XMLSmartHelper ver. 1.1.0

The HowTo Guide

By Nicola Martella CEO Nick4Name di Nicola Martella & F. s.a.s

Summary

Copyright

Email Author: n4ngit@gmail.com

Download: http://www.nick4name.eu/xmlsmarthelper/

XMLSmartHelper: Why?

XMLSmartHelper is born because I find very boring the implemented mode from XML frameworks to reach a particular element into the hierarchy. In the better ipotesys you can use XPath if you have practice with its syntax. Other chance, you can iterate all DOM's elements, or one of its hierarchy sub-section, until you find that you search.

My approch to the problem is more direct. Why doesn't manage an XML like a db table? What is missing to an XML file to be managed like a db? Simple. The index.

XMLSmartHelper: Indexed XML

When you create a new element by **XMLSmartHelper** framework it attributes to it a unique id assigned to the *elID* attribute

```
<MYROOT elID="6d0840cb-9788-49aa-94b1-ce45b7d56121" />
```

Each new element has to refer an existing *elID* index and for that element it will be a child

```
<MYROOT elID="6d0840cb-9788-49aa-94b1-ce45b7d56121">
  <LEVEL1 elID="9b0a4c5e-3bff-4168-b9d0-b3e488534a61">
    <LEVEL2 elID="93b6d314-27bc-4ef7-9b64-aca282ce9a3c" />
  </LEVEL1>
</MYROOT>
```

Of course, an element can be enriched by one or more attributes

```
<MYROOT elID="6d0840cb-9788-49aa-94b1-ce45b7d56121">
  <LEVEL1 elID="9b0a4c5e-3bff-4168-b9d0-b3e488534a61" attr1="val1">
    <LEVEL2 elID="93b6d314-27bc-4ef7-9b64-aca282ce9a3c" attr1="val1" attr2="val2" />
  </LEVEL1>
</MYROOT>
```

The final hierarchy could be look like this

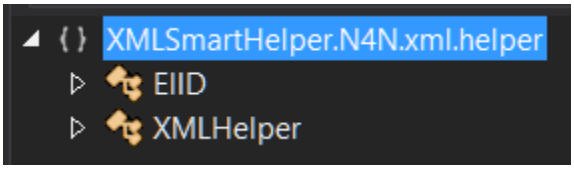
```
<MYROOT elID="6d0840cb-9788-49aa-94b1-ce45b7d56121">
  <LEVEL1 elID="9b0a4c5e-3bff-4168-b9d0-b3e488534a61" attr1="val1">
    <LEVEL2 elID="93b6d314-27bc-4ef7-9b64-aca282ce9a3c" attr1="val1" attr2="val2" />
  </LEVEL1>
  <LEVEL1A elID="ef0d3c3c-7851-47d3-8f5b-b0a0718b9830">
    <LEVEL2A elID="c96b5cd3-877d-4642-9482-35304abcdc32">
      <LEVEL3A elID="c81702b4-fa7c-48ce-b8f1-c91d16793291" />
      <LEVEL3B elID="f8f706d3-5c74-46d2-8654-5c5c82458c33">
        <LEVEL4A elID="e4bbe8dd-e9d9-47ca-bd98-999b78b65e64" attr1="val1" />
      </LEVEL3B>
      <LEVEL3C elID="448730de-b97f-42a2-afb8-6991352bc0d6" />
    </LEVEL2A>
  </LEVEL1A>
</MYROOT>
```

Just a little bit less clear but really comfortable to manage. Discover why!

The Framework

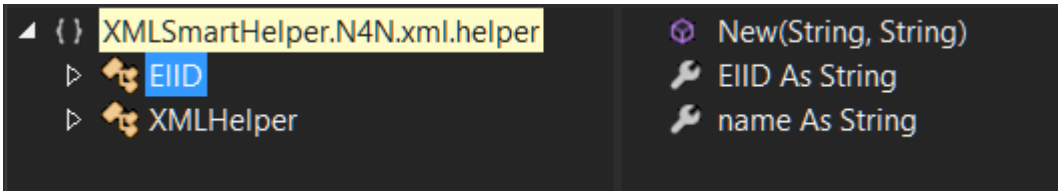
The **XMLSmartHelper** library is developed in VB.NET and is based on LINQ framework.

