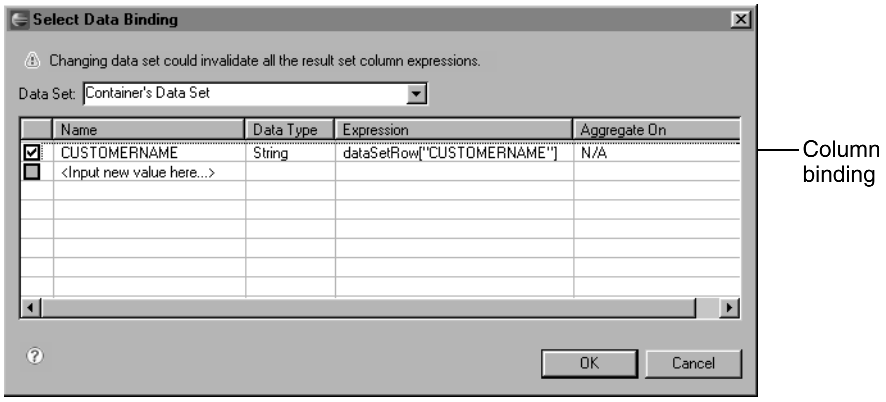
Chapter

8

Binding Data

The data set or data sets you create return the data you want to use in a report. Before you can use or display data set data in a report, you must first create the necessary data bindings. As the first tutorial demonstrated, to display the data in a report, you simply drag data set fields from Data Explorer to a table in the layout editor.

You might also remember in that same tutorial that when you dragged a data set field to the layout editor, BIRT Report Designer first displayed a dialog called Select Data Binding. This dialog, shown in Figure 8-1, shows the data binding that BIRT Report Designer creates each time you insert a data set field. This data binding, called a column binding, defines an expression that specifies what data to display. The column binding also defines a name that report elements use to access data.



Changing data set could invalidate all the result set column expressions.

Data Set: Container's Data Set

	Name	Data Type	Expression	Aggregate On
✓	CUSTOMERNAME	String	dataSetRow["CUSTOMERNAME"]	N/A
☐	<Input new value here...>			

Figure 8-1 Select Data Binding showing a column binding

Understanding column bindings

For each piece of data to display in a report, there must be a column binding. For this discussion, note that data refers to dynamic data, and not static text that you type for a label. Dynamic data is data from a data set, or data that is calculated from a function or a formula. The data is dynamic, because the values are not fixed at design time.

The default column binding, which BIRT Report Designer creates for a data set field, uses the data set field name as the name of the column binding. You can change the name of the column binding, and you might want to do so if the data set field name is not descriptive. In Figure 8-1, the expression defined for the column binding is dataSetRow["CUSTOMERNAME"]. This expression indicates that the column binding accesses data from the data set field, CUSTOMERNAME. In the layout editor, the column-binding name appears within square brackets ([ ]) in the report, as shown in Figure 8-2.

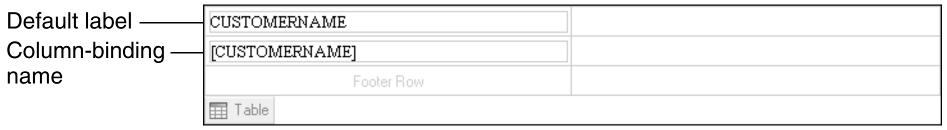


Figure 8-2 Data element displaying the column-binding name

Column bindings form an intermediate layer between data set data and report elements—such as chart, data, dynamic text, and image elements—that display data. Figure 8-3 illustrates this concept. Report elements can access data only through column bindings.

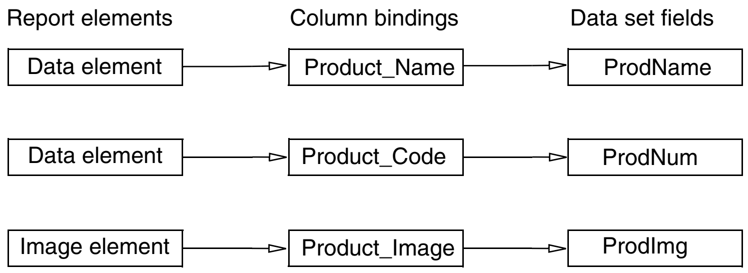
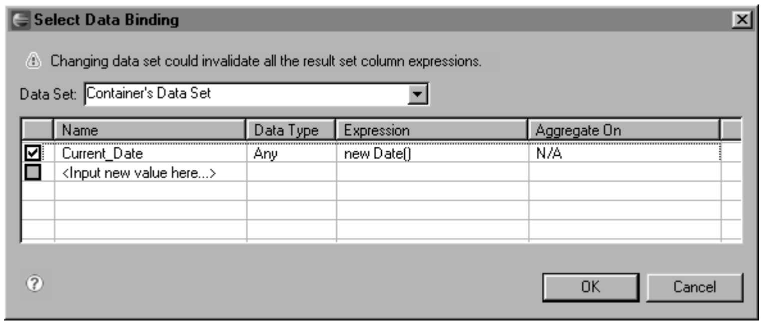


