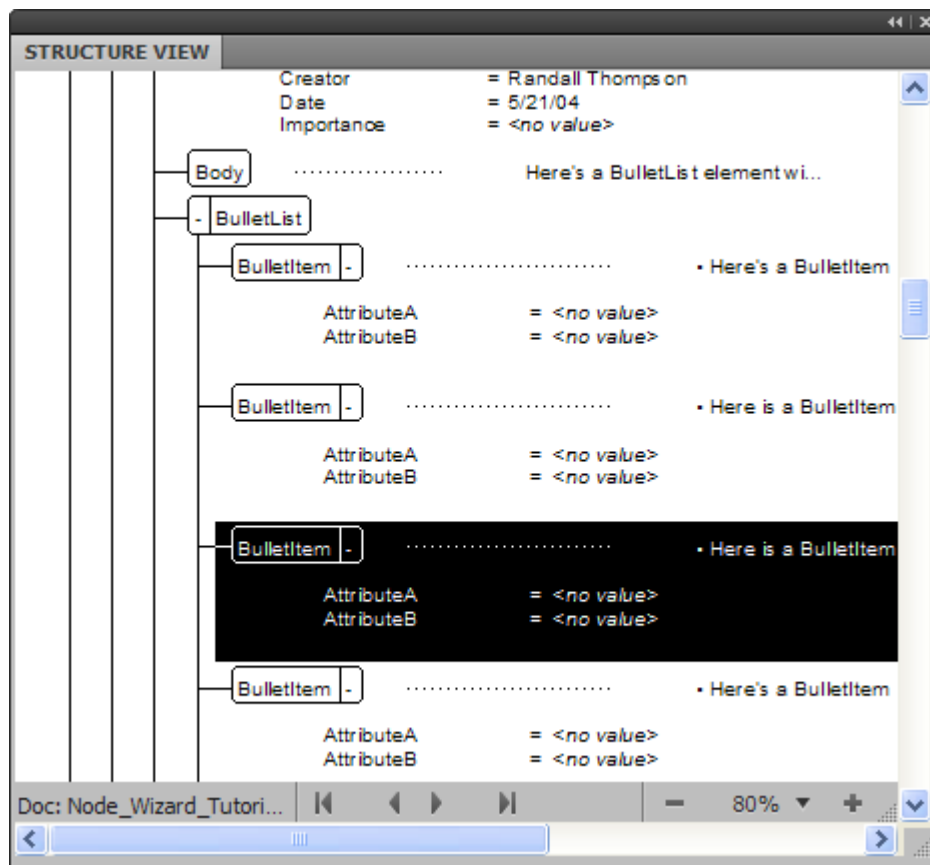
10. Click **Match First** and **Match Next**, noting the results.

11. In the **XPath** box, enter the following expression:

```
//*[ . = "Here is a BulletItem" and position() = 3 ]
```

This XPath means literally, “Match all elements that contain only and exactly the text ‘Here is a BulletItem’, and are in the third position of their respective branches.”

12. Click **Match First** and **Match Next**, noting the results.



13. In the **XPath** box, enter the following expression:

```
//node()[contains(., "men")]
```

This XPath means literally, “Match all element and text nodes that contain the text fragment “men”.

14. Click **Match First** and **Match Next**, noting the results. The words “element” and “comment” contain the text fragment “men”.

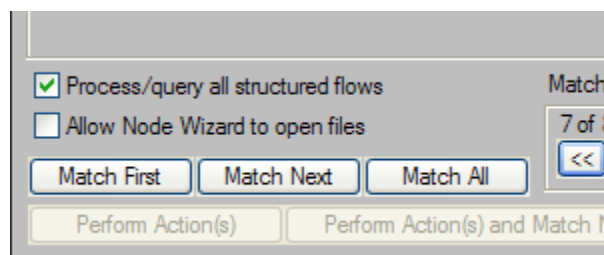
15. In the **XPath** box, enter the following expression:

```
//*[not(self::BulletItem) and not(self::Body)]
```

This XPath means literally, “Match all elements that are not BulletItem or Body elements.

16. Click **Match First** and **Match Next**, noting the results.

17. In the Node Wizard, check the **Process/query all structured flows** checkbox.



18. In the **XPath** box, enter the following expression:

//Body

19. Click **Match First** and **Match Next**, noting the results.

With this option checked, FrameSLT will query all structured flows, as it finds them. There is unconnected, unnamed flow at the end of the file with some `Body` elements that will be matched. You may also note that the match history will shuffle between flows as applicable.

20. In the Node Wizard, uncheck the **Process/query all structured flows** checkbox.

**IMPORTANT:** You must disable the setting at this point, otherwise the remainder of the tutorial will not work correctly.

## Part 3 - Performing element actions

In this part, you will use the Node Wizard to perform some sample element actions. These procedures are for demonstration purposes only and do not necessarily reflect any prudent end use of the software.

1. In the sample file, `Node_Wizard_Tutorial_Sample.fm`, note the `Strong` elements, indicated by bold text. Currently, they are children of `Body` elements.

