Selecting Data Using the SqlDataSource Control

You can use the SqlDataSource control to retrieve data from a database with little or no code. The SqlDataSource control can work with any database that has an associated ADO.NET provider configured in the DbProviderFactories section of the configuration, including Microsoft SQL Server, Oracle, ODBC, or OLE DB databases such as Microsoft Access. The database you use will dictate the syntax of the SQL statements you configure the SqlDataSource to use and whether you can use more advanced database features such as stored procedures. However, the data source control operates the same for all databases.

To retrieve data from a database using the SqlDataSource control, you need to set at least the following properties:

- ProviderName Set to the name of the ADO.NET provider that represents the database you are working with. If you are working with Microsoft SQL Server, set the ProviderName property to "System.Data.SqlClient"; if you are working with an Oracle database, set the ProviderName property to "System.Data.OracleClient"; and so on.
- ConnectionString Set to a connection string that works for your database.
- SelectCommand Set to an SQL query or stored procedure that returns data from the database. The query that you set for the SelectCommand property is the same query that you set for the CommandText property of an ADO.NET IDbCommand object when writing ADO.NET data-access code. The actual syntax of the SQL query depends on the schema of your data and which database you are using.

The following sections describe these properties in more detail.

Specifying a Provider Name

You set the ProviderName property to the name of the ADO.NET provider associated with the type of database in which your data is stored. The list of allowable providers is registered in the DbProviderFactories section of the configuration file, either in the Machine.config or Web.config file. By default, the SqlDataSource control uses the System.Data.SqlClient ADO.NET provider, which corresponds to Microsoft SQL Server. Therefore, if you are connecting to a SQL Server database, you do not need to explicitly specify a provider. However, you can also specify the System.Data.OracleClient, System.Data.Odbc, or System.Data.OleDb providers. For more information, see ADO.NET.

Note
Do not set the ProviderName property to the value of an unmanaged ADO provider, such as **SQLOLEDB** or **MSDAORA**.

Specifying a Connection String

You set the ConnectionString property to a connection string used for a specific database. However, setting ConnectionString property of a SqlDataSource control to a specific connection string is not a very maintainable strategy for large sites.

Additionally, the connection string is then stored in plain text in the ASP.NET page. To make your Web application more maintainable and more secure, it is recommended that you store connection strings in the connectionStrings element in the application's configuration file. You can then reference the stored connection string using a connection expression like that in the following example:

```
<asp:SqlDataSource
  ID="SqlDataSource1"
  runat="server"
  ConnectionString="<%$ ConnectionStrings:NorthwindConnectionString %>"
   SelectCommand="SELECT * FROM [Categories]">
</asp:SqlDataSource>
```

For additional security, you can encrypt the contents of the **<connectionStrings>** configuration section. For more information, see Encrypting and Decrypting Configuration Sections.

Specifying the Select Command

You can specify an SQL query for the SqlDataSource control to execute by setting its SelectCommand property. The following example demonstrates an SQL query that retrieves a result set consisting of the last names of all the employees in an `Employees` table:

```
SELECT LastName FROM Employees;
```

The following code example shows how you can set the ConnectionString and SelectCommand properties of a SqlDataSource control to display the Employees data in a GridView control:

C#

```
<%@ Page language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
  <head runat="server">
    <title>ASP.NET Example</title>
</head>
<body>
    <form id="form1" runat="server">
      <asp:SqlDataSource
          id="SqlDataSource1"
          runat="server"
          DataSourceMode="DataReader"
          ConnectionString="<%$ ConnectionStrings:MyNorthwind%>"
          SelectCommand="SELECT