


# 5

## Handling BIRT data

## Slide 23 Handling BIRT data



## Slide 24 Overview

A presentation slide titled "Overview" with a grey header. The main content area is white and contains two bullet points. The footer is black and contains the "ActuateOne" logo and a copyright notice.

Overview

- > About Handling BIRT data
- > What you learn

ActuateOne

© 2010 by Actuate Corporation

### About Handling BIRT data

The JSAPI can use basic data analysis tools to work on data in a table or chart.

### What you learn

In this chapter, you learn to:

- Filter and sort data in tables using the `actuate.data` subclasses.
- Filter data in charts using the `actuate.data.filter` class.
- Enable JSAPI data controls using BIRT chart interactivity.

## Slide 25 Using the Actuate JavaScript API data classes

### Using the Actuate JavaScript API data classes

- > Accessing the JSAPI data classes
- > Using a filter
- > Using a sorter



© 2010 by Actuate Corporation

### Accessing the JSAPI data classes

BIRT report items that display data accept filters and sorting on that data. The `actuate.data` subclasses are loaded when the viewer loads, so an additional `actuate.load()` call is not required to use them.

### Using a filter

Apply a data filter to data or elements in a report, such as charts or tables, to extract specific subsets of data. For example, the callback function to view only the rows in a table with the CITY value of NYC, uses code similar to the following function:

```
function filterCity(pagecontents){
    var myTable = pagecontents.getTableByBookmark("mytable");

    // create filter
    var filters = new Array( );
    var city_filter = new actuate.data.Filter("CITY", actuate.data.Filter.EQ,
        "NYC");
    filters.push(city_filter);

    // add filter to table and refresh
    myTable.setFilters(filters);
    myTable.submit(nextStepCallback);
}
```

In this example, the operator constant `actuate.data.filter.EQ` indicates an equals (=) operator.

## Using a sorter

A data sorter can sort rows in a report table or cross tab based on a specific data column. For example, to sort the rows in a table in descending order by quantity ordered, use code similar to the following function as the callback function after submitting the viewer:

```
function sortTable( ){
    var btable = this.getViewer( ).getCurrentPageContent( ).
        getTableByBookmark("mytable");

    // create sorter
    var sorter = new actuate.data.Sorter("QUANTITYORDERED", false);
    var sorters = new Array( );
    sorters.push(sorter);


    // add sorter to table and refresh
    btable.setSorters(sorters);
    btable.submit( );
}
```

The first line of `sortTable( )` uses the `this` keyword to access the container that contains this code. Use the `this` keyword when you embed code in a report or report element. This keyword doesn't provide reliable access to the current viewer when called directly from a web page.

## Slide 26 Using the Actuate JavaScript API in chart interactive features

Using the Actuate JavaScript API in chart interactive features

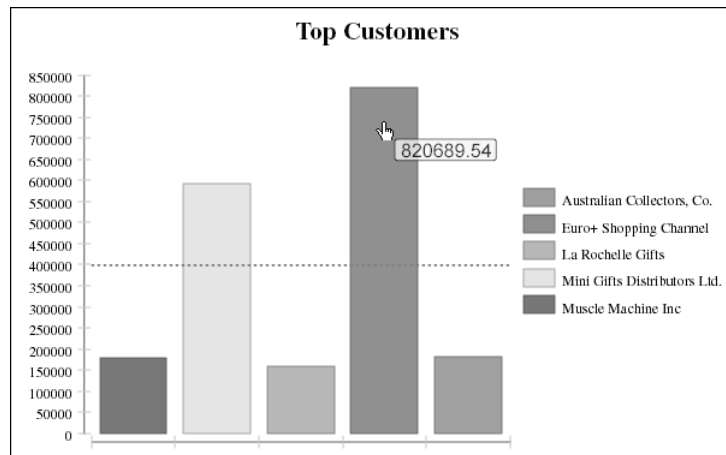
- > Understanding chart interactive features
- > Accessing chart interactive features
- > Scripting chart interactivity

© 2010 by Actuate Corporation

### Understanding chart interactive features

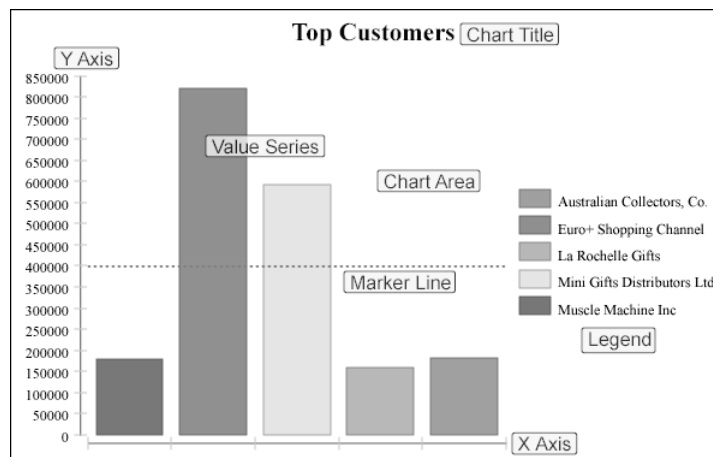
BIRT reports support adding interactive features to a chart to enhance the behavior of a chart in the viewer. The interactive chart features are available through the chart builder. You can implement Actuate JavaScript API functions within interactive features.

