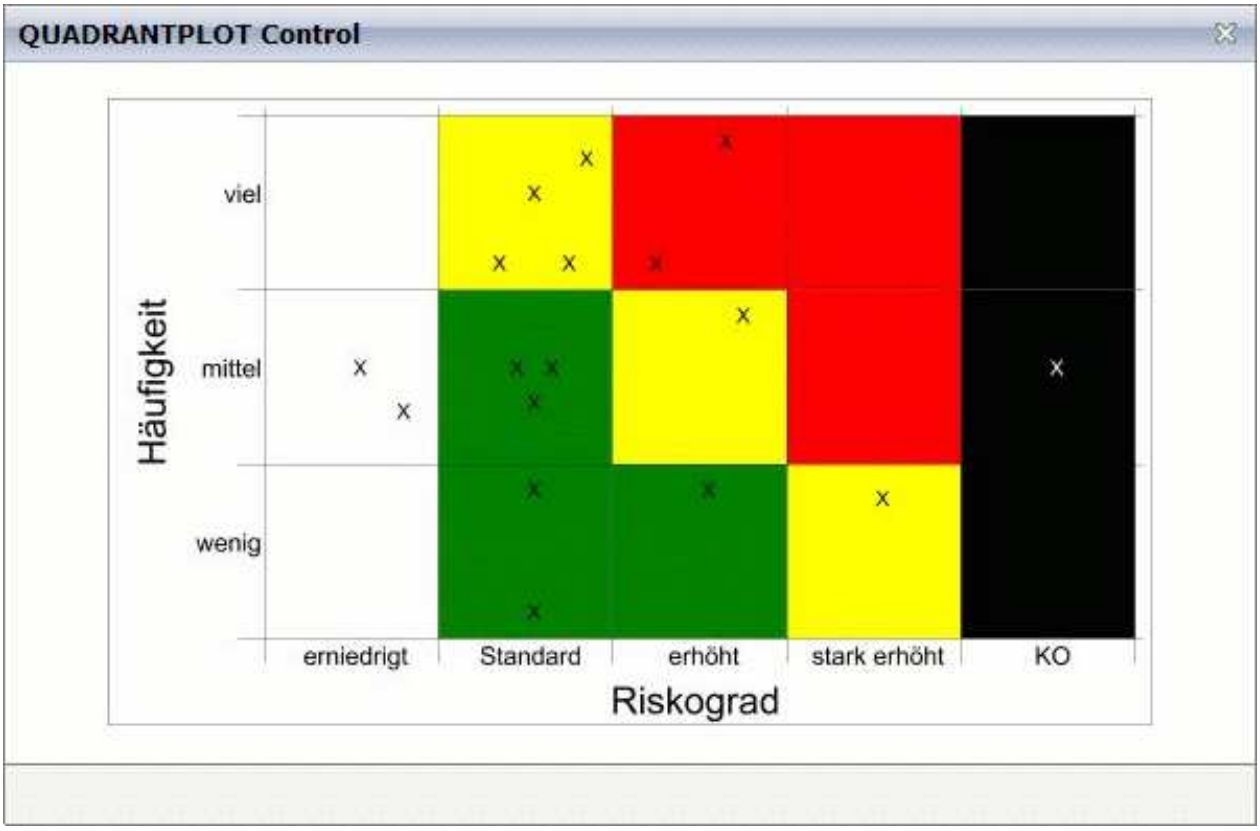
Using the Special Chart Control QUADRANTPLOT

The QUADRANTPLOT control represents an extension to the normal SVG chart. It allows you to divide the displayed chart into quadrants and to react on clicks.

This chapter covers the following topics:

- Simple Example
- Properties

Simple Example



The XML layout definition is:

```
<pagebody takefullheight="true">
  <itr takefullwidth="true" valign="middle">
  <quadrantplot quadrantplotprop="drawArea">
  </quadrantplot>
  </itr>
</pagebody>
```

