User Interface - Feature #4199

implement a data visualization widget which can be used for a wide range of data-bound charts and animations

08/09/2019 05:17 PM - Greg Shah

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**vendor_id:**	GCD

Description

History

#1 - 08/09/2019 05:42 PM - Greg Shah

We have a customer application that uses the Infragistics UltraChart .NET control (and related .NET classes) to provide charting. As part of the conversion project we must replace this usage.

The customer's application used this control to provide the following features:

- chart types
  - column
  - column 3D
  - line
  - line 3D
  - area
  - area 3D
  - pie
  - pie 3D
- options
  - scale auto (not sure what this means)
  - always on top
- save to PDF
- print
- the control is linked to the data in a browse
- user can hover over a chart element which highlights and displays a tooltip with the value
- the charts have axis labels and a legend

I don't intend to provide something that only does this specific set of charts. Instead, the objective here is to provide a full fledged data visualization widget, which covers all the needed features.

The core idea is to use the JavaScript D3 framework (we already use this in the Analytics, see #3277-5 for some examples) inside an embedded web browser control. By using D3, we can enable the full range of data visualization including highly interactive animations.

From my perspective, I think there are key integration points that must be handled:

- data binding - D3 is all about binding the data model to the visualization implementation, as the data model changes the visualization can be easily refreshed to dynamically update
- event processing - we will need to implement a range of basic callbacks for common events, especially related to user interaction
- customizable/replaceable JS portion - we will implement the charting first, but it should be possible to do a completely custom JS implementation

I think the same integration techniques being delivered in #3910 and #3821 will be naturally used here.

The core idea is that this 4GL widget will bind 4GL data sources with a customizable JS data visualization implementation with full interactivity.

We will probably need to implement a conversion-time mapping from the Infragistics UltraChart .NET control into this widget, similar to how we handle OCX mapping today.

Let me know if there are any thoughts or concerns.

#2 - 03/02/2020 07:54 AM - Constantin Asofiei

For chart, in the end I'm working on adding support for calls like:

```
c:axis:x:labels:FontColor = 0x000000.
```

This will allow the chart to behave like a widget, while allowing access to its internal structure (for axis, labels, etc) in a Java style, like c.unwrapChart().x.labels.FontColor = 0x000000.

The chaining (for chart setup) is pretty deep and is easier to allow this instead of exploding everything with keywords.

#3 - 03/02/2020 07:57 AM - Constantin Asofiei

I'm placing everything related to chart in com.goldencode.p2j.ui.chart, except main widget class (ui.ChartWidget).

Every access is like this:

```
def var ultraChart as com.goldencode.p2j.ui.ChartWidget.
def var c as handle.
create chart c.
ultraChart = com.goldencode.p2j.util.handle:getResource(c). // get the underlying Java instance

// from now on, everything is Java-style
ultraChart:axis:x:labels:FontColor = integer(0x0000).
ultraChart:axis:x:labels:FontColor = i.
ultraChart:axis:x:labels:FontColor = 1.
```

This allows for the resource to act like an widget in terms of size, visibility, frame, etc; and any other chart-specific access will be made directly.

Greg, one concern: the Java field access is made directly in original 4GL code, as these are defined as properties. Maybe it makes sense to mark (at the Java field definition label) via an annotation, so that the access is emitted via Java getter/setter? This would complicate things a little.

The alternative would be instead of direct access to write the Java-style code in 4GL directly via getter/setter (and make the variables private at the Java level).

#4 - 03/02/2020 07:58 AM - Hynek Cihlar

Constantin Asofiei wrote:

> The chaining (for chart setup) is pretty deep and is easier to allow this instead of exploding everything with keywords.

I suppose this could be used as the basis for the direct OCX to Java conversion without the need to introduce method/attribute keywords.

#5 - 03/02/2020 08:08 AM - Constantin Asofiei

Hynek Cihlar wrote:

> Constantin Asofiei wrote:
>
> > The chaining (for chart setup) is pretty deep and is easier to allow this instead of exploding everything with keywords.
>
> I suppose this could be used as the basis for the direct OCX to Java conversion without the need to introduce method/attribute keywords.

Yes, I think so. But I'm not working on automating this via conversion now, I'm re-writing the code by hand.

#6 - 03/02/2020 09:59 AM - Greg Shah

> I'm placing everything related to chart in com.goldencode.p2j.ui.chart, except main widget class (ui.ChartWidget).

This is probably OK for now, but we are really trying to achieve a generic data visualization widget, not just charting.

> Greg, one concern: the Java field access is made directly in original 4GL code, as these are defined as properties. Maybe it makes sense to mark (at the Java field definition label) via an annotation, so that the access is emitted via Java getter/setter? This would complicate things a little.

I like the idea of using direct Java access, but I prefer to avoid the OO 4GL property approach. I want to minimize the unnecessary 4GL nonsense in these UI classes.

Or are you just suggesting pure-Java getters/setters?

#7 - 03/02/2020 10:41 AM - Constantin Asofiei

Greg Shah wrote:

> This is probably OK for now, but we are really trying to achieve a generic data visualization widget, not just charting.