The object model is very simple



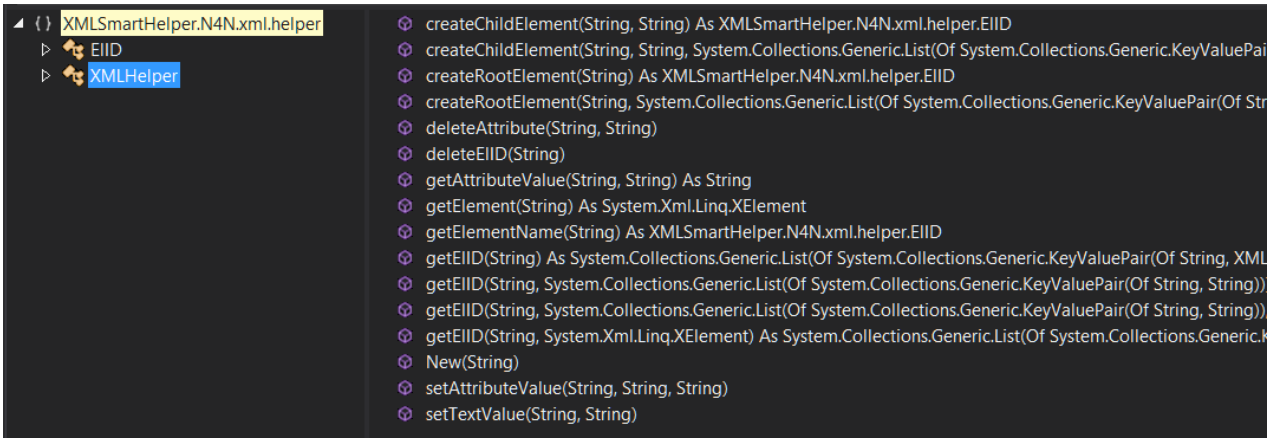
Finished!

The **EIID** class represents the XML element and its index



`ElID As String`	Unique ID for the element
`name As String`	Element name

The **XMLHelper** class contains the methods to interact with the XML



Constructors	
`New(String filename)`	Creates the XML instance for/from the existing/new file
Creation methods	
`ElID createRootElement(String name)`	Returns the *ElID* instance for a new element named *name*
`ElID createRootElement(String name, List(Of KeyValuePair(Of String, String)) attribs)`	Returns the *ElID* instance for a new element named *name* enriched with attibutes listed in the *KeyValuePair* list. Each *KeyValuePair* list element is composed by *attribute name* and *attribute value*.

`ElID createChildElement(String elIDparent, String name)`	Returns the *ElID* instance for a new element named *name*, child of the element with ElID *elIDparent*
`ElID createChildElement(String elIDparent, String name, List(Of KeyValuePair(Of String, String)) attribs)`	Returns the *ElID* instance for a new element named *name*, child of the element with ElID *elIDparent*. The element is enriched with attibutes listed in the *KeyValuePair* list. Each *KeyValuePair* list element is composed by *attribute name* and *attribute value*.
Update methods	
`setTextValue(String elID, String value)`	Sets/changes the text value for the element with ElID *elID*
`setAttributeValue(String elID, String value, String attrib)`	Sets/changes the value *value* for the attribute named *attrib* referred to the element with ElID *elID*
Delete methods	
`deleteElID(String elID)`	Deletes the element with ElID *elID*
`deleteAttribute(String elID, String attrib)`	Deletes the attribute named *attrib* related to the element with ElID *elID*
Retrive methods	
`XElement getElement(String elID)`	Returns the *System.Xml.Linq.XElement* related to the element with ElID *elID*
`ElID getElementName(String elID)`	Returns the *ElID* instance for the element with ElID *elID*
`String getAttributeValue(String elID, String attrib)`	Returns the value of the attribute named *attrib* related to the element with ElID *elID*
`List(Of KeyValuePair(Of String, ElID)) getElID(String name)`	Returns the *List* of *ElID* instances of elements named *name*. The *KeyValuePair* instance is represented by the *ElID* instance found and the related key is his Elld string value.
`List(Of KeyValuePair(Of String, ElID)) getElID(String name, XElement source)`	Returns the *List* of *ElID* instances of elements named *name* belonging to the *System.Xml.Linq.XElement* element with ElID *elID*. The *KeyValuePair* instance is represented by the *ElID* instance and the related key is his ElID string value. Use this powerfull method to dissect research to a XML section instead of the entire hierarchy if the section is known.
`List(Of KeyValuePair(Of String, ElID)) getElID(String name, List(Of KeyValuePair(Of String, String)) attribs)`	Returns the *List* of *ElID* instances of elements named *name* and having attributes referred in the *KeyValuePair List* valued as the value of the kvp value. The *KeyValuePair* of the returned *List* is represented by the *ElID* instance and the related key is his ElID string value.
`List(Of KeyValuePair(Of String, ElID)) getElID(String name, List(Of KeyValuePair(Of String, String)) attribs, XElement source)`	Returns the *List* of *ElID* instances of elements named *name* and having attributes referred in the *KeyValuePair List* valued as the value of the kvp value. The hierarchy section evaluated is referred by the *System.Xml.Linq.XElement source*. The *KeyValuePair* of the returned *List* is represented by the *ElID* instance and the related key is his ElID string value.

