LibreOffice from COLLABORA

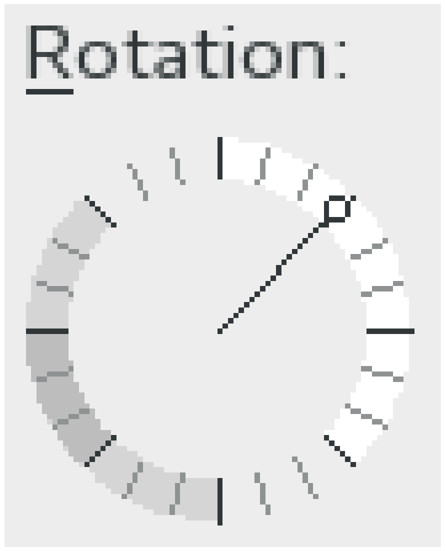
How to Create a Custom Widget?

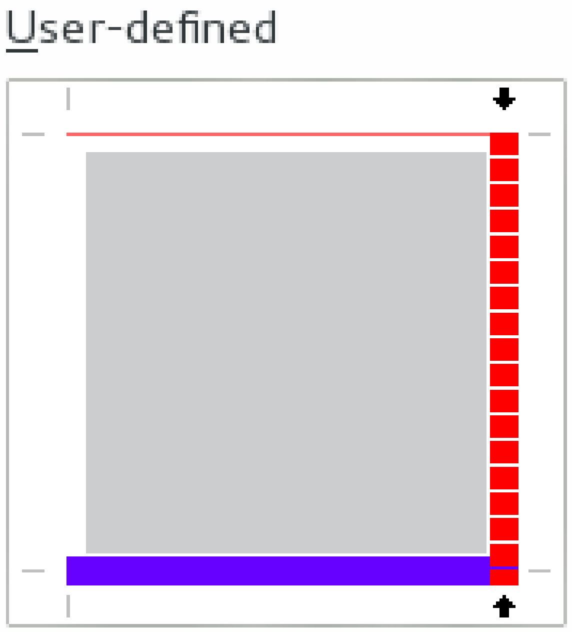
Jan Holesovsky <kendy@collabora.com>

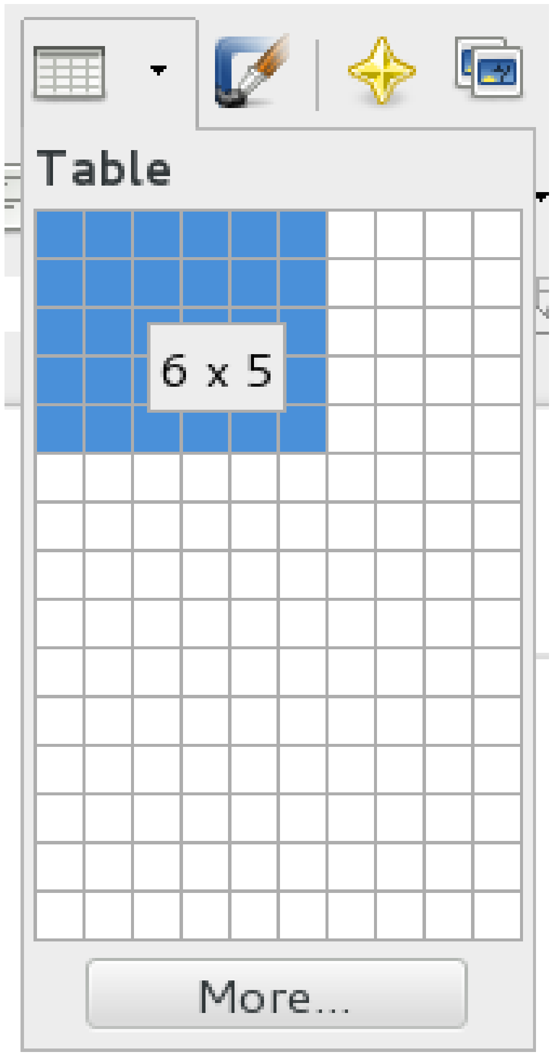
kendy, #libreoffice-dev, irc.freenode.net

