Seagate Software

Seagate Crystal Reports

Crystal Reports Java Viewer Bean White paper

Contents

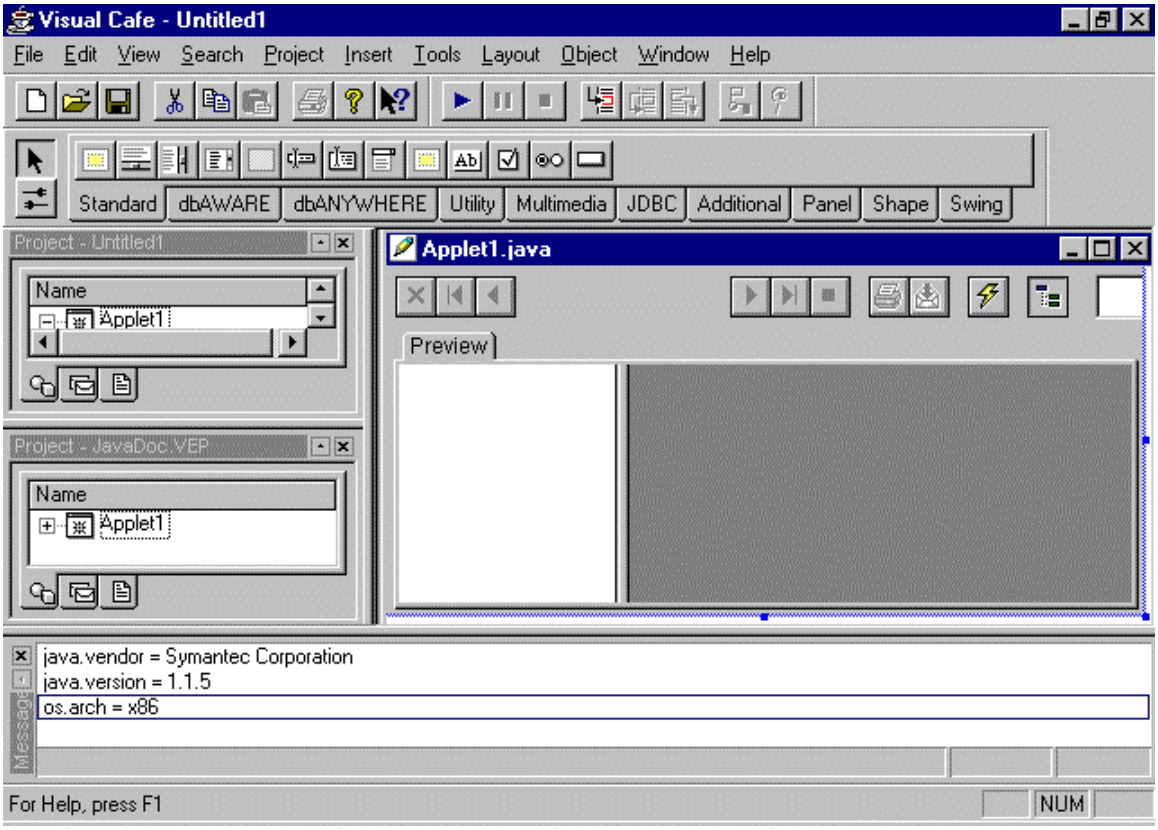
What is the Report Viewer Bean?

A JavaBean is a collection of one or more Java classes that are usually bundled into a single Java Archive (.Jar) file. The Bean is meant to be a component that can be reused in development environments. A JavaBean will usually have properties and methods that can be adjusted by the developer at design and run-time. Like other JavaBeans the Report Viewer Bean is a component that can be added to your IDE project and adjusted through its properties accordingly. The Bean is designed to work only in a development environment that supports adding a JavaBean as a component to your project. The Bean will work as an Applet within a browser or it can be compiled into an executable file and run independently (although your application will still have to have access to both a TCP/IP connection and a Web-server in order to access the report)

Implementing the Crystal Reports Java Bean into an Integrated Design Environment

After the installation of Crystal Reports you will be able to insert the java bean as a component into an IDE project. Use the following steps to implement the Bean:

In your Project Window select the component that is located under the ReportViewerBean folder. You should now see the JavaBean appear in your project like the following example in Visual Cafe:



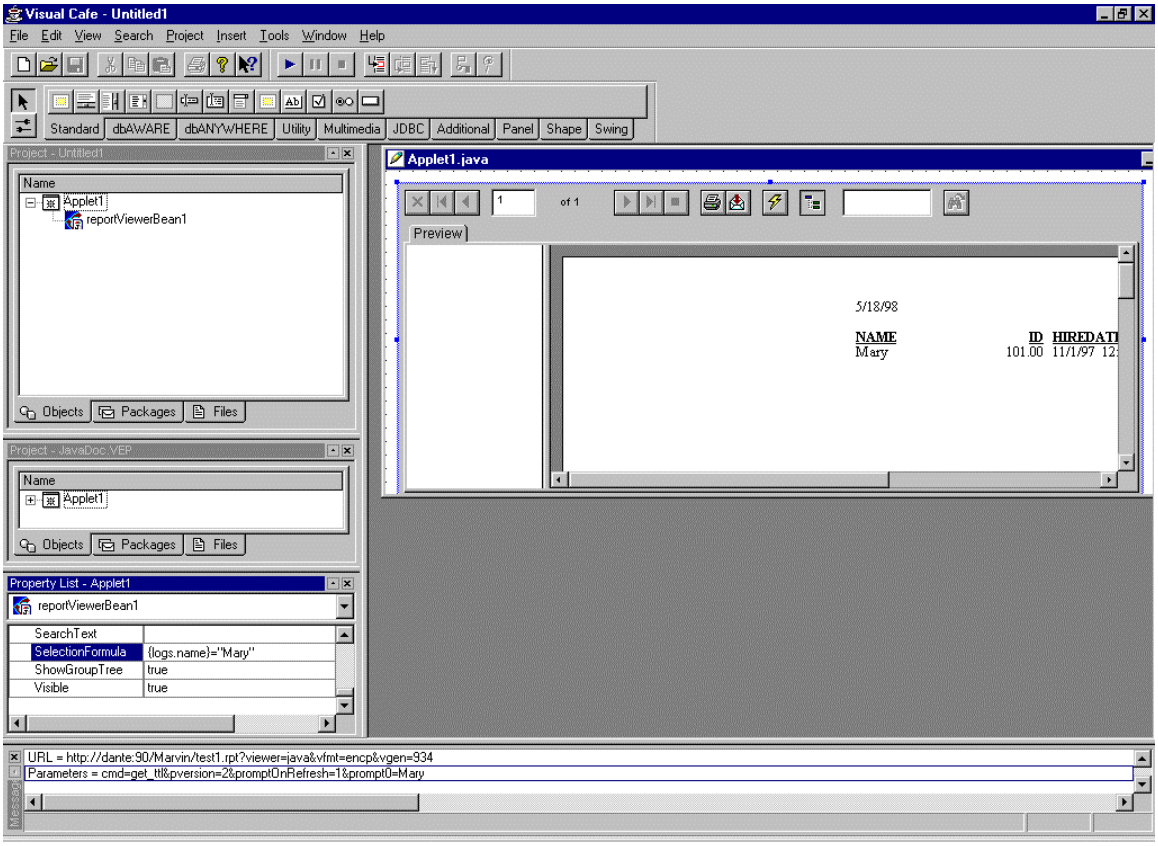
Now that the component is in your Project you can use the Property list to make desired adjustments to the Bean. The Smart Viewer Bean properties may have one or more of the characteristics listed below:

- **read** - You can get the current value.
- **write** - You can set the value.
- **bound** - You can get a notification every time the value changes.
- **constrained** - You can veto a request to change the value.
- **Property Description** - Read(r), Write(w), Bound(b), Constrained(c)
- **busy** - Boolean. True if the ReportViewer is currently processing a command initiated by user action, method call, or property change. r,
- **canCloseCurrentView** - Boolean. True if the current views tab can be closed. The initial ("Preview") tab can not be closed. Refer to method closeCurrentView. r, b
- **canDrillDown** - Boolean. True if drill down views can be opened. Normally, clicking on a hidden group in the group tree (indicated by a magnifying glass icon next to the group name) or double-clicking on a chart, map, or group section in the page will open a drill down view. r, w, b, c
- **currentMessage** - string. The message currently displayed in the status area of the toolbar. r, b
- **currentPageNumber** - int. The number of the page currently being viewed. r, b
- **currentTip** - string. The "tool tip" currently displayed in the status area of the toolbar. Tool tips temporarily override any message in the status area. r, b
- **currentViewName** - string. The name of the view whose tab is selected. r, b
- **exportingPossible** - Boolean. False if exporting is not possible because the user has denied the bean permission to write to the local disk. r
- **hasExportButton** - Boolean. True if the Export button can be made visible in the toolbar. If exporting is not possible (refer to property exportingPossible), requests to set this property to true will be vetoed. r, w, b, c
- **hasGroupTree** - Boolean. If set to True then the GroupTree toggle button is made visible in the toolbar and the GroupTree can be displayed (refer to property showGroupTree). r, w, b, c
- **hasPrintButton** - Boolean. If True then the Print button will be visible in the toolbar. If printing is not possible (refer to property printingPossible) then requests to set this property to True will be vetoed. r, w, b, c
- **hasRefreshButton** - Boolean. If True then the Refresh button is visible in the toolbar. r, w, b, c
- **hasSelectExpert** - Boolean. If True then the SelectExpert button is visible in the toolbar. r, w, b, c
- **hasToolBar** - Boolean. If True then the toolbar is visible. r, w, b, c