An interactive chart feature supports a response to an event, such as the report user choosing an item or moving the mouse pointer over an item. The response can trigger an action, such as opening a web page, drilling to a detail report, or changing the appearance of the chart. For example, you can use a tooltip to display the series total when a user places the mouse over a bar in a bar chart, as shown in Figure 5-1.



**Figure 5-1** Chart showing a tooltip

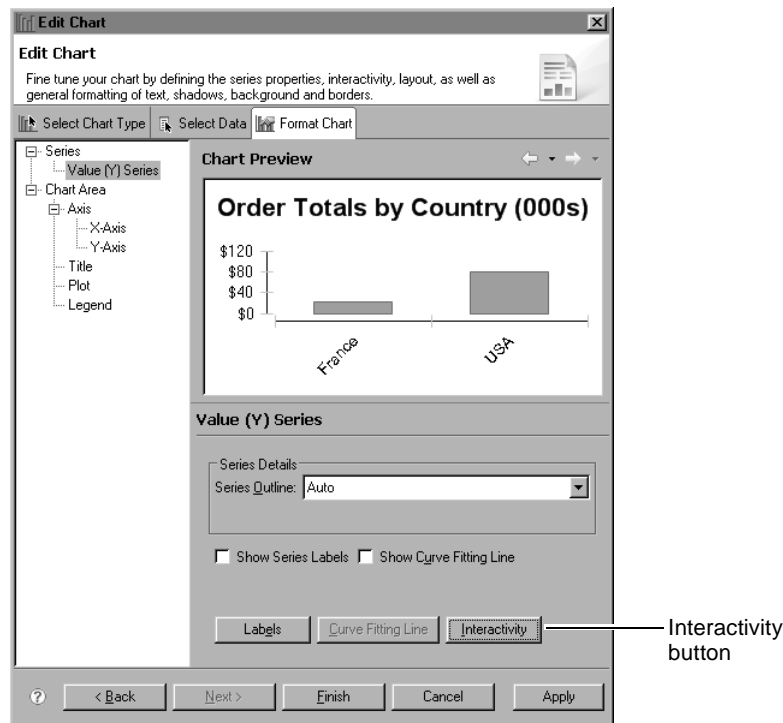
You can add an interactive feature to a value series, the chart area, a legend, marker lines, the x- and y-axis, or a title. Figure 5-2 identifies these elements.



**Figure 5-2** Elements selectable for chart interactivity

## Accessing chart interactive features

To access the chart interactive features, choose Format Chart in the chart builder, then select a chart element to make interactive. Figure 5-3 shows the location of the Interactivity button for a value series.

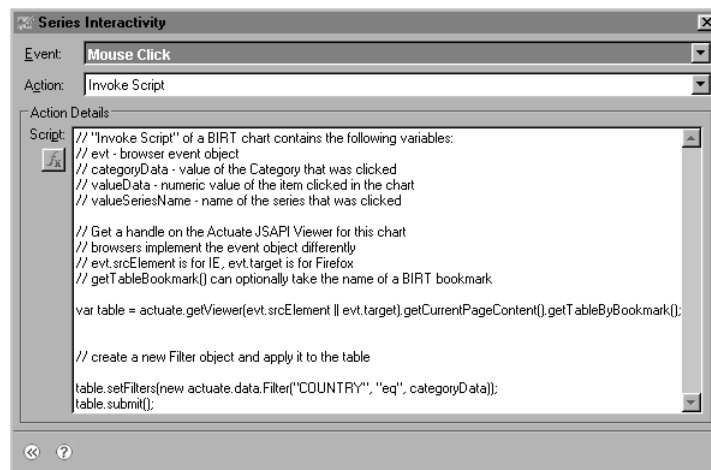


**Figure 5-3** Accessing interactivity for a value series

The location of the Interactivity button varies by chart element. Click the Interactivity button to display the interactivity editor.

## Scripting chart interactivity

Figure 5-4 shows the interactivity editor. The Action Details window displays a script that runs when the user clicks an item in the series.



