Writing Java report chapters

Revised: April 12, 2011

Created: May 7, 2010

Author: Noé Lavallée

CONTENTS

Summary

Starting from Mega 2009 SP4 it is possible to write report chapters in Java. This article explains how to do so, the new architecture and the differences with VBS analyses.

INTRODUCTION TO JAVA REPORT CHAPTERS

Generalities

Starting with MEGA 2009 SP4, a report chaptercan be written in Java.

It is highly recommended to write new reports in this language in order to benefit from the platform improvements and also to better handle PDF/RTF document generation. The old report platform will not be improved whereas the Java platform will be.

Compared to VB Script reports, Java reports are written differently as described in this article.

Both use the same metamodel described in the product documentation.

Data and view separation

VB Script report chapter macros generate HTML.

Java report chapter macros generate an object structure describing datasets (report data) and the type of view to apply to these datasets.

For more details on this structure, you should refer to the technical article "Java report data structure".

Conversion from this object structure to a report output (HTML or PDF for example) is handled by the report engine using specific renderers. Renderers are the subject of another technical article "Writing Java report renderers".

> This new architecture allows for better management of different output formats and more flexibility and modularity. It is therefore highly recommended to write new report chapter macros in Java, using the method described hereafter.

PREREQUISITES

Report modeling

This technical article does not cover modeling of reports, report templates, report parameters and report chapters.

Modeling is the same as for VB Script reports. You should refer to product documentation on this subject.

Setup a Java development environment

A Java report chapter macro is a Java Plug-in which uses Mega APIs.

You should first refer to the technical article "Creating MEGA Plug-ins with Java" to set up a Java development environment adapted to MEGA.

Referencing jars and javadoc

Following the method described in "Creating MEGA Plug-ins with Java" you should reference the following jars in your development environment:

- mj_toolkit.jar
- mj_api.jar
- mj_anls.jar

To allow for contextual help in Eclipse, you should also reference the corresponding javadocs:

- mj_api.doc.zip
- mj_anls.doc.zip

Interfaces to implement

Report chapters

A Java report chapter must implement one of the following interfaces:

- `com.mega.modeling.analysis.AnalysisReport`
- `com.mega.modeling.analysis.AnalysisReportWithContext`

The `getReportContent` method is the only method required to implement these interfaces. Its parameters are:

- a Mega Root,
- a Map of parameter Lists referenced by the HexaIdAbs of the "Analysis Parameter" repository object,
- an optional userData object,
- and in the case of `AnalysisReportWithContext` an extra Analysis object which provides contextual information.

It produces the report content in the form of a ReportContent object. This is an instance of the ReportContent Java class, to which all data and view information is given in order to pass it to the analysis engine and its renderers.

More information on ReportContent objects is available in the "Java analysis data structure" technical article and in the engine Javadoc.

This typically results in either of the following structures:

```
public class MyReport implements AnalysisReport {
  @Override
  public ReportContent getReportContent(final MegaRoot root, final Map<String,
List<AnalysisParameter>> parameters, final Object userData) {
    …
  }
}
```

```
public class MyReport implements AnalysisReportWithContext {
  @Override
  public ReportContent getReportContent(final MegaRoot root, final Map<String,
List<AnalysisParameter>> parameters, final Analysis analysis, final Object
userData) {
    …
  }
}
```

Callbacks

Callback calls allow for user interactivity in tables and trees, by allowing the execution of a Callback function when the user clicks on a cell. The callback function subsequently updates the cell content.

The Java implementation of the macro that handles callback calls must implement the `com.mega.modeling.analysis.AnalysisCallback` interface.

This can be the same macro and Java class as the report chapter itself or a different macro.

The `Callback` method is the only method required to implement the AnalysisCallback interface. Its parameters are:

- a Mega Root,
- the HTML content of the cell that was clicked,
- MegaCollections of the line and column objects of the cell,
- an optional userData.

It produces the new content to be rendered in the clicked cell, in the form of an HTML string.

For example:

```
public class MyReportCallback implements AnalysisCallback {
  @Override
  public String Callback(final MegaRoot root, final String
HTMLCellContent, final MegaCollection ColumnMegaObjects, final
MegaCollection LineMegaObjects, final Object userData) {
    …
  }
}
```

Implementing the macro

General steps

The implementation of the getReportContent function follows the following general steps:

- Create a new ReportContent object that will contain all report data and views:

```
final ReportContent reportContent = new ReportContent("");
```

- Create one or more Datasets (Java objects that contain all the data to be rendered) and add them to the ReportContent, getting the ID of the dataset for future use:

```
final Dataset d = new Dataset("");
final int datasetID = reportContent.addDataset(d);
```

- Create one or more Dimensions (the equivalent of x, y, etc. axis in a diagram) and add them to the Dataset(s):

```
final Dimension dim = new Dimension("");
d.addDimension(dim);
```

- Create one or more items and add them to the Dimension(s).This is the equivalent of line or column headers in a table.

```
dim.addItem(new Text("Some text…", false));
```

- Create one or more items and add them to the Dataset(s):

```
d.addItem(new Value((double) n),"1”);
```

- Create one or more Views (or Texts or MegaObjectProperties), using the ID of the dataset they should represent, and add them to the ReportContent. A view links a dataset to a renderer in order to define where and how the dataset should be rendered.

```
final View v = new View(datasetID);
reportContent.addView(v);
```

- Add one or more renderers to the View(s) to specify how the view dataset should be shown (here, as a table):

```
v.addRenderer(AnalysisReportToolbox.rTable);
```

- Return the now complete ReportContent:

```
return reportContent;
```

> More information on the data structure of ReportContent, Dataset, Dimension, View, Item, etc. is available in the "Java report data structure" technical article and in the engine Javadoc.

