

PlotAction

November 11, 2016

The goal is to add an action to the plot window For example : - shift the selected curve - invert x and y - ...

1 create your own plot action

- heritate from PlotAction
- redefine the triggered function
- http://www.silx.org/doc/silx/dev/modules/gui/plot/plotactions_examples.html

```
In [1]: %gui qt
```

```
In [2]: from silx.gui.plot.PlotActions import PlotAction
class ShiftUpAction(PlotAction):
    """QAction shifting up a curve by one unit

    :param plot: :class:`.PlotWidget` instance on which to operate
    :param parent: See :class:`QAction`
    """
    def __init__(self, plot, parent=None):
        PlotAction.__init__(self,
                             plot,
                             icon='shape-circle',
                             text='Shift up',
                             tooltip='Shift active curve up by one unit',
                             triggered=self.shiftActiveCurveUp,
                             parent=parent)

    def shiftActiveCurveUp(self):
        """Get the active curve, add 1 to all y values, use this new y
        array to replace the original curve"""
        # By inheriting from PlotAction, we get access to attribute self.plot
        # which is a reference to the PlotWindow
        activeCurve = self.plot.getActiveCurve()

        if activeCurve is not None:
            # Unpack curve data.
```

```

# Each curve is represented by a tuple of 5 variables:
# - x and y are the array of abscissa and ordinate values
# - legend is a unique text identifying a curve
# - info and params are dictionaries of additional data
#   (user defined, curve style and color...)
x0, y0, legend, info, _params = activeCurve

# Add 1 to all values in the y array
# and assign the result to a new array y1
# (do not modify y0 if you want to preserve the original curve)
y1 = y0 + 1.0

# Re-using the same legend causes the original curve
# to be replaced
self.plot.addCurve(x0, y1, legend=legend,
                   info=info)

```

2 Integrate the action in a plot window

2.1 define the plot window

```

In [3]: from silx.gui.plot import Plot1D
import numpy
def createPlot():
    plot=Plot1D()
    x=numpy.linspace(0, numpy.pi, 1000)
    y=numpy.sin(x)
    plot.addCurve(x, y, legend='curve')
    return plot

```

2.2 add it to the toolBar

```

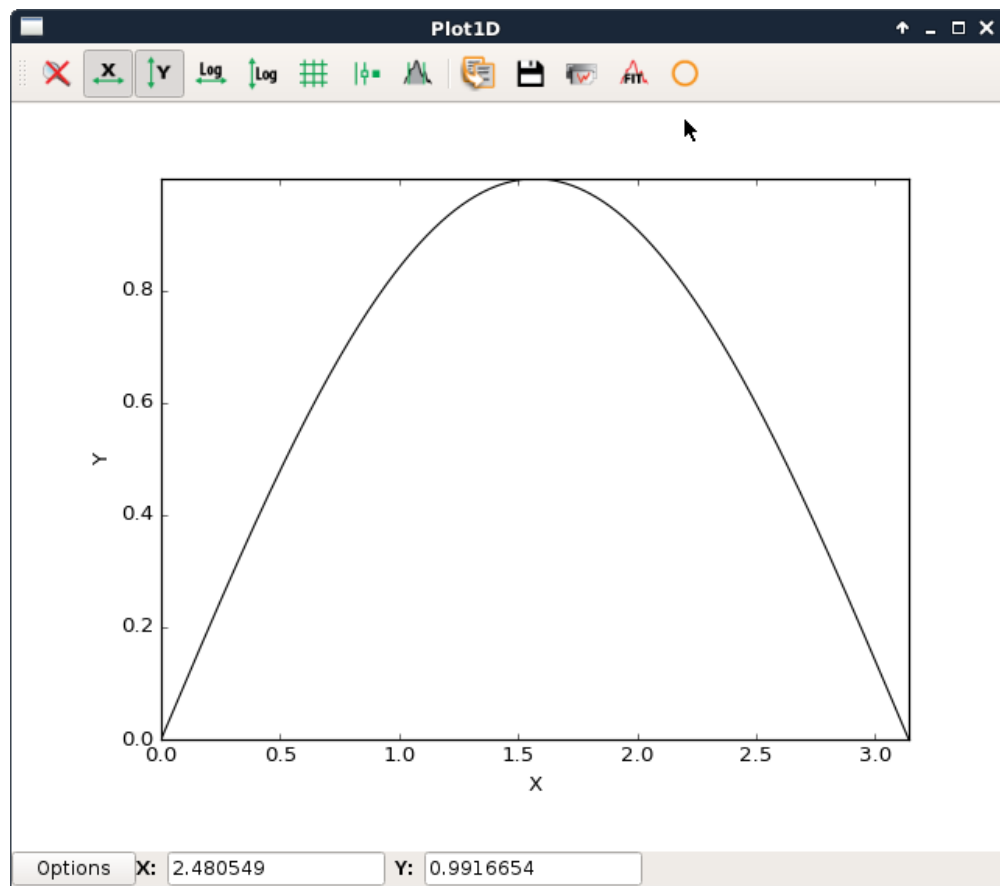
In [ ]: plot=createPlot()
        toolBar=plot.toolBar()
        myaction = ShiftUpAction(plot)
        toolBar.addAction(myaction)
        plot.show()

```

```

In [ ]:

```



plot action integrated into a toolBar