Each method *create*, set** or *delete** itself commit into the file.

XML like a db

You could argue "*Why like a DB. Perhaps you want to say Table?*" No. I want to say database an soon you'll understand why.

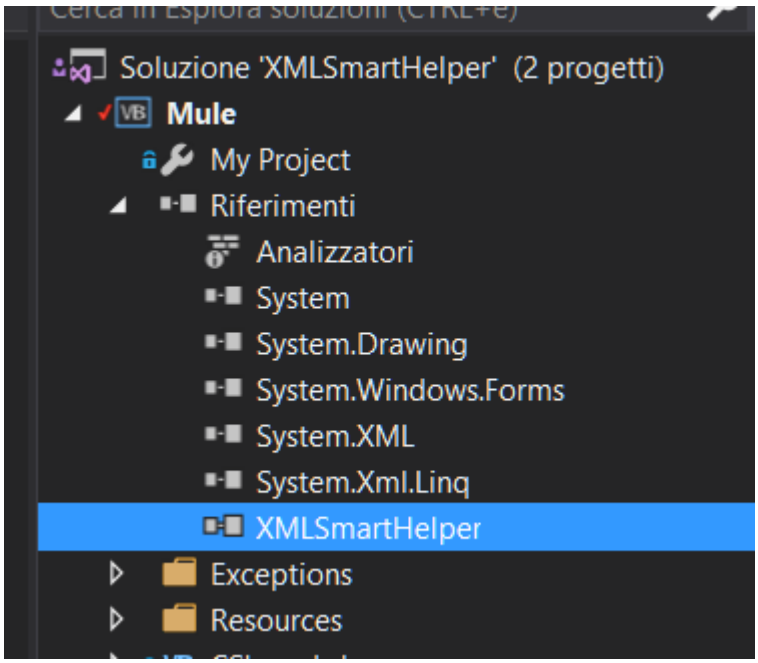
With **XMLSmartHelper** framework exposed below is quite simple realize programmatically something as a hierarchical database like this

```
<SMARTDB elID="18faeaae-6e89-4388-90d7-b13087bd7fae">
  <PERSON elID="1ff3a5d4-d3a9-4451-bbf1-2d04993717e5" surname="Martella" name="Nicola">
    <DOCUMENT elID="aa5c268f-16c8-47f2-8e47-d1543f737356" type="Passport">123456</DOCUMENT>
    <DOCUMENT elID="dd3bed3c-9421-4589-b8e6-e7337bd2f9fc" type="Driver's">789456</DOCUMENT>
  </PERSON>
  <PERSON elID="34539f92-9035-4896-a100-90f6f7c46d5a" surname="Bianchi" name="Carlo">
    <DOCUMENT elID="8bd97fdb-26c6-4b68-bb76-f4719d30e1bf" type="Driver's">456123</DOCUMENT>
  </PERSON>
</SMARTDB>
```

After showed, further, the *XMLSmartHelper Mule* we can leave "quite" from the previous sentence.