- **hasTextSearchControls** - Boolean. If true then the Text Search field and the Find Next button are visible in the toolbar. r, w, b, c
- **language** - string. Contains the 2 letter international Standard code for the language to be used for the user interface. Languages currently supported are: English—en, French—fr, German—de, Japanese—ja, Italian—it, Spanish —es, Portuguese—pt. r, w, b, c
- **lastPageNumber** - int. Indicates to the highest numbered page currently available in the report. This may or may not be the final page. (refer to property lastPageNumberKnown). r, b
- **lastPageNumberKnown** - Boolean. True if the number of the final page in the report is known. If False then there are more pages in the report than the lastPageNumber property indicates. r, b
- **printingPossible** - Boolean. True if printing is possible. If False then either the Java implementation doesn't support it or the user has denied the bean permission to print. R
- **reportName** - string. The URL of the report to be viewed. For example: http://server_name/report_dir/report.rpt

  Setting this property causes the ReportViewer to request page 1 of the report from the server.r, w, b, c
- **searchText** - string. Contains the text most recently searched for, or the text being typed by the user int to the Text Search field in the toolbar. r, b
- **selectionFormula** - string. The current selection formula to be used for subsequent commands. The formula is expressed in the Crystal Reports formula language. (e.g. {Table.Field}=”Test”) Setting this property closes all views except the initial one (the "Preview" view), discards all information cached for the report, and re-requests the current page of the report. r, w, b, c
- **showGroupTree** - Boolean. If True and the hasGroupTree property is True then the GroupTree will be visible. r, w, b, c

In the following example a selection formula is added in the properties box in a Visual Café project:



Please note that the Selection Formula must correspond to the {Table.Field}=”Value” format.

Changing Report Properties at Runtime

As we’ve seen, the report properties in the Bean can be changed by hard-coding values into the properties window. However, it is more likely in a development environment that you would like to change a portion of these values at runtime. To accomplish this in your development environment you will need to add choice and buttons to your form and then script events that pass variables to specific properties within the Report Viewer Bean.

The following is a sample of Java code that sends a string variable to a selection formula. The selection variable is then enacted with a mouse-clicked event on a button.

```
String s = new String(choice1.getSelectedItem());

try{
reportViewerBean1.setSelectionFormula("{Logs.name}=\""+s+"\
"");

              }catch(java.beans.PropertyVetoException
e){System.err.println("Property Veto Exception");}
```

The first portion of the code:

```
“String s = new String(choice1.getSelectedItem());”
```

creates a string variable for the items that are stored in the choice box called “s”.

The second section of the code:

```
"try{
reportViewerBean1.setSelectionFormula("{Logs.name}=\""+s+"\
"");"
```