Example

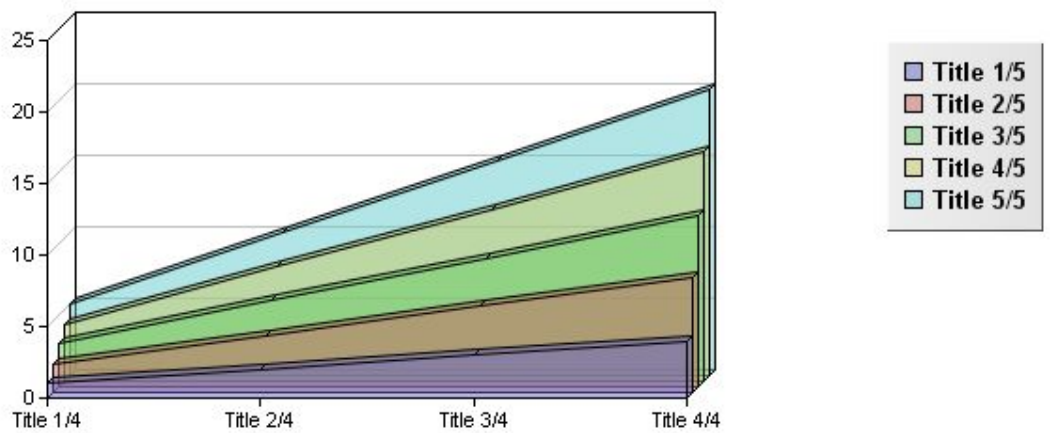
A basic working example is as follows:

```
import java.util.List;
import java.util.Map;

import com.mega.modeling.analysis.AnalysisParameter;
import com.mega.modeling.analysis.AnalysisReport;
import com.mega.modeling.analysis.AnalysisReportToolbox;
import com.mega.modeling.analysis.content.Dataset;
import com.mega.modeling.analysis.content.Dimension;
import com.mega.modeling.analysis.content.ReportContent;
import com.mega.modeling.analysis.content.Text;
import com.mega.modeling.analysis.content.Value;
import com.mega.modeling.analysis.content.View;
import com.mega.modeling.api.MegaRoot;

/**
 * This is a basic demonstration report.
 * @author NLE
 */

public class MyReport implements AnalysisReport {
  public ReportContent getReportContent(final MegaRoot root, final Map<String,
List<AnalysisParameter>> parameters, final Object userData) {
    final ReportContent reportContent = new ReportContent("");

    // Creating a 2D Dataset and its dimensions
    final Dataset d2 = new Dataset("");
    final Dimension dim21 = new Dimension("");
    final Dimension dim22 = new Dimension("");
    // Set the dimension sizes
    // (compulsory only when no Items are added to the dimension)
    dim21.setSize(4);
    dim22.setSize(5);
    // Add the dimensions to the Dataset
    d2.addDimension(dim21);
    d2.addDimension(dim22);

    // Filling in the dataset and its dimensions
    // Arbitrary data is used here
    for (int i = 1; i <= 4; i++) {
      dim21.addItem(new Text("Title " + i + "/4", false));
      for (int j = 1; j <= 5; j++) {
        d2.addItem(new Value((double) i * j), i + "," + j);
      }
    }
    for (int j = 1; j <= 5; j++) {
      dim22.addItem(new Text("Title " + j + "/5", false));
    }
```

```
    // Add the dataset to the report
    final int datasetID = reportContent.addDataset(d2);

    // Add a table and list view of the dataset
    // List will only be used if table cannot be used
    final View v1 = new View(datasetID);
    v1.addRenderer(AnalysisReportToolbox.rTable);
    v1.addRenderer(AnalysisReportToolbox.rList);
    reportContent.addView(v1);

    // Add an area chart view, with the same dataset (not duplicated)
    final View v2 = new View(datasetID);
    v2.addRenderer(AnalysisReportToolbox.rAreaChart);
    reportContent.addView(v2);

    return reportContent;
  }
}
```

This code will typically result in the following rendering:

	Title 1/5	Title 2/5	Title 3/5	Title 4/5	Title 5/5
Title 1/4	1.0	2.0	3.0	4.0	5.0
Title 2/4	2.0	4.0	6.0	8.0	10.0
Title 3/4	3.0	6.0	9.0	12.0	15.0
Title 4/4	4.0	8.0	12.0	16.0	20.0



Going further

You should refer to the Javadoc for all possible options and possibilities. The Javadoc is accessible contextually in Eclipse if you configured it as suggested in the above chapter, as well

as in HTML format in a zip file “mj_anls.doc.zip" in the "java\doc" directory of your MEGA installation.

You can of course make use of all MEGA Java APIs (documented in the “mj_api.doc.zip" javadoc) to handle MegaObjects and MegaCollections.

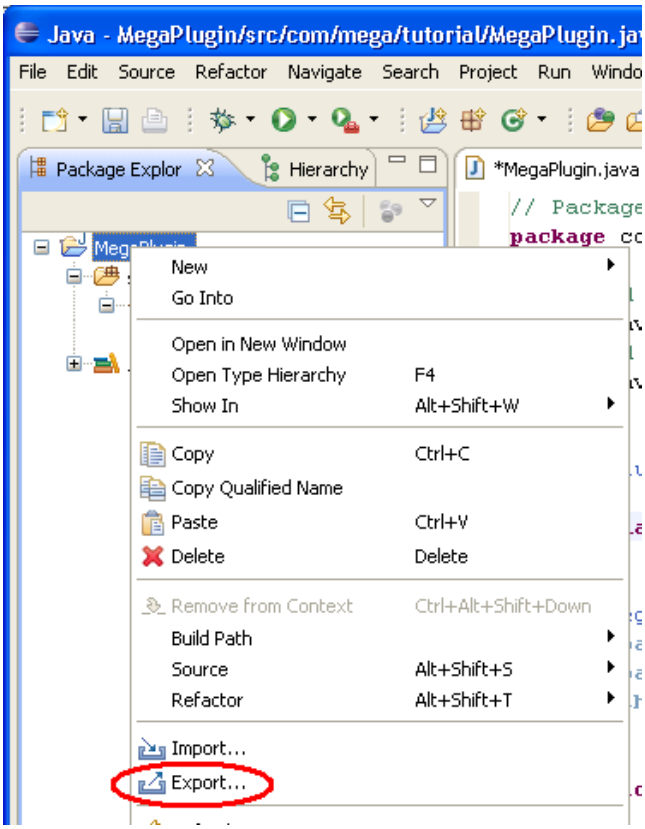
A more complex example is provided at the end of this document.