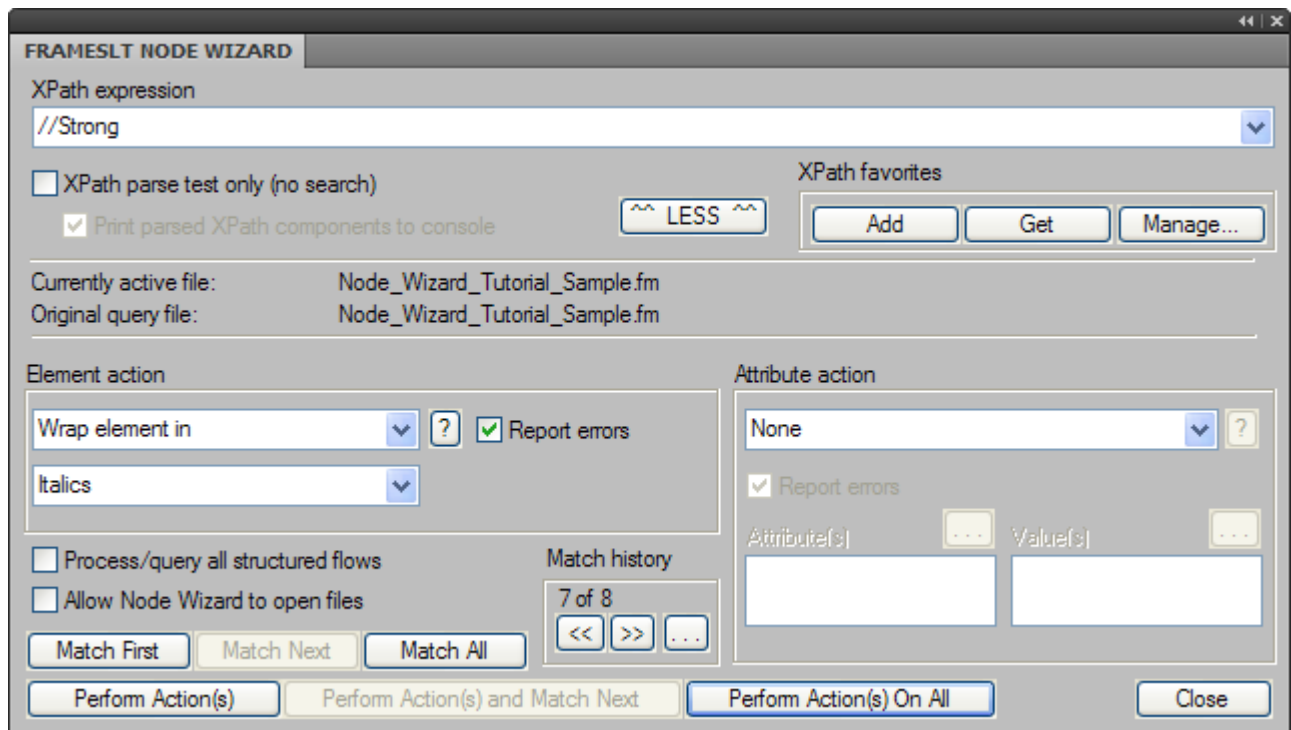
### Wrap all `Strong` elements in `Italics` elements

2. In the **XPath** box of the Node Wizard, enter the following expression:

//Strong

This XPath means literally, "Match all `Strong` elements in the document."

3. Under **Element action**, select **Wrap element in**.
4. In the box to the right, select **Italics**.



5. Click **Perform Actions On All**, then **Yes** to confirm.

Note the results. All `Strong` elements are now wrapped in `Italics` elements, as indicated by the newly italicized text.

## Unwrap the new *Italics* elements

6. In the **XPath** box, enter the following expression:

```
//Italics
```

7. Under **Element action**, select **Unwrap**.
8. Click **Perform Actions On All**, then **Yes** to confirm.

Note the results. All the new *Italics* elements are gone, and the document is back the way it was.