Figure 8-3 Report elements access data set data through column bindings

The preceding examples show column bindings that access data set data. Column bindings can also access data derived from functions or user-defined formulas. For example, you can use a data element to display the current date derived from the JavaScript Date object. You would create a column binding that used the following expression:

```
new Date()
```

Figure 8-4 shows this column binding defined in Select Data Binding.



Changing data set could invalidate all the result set column expressions.

Data Set: Container's Data Set

	Name	Data Type	Expression	Aggregate On
☑	Current_Date	Any	new Date()	N/A
☐	<Input new value here...>			

Figure 8-4 User-defined column binding

Descriptive names

One of the benefits of using column bindings is that you control the names used in the report. Instead of displaying data set field names, which are often not descriptive enough, or formulas, which can be long, you can specify short and descriptive names. If you share report designs with other report developers, descriptive names make that design much easier to understand. Modifying and maintaining a report design that has user-friendly names is easier. When deciding what names to use, consider that, in the layout editor, elements display up to 20 characters.

Dynamic updates of calculated data

Another advantage of column bindings becomes apparent when you work with calculated data. When a report needs to display a series of related calculated data, column bindings enable you to create and update calculations easily. For example, assume a report contains the following four data elements:

- The first data element uses column binding, Total_National_Sales, which defines the following expression to calculate the sum of all order amounts:

  ```
  Total.sum(row["Order_Amount"])
  ```

- The second data element uses column binding, Total_State_Sales, which defines the following expression to calculate the sum of order amounts in each state:

  ```
  Total.sum(row["Order_Amount"])
  ```

  This expression is the same as the previous expression, but both return different results, because they perform the aggregate calculation over different sets of rows.

- The third data element uses column binding, Percent_State_Sales, which uses the previous column bindings to calculate a state's total sales as a percentage of the overall sales. The expression is:

  ```
  row["Total_State_Sales"]/row["Total_National_Sales"] * 100
  ```

Without column bindings, the calculation would have to be more complicated:

```
Total.sum(row["Order_Amount"], null, "state")/
   Total.sum(row["Order_Amount"], null, "overall") * 100
```

- The fourth data element uses column binding, National_Profit, which contains this expression:

  ```
  row["Total_National_Sales"] - row["Total_Costs"]
  ```

  Without column bindings, this calculation would also be more complicated:

  ```
  Total.sum(row["Order_Amount"], null, "overall") -
     Total.sum(row["Item_Cost"])
  ```

You have already seen how column bindings make expressions shorter and more readable. Now, consider the case where you need to update one calculation that is used by other calculations. Suppose you need to change how the total national sales is calculated, from

```
Total.sum(row["Order_Amount"])
```

to

```
Total.sum(row["Order_Amount"]) -
   Total.sum(row["Order_Amount"]) * 0.08
```

Because the third and fourth data elements use the total national sales calculation in their calculations, without column bindings, you would have to manually edit those calculations as well. Without column bindings, you would have to revise the expression for the fourth element as follows:

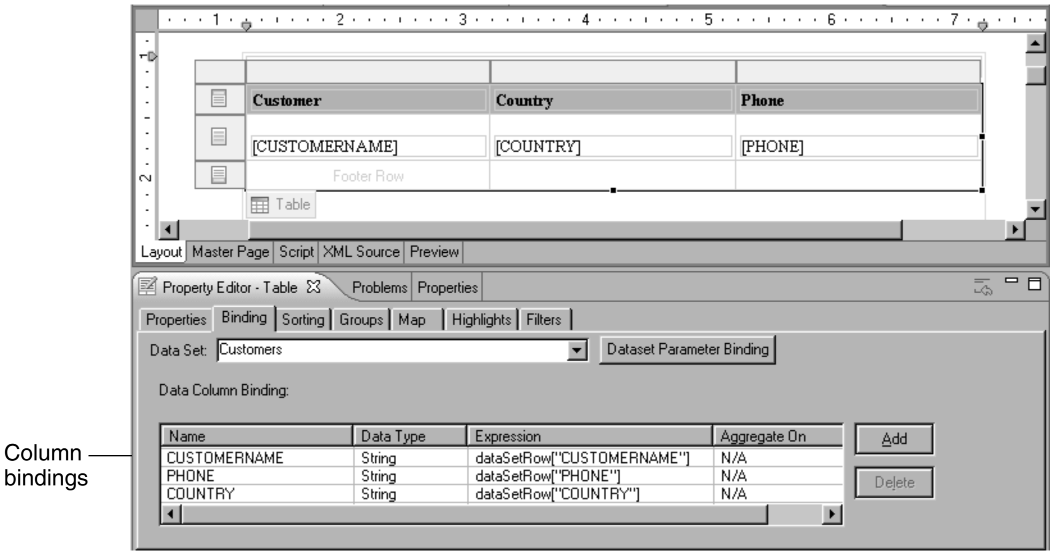
```
(Total.sum(row["Order_Amount"]) -
   Total.sum(row["Order_Amount"]) * 0.08) -
   Total.sum(row["Item_Cost"])
```