The project's components

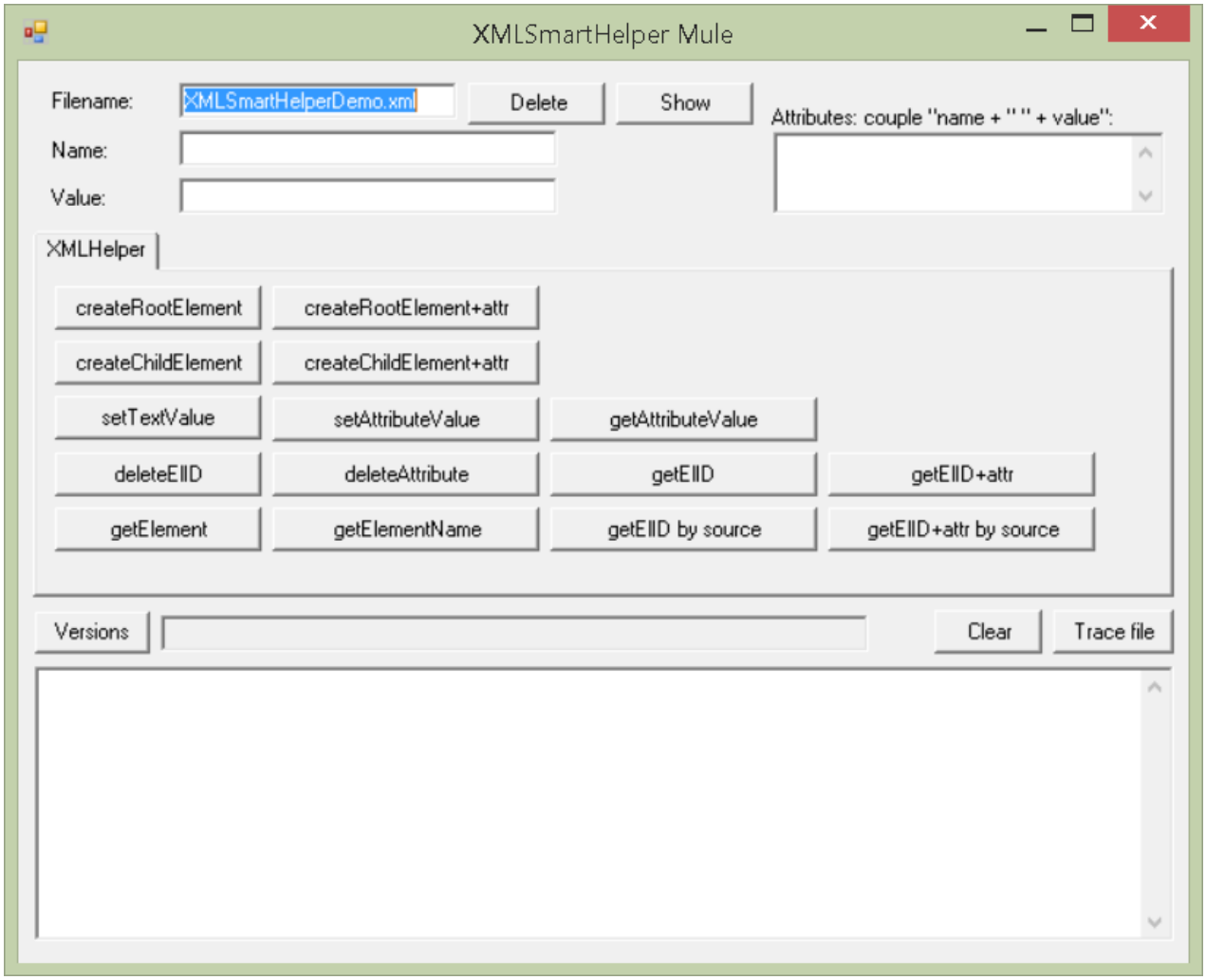
It's really simple. To power your application with **XMLSmartHelper** framework it's enough to add the reference to *XMLSmartHelper.dll* to your project



The Mule

The *XMLSmartHelper Mule* is the framework tester in which every method is fired by a button that uses some text field to take the value for parameters. The methods return is shown in the textbox down. In this way under each button you can see the correct way to use the method.

To know in which textbox set the value parameter for a method, press the button without values and left to drive from the interface.



A bit Training

The following is a little training for the *Mule* usage in which we create an XML DB called *XMLSmartHelperDemo.xml*.

DB initialization

- Write **XMLSmartHelperDemo.xml** in *Filename* textbox.
- Press *Delete* to sure the deletion of previous file.

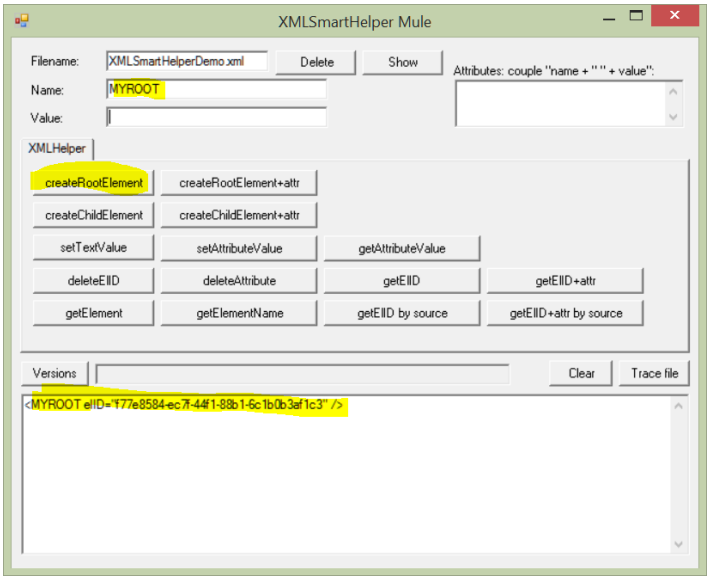
Result

The file will be create in *XMLSmartHelper\Mule\bin* path.