**Figure 5-4** Interactivity editor

The script adds a filter to the table that displays below the chart. The filter restricts the data by the selected element. The code performs the following three tasks to handle this interactivity:

- Obtains the bookmark for the table when the event occurs

```
var table = actuate.getViewer(evt.srcElement ||
    evt.target).getContentPane().getTableByBookmark( );
```



The event is taken from the Invoke Script action of a BIRT chart. Set the Invoke Script action in the second field of the interactivity editor. The Invoke Script action contains the following variables:

- evt: browser event object
- categoryData: value of the selected category
- valueData: numeric value of the selected item
- valueSeriesName: name of the selected series

The code above uses `getViewer` and the `evt` object to obtain a handle for the viewer when an event occurs. The Firefox and Internet Explorer browsers implement the event differently. For Firefox, `evt.target` contains the name of the viewer object. For Internet Explorer, `evt.srcElement` contains the name of the viewer object.

The `getCurrentPageContent.getTableByBookmark()` function retrieves the table object for the first table in the viewer. To target a different table, use a specific table bookmark as the input parameter for `getTableByBookmark()`.

- Performs an operation on the target

```
table.setFilters(new actuate.data.Filter("COUNTRY", "eq", categoryData));
```

This code example creates a new filter using the `actuate.data.Filter` constructor. The constructor takes three arguments:

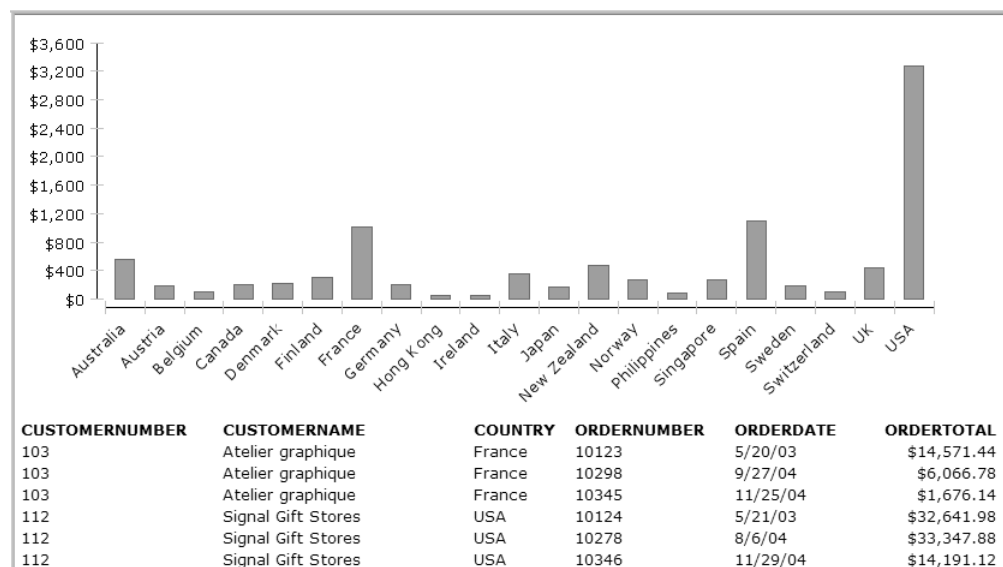
- column name: The column name is the name of the series. In this case, the y-axis is a list of countries, so a mouse click filters the table according to the COUNTRY column.
- operator: `eq` is the reserved operator for equal to.
- value: The value of the `categoryData` object generated by the event, which is a country. The filter returns rows with a COUNTRY value that matches the value selected by the user. The target must have COUNTRY as a data series.

- Submits the action for processing

```
table.submit();
```

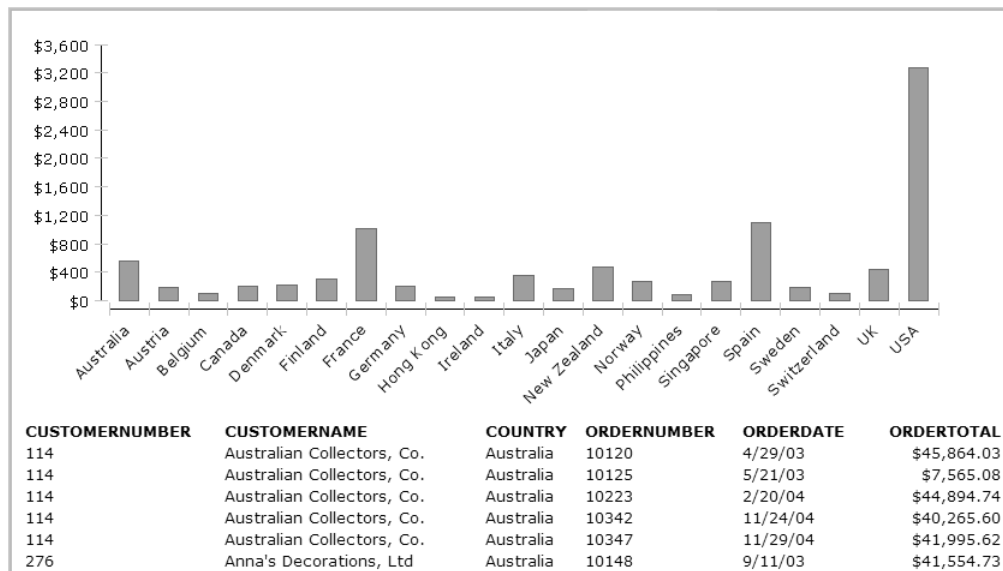
The Actuate JavaScript API processes operations asynchronously. Actions are performed when you call `submit()`.