USING YOUR IMPLEMENTATION FROM THE MEGA REPORT MACRO

Compile

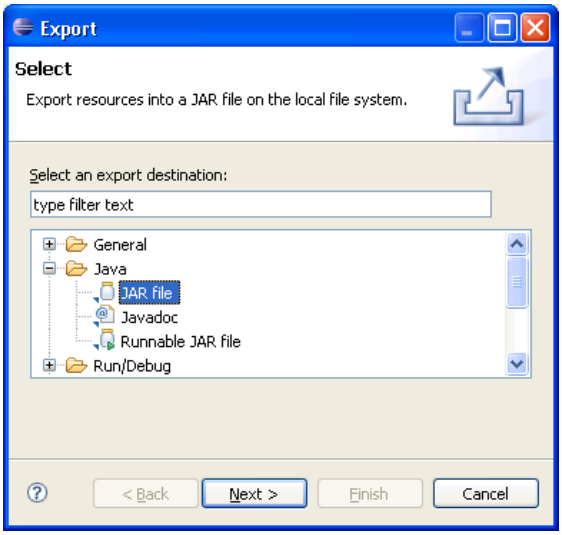
In order to use your Java report chapter, it must be exported in a .jar in the java/lib directory of your MEGA installation.

Compilation of the Java component in the form of a JAR file is via the "Export" menu of the Java project:



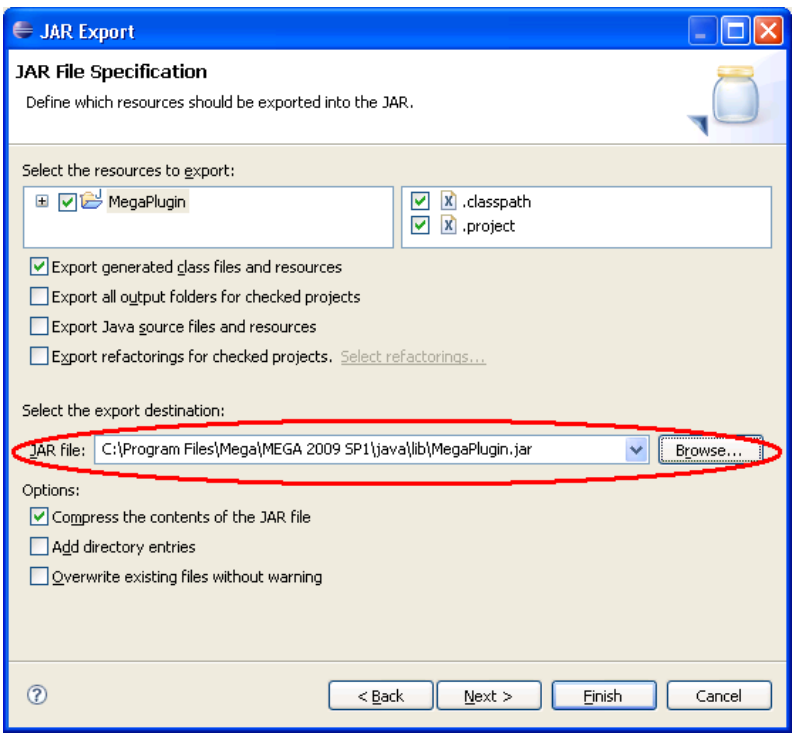
A JAR can contain as many Java report chapter implementing classes as you wish.

"JAR File" export in the Java directory should be selected:



Indicate the location of the JAR file to be generated and click "Finish":

The JAR file **must** be generated (or copied after generation) in the "java\lib" directory of the MEGA installation site.

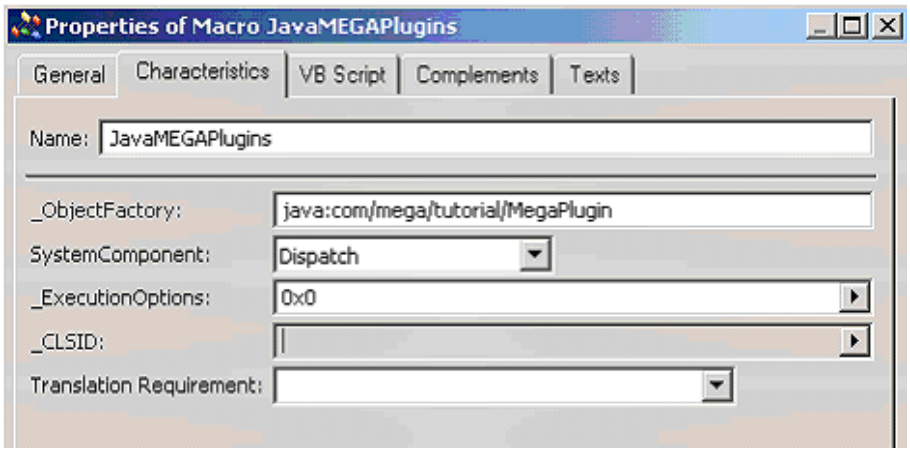


The name of the JAR file itself is not significant; you should use a name that makes sense in your project.

