TextualDisplay

TextualDisplay is a utility for displaying data as styled text. Most commonly, it can be used to display scientific data as HTML tables.

Requirements

Developer's Guide

Abstract

This guide describes in detail how to use TextualDisplay in a Swing application. In general, you serialize your object to XML, transform XML to HTML via an XSLT style sheet, and display HTML in a Swing widget.

Overview

We recommend the following steps:

1. #Check out TextualDisplay from CVS.
2. #Build TextualDisplay and run an example application.
3. #Serialize your exemplary object to XML.
4. #Write an XSLT style sheet that transforms your XML to HTML.
5. #Use TextDispViewer to look at the result of your XSLT transformation.
6. #Integrate TextualDisplay into your application.

Check out TextualDisplay from CVS

Make sure you have access to LCLS CVS!
Check out safely from the CVS head:

```
bash-3.00$ cvs co physics/textdisp
bash-3.00$ cd physics/textdisp
```

Build TextualDisplay and run an example application

Make sure you have ANT version 1.6 (or newer)!
In the top directory of TextualDisplay, type

```
ant
```

if BUILD is successful, run the example application:

```
java -cp textdisp.jar:lib/core-renderer-R7final.jar:lib/xstream-1.2.2.jar edu.stanford.slac.util.textdisp.
example.ExampleApplication
```

Some tips on how to use the example application:

- The main window consists of two panels, separated by a drag-able bar (if you see only one panel, just drag the bar).
- The left panel contains the serialized object (XML), the right panel the styled data from the same object (HTML). All tags in the XML file are self-explanatory.
- The command line output of the application contains some useful info, e.g. when you resize the main window.

If you want to look at the source code of the example, check out these classes:

- edu.stanford.slac.util.textdisp.example.ExampleApplication
  (the main method contains a good example of how to use TextualDisplay)
- edu.stanford.slac.util.textdisp.example.ExampleData
  (an instance of this class is serialized to XML that is displayed in the left panel of the main application window)

Serialize your exemplary object to XML

Add this code to your test class:

```
Object myData = ...;
ObjectSerializer objSerializer = new ObjectSerializer();
String xml = objSerializer.toXML(myData);
System.out.println(xml);
```

You can tweak the XML output via the XStream API, which you get from ObjectSerializer:

```
XStream xstream = objSerializer.getXStream();
```

Some of the most useful methods are:

```
xstream.alias(...);
xstream.omitField(...);
```

More info:

- http://xstream.codehaus.org/manual-tweaking-output.html
- XStream JavaDoc

When you are pleased with your exemplary XML, proceed to the next step.

Write an XSLT style sheet that transforms your XML to HTML

This step is the most specialized of all. Basically, I can only recommend to look at:

- The example file edu/stanford/slac/util/textdisp/example/example.xsl
- The tutorial at http://w3schools.com/xsl/default.asp
- A good article on embedding CSS into XSLT at http://www.ibm.com/developerworks/xml/library/x-xslt5.html

Of course, you can always ask an XSLT expert (i.e. me) to write you a style sheet...

Use TextDispViewer to look at the result of your XSLT transformation

Copy your style sheet to the directory of your test class and add this code to your test class:

```
String xml = ...;
URL url = TestClass.class.getResource("myxslt.xsl");

TextDispModel textDispModel = null;
try {
textDispModel = TextDispModel.newDefault(xml, url);
} catch (IOException e) {
e.printStackTrace();
} catch (TransformerException e) {
e.printStackTrace();
}

TextDispViewer textDispViewer = new TextDispViewer();
textDispViewer.setModel(textDispModel);
```

Tweak your XSLT, until you're satisfied with what you see.

Integrate TextualDisplay into your application

Below are some things to consider when integrating TextualDisplay into your application.

HTMLScrollPane

To get the maximum support available in Swing, you should display your HTML only on the edu.stanford.slac.util.textdisp.ui.HTMLScrollPane widget, which is a subclass of javax.swing.JScrollPane and wraps the Flying Saucer web browser. Also, you might want to turn off the logging messages:

```
System.setProperty("xr.util-logging.java.util.logging.ConsoleHandler.level", "OFF"); //do this ASAP
HTMLScrollPane htmlScrollPane = new HTMLScrollPane();
Container myContainer = ...;
... //do layout
myContainer.add(htmlScrollPane, ...); /
String html = ...;
htmlScrollPane.setStyledData(html);
```

You have access to the wrapped browser widget:

```
HTMLScrollPane htmlScrollPane = ...;
htmlScrollPane.xhtmlPanel.incrementFontSize();
```

Note: TextDispViewer uses HTMLScrollPane.

Upload XSLT style sheets to a web server

When the infrastructure is ready, you might want to upload your XSLT style sheets to a web server. You can easily construct a URL in such case:

```
URL url = new URL("http://...");
```

We recommend that you externalize all your URLs (or any strings, for that matter).