Figure 5-5 shows the chart before interaction.



**Figure 5-5** An interactive chart and table before any user action

When the user selects the bar for Australia in the value series, the table is filtered for Australia, as shown in Figure 5-6.



**Figure 5-6** An interactive chart and table after the user selects Australia

## Slide 27   Exercise 4

Exercise: Using HTML buttons to apply filters a chart

You learn to

- > write JSAPI scripts that apply filters to a chart
- > write JSAPI scripts that remove filters from a chart

## Exercise 4 Using HTML buttons to apply filters to a chart

**Overview** In this exercise, you add multiple HTML buttons to an existing report that each add a filter for a different product line. An additional HTML button removes all filters to display data for all product lines.

**What you learn** In this exercise, you learn how to:

- Write JSAPI scripts that apply filters to a chart.
- Write JSAPI scripts that remove filters from a chart.

**What you do** In this exercise, you perform the following tasks:

- Add a filter button to the report.
- Add HTML buttons for the remaining product lines.
- Add the final HTML button to the report.
- Test the report.

**What you use** To complete this exercise, you use:

- Classic Models sample database
- ButtonFilterChartStart.rptdesign

### Task 1: Add a filter button to the report

In this task, you preview the report and add the first HTML button that implements event handlers to apply a filter to the chart.

- 1 In Navigator, expand the JSAPI project, and open ButtonFilterChartStart.rptdesign.
- 2 Save the report design as ButtonFilterChart.rptdesign.
- 3 Choose Run>View Report>In Web Viewer to view the report, as shown in Figure 5-7.

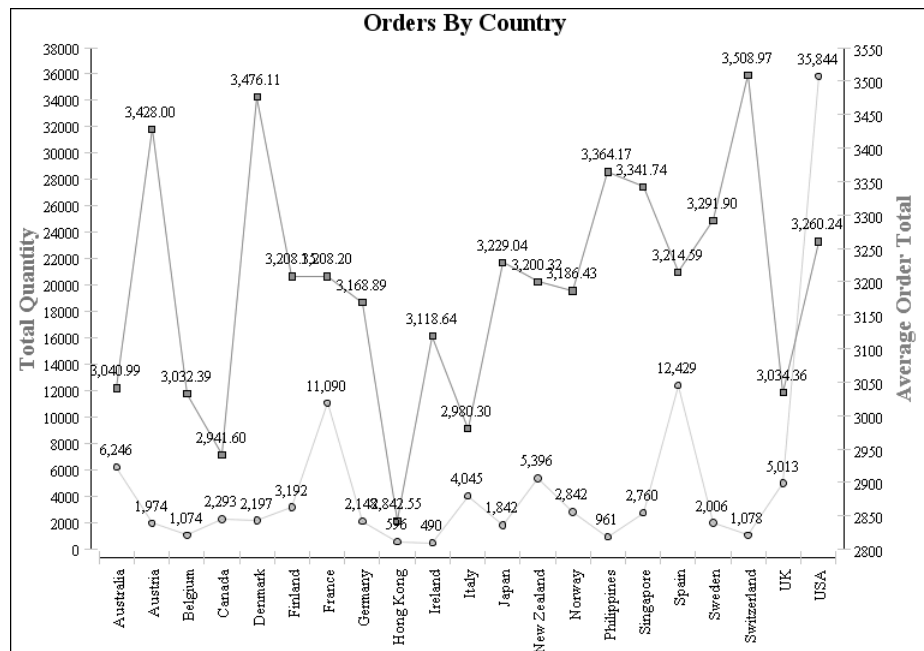
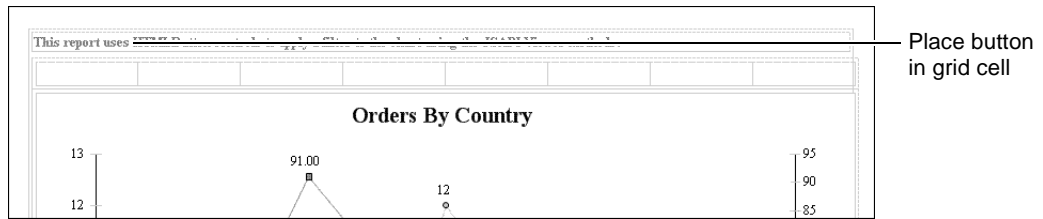


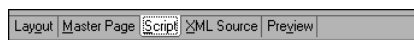
Figure 5-7 Previewing the line chart report

- 4 Close the Web Viewer window. From Palette, drag an HTML button element into the first grid cell, as shown in Figure 5-8.