The Java code of the adapter is:

```
public class QuadrantPlotAdapter
  extends Adapter
  implements IQUADRANTPLOTListener
  // Interface for the click event of the QUADRANTPLOTInfo.

{
  QUADRANTPLOTInfo m_drawArea = new QUADRANTPLOTInfo(this);
  public QUADRANTPLOTInfo getDrawArea() { return m_drawArea; }
  public void setDrawArea(QUADRANTPLOTInfo value) { m_drawArea = value; }

  /** Interface IQUADRANTPLOTListener
   *  Method is call after a click in a quadrant with an ID.<br>
   *  The ID is passed as argument. */
  public void reactOnClick(String id)
  {
    outputMessage (MT_SUCCESS,"Quadrant "+ id + " was clicked!");
  }

  public void initData()
  {
  // set x axis quadrants description
  String[] xAxisValues = new String[]
  {
  "decreased", "standard", "increased", "highly increased", "none"
  };

  // set y axis quadrants description
  String[] yAxisValues = new String[]
  {
  "low", "mid", "high"
  };

  m_drawArea.clearDrawAreaData();
  m_drawArea.rotateXAxisScaleText(-0,0);
  m_drawArea.setLeftYAxisDistance(400);
  m_drawArea.printFrameAround(true);

  // set x axis description
  m_drawArea.setXAxisDescription("Risk","black", 18, 250);
  m_drawArea.setYAxisDescription("Occurrences","black", 18, 250);

  m_drawArea.setXAxis(xAxisValues);
  m_drawArea.setYAxis(yAxisValues);

  // added quadrants
  m_drawArea.addQuadrantInfo(0, 0, "white", "area_0_0");
  m_drawArea.addQuadrantInfo(0, 1, "white", "area_0_1");
  m_drawArea.addQuadrantInfo(0, 2, "white", "area_0_2");

  m_drawArea.addQuadrantInfo(1, 0, "green", "area_1_0");
  m_drawArea.addQuadrantInfo(1, 1, "green", "area_1_1");
  m_drawArea.addQuadrantInfo(1, 2, "yellow", "area_1_2");

  m_drawArea.addQuadrantInfo(2, 0, "green", "area_2_0");
  m_drawArea.addQuadrantInfo(2, 1, "yellow", "area_2_1");
  m_drawArea.addQuadrantInfo(2, 2, "red", "area_2_2");

  m_drawArea.addQuadrantInfo(3, 0, "yellow", "area_3_0");
  m_drawArea.addQuadrantInfo(3, 1, "red", "area_3_1");
  m_drawArea.addQuadrantInfo(3, 2, "red", "area_3_2");
```

```
  m_drawArea.addQuadrantInfo(4, 0, "black", "area_4_0");
  m_drawArea.addQuadrantInfo(4, 1, "black", "area_4_1");
  m_drawArea.addQuadrantInfo(4, 2, "black", "area_4_2");

  // added values
  m_drawArea.addPlotData(0.5, 1.5, "X", 10, "black");
  m_drawArea.addPlotData(0.75, 1.25, "X", 10, "black");

  m_drawArea.addPlotData(1.5, 0.1, "X", 10, "black");
  m_drawArea.addPlotData(1.5, 0.8, "X", 10, "black");

  m_drawArea.addPlotData(1.5, 1.3, "X", 10, "black");
  m_drawArea.addPlotData(1.4, 1.5, "X", 10, "black");
  m_drawArea.addPlotData(1.6, 1.5, "X", 10, "black");

  m_drawArea.addPlotData(1.3, 2.1, "X", 10, "black");
  m_drawArea.addPlotData(1.7, 2.1, "X", 10, "black");
  m_drawArea.addPlotData(1.5, 2.5, "X", 10, "black");
  m_drawArea.addPlotData(1.8, 2.7, "X", 10, "black");

  m_drawArea.addPlotData(2.5, 0.8, "X", 10, "black");

  m_drawArea.addPlotData(2.7, 1.8, "X", 10, "black");

  m_drawArea.addPlotData(2.2, 2.1, "X", 10, "black");
  m_drawArea.addPlotData(2.6, 2.8, "X", 10, "black");

  m_drawArea.addPlotData(3.5, 0.75, "X", 10, "black");

  m_drawArea.addPlotData(4.5, 1.5, "X", 10, "white");

  // get the svg plot
  m_drawArea.setSVGOutputWidth(300);
  }

  /** initialisation - called when creating this instance*/
  public void init()
  {
    initData();
  }
}
```

Properties

Basic			
quadrantplotprop	Name of the adapter property that represents the control on server side. Return type must be "QUADRANTPLOTInfo". Pay attention: The QUADRANTPLOTInfo Constructor needs a valid Adapter i.e. QUADRANTPLOTInfo m_qpi = new QUADRANTPLOTInfo(this) ;	Obligatory	
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percantage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of "50%" then the parent element (e.g. an ITR row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%