By using column bindings, any change to the first calculation automatically applies to the third and fourth calculations. Instead of modifying three expressions, you modify only one. Your work is faster and less error-prone.

Creating column bindings

As discussed previously, when you drag a data set field from Data Explorer to a table in the layout editor, BIRT Report Designer creates the column binding. The table that contains the data set fields displays the list of column bindings when you select the table and choose the Binding tab on Property Editor. Figure 8-5 shows an example of a table's binding information.

In Figure 8-5, the CUSTOMERNAME, PHONE, and COUNTRY column bindings were generated by BIRT Report Designer when the data set fields, CUSTOMERNAME, PHONE, and COUNTRY were inserted in the table.



Data Set: Customers

Data Column Binding:

Name	Data Type	Expression	Aggregate On
CUSTOMERNAME	String	dataSetRow["CUSTOMERNAME"]	N/A
PHONE	String	dataSetRow["PHONE"]	N/A
COUNTRY	String	dataSetRow["COUNTRY"]	N/A

Figure 8-5 Binding information for a selected table

When you insert a dynamic text, text, image, or data element from the palette, you must manually create the column binding if you want the element to display dynamic data. If the information to display is static—for example, a specific image stored in a file system, or literal text—then column binding is not applicable.

How to create a column binding

This procedure shows an example of creating a column binding for a data element that was added to the report shown in Figure 8-5.

1 Drag a data element from the palette and drop it in the table. Select Data Binding displays all the column bindings defined previously for elements in the table.

2 Create a new column binding:

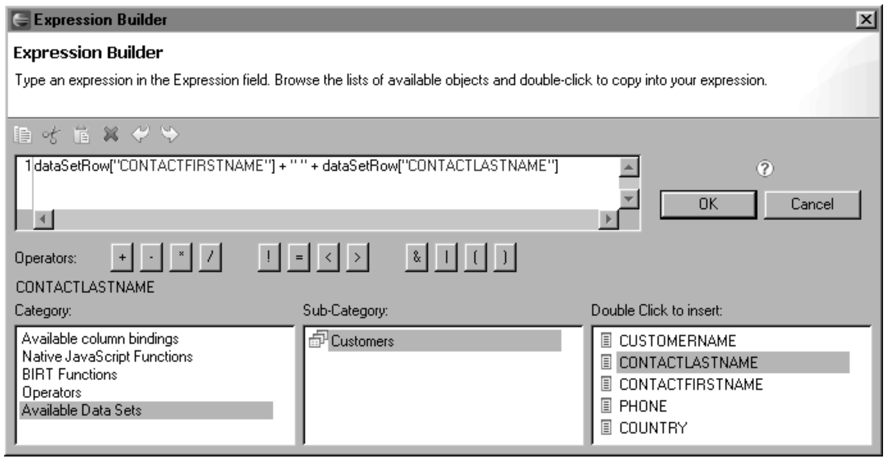
   1 Select the first empty row and, in Name, specify a unique name for the column binding.

   2 In Data Type, select a data type appropriate for the data returned by the expression you specify next. If you are not sure what the data type is, use the default type, Any.

   3 In Expression, specify the expression that indicates the data to return, using one of the following methods:

      - Type the expression directly in the Expression field.
      - If you need help constructing the expression, click in the empty field, then choose the ellipsis button to launch Expression Builder.

Figure 8-6 shows an expression in Expression Builder that combines the values of two data set fields selected from the Customers data set. Choose OK when you finish constructing the expression.

