

Tutorial 2. How to modify visual elements using embedded script language

Contributed by Michael
Tuesday, 18 August 2009
Last Updated Monday, 24 August 2009

FreeSCADA has embedded language based on Python syntax which allows you to customize your scheme in various ways. In this example I will show how to handle events from visual elements and modify visual state of these objects.

FreeSCADA has embedded language based on Python syntax which allows you to customize your scheme in various ways. In this example I will show how to handle events from visual elements and modify visual state of these objects.

First, we need to create new schema and put Rectangle element on it.

Using scripting, I'd like to change color of this rectangle to blue when mouse cursor in this rectangle and to red when it outside of this area. So, we will need 3 events:

- MouseEnter – occurs when mouse pointer enters to this element.
- MouseLeave – occurs when mouse pointer is moved outside of the element.
- Loaded – This event occurs at very beginning when schema just loaded into memory. In our case it can be used to handle initialization stage.

Now let's create handlers for these events. Go to Property Browser window and click "Event" button.

You will see full list of event. Find "MouseEnter" row and type your handler name in editable field. At the end hit Enter to create this handler.

By pressing Enter Designer will create a placeholder for this event and open Script editor window. But let's switch back to schema and create the rest two handlers: MouseLeave and Loaded.

Finally you get see the following script text with 3 placeholders for our handlers.

The last step is to create some logic for them. I will set color property for the rectangle using "Colors" class from WPF class library. Therefore I need to add reference to WPF in the script (please refer to IronPython documentation for details).

```
import
clr
clr.AddReferenceByPartialName("PresentationCore")
from
System.Windows.Media import *
```

This code snippet says that we import core python library "clr" and add reference to "PresentationCore" .Net library. Since I don't want to write full name for "Colors" class (like System.Windows.Media.Colors) I imported its namespace into script namespace.

Now we can enrich visual element handler by some code:

```
#
Arguments: Object sender, MouseEventArgs e
def
OnMouseEnter(sender, e):

sender.Fill.Color = Colors.LightSkyBlue

# Arguments:
Object sender, MouseEventArgs e
def
OnMouseLeave(sender, e):

sender.Fill.Color = Colors.Red

# Arguments:
Object sender, RoutedEventArgs e
def
OnLoaded(sender, e):

sender.Fill.Color = Colors.Red
```

Looks pretty simple, isn't it? Now click "check syntax" button just to make sure that everything is correct with our code.

Finally the script should look like on the picture below.

Now you can save this project and open it in Runtime module.