The point here is I'm trying to avoid refactoring the logic behind the code - and just assume there are equivalents for (almost) every .NET field/method in the Java implementation. The other approach would be to find and define the (common) APIs behind this 'data visualisation' and map them to chart.

> Or are you just suggesting pure-Java getters/setters?

I was thinking to add a (unrelated) annotations to these fields, so that conversions knows to use pure Java-style getters/setters instead of the direct field access. For example, if you write:

```
ultraChart:axis:x:labels:FontColor = 1.
```

you would get:

```
ultraChart.getAxis().getX().getLabels().setFontColor(1)
```

instead of

```
ultraChart.getAxis().getX().getLabels().fontColor = 1.
```

Or maybe do this automatically if the field is private... OTOH, the .NET fields are camel-cased, and I don't want to do this in Java, too.

The idea behind this is to minimize the 'refactored' code in existing projects - I'd like to avoid duplicating this just to lowercase the first letter from the .NET fields/methods.

#8 - 03/02/2020 05:20 PM - Greg Shah

Constantin Asofiei wrote:

> Greg Shah wrote:
>
> > This is probably OK for now, but we are really trying to achieve a generic data visualization widget, not just charting.
>
> The point here is I'm trying to avoid refactoring the logic behind the code - and just assume there are equivalents for (almost) every .NET field/method in the Java implementation.

This is fine. We probably will want to implement a chart subclass of the data visualization control, which makes it easier to use a set of predefined chart features instead of the more generic data visualization which will probably require more customization.

> Or are you just suggesting pure-Java getters/setters?
>
> I was thinking to add a (unrelated) annotations to these fields, so that conversions knows to use pure Java-style getters/setters instead of the direct field access. For example, if you write:
> [...]
> you would get:
> [...]
> instead of
> [...]

OK. I'm fine with using Java getters/setters.

#9 - 03/04/2020 05:58 AM - Constantin Asofiei

Greg Shah wrote:

OK. I'm fine with using Java getters/setters.

I haven't fixed this yet. The solution should work automatically without editing the existing 4GL code, which accesses the Java field directly, but I'm not sure how easy it is.

#10 - 03/04/2020 06:53 AM - Constantin Asofiei

The list of files I've added for charting are:

```
src/com/goldencode/p2j/ui/ChartWidget.java
src/com/goldencode/p2j/ui/DockStyle.java
src/com/goldencode/p2j/ui/ImageLayout.java
src/com/goldencode/p2j/ui/StringAlignment.java
src/com/goldencode/p2j/ui/TextOrientation.java
src/com/goldencode/p2j/ui/chart/
src/com/goldencode/p2j/ui/chart/AxisAppearance.java
src/com/goldencode/p2j/ui/chart/AxisLabelAppearance.java
src/com/goldencode/p2j/ui/chart/AxisLabelAppearanceBase.java
src/com/goldencode/p2j/ui/chart/AxisLabelLayoutAppearance.java
src/com/goldencode/p2j/ui/chart/AxisLabelLayoutBehaviors.java
src/com/goldencode/p2j/ui/chart/AxisRangeType.java
src/com/goldencode/p2j/ui/chart/AxisSeriesLabelAppearance.java
src/com/goldencode/p2j/ui/chart/AxisTickStyle.java
src/com/goldencode/p2j/ui/chart/ChartGridAppearance.java
src/com/goldencode/p2j/ui/chart/ChartTextAppearance.java
src/com/goldencode/p2j/ui/chart/ChartType.java
src/com/goldencode/p2j/ui/chart/ColorAppearance.java
src/com/goldencode/p2j/ui/chart/ColorModels.java
src/com/goldencode/p2j/ui/chart/ColorScaling.java
src/com/goldencode/p2j/ui/chart/DataAppearance.java
src/com/goldencode/p2j/ui/chart/EffectsAppearance.java
src/com/goldencode/p2j/ui/chart/GradientEffect.java
src/com/goldencode/p2j/ui/chart/GridlinesAppearance.java
src/com/goldencode/p2j/ui/chart/LabelRenderer.java
src/com/goldencode/p2j/ui/chart/LegendAppearance.java
src/com/goldencode/p2j/ui/chart/LegendLocation.java
src/com/goldencode/p2j/ui/chart/LineDrawStyle.java
src/com/goldencode/p2j/ui/chart/PaintElement.java
src/com/goldencode/p2j/ui/chart/PaintElementType.java
src/com/goldencode/p2j/ui/chart/PieChartAppearance.java
src/com/goldencode/p2j/ui/chart/PieLabelAppearance.java
src/com/goldencode/p2j/ui/chart/TitleAppearance.java
src/com/goldencode/p2j/ui/chart/TooltipAppearance.java
src/com/goldencode/p2j/ui/chart/TooltipOverflow.java
src/com/goldencode/p2j/ui/chart/View3DAppearance.java
```

Most of these are used directly from 4GL code - I'll document these as such, as once we release them, we must not change the Java class name, fields, methods, etc.

#11 - 03/04/2020 02:30 PM - Constantin Asofiei

The chart-related classes are in 4335a rev 11452

#12 - 03/13/2020 03:43 PM - Greg Shah

Task branch 4335a was merged to trunk as revision 11345.