Creation methods

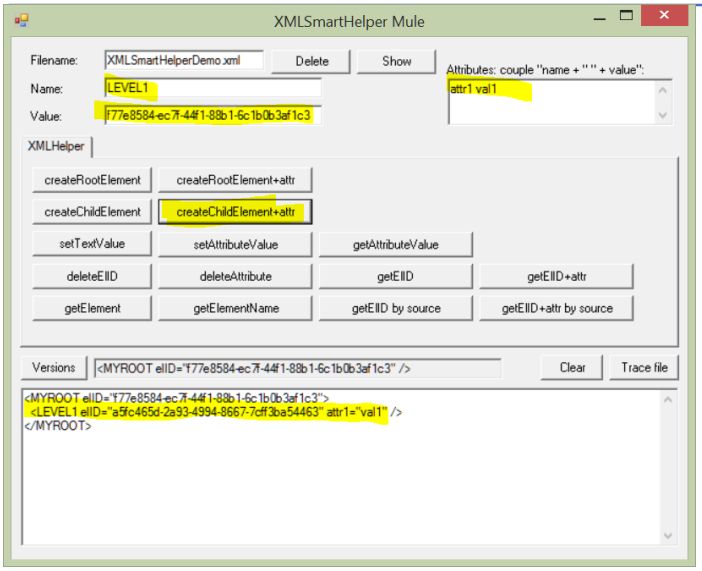
Creation root element

Write **MYROOT** in *Name* textbox and press *createRootElement*.



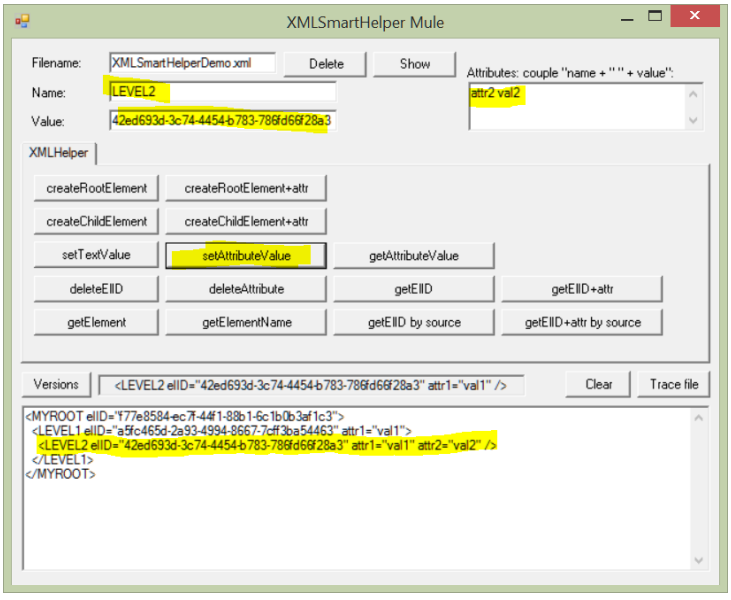
Creation child's root with attribute

- Write **LEVEL1** in *Name* textbox;
- Copy&paste the *MYROOT elID* value in *Value* textbox;
- Write **attr1 val1** in *Attributes* (pay attention to the space);
- Press *createChildElement+attr*.



Creation nodes with more attributes

- Write **LEVEL2** in *Name* textbox;
- Copy&paste the *LEVEL1 elID* value in *Value* textbox;
- Write **attr1 val1** in *Attributes* (pay attention to the space);
- Press *createChildElement+attr*;
- Write **attr2 val2** in *Attributes* (Mule doesn't support a list of attributes ☹ but *createChildElement()* yes ☺);
- Copy&paste the *LEVEL2 elID* value in *Value* textbox;
- Press *setAttributeValue*.