## Apply a condition tag to all *Comment* elements by “Jazmyn Campbell.”

9. In the **XPath** box, enter the following expression:

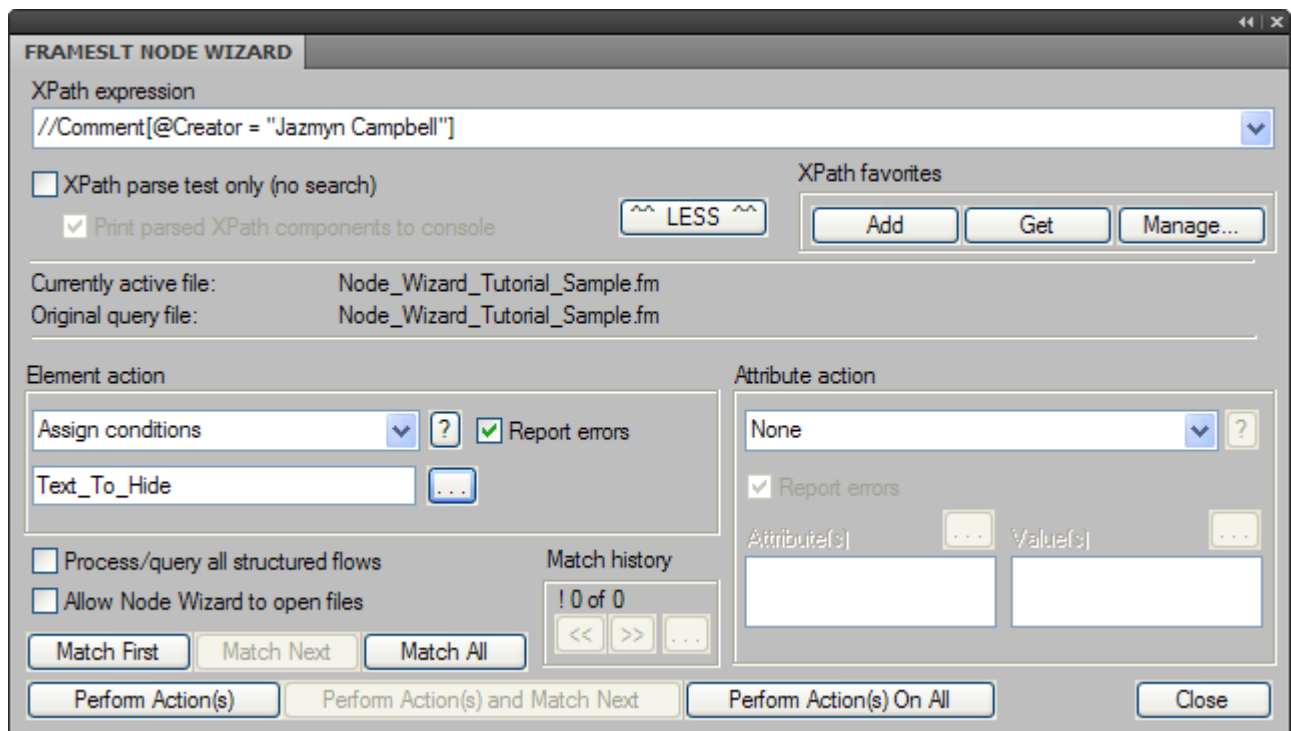
```
//Comment[@Author = "Jazmyn Campbell"]
```

This XPath means literally, “Find all *Comment* elements whose *Author* attribute is set to ‘Jazmyn Campbell’.”

10. Under **Element action**, select **Assign conditions**.
11. To the right, click the “...” button.
12. In the **Conditions To Assign** dialog, click **Add**.
13. In the **New Item** prompt, select **Text\_To\_Hide**.

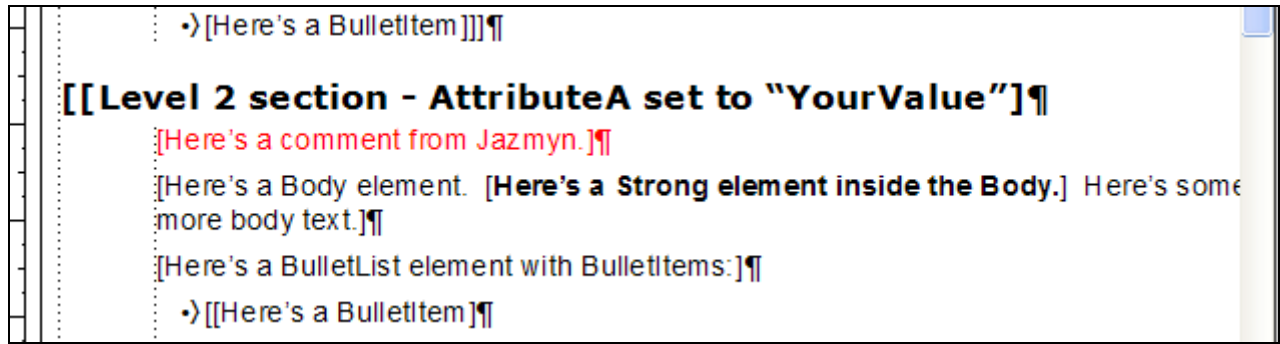
“Text\_To\_Hide” is a simple condition tag that currently exists in the document. FrameSLT automatically scanned the file and populated the prompt dropdown list with any condition tags it found.

14. Click **OK**, then **OK** to return to the main Node Wizard.



15. Click **Perform Actions On All**, then **Yes** to confirm.

Note the results. All the applicable *Comment* elements are now colored red, according to the settings of the applied condition tag. If you wanted to hide all of Jazmyn’s comments now, you could simply hide the “Text\_To\_Hide” condition tag.