**Figure 5-8** Viewing the location of the first HTML button

- 5 In HTML Button, type the following text for Value:  
Classic Cars
- 6 Choose OK. If a warning appears displaying a message about adding functionality, choose OK.
- 7 Choose Script, as shown in Figure 5-9, to access the script editor.



**Figure 5-9** Choosing Script

- 8 In New Event Function, select onclick.
- 9 In the function body for the onclick event handler, copy the code for the Classic Cars button shown in Listing 5-1.

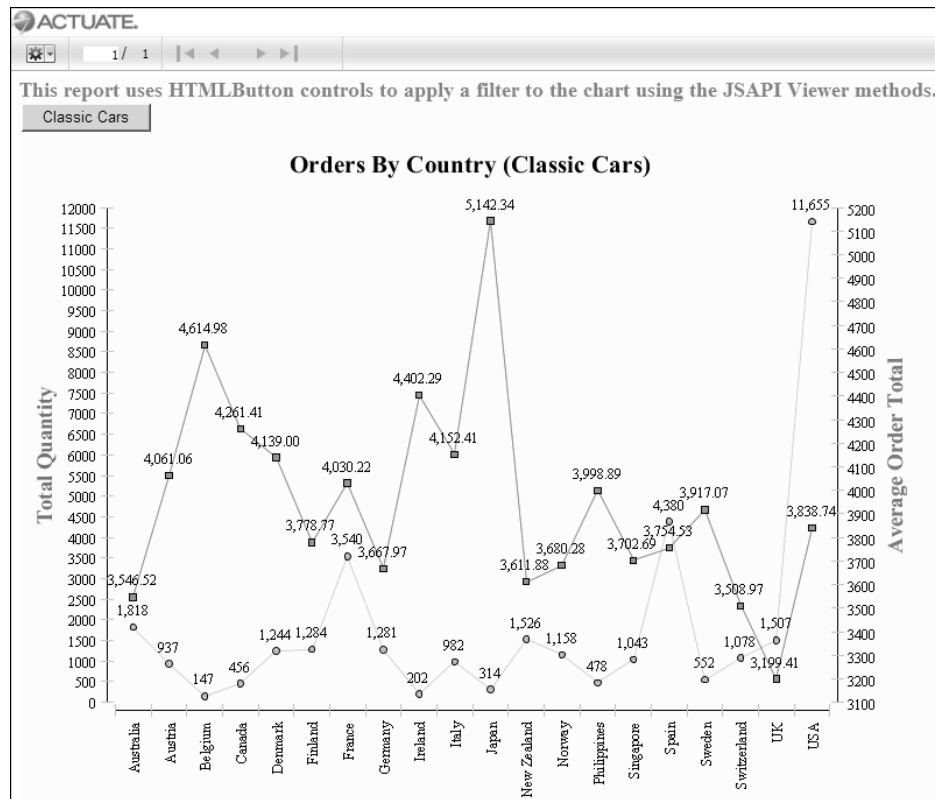
**Listing 5-1** Classic Cars JSAPI code

```
var bviewer = this.getViewer( );
var bpagecontents = bviewer.getCurrentPageContent( );
var bchart = bpagecontents.getChartByBookmark("ChartBookmark");

if (bchart == null) return; // unable to get handle to chart in case where
                           // chart becomes hidden
var filter = new actuate.data.Filter("PRODUCTLINE", "=", "Classic Cars");
var filters = new Array( );
filters.push(filter);

bchart.setFilters(filters);
bchart.setChartTitle("Orders By Country (Classic Cars)");
bchart.submit( );
```

- 10 Preview the report in the web viewer by choosing Run➤View Report➤In Web Viewer. Click on the Classic Cars HTML button that appears in the top left corner and the filtered chart appears as shown in Figure 5-10.



**Figure 5-10** Previewing the report with the Classic Cars data

11 Close Actuate Viewer.

## Task 2: Add HTML buttons for the remaining product lines

In this task, you add six HTML buttons, one for each of the remaining product lines.

- 1 Choose Layout to return to the layout editor. From Palette, drag an HTML button element into the next available grid cell. In HTML Button, for Value, type:

Motorcycles

Choose OK. If a warning appears displaying a message about adding functionality, choose OK. The HTML button appears in the layout editor.

- 2 Choose the Script tab. In New Event Function, select onclick.
- 3 Copy the code for the Classic Cars button shown in Listing 5-1, and paste the code in the function body of the onclick event handler.
- 4 In Script, replace Classic Cars with Motorcycles in the following two lines:

```
var filter = new actuate.data.Filter("PRODUCTLINE", "=", "Classic Cars");
and:
```

```
bchart.setChartTitle("Orders By Country (Classic Cars)");
```

The edited event handler appears as shown in Listing 5-2.