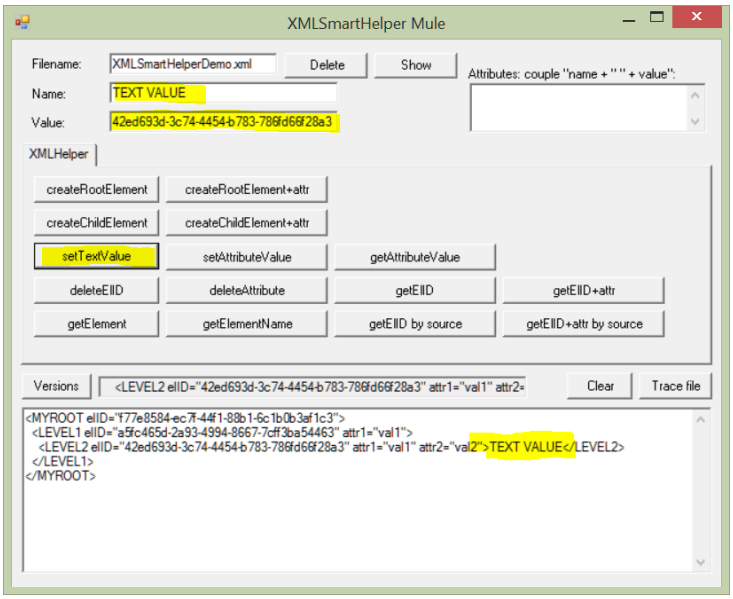
Choosing the appropriate node you can quickly create a hierarchy with *brothers* node in every level you want. For example, this:

```
- <MYROOT elID="6d0840cb-9788-49aa-94b1-ce45b7d56121">
  - <LEVEL1 elID="9b0a4c5e-3bff-4168-b9d0-b3e488534a61" attr1="val1">
      <LEVEL2 elID="93b6d314-27bc-4ef7-9b64-aca282ce9a3c" attr1="val1" attr2="val2"/>
    </LEVEL1>
  - <LEVEL1A elID="ef0d3c3c-7851-47d3-8f5b-b0a0718b9830">
    - <LEVEL2A elID="c96b5cd3-877d-4642-9482-35304abcdc32">
        <LEVEL3A elID="c81702b4-fa7c-48ce-b8f1-c91d16793291"/>
      - <LEVEL3B elID="f8f706d3-5c74-46d2-8654-5c5c82458c33">
          <LEVEL4A elID="e4bbe8dd-e9d9-47ca-bd98-999b78b65e64" attr1="val1"/>
        </LEVEL3B>
        <LEVEL3C elID="448730de-b97f-42a2-afb8-6991352bc0d6"/>
      </LEVEL2A>
    </LEVEL1A>
</MYROOT>
```

Update methods

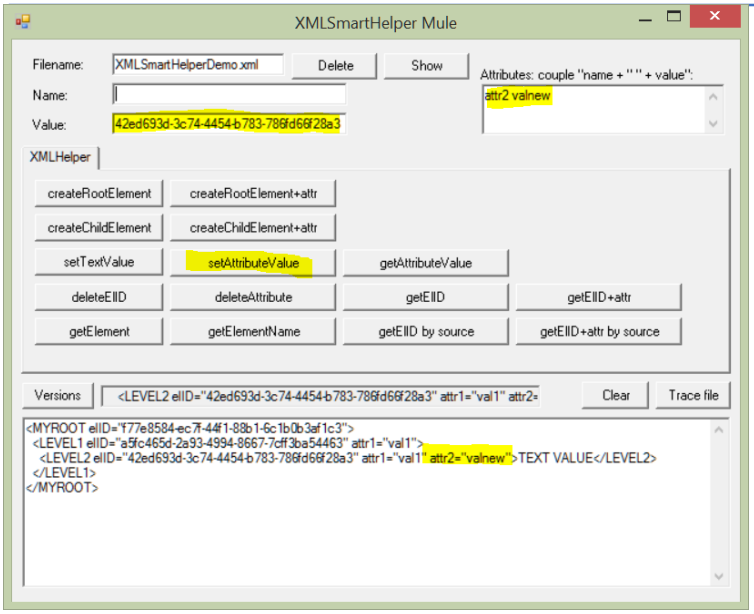
Set a text value for a node

- Write **TEXT VALUE** in *Name* textbox;
- Copy&paste the *LEVEL2 elID* value in *Value* textbox;
- Press *setTextValue*.



Change the attribute value

- Copy&paste the *LEVEL2 elID* value in *Value* textbox;
- Write **attr2 valnew** in *Attributes*;
- Press *setAttributeValue*.



	In the same way you can delete nodes and attributes. It's enough to set the correct elID.

	If you delete a parent node you destroy all his children.