Configure the Macro

Once you have added the JAR file to the "java\lib" directory, restart Mega and edit the properties of your report macro in order to reference your Java class.

In the macro properties dialog box, assign the "Dispatch" value to the "SystemComponent" attribute, then specify the "_ObjectFactory" attribute using the report Java class. For example, the value "java:com/mega/tutorial/MegaPlugin" identifies the "MegaPlugin" class of the "com.mega.tutorial" package.



The VB Script of the macro should be left empty.

CODE EXAMPLE

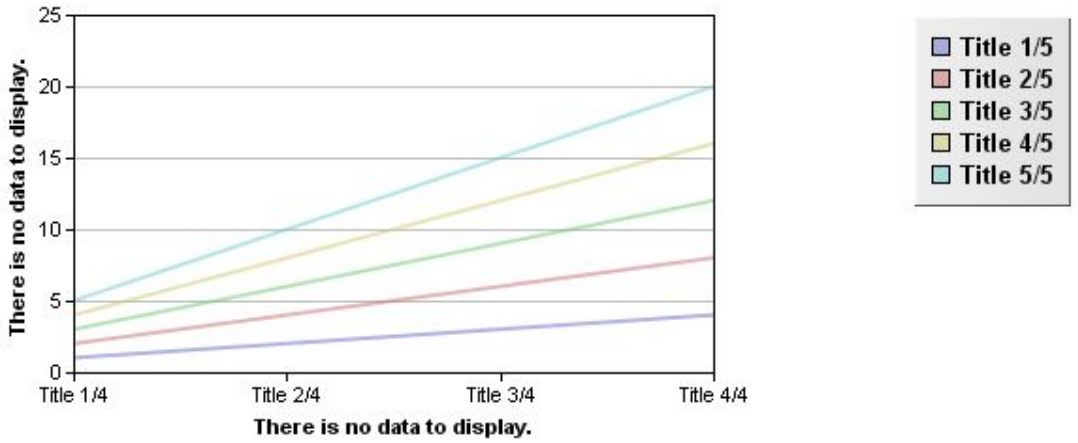
```
import java.util.Date;
import java.util.List;
import java.util.Map;

import com.mega.modeling.analysis.AnalysisCallback;
import com.mega.modeling.analysis.AnalysisParameter;
import com.mega.modeling.analysis.AnalysisReport;
import com.mega.modeling.analysis.AnalysisReportToolbox;
import com.mega.modeling.analysis.content.Dataset;
import com.mega.modeling.analysis.content.Dimension;
import com.mega.modeling.analysis.content.Item;
import com.mega.modeling.analysis.content.MegaObjectProperty;
import com.mega.modeling.analysis.content.ReportContent;
import com.mega.modeling.analysis.content.Text;
import com.mega.modeling.analysis.content.Value;
import com.mega.modeling.analysis.content.View;
import com.mega.modeling.api.MegaCollection;
import com.mega.modeling.api.MegaObject;
import com.mega.modeling.api.MegaRoot;
import com.mega.modeling.api.MegaAttribute.OutputFormat;
import com.mega.toolkit.errmngt.ErrorLogFormater;

/**
 * This is a demonstration report.
 * @author NLE
 */

public class MyReport implements AnalysisReport, AnalysisCallback {
  @Override
  public ReportContent getReportContent(final MegaRoot root, final
Map<String, List<AnalysisParameter>> parameters, final Object
userData) {
    // Error Management exemple
    final ErrorLogFormater err = new ErrorLogFormater();
    err.openSession(root);
    // Do not forget to update this line
    err.addSessionInfo("Component", "(Java) New Analysis Engine:
TestReport: getReportContent");

    // Initialize the report content
    final ReportContent reportContent = new ReportContent("");

    try {
```

Demo 1: Charts using 2D datasets

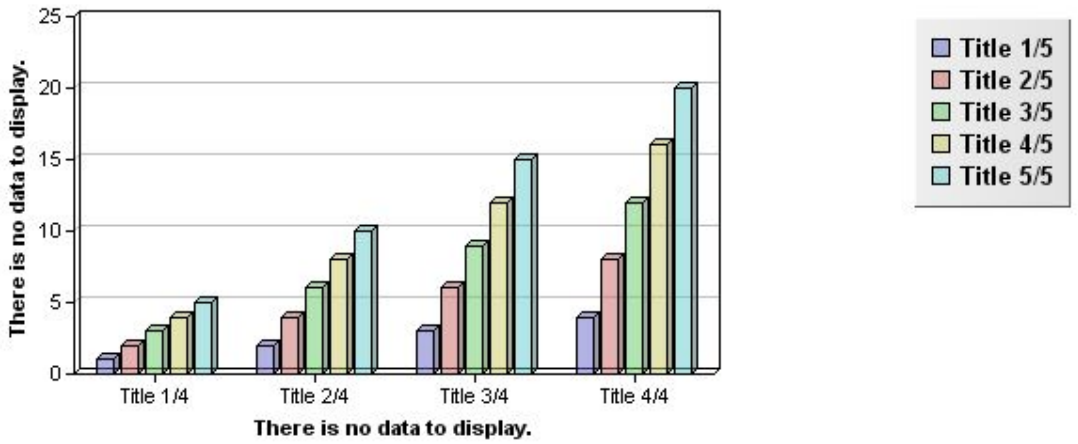
```
      final Dataset d2 = new Dataset("~LshNx7Mw6zE0[No Data To
Display]");
      final Dimension dim21 = new Dimension("~LshNx7Mw6zE0[No Data
To Display]");
      final Dimension dim22 = new Dimension("~LshNx7Mw6zE0[No Data
To Display]");
      dim21.setSize(4);
      dim22.setSize(5);
      d2.addDimension(dim21);
      d2.addDimension(dim22);

      // Filling in the dataset (here with arbitrary data)
      for (int i = 1; i <= 4; i++) {
        dim21.addItem(new Text("Title " + i + "/4", false));
        for (int j = 1; j <= 5; j++) {
          d2.addItem(new Value((double) i * j), i + "," + j);
        }
      }
      for (int j = 1; j <= 5; j++) {
        dim22.addItem(new Text("Title " + j + "/5", false));
      }

      // Add the Dataset to the report
      final int datasetID = reportContent.addDataset(d2);

      // Add the Dataset to many different views
      final View v21 = new View(datasetID, true, false);
      v21.addRenderer(AnalysisReportToolbox.rAreaChart);
      reportContent.addView(v21);

      final View v22 = new View(datasetID);
      v22.addRenderer(AnalysisReportToolbox.rLineChart);
      reportContent.addView(v22);

      final View v23 = new View(datasetID);
      v23.addRenderer(AnalysisReportToolbox.rBarChart);
      reportContent.addView(v23);

      final View v24 = new View(datasetID);
      v24.addRenderer(AnalysisReportToolbox.rRadarChart);
      reportContent.addView(v24);
```