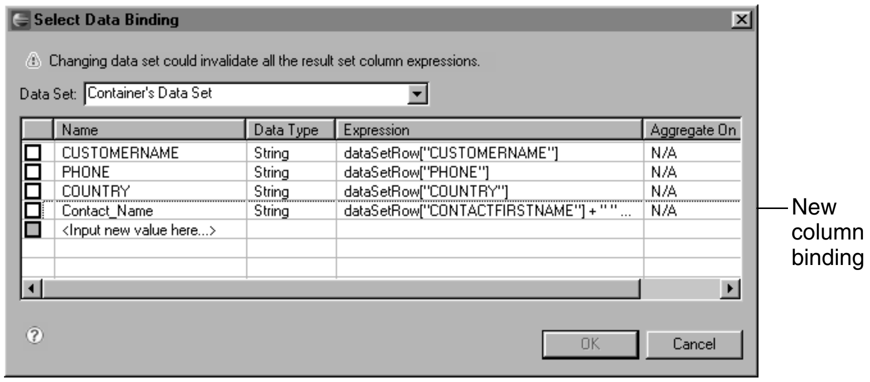


```
dataSetRow["CONTACTFIRSTNAME"] + " " + dataSetRow["CONTACTLASTNAME"]
```

Figure 8-6 Expression Builder showing a column-binding expression

4 In Aggregate On, use the default value, N/A. The Aggregate On option applies only to aggregate expressions that perform calculations over a specified set of data rows.

Figure 8-7 shows an example of a completed column binding.


Select Data Binding
Changing data set could invalidate all the result set column expressions.
Data Set: Container's Data Set
OK
Cancel
New column binding

	Name	Data Type	Expression	Aggregate On
☐	CUSTOMERNAME	String	dataSetRow["CUSTOMERNAME"]	N/A
☐	PHONE	String	dataSetRow["PHONE"]	N/A
☐	COUNTRY	String	dataSetRow["COUNTRY"]	N/A
☐	Contact_Name	String	dataSetRow["CONTACTFIRSTNAME"] + " "...	N/A
	<Input new value here...>			

Figure 8-7 Example of a user-defined column binding

3 Select the column binding you created by clicking the check box next to the column-binding name, then choose OK. The data element uses the selected column binding. When you preview the report, the data element displays the data defined in the column binding.

Editing and deleting column bindings

Be careful when editing or deleting column bindings. More than one element can use a column binding, and a column binding can refer to other column bindings. Renaming or deleting a column binding that is used by multiple elements often results in errors in the report design. Consider the following scenario:

BIRT Report Designer creates a column binding named CUSTOMER_COUNTRY at the table level that refers to the data set field, CUSTOMER_COUNTRY, when you drop the field in a table. You create a sort condition to display rows alphabetically by country names. The sort expression uses the CUSTOMER_COUNTRY column binding. Later, you select the data element that uses CUSTOMER _COUNTRY, and you change the column-binding name to COUNTRY. When you run the report, you get an error message, because the sort expression still refers to the CUSTOMER_COUNTRY column binding, which no longer exists. Similarly, if you delete the CUSTOMER_COUNTRY column binding, the same error occurs, because the sort expression now refers to a non-existent column binding.

If you edit the name of a column binding, you need to manually update all other elements that refer to the column binding. In the previous example, you would need to edit the table's Sort Key value to use the renamed column binding. Do not confuse this type of change with changes to a column binding's defined expression. Earlier in this chapter, you saw examples of how a change to a calculated-data expression cascaded to other expressions that used the higher level expression. Remember these guidelines:

- A change to a column binding's expression applies to other column bindings that refer to that column binding.
- A change to a column binding's name does not change other references to that column binding. Column-binding names are case sensitive. COUNTRY and Country are two different names.

More about column-binding expressions

When you write an expression for a column binding, the expression can refer to data set fields, other column bindings, functions, and operators. Until you become familiar with writing expressions, you should use Expression Builder to construct expressions. Expression Builder displays the items—data set fields, column bindings, functions, and so on—that you can use in a column-binding expression.

The items available in Expression Builder change depending on where you define the column binding. For example, if you insert a data element in a table that contains other column bindings, the data element can access those column bindings. If you insert a data element directly on the report page, the data element cannot access column bindings defined for the table or any other report element.

When you select an item in Expression Builder, Expression Builder adds the item to the expression with the proper syntax. When a column-binding expression refers to a data set field, the syntax is:

```
dataSetRow["datasetField"]
```

When a column-binding expression refers to another column binding, the syntax is:

```
row["columnBinding"]
```

If you use Expression Builder to construct column-binding expressions, you do not need to remember what syntax to use. You will find it helpful, though, to understand what each syntax means, because the expression examples that appear throughout the book use both syntaxes.