## Remove any assigned condition tags from the whole document

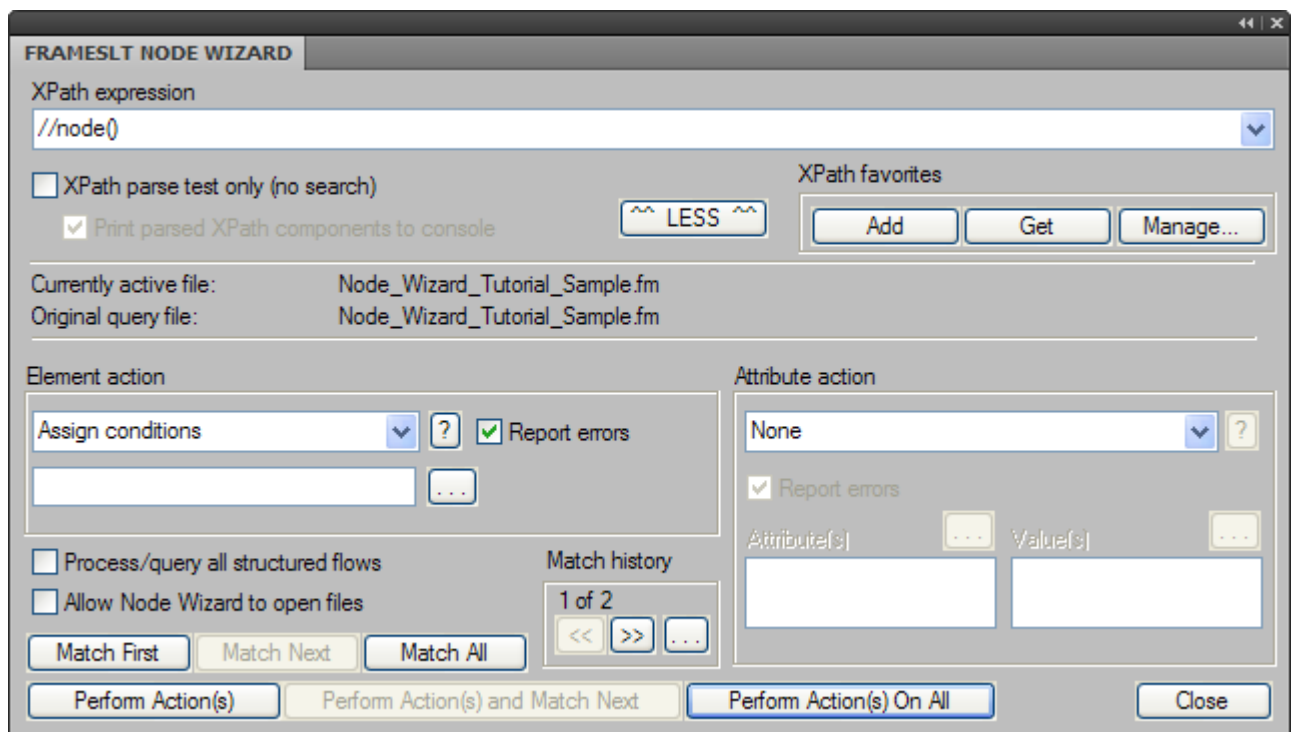
16. In the **XPath** box, enter the following expression:

```
//node()
```

This XPath means literally, "Match all elements in the document."

17. In the **Element action**, click the "... " button again and use the tools to remove the **Text\_To\_Hide** conditions list.

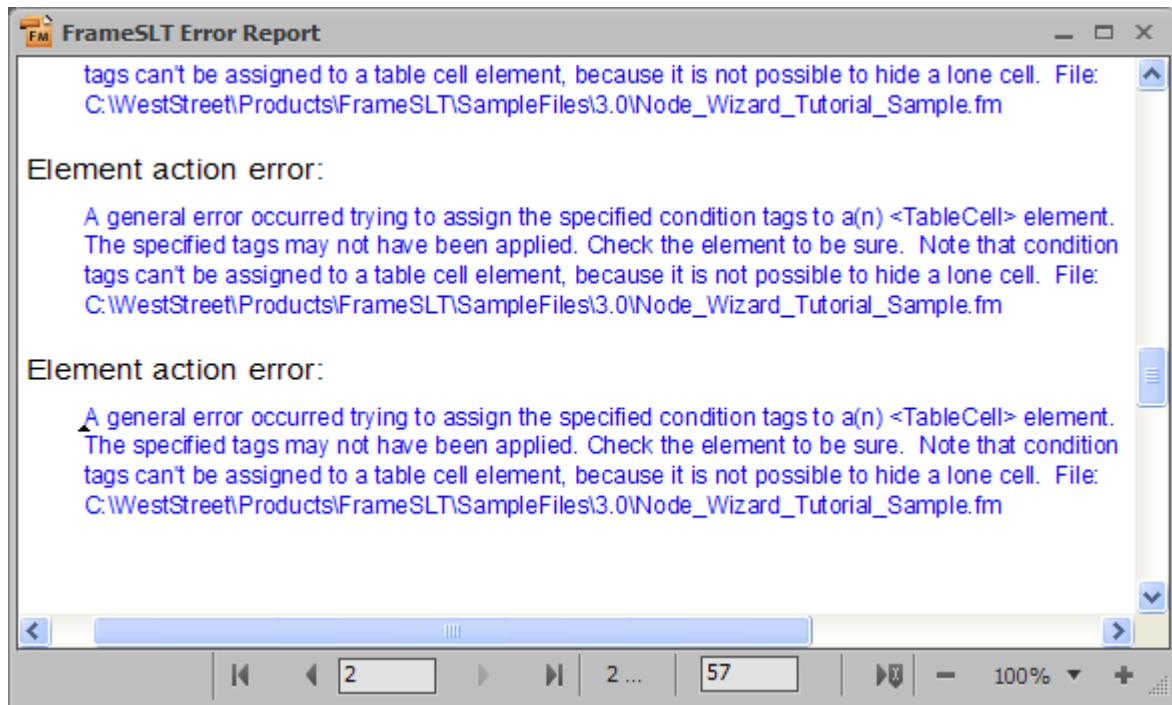
The list should now be empty.