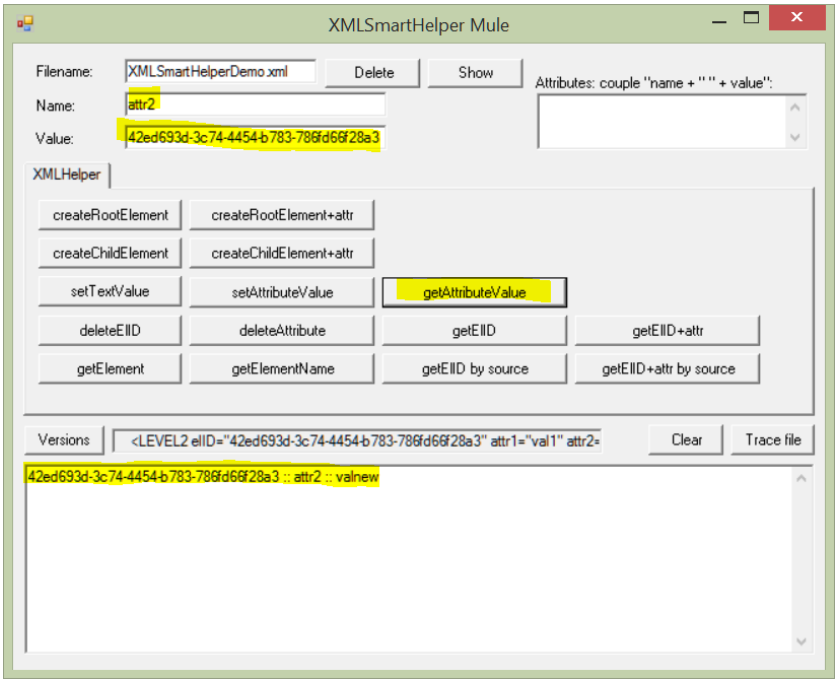
Retrieve methods

To show the *retrieve methods*, please, copy in *XMLSmartHelperDemo.xml* file by your text editor, the follow XML and press *Show* button on the *Mule*.

```
<MYROOT elID="6d0840cb-9788-49aa-94b1-ce45b7d56121">
  <LEVEL1 elID="9b0a4c5e-3bff-4168-b9d0-b3e488534a61" attr1="val1">
    <LEVEL2 elID="93b6d314-27bc-4ef7-9b64-aca282ce9a3c" attr1="val1" attr2="val2" />
    <LEVEL2 elID="7aee2183-ec33-45e7-87ac-9d8cb13c9cdd" attr1="val1a" />
  </LEVEL1>
  <LEVEL1A elID="ef0d3c3c-7851-47d3-8f5b-b0a0718b9830">
    <LEVEL2A elID="c96b5cd3-877d-4642-9482-35304abcdc32">
      <LEVEL3A elID="c81702b4-fa7c-48ce-b8f1-c91d16793291" />
      <LEVEL3B elID="f8f706d3-5c74-46d2-8654-5c5c82458c33">
        <LEVEL4A elID="e4bbe8dd-e9d9-47ca-bd98-999b78b65e64" attr1="val1" />
        <LEVEL2 elID="3e3dcda5-d8a2-4ea8-acae-898ed84926be" attr1="val1a" />
      </LEVEL3B>
      <LEVEL3C elID="448730de-b97f-42a2-afb8-6991352bc0d6" />
    </LEVEL2A>
  </LEVEL1A>
</MYROOT>
```

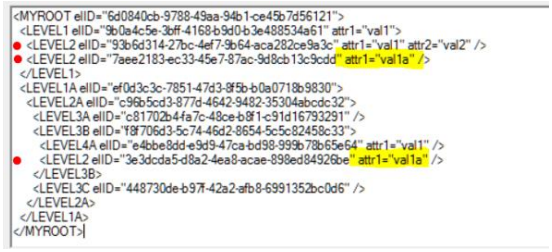
Return the attribute value

- Copy&paste the *LEVEL2 elID* value in *Value* textbox;
- Write **attr2** in *Name*;
- Press *getAttributeValue*.



Return a list of nodes for attribute value

- Write the node name to search, **LEVEL2**, in *Name*;
- Write the attribute name and value to search, **attr2 val1a** in *Attributes*;
- Press *getElID+attr*