### Listing 5-2 Motorcycles JSAPI code

```
var bvviewer = this.getViewer( );
var bpagecontents = bvviewer.getCurrentPageContent( );
var bchart = bpagecontents.getChartByBookmark("ChartBookmark");
```

```

if (bchart == null) return;// unable to get handle to chart in case where
    chart becomes hidden
var filter = new actuate.data.Filter("PRODUCTLINE", "=", "Motorcycles");
var filters = new Array( );
filters.push(filter);

bchart.setFilters(filters);
bchart.setChartTitle("Orders By Country (Motorcycles)");
bchart.submit( );

```

5 Repeat steps 1 through 4 of this task for the following five buttons and data values:

- Planes
- Ships
- Trains
- Trucks and Buses
- Vintage Cars

When complete, the report layout appears as shown in Figure 5-11.

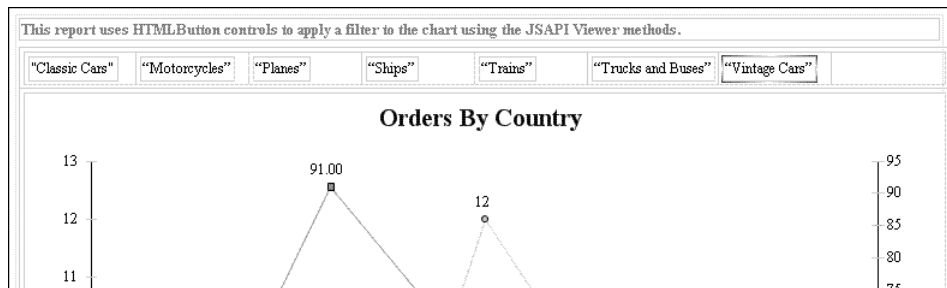


Figure 5-11 Viewing filter buttons in Layout

### Task 3: Add the final HTML button to the report

In this task, you add the final filter button to the report. This button is different from the previous buttons, in that it will clear any filter and display summary data for all product lines.

- 1 From Palette, drag an HTML button element into the remaining grid cell. In HTML Button, in Value, type:  
Show All  
Choose OK.
- 2 Choose Script. In New Event Function, select onclick.
- 3 In the function body for the onclick event handler, copy the code for the Show All button shown in Listing 5-3.

#### Listing 5-3 JSAPI code to remove filters from chart

```

var bviewer = this.getViewer( );
var bpagecontents = bviewer.getCurrentPageContent( );
var bchart = bpagecontents.getChartByBookmark("ChartBookmark");

if (bchart == null) return;// unable to get handle to chart in case where
    chart becomes hidden

bchart.clearFilters("PRODUCTLINE");
bchart.setChartTitle("Orders By Country");
bchart.submit( );

```

- 4 Choose Layout. The Show All button appears as shown in Figure 5-12.

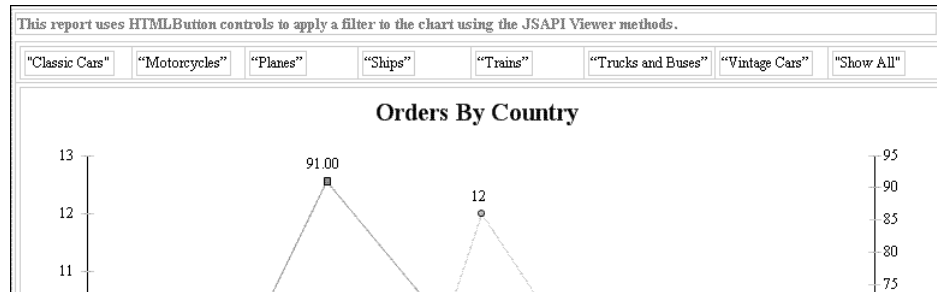


Figure 5-12 Viewing all buttons in Layout

## Task 4: Test the report

In this task, you test the report by selecting the various product line buttons.

- 1 Choose Run→View Report→In Web Viewer.
- 2 Choose the Planes HTML button. The chart changes, as shown in Figure 5-13.