18. Click **Perform Actions On All**, then **Yes** to confirm.

Note the results. All conditional tag assignment is removed from the document, because the Node Wizard stopped at every element and applied "no tags," which is equivalent to removing any existing tags.

In addition, the FrameSLT error report should have appeared. A hyperlinked warning was printed each time that FrameSLT attempted to apply (or unapply, in this case) condition tags to the table cell elements in the document. Condition tags cannot be assigned to individual table cells in FrameMaker. However, all other elements, including Jazmyn's comments, should now be unconditional. You can close the error report.

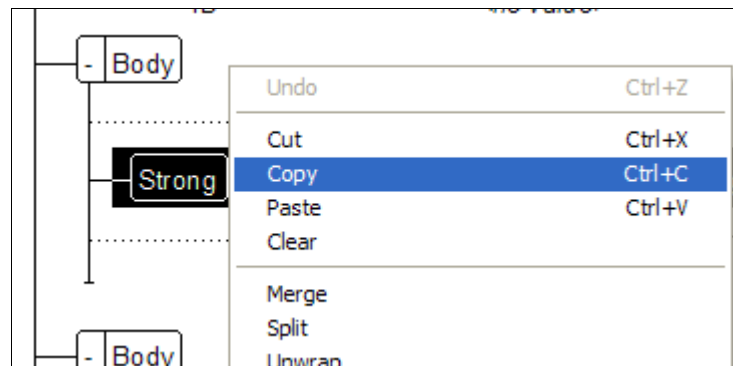


## Perform a clipboard-related element action, and then undo it

19. In the **XPath** box, enter the following expression:

`//Strong`

20. In the document, completely select any `Strong` element and select **Edit > Copy** to put the element on the clipboard.



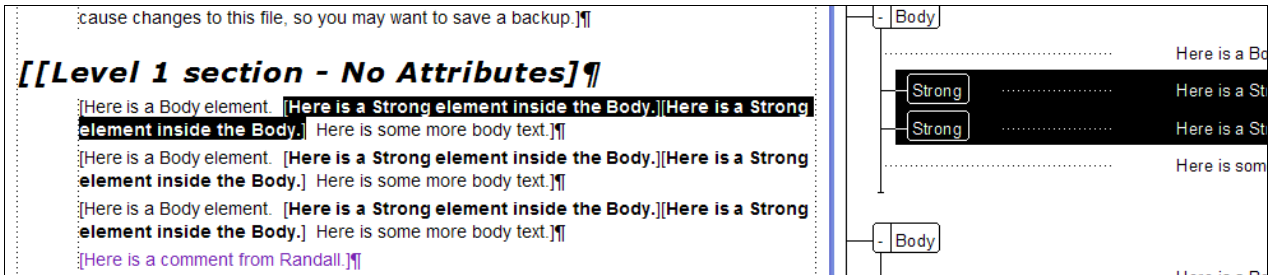
21. Under **Element action**, select **Paste CB after**.

With this action, for each match of the XPath (`Strong` elements), the contents of the clipboard (another `Strong` element) will be pasted on the branch directly after the matched element.

22. Click **Perform Actions On All**, then **Yes** to confirm.

Note that every `Strong` element in the document now has an identical sibling.





23. In the **XPath** box, enter the following expression:

```
//Strong[preceding-sibling::Strong]
```

This XPath means literally, “Match all `Strong` elements in the document that have a preceding-sibling element that is a `Strong`.”

24. Under **Element action**, select **Delete element** in the conditions list.

This action will delete all elements matched by the XPath.

25. Click **Perform Actions On All**, then **Yes** to confirm.

Note that the document is returned to its original state.

## Part 4 - Performing an attribute action

The following parts of the tutorial will demonstrate different Node Wizard features associated with attribute actions. Attribute management with FrameSLT is extremely flexible and allows a substantial variety of management tools, and therefore can become confusing. This tutorial will give you a very small sampling of FrameSLT’s capabilities. Please refer to your *User Guide* for more information on the full scope of features available.