There is no data to display.

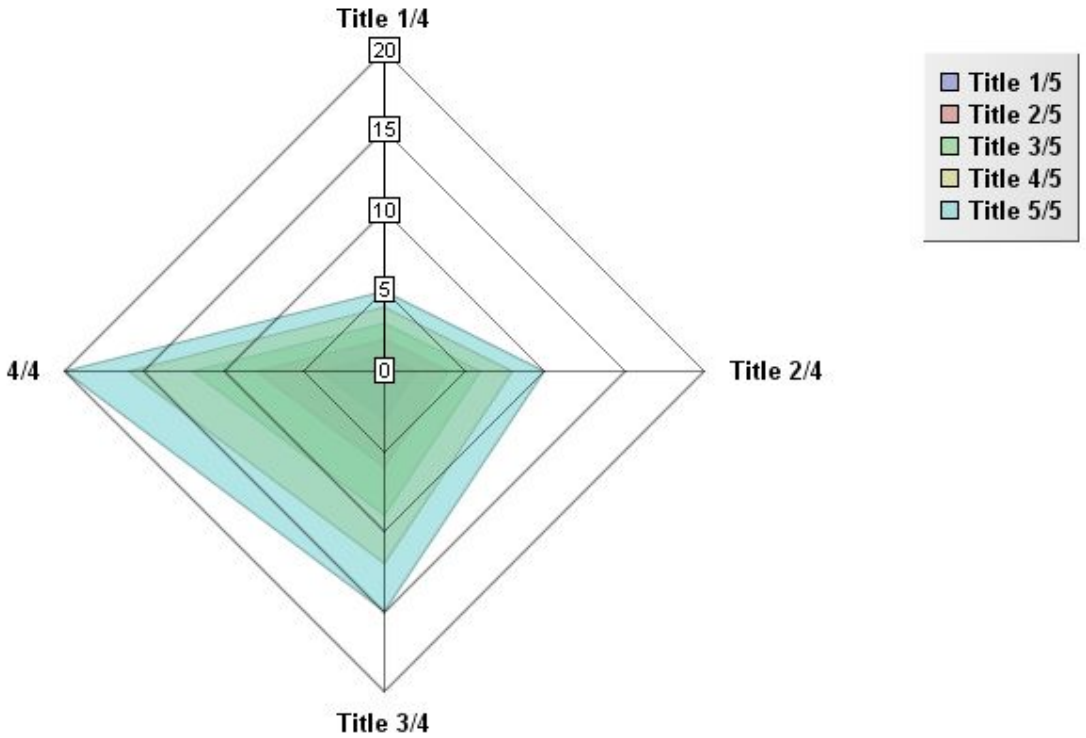
There is no data to display.



There is no data to display.



There is no data to display.



Demo 2: Parameters in tables with vertical headers

```
      // Going through parameters
      for (final String paramType : parameters.keySet()) {
        reportContent.addText(new Text("" +
root.getObjectFromID(paramType).getAttribute("~Z20000000D60[Short
Name]").getFormated(OutputFormat.html, ""), false, 3));

        // Going through its values
        for (final AnalysisParameter paramValue :
parameters.get(paramType)) {
          reportContent.addText(new
Text(paramValue.getParameterObject().getAttribute("~Z20000000D60[Sh
ort Name]").getFormated(OutputFormat.html, ""), false, 4));

          // and the actual individual values
          final Dataset paramDataset = new Dataset("");
          final Dimension dim1 = new Dimension("");
          final Dimension dim2 = new Dimension("");
          dim2.setSize(1);
          final Item title = new Text("Column Title", false);
          // Set the title column header to allow ordering of
column
```

```
          title.setOrderable(true);
          dim2.addItem(title);
          int i = 1;
          for (final MegaObject value : paramValue.getValues()) {
            paramDataset.addItem(new
MegaObjectProperty(value.megaField(), "~Z20000000D60[Short Name]"),
i + ",1");
            i++;
          }
          dim1.setSize(i - 1);
          paramDataset.addDimension(dim1);
          paramDataset.addDimension(dim2);

          final int paramDatasetID =
reportContent.addDataset(paramDataset);

          final View paramView1 = new View(paramDatasetID, true,
false);

paramView1.addRenderer(AnalysisReportToolbox.rVerticalTextTable);
          reportContent.addView(paramView1);

        }
      }
```