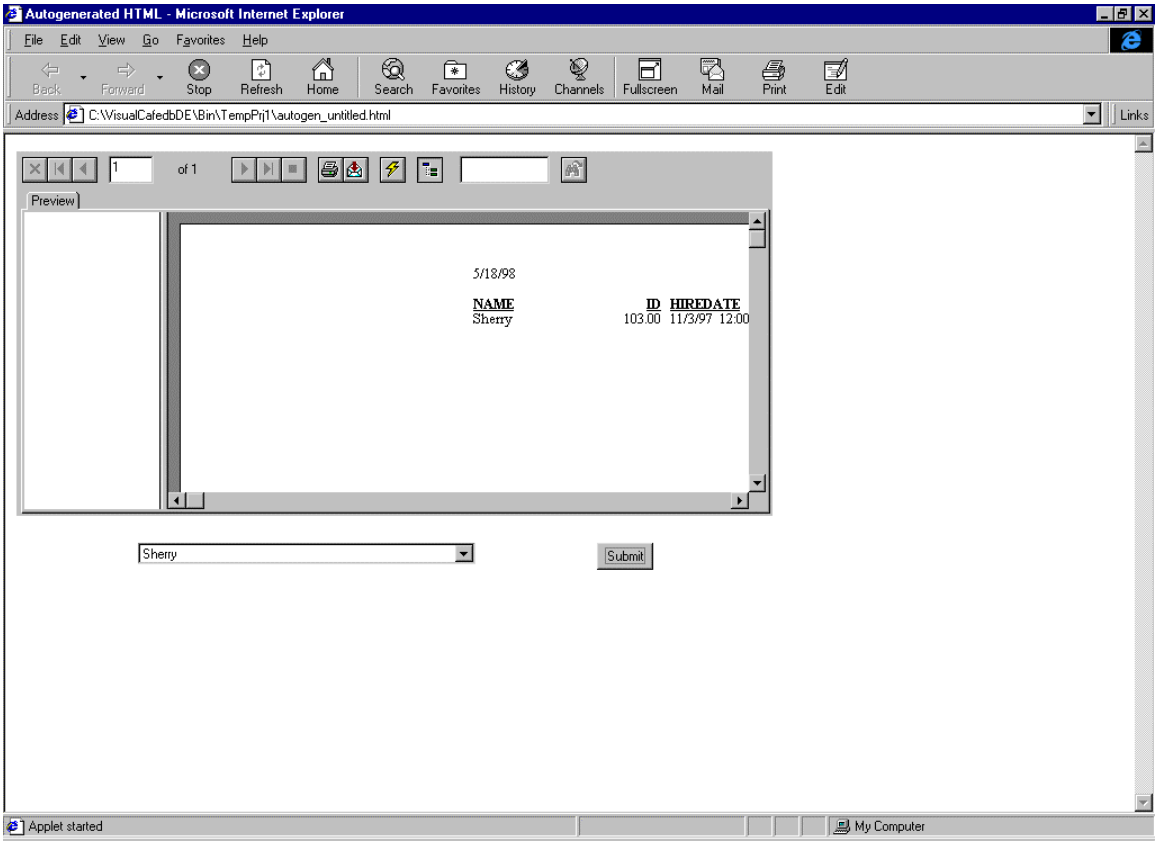
passes the string variable “s” to the selection formula in the report viewer bean. Note that since the selection formula requires double quotes for the value you will need to use the \""+s+"\" syntax.

The third section of the code:

```
catch(java.beans.PropertyVetoException
e){System.err.println("Property Veto Exception");}
```

A PropertyVetoException is thrown when a proposed change to a property represents an unacceptable value.

Now that the mouse click event has been coded the Bean can run in a browser and have different values passed to it:



It is feasible that your user would like to get all the record-set back once they begin to submit items from the choice box. To do this you will need to set up a value in the choices box that flags a null value in your code (you can use an

If…Then or Else statement to do this). A null value is equal to “”. When this is submitted to the Bean, all the record-set should be returned.

Notes regarding the use of the Report Viewer Bean in Jbuilder

If you are applying the Report Viewer Bean as a component in a Jbuilder project you must first initialize the bean in the Jbuilder code. To do this go to the JbInit() section of your applet code and initialize the Report Viewer Bean with the following line:

```
ReportViewerBean1.start();
```

ReportViewerBean1 simply refers to the name that you have given the Java Bean. After adding this line to the jbInit() section of your Applet code it should look like the following:

```
private void jbInit() throws Exception {
    this.setSize(400,300);
    this.getContentPane().add(reportViewerBean1,
BorderLayout.CENTER);
    reportViewerBean1.start();
```

Once this process has been completed you will be able to change the properties and add events for the Bean using the same code that we discussed in our Visual Café example.

Contacting Seagate Software for Technical Support

Along with this document, and the *Seagate Crystal Reports User’s Guide*, we recommend that you visit our Technical Support web site for further resources and sample files. For further assistance, visit us at the websites below.

Technical Support web site:

http://support.seagatesoftware.com/homepage/

Answers By Email Support:

http://support.seagatesoftware.com/support/answers.asp

Phone and Fax Support:

Tel: (604) 669-8379 Fax: (604) 681-7163