### A simple attribute action on a “specified” attribute

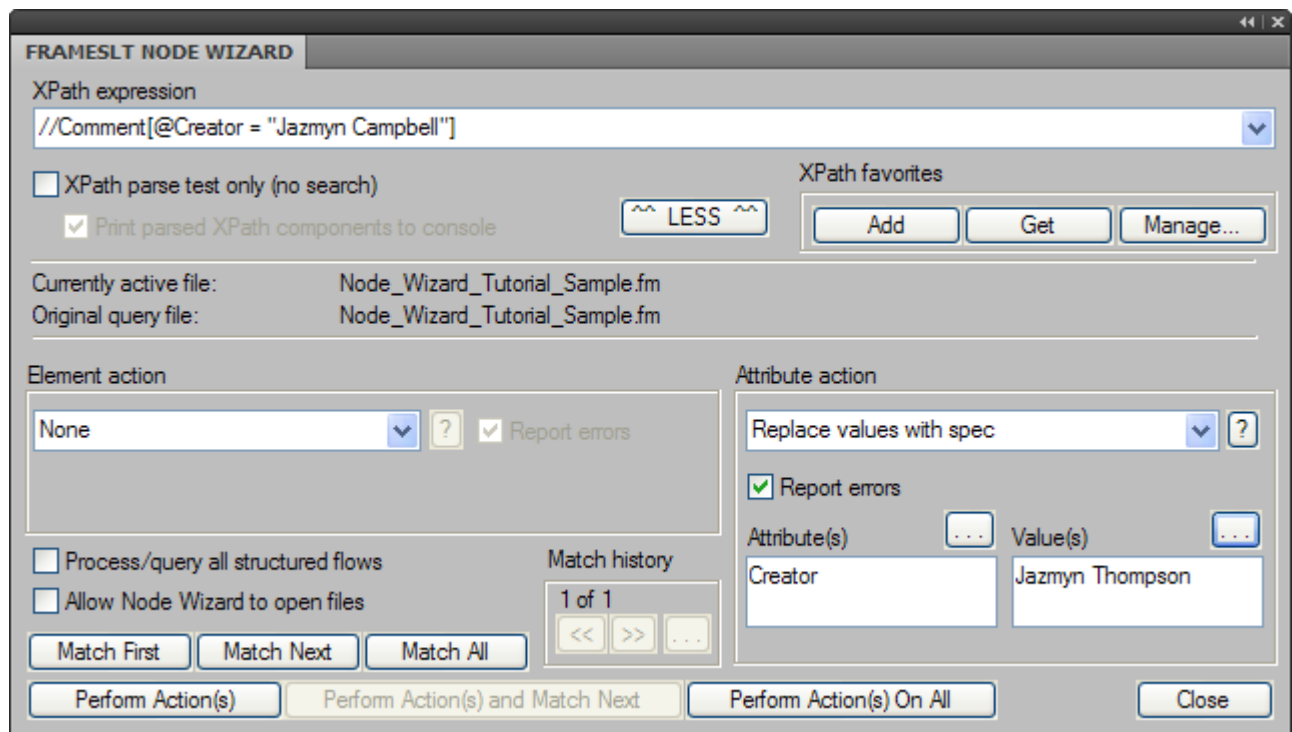
Assume that Jazmyn Campbell got married to Randall Thompson, and has changed her last name. As such, she wants to use the Node Wizard to change the names assigned to her comments throughout the document. Recall that the document contains `Comment` elements inserted by her, with her original name populated in the `Author` attribute.

1. In the **XPath** box, enter the following expression:

```
//Comment[@Author = "Jazmyn Campbell"]
```

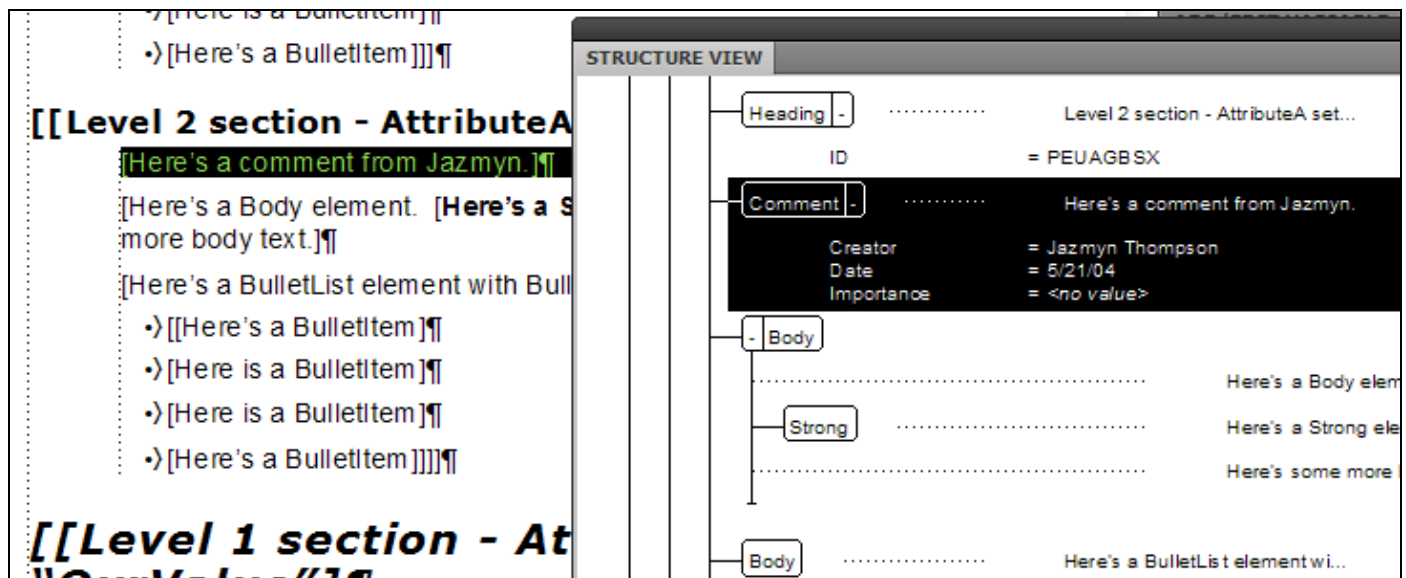
Again, FrameSLT will match all `Comment` elements whose `Author` attribute is set to ‘Jazmyn Campbell’.