LibreOffice

What is a Custom Widget







Creating Custom Widgets

- Don't do that in the first place! :-)
  - Always try to use what is already existing
  - Standard widgets set – ideally what is available via glade for the .ui creation
- But unfortunately sometimes it makes sense
  - More convenient than the stock ones
  - Special functionality – like the Start Center document previews

Ways to Create One

- Subclass an existing widget + specialize
- Create from scratch
  - And draw the content using the VCL methods
  - Or draw the content using DrawingLayer

Subclassing an Existing Widget

- Eg. SelectionListBox – in Writer, just to change the behavior slightly
  - class SelectionListBox : public ListBox
- You just take the existing class, and change the virtual methods
  - Different behavior on click
  - Different drawing
  - Etc.

Creating from Scratch

- Instead of subclassing an existing widget, you subclass directly the VCL's class 'Control'
- Then you have to provide all the functionality
  - Drawing it
  - Behavior when mouse is over / clicked etc.

Drawing

- Use DrawingLayer (or direct VCL calls) to draw it
  - DrawingLayer has the advantage that it provides also antialiasing; though a bit more complex to write
  - Cf. my yesterday's presentation :-)
- virtual void Paint(const Rectangle&) SAL_OVERRIDE;

DrawingLayer Way of Drawing

svx/source/xoutdev/xtabhtch.cxx:121

```
const basegfx::BColor aBlack(0.0, 0.0, 0.0);
const drawinglayer::primitive2d::Primitive2DReference aHatchPrimitive(
    new drawinglayer::primitive2d::PolyPolygonHatchPrimitive2D(
        basegfx::B2DPolyPolygon(aRectangle),
        aBlack,
        aFillHatch));

const drawinglayer::primitive2d::Primitive2DReference aBlackRectanglePrimitive(
    new drawinglayer::primitive2d::PolygonHairlinePrimitive2D(
        aRectangle,
        aBlack));

// prepare VirtualDevice
VirtualDevice aVirtualDevice;
// ... some aVirtualDevice.SetDrawMode()'s etc. ...

// create processor and draw primitives
const drawinglayer::geometry::ViewInformation2D aNewViewInformation2D;
boost::scoped_ptr<drawinglayer::processor2d::BaseProcessor2D> pProcessor2D(
        drawinglayer::processor2d::createPixelProcessor2DFromOutputDevice(
            aVirtualDevice,
            aNewViewInformation2D));

if(pProcessor2D)
{
    drawinglayer::primitive2d::Primitive2DSequence aSequence(2);

    aSequence[0] = aHatchPrimitive;
    aSequence[1] = aBlackRectanglePrimitive;
    pProcessor2D->process(aSequence);
    pProcessor2D.reset();
}

// get result bitmap and scale
aRetval = aVirtualDevice.GetBitmap(Point(0, 0), aVirtualDevice.GetOutputSizePixel());
```

Creation of the Hatch Primitive (to add to a kind of display list, to render later).

Creation of the Hairline Primitive (rectangle)

Processor to render the “display list” later.

The “display list”.

The rendering itself.

Mouse Behavior

- virtual void MouseButtonDown(const MouseEvent& rMEvt) SAL_OVERRIDE;
- virtual void MouseButtonUp(const MouseEvent& rMEvt) SAL_OVERRIDE;
- virtual void MouseMove(const MouseEvent& rMEvt) SAL_OVERRIDE;
- If you need to update parts of the widget after the mouse action, use Invalidate()
  - Ideally with specifying the area to invalidate, to avoid blinking / redrawing just everything

Keyboard Behavior, Accessibility

- virtual void KeyInput(const KeyEvent& rKEvt) SAL_OVERRIDE;
  - Usually you want to implement at least behavior of the arrows, Tab, Enter

Layout Related Functions

- virtual void Resize() SAL_OVERRIDE;
- virtual Size GetOptimalSize() const SAL_OVERRIDE;

How to Make Use of It

- Add it to extras/source/glade/libreoffice-catalog.xml.in
  - So that glade sees it / allows you to work with it
  - Derive it from the closest widget
- Edit the dialog's .ui in glade, and place the widget
- Implement make...() method
  - Like makeSelectionListBox()

Thank You for Your Attention!