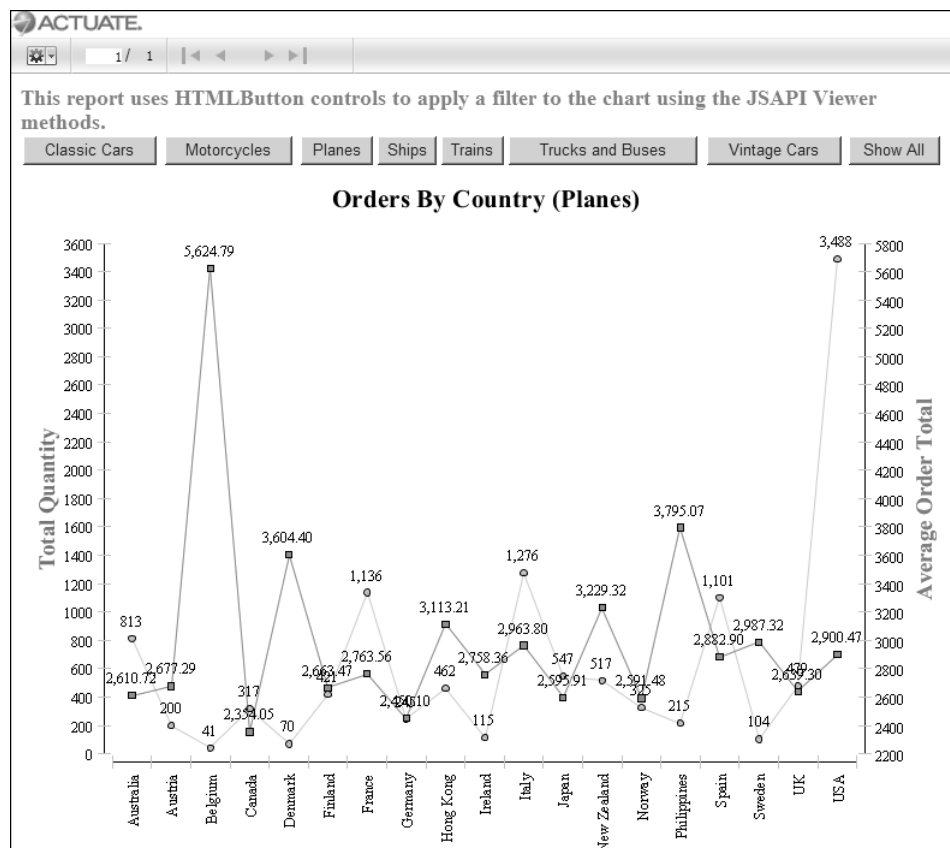


Figure 5-13 Viewing the report after selecting Planes HTML button



## Slide 28 Exercise 5

### Exercise: Adding an chart interactive filter to a BIRT Report

You learn to

- > write a script for chart interactive features
- > use the JSAPI `actuate.data.filter` class to change other charts in the report

## Exercise 5 Adding an interactive chart filter to a BIRT report

**Overview** In this exercise, you add an interactive chart control to a BIRT report design that implements a filter on the other charts in the report design.

**What you learn** In this exercise, you learn how to:

- Write a script for chart interactive features.
- Use the JSAPI `actuate.data.filter` class to change other charts in the report.

**What you do** In this exercise, you perform the following tasks:

- Adding bookmarks
- Adding a filter script to chart interactivity

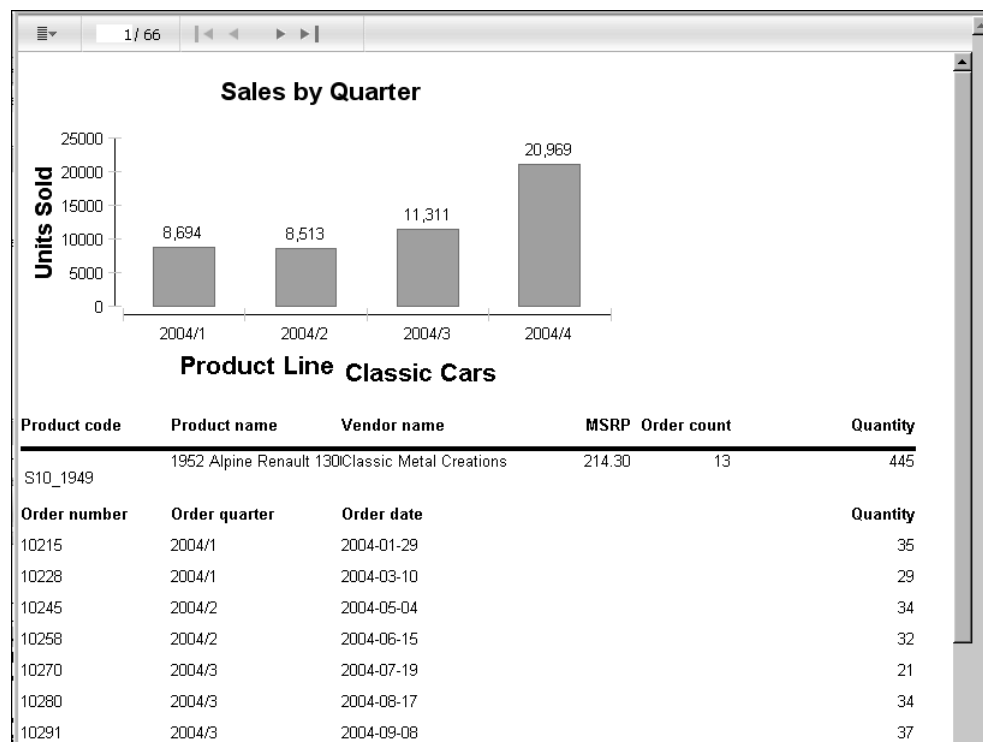
**What you use** To complete this exercise, you use the following resources:

- `Start\ChartandTable.rptdesign`

### Task 1: Adding bookmarks

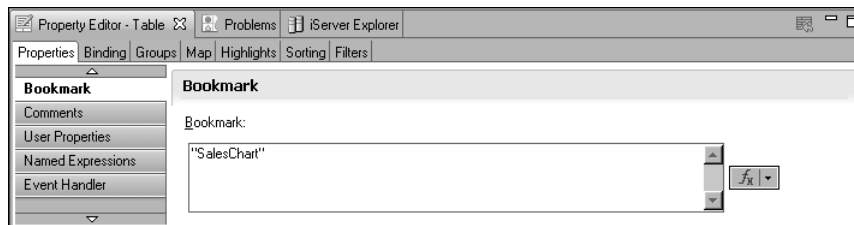
In this task, you review a BIRT report design and add a bookmark to the chart and table.

- 1 In Navigator, expand the JSAPI project.
- 2 Open `ChartandTable.rptdesign`. Save the file in the JSAPI project as `InteractiveChartandTable.rptdesign`.
- 3 Choose **Run**→**View Report**→**In Web Viewer** to view the report, as shown in Figure 5-14.



**Figure 5-14** Previewing the report

- 4 Choose **Layout** to return to the layout editor.
- 5 Select the chart entitled **Sales by Quarter**. In the property editor, open **Properties**→**Bookmark**. Set the bookmark value to "SalesChart" as shown in Figure 5-15.



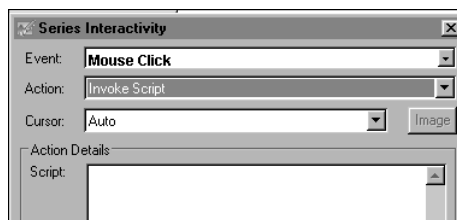
**Figure 5-15** Setting the chart bookmark property

- 6 Repeating the process of step 5, set the bookmark for the table entitled Product Line the bookmark value to "ProductTable".

## Task 2: Adding a filter script to chart interactivity

In this task, you add a filter script to the Sales by Quarter chart to affect the other charts.

- 1 Double-click on the Sales by Quarter chart. In Edit Chart, select Format Chart>Series>Value (Y) Series. Then choose Interactivity.
- 2 On series interactivity, select Mouse Click for event, and Invoke Script for action, as shown in Figure 5-16.



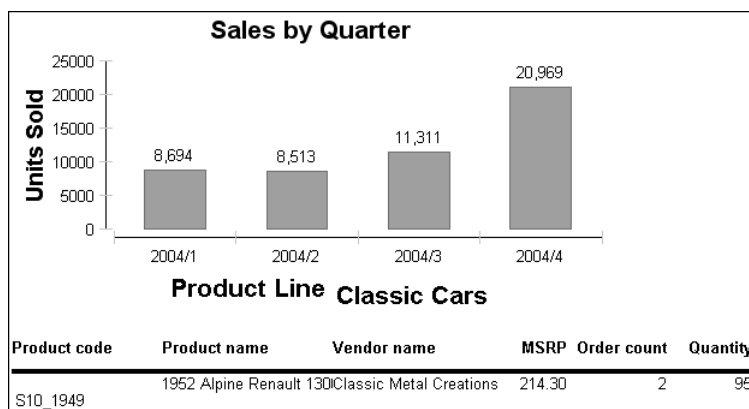
**Figure 5-16** Interactivity settings for Invoking a Script on Mouse Click

- 3 In the Script: text box, add the following code:

```
var atable = actuate.getViewer(evt.srcElement || evt.target).
    getCurrentPageContent( ).getTableByBookmark("ProductTable" );
atable.setFilters(new actuate.data.Filter("QUANTITYORDERED", "gt",
    valueData/200));
atable.submit();
```

Choose Finish.

- 4 View the report by choosing Run>View Report>In Web Viewer.
- 5 Select a bar in the table to activate the filter, as shown in Figure 5-17.



**Figure 5-17** Filtered product table after selecting a chart value

## Slide 29 Summary

### Summary

- > You learned to
  - > Filter and Sort data in tables using the `actuate.data` subclasses.
  - > Filter data in charts using the `actuate.data.filter` class.
  - > Enable JSAPI data controls using BIRT chart interactivity.
- > Next step  
Controlling the User Interface

ActuateOne

© 2010 by Actuate Corporation

- Quiz**
- 1 Which JSAPI data tools apply to which BIRT report items?
  - 2 Which variable contained in the BIRT chart interactivity Invoke Script handler contains the numeric value for a selected item?
    - a. `valueData`
    - b. `categoryData`
    - c. `evt`
    - d. `valueSeriesName`