- Under **Element action**, select **None**.
- Under **Attribute action**, select **Replace values with spec.**  
For each XPath match, this action will replace attribute values with the ones you specify. In the next steps, you will specify these values, and the particular attribute on which to set them.
- Above the **Attribute(s)** box, click the “...” button.
- In the **Action Attributes** dialog box, click the **Add** button.
- In the **New Item** prompt, enter or select **Author** and click **OK**, then **OK** again to return to the main Node Wizard.
- Above the **Value(s)** box, click the “...” button.
- In the **Action Values** dialog box, click the **Add** button.
- In the **New Item** prompt, enter **Jazmyn Thompson** and click **OK**, then **OK** again to return to the main Node Wizard.



10. Click **Perform Actions On All**, then **Yes** to confirm.

Note the results. The Node Wizard found all of Jazmyn's `Comment` elements, located the `Author` attribute, and replaced any existing values with "Jazmyn Thompson."



### An attribute action on a "matched" attribute

After the marriage, assume that Jazmyn has gotten tired of everyone misspelling her first name, and has decided to change it to Jane. Reluctantly, Randall agrees. After the change, Jane again wants to update the `Author` attributes of her `Comment` elements. Being the type who likes variety, she decides to use the Node Wizard in a different manner this time.

1. In the **XPath** box, enter the following expression:

```
//Comment/@Author
```

This XPath will match all `Author` attributes on `Comment` elements. This expression is different than all others so far, in that it directly matches an attribute node, rather than an element node.

2. Under **Element action**, select **None**.

3. Under **Attribute action**, select **Search and replace string**.

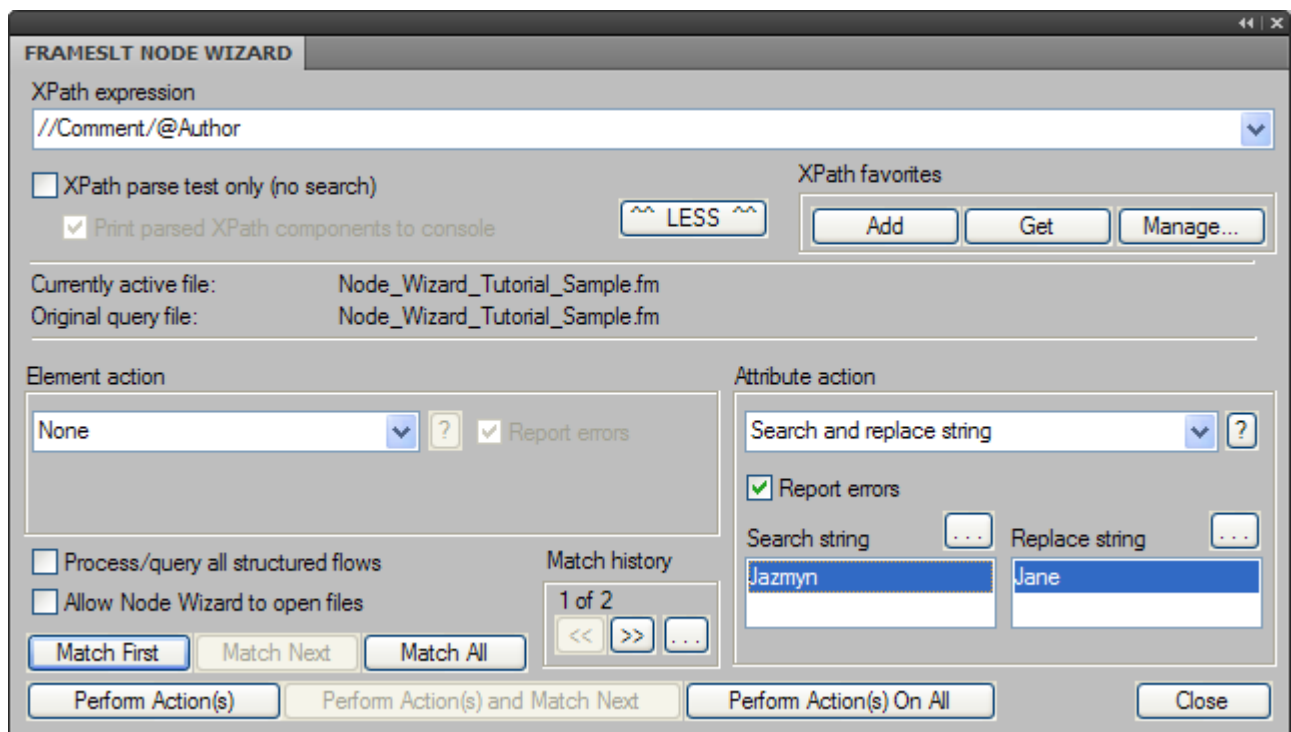
This action allows FrameSLT to search for a string fragment within an attribute value and replace it with another string. It always acts upon the attribute(s) matched by the XPath expression.

4. Under **Search string**, delete any values currently in the box and add the string “Jazmyn”.

Tip: You can double-click a value in the box to edit it.

5. Under **Replace string**, delete any values currently in the box and add the string “Jane”.

The Node Wizard is now set up to replace any instances of the string “Jazmyn” with the string “Jane”, on any attributes matched by the XPath. You have not had to specify an attribute name, because this particular action works on the matched attribute only.



6. Click **Perform Actions On All**, then **Yes** to confirm.

Note that Jazmyn’s `Comment` elements now reflect Jane Campbell.