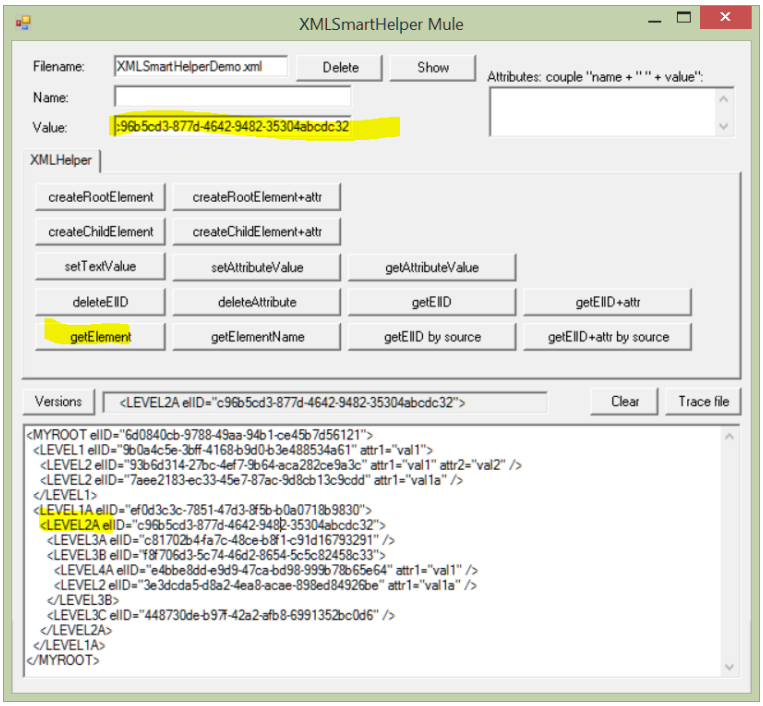


Result

```
LEVEL2 :: 7aee2183-ec33-45e7-87ac-9d8cb13c9cdd
LEVEL2 :: 3e3dcda5-d8a2-4ea8-acae-898ed84926be
```

Get the XElement section of the XML

- Copy&paste the *LEVEL2A elID* value in *Value* textbox;
- Press *getElement*



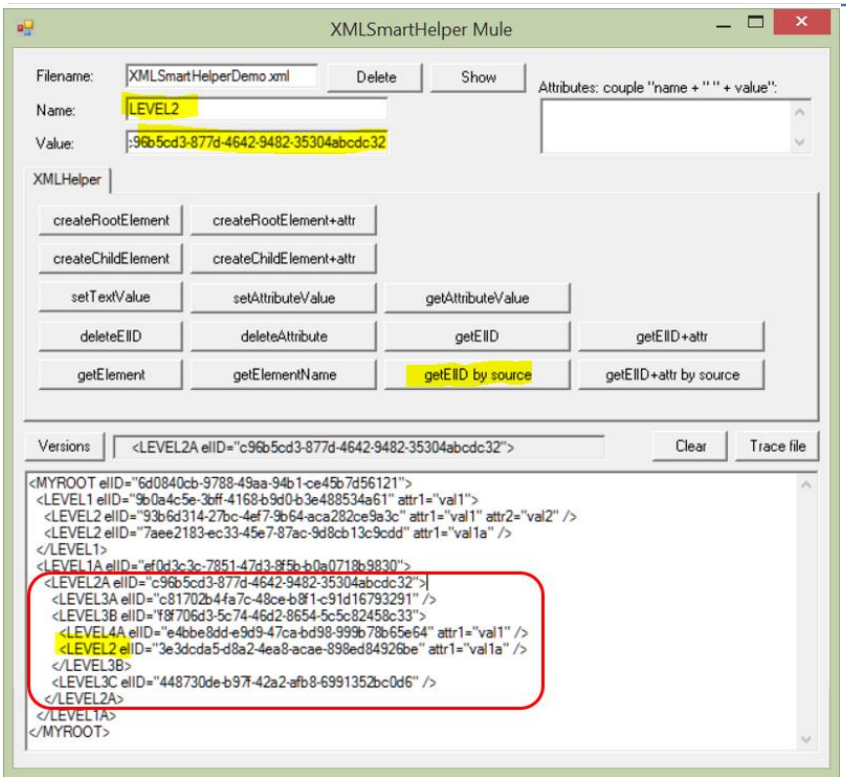
Result

```
c96b5cd3-877d-4642-9482-35304abcdc32 ::
<LEVEL2A elID="c96b5cd3-877d-4642-9482-35304abcdc32">
  <LEVEL3A elID="c81702b4-fa7c-48ce-b8f1-c91d16793291" />
  <LEVEL3B elID="f8f706d3-5c74-46d2-8654-5c5c82458c33">
    <LEVEL4A elID="e4bbe8dd-e9d9-47ca-bd98-999b78b65e64" attr1="val1" />
    <LEVEL2 elID="3e3dcda5-d8a2-4ea8-acae-898ed84926be" attr1="val1a" />
  </LEVEL3B>
  <LEVEL3C elID="448730de-b97f-42a2-afb8-6991352bc0d6" />
</LEVEL2A>
```

Return a node list starting search by a certain level

- Write the node name to search, **LEVEL2**, in *Name*;
- Copy&paste the *LEVEL2A elID* value in *Value* textbox;
- Write the attribute name and value to search, **attr2 val1a** in *Attributes*;
- Press *getElID by source*

You can enrich the search with attribute(s) value and press *getElID+attr by source*



Result

```
LEVEL2 :: 3e3dcda5-d8a2-4ea8-acae-898ed84926be
```

That's all

I hope this work helps you in your next XML develop.

Enjoy with **XMLSmartHelper** framework!