Formatted Strings

Introduction

ReportMill supports a number of data types for data substitution: String, Number, Date, byte[], double, float, int, etc. Formatting the display of these values is normally done in the layout application - simply set the font, color or size of the key (and perhaps even add a money/date formatter) and the resulting value will have that format. However, sometimes with text data it is desirable to provide additional formatting of specific words or ranges (like displaying all proper names in bold). There are two different methods for accomplishing this.

Method 1: HTML Formatted Strings

By far the easiest way to provide ReportMill with custom formatted strings is to use familiar HTML formatting tags. At runtime, any string beginning with "<HTML>" will be parsed and the specified formatting in the string will be preserved. This formatted HTML string could come straight from your datasource or could be provided by a custom method in your business object. If your string doesn't start with "<HTML>", you can use a key like @"<html>"+myKey@ to add it.

To illustrate this concept, here is a simple method that returns a formatted version of a plot summary where the movie title will always appear in bold, red type.

```
public String plotSummaryFormatted() {
  StringBuffer stringBuf = new StringBuffer(plotSummary)
  int startIndex = stringBuf.indexOf(title);
  while(index>=0) {
    stringBuf.replace(startIndex, startIndex + title.length(),
        "<font face=Times-Bold color=red>" + title + "</font>");
    startIndex = stringBuf.indexOf(title, startIndex + title.length());
  }

  return "<html>" + stringBuf.toString();
}
```

Currently, not all available HTML tags are supported. Unsupported tags will be skipped, but the text contained by those tags will still be rendered. The tags supported in the current release of ReportMill are:

- B or STRONG for bold formatting
- EM or I for italicized formatting
- U for underlining
- UL, OL, LI for lists
- FONT using COLOR, SIZE, and FACE attributes to specify font settings
  - COLOR can be any RGB hex value in the format "#RRGGBB" (e.g., #FF0000 for red)
    You can also use the standard 16 color names: black, silver, gray, white, maroon, red, purple, fushia, green, lime, olive, yellow, navy, blue, teal, or aqua
  - SIZE is a relative control specificied using values 1 through 7 where 7 is the largest relative size (200% of the font size of the key in the template) and 1 is the smallest size (50% of the font size of the key in the template).
  - FACE can be any available font name. Multiple font names can be specified to give an order of preference in the event a font is not available (e.g., "Copperplate, Arial")
- BR and P for line break control

Formatted Strings (continued)

Method 2: Providing ReportMill An XString (String with Attributes)

In previous versions of ReportMill, the only way to provide formatted strings was to create a custom method in your business logic that returned your string as a com.reportmill.text.RMXString object. For various reasons, you may find that you still prefer this method over the HTML formatting approach. Here is an example:

```
import com.reportmill.base.*;
import com.reportmill.graphics.*;
import com.reportmill.text.*;

public class Movie {

  // Ivars
  public String title;
  public String plotSummary;

  // Returns the plot summary string as an xstring with title name occurrances bolded
  public String getPlotSummaryFormatted() {

    // Get return xstring
    RMXString xstring = new RMXString(plotSummary);
    int index = plotSummary.indexOf(title);

    while(index>=0) {
      RMFont font = RMFont.getFont("Times-Bold", 12);
      RMColor color = new RMColor(1, 0, 0);
      xstring.addAttribute(font, index, index + title.length());
      xstring.addAttribute(color, index, index + title.length());
      index = plotSummary.indexOf(title, index + title.length());
    }

    return xstring;
  }
}
```

This example shows the String attribute plotSummary being returned as ReportMill's formatted string class, RMXString. The constructor takes the plain string. Then the method addAttribute is called with an attribute object, the start index and the end index. The objects need to be either an RMFont instance (its constructor takes a font name and size) or RMColor instance (its constructor takes three float values which range from 0-1 and represent amounts of red, green and blue).