## Another attribute action on a “matched” attribute

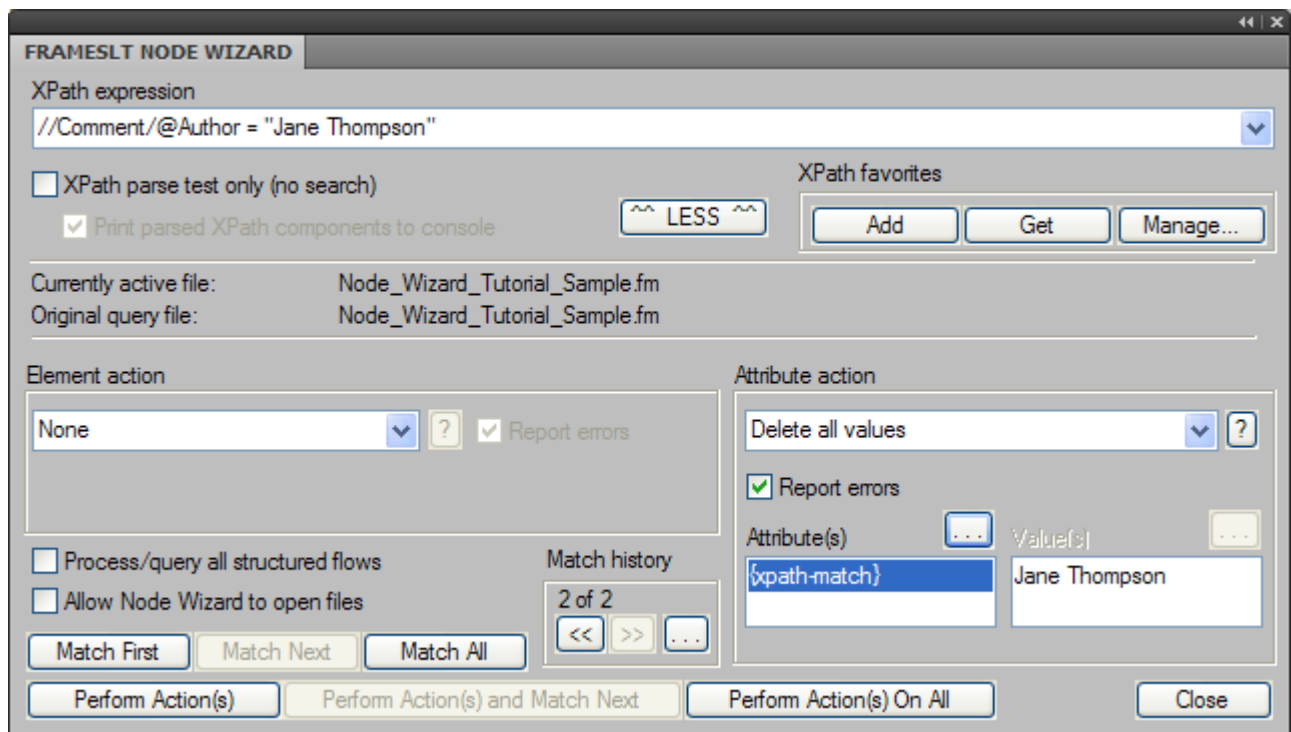
Now after all that, assume that the marriage didn’t work out because Randall couldn’t stand doing all the housework while Jane sat around watching TV. Following the split, Jane caused so much trouble in the workplace that she was eventually fired. Randall now wants to erase any memory of her, including her name in the `Comment` element attributes of their FrameMaker files. He decides to use the Node Wizard with an attribute action that can work on specified and/or matched attributes, opting ultimately to have it act on matched attributes.

1. In the **XPath** box, enter the following expression:

```
//Comment/@Author = "Jane Thompson"
```

This is the same expression from earlier, except now we are looking for `Author` attributes that specifically contain the text ‘Jane Thompson’.

2. Under **Element action**, select **None**.
3. Under **Attribute action**, select **Delete all values**.  
This action will delete all values from the specified and/or matched attribute(s), depending on what is specified below.
4. Under **Attribute(s)**, click “. . .” and delete any values in the list, then add **{xpath-match}** (the first item in the **New Item** prompt dropdown list).  
This value is a flag that tells FrameSLT to perform the action on the matched attribute, rather than any specified attribute. In an earlier part of this tutorial, you explicitly specified the value “Author” in this box, which directed FrameSLT to look specifically for an attribute named `Author` on the element matched by the XPath. “{xpath-match}”, on the other hand, does not indicate any specific attribute by name, rather it indicates whatever attribute may be matched by the XPath. Naturally, you must have an expression that matches attribute nodes for this flag to have any effect.



5. Ignore anything specified under **Value(s)**. This parameter is not applicable to the “Delete all values” action.
6. Click **Perform Actions On All**, then **Yes** to confirm.  
Note that all Jazmyn’s comments now have no value for the `Author` attribute.

## Finishing up

Congratulations, you have finished the tutorial. Feel free to experiment with the sample file as needed to learn more. Remember that the document has now been altered, so to restore it to its original state, you should close it without saving changes and reopen it.

This tutorial has touched on only a small part of Node Wizard functionality. For more information on the full scope of features supported, please see your *User Guide*.