Prohibited Technology

Prohibited Technology

⊞

Accepted Technology

Accepted Technology

⊞

Expected Technology

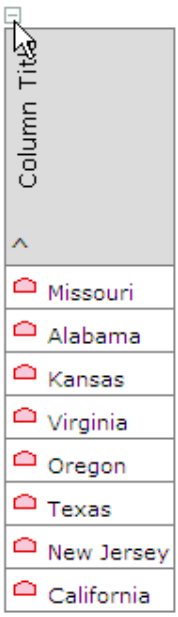
Expected Technology

⊞

Information System

American States

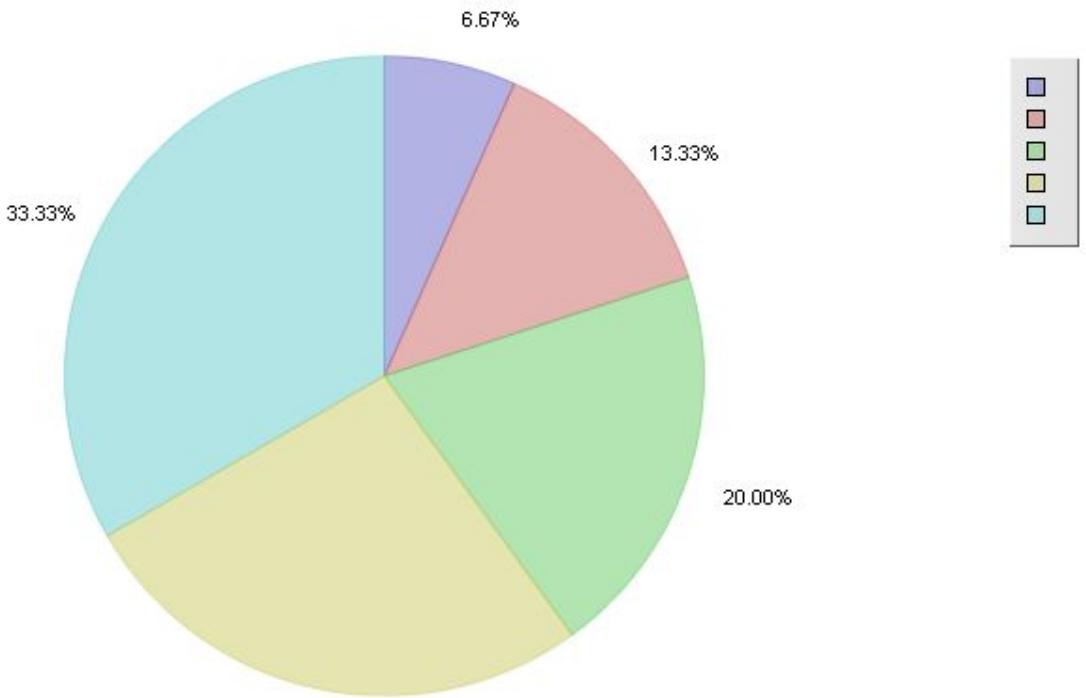
⊟



Demo 3: Pie chart

```
final Dataset d1 = new Dataset("");
final Dimension dim11 = new Dimension("");
dim11.setSize(5);
d1.addDimension(dim11);
```

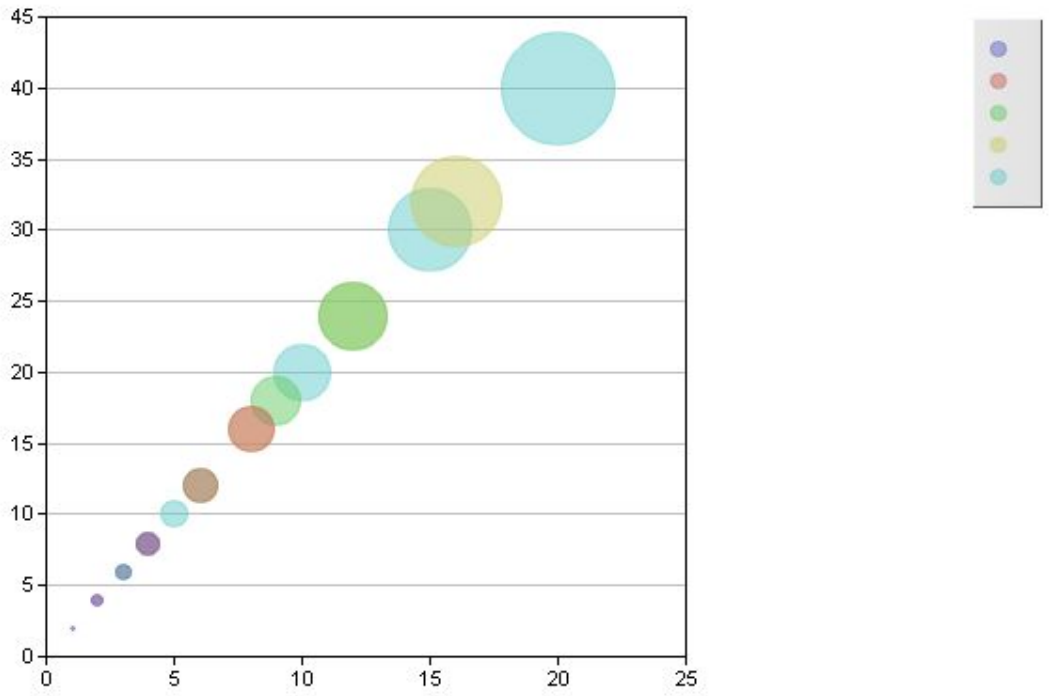
```
      for (int i = 1; i <= 5; i++) {
        d1.addItem(new Value((double) i), i + "");
      }
      final View v1 = new View(reportContent.addDataset(d1));
      v1.addRenderer(AnalysisReportToolbox.rPieChart);
      reportContent.addView(v1);
```



Demo 4: Bubble chart

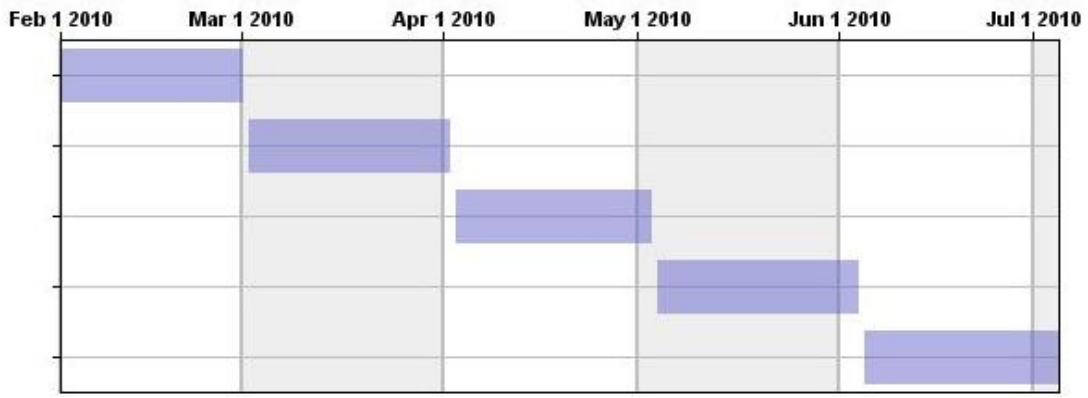
```
      final Dataset d3 = new Dataset("");
      final Dimension dim31 = new Dimension("");
      final Dimension dim32 = new Dimension("");
      final Dimension dim33 = new Dimension("");
      dim31.setSize(5);
      dim32.setSize(4);
      dim33.setSize(3);
      d3.addDimension(dim31);
      d3.addDimension(dim32);
      d3.addDimension(dim33);
      for (int i = 1; i <= 5; i++) {
        for (int j = 1; j <= 4; j++) {
          for (int k = 1; k <= 3; k++) {
            d3.addItem(new Value((double) i * j * k), i + "," + j +
"," + k);
          }
```

```
        }
      }
      final View v3 = new View(reportContent.addDataset(d3));
      v3.addRenderer(AnalysisReportToolbox.rBubbleChart);
      reportContent.addView(v3);
```



Demo 5: Simple Gantt chart

```
      final Dataset dGantt = new Dataset("");
      final Dimension dimGantt = new Dimension("");
      dimGantt.setSize(5);
      dGantt.addDimension(dimGantt);
      for (int i = 1; i <= 5; i++) {
        dGantt.addItem(new Value(new Date(2010 - 1900, i, i), new
Date(2010 - 1900, i + 1, i)), i + "");
      }
      final View vGantt = new
View(reportContent.addDataset(dGantt));
      vGantt.addRenderer(AnalysisReportToolbox.rGanttChart);
      reportContent.addView(vGantt);
```



Demo 6: Tree of parameters

```
      final Dataset paramDataset = new Dataset("");
      // Callback: set the Macro to be called back, in this exemple
you should
      // reference the macro referencing this Java class
      paramDataset.setCallback("~Jq(Ipv4W4P50[Analysis - Set of
Parameters]");
      final Dimension dim1 = new Dimension("");
      final Dimension dim2 = new Dimension("");
      dim2.setSize(1);
      int i = 0;
      for (final String paramType : parameters.keySet()) {
        dim1.addItem(new
MegaObjectProperty(root.getObjectFromID(paramType).megaField(),
"~Z20000000D60[Short Name]"), 1);
        i++;
        paramDataset.addItem(new
MegaObjectProperty(root.getObjectFromID(paramType).megaField(),
"~210000000900[Name]"), i + ",1");
        // Going through its values
        for (final AnalysisParameter paramValue :
parameters.get(paramType)) {
          dim1.addItem(new
MegaObjectProperty(paramValue.getParameterObject().megaField(),
"~Z20000000D60[Short Name]"), 2);
          i++;
          paramDataset.addItem(new
MegaObjectProperty(paramValue.getParameterObject().megaField(),
"~210000000900[Name]"), i + ",1");
          // and the actual individual values!!
          for (final MegaObject value : paramValue.getValues()) {
```

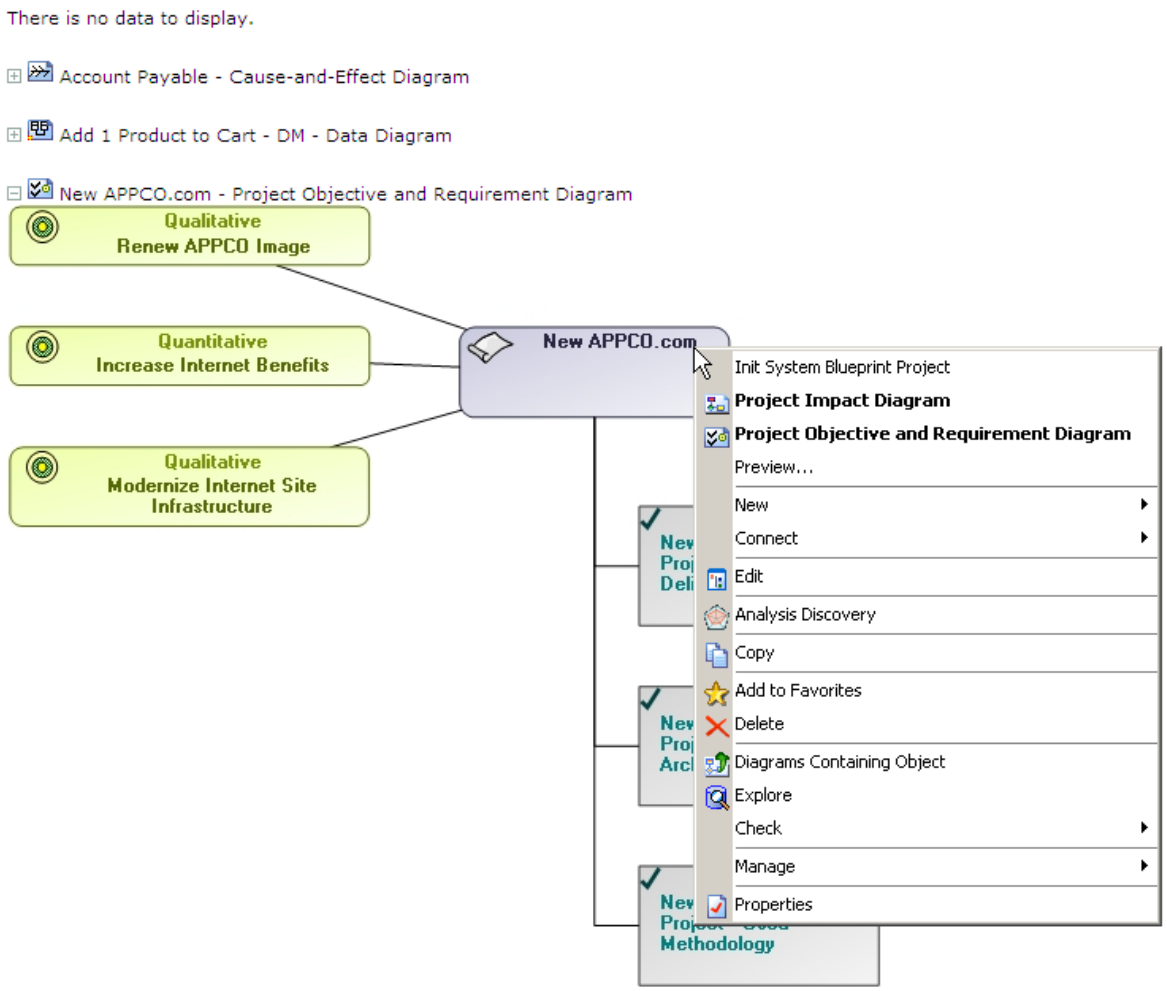
```
            dim1.addItem(new MegaObjectProperty(value.megaField(),
"~Z20000000D60[Short Name]"), 3);
            i++;
            paramDataset.addItem(new
MegaObjectProperty(value.megaField(), "~210000000900[Name]"), i +
",1");
          }

        }
      }
      paramDataset.addDimension(dim1);
      paramDataset.addDimension(dim2);
      final View treeView = new
View(reportContent.addDataset(paramDataset));
      treeView.addRenderer(AnalysisReportToolbox.rTree);
      treeView.addRenderer(AnalysisReportToolbox.rTable);
      reportContent.addView(treeView);
```

Prohibited Technology	Infrastructure Compliance::Prohibited Technology
Accepted Technology	Infrastructure Compliance::Accepted Technology
Expected Technology	Infrastructure Compliance::Expected Technology
Information System	Infrastructure Compliance::Information System
American States	.Net Architecture Compliance::American States
Missouri	Missouri
Alabama	Alabama
Kansas	Kansas
Virginia	Virginia
Oregon	Oregon
Texas	Texas
New Jersey	New Jersey
California	California

Demo 7: Clickable diagrams

```
      final Dataset dDiags = new Dataset("~LshNx7Mw6zE0[No Data To
Display]");
      final Dimension dimD1 = new Dimension("");
      dDiags.addDimension(dimD1);
```

```
      // filling in the dataset
      dimD1.addItem(new MegaObjectProperty("~uGEJcPMZ4fM0[Account
Payable - Cause-and-Effect Diagram]", ""));
      dimD1.addItem(new MegaObjectProperty("~B9QnnNye9940[Add 1
Product to Cart - DM - Data Diagram]", ""));
      dimD1.addItem(new MegaObjectProperty("~vU3v8M(a9L50[New
APPCO.com - Project Objective and Requirement Diagram]", ""));
      final int datasetDiagID = reportContent.addDataset(dDiags);

      final View vDiag = new View(datasetDiagID);
      vDiag.addRenderer(AnalysisReportToolbox.rDiagrams);
      reportContent.addView(vDiag);
```



Demo 8: Clickable illustrating diagrams

```
      final Dataset diDiags = new Dataset("");
```

```
      final Dimension dimiD1 = new Dimension("");
      diDiags.addDimension(dimiD1);

      // filling in the dataset with the objects we want to find in
the diagram
      // and the color to apply to them
      dimiD1.addItem(new MegaObjectProperty("~4YRLwW5V49o0[Creation
of an imaginary supplier]", ""));
      final int datasetiDiagID = reportContent.addDataset(diDiags);
      diDiags.addItem(new Value((short) 200, (short) 0, (short) 0),
"1");

      final View viDiag = new View(datasetiDiagID);

viDiag.addRenderer(AnalysisReportToolbox.rIllustratingDiagrams);
      reportContent.addView(viDiag);

      // End of error management
    } catch (final Exception e) {
      err.logMessage("Report generation failed:");
      err.logError(e);
      err.closeSession();
    }
    return reportContent;
  }

  @Override
  public String Callback(final MegaRoot root, final String
HTMLCellContent, final MegaCollection ColumnMegaObjects, final
MegaCollection LineMegaObjects, final Object UserData) {
    // Exemple of callback in a table or tree
    return "[TEST] Was called back successfully, was[" +
HTMLCellContent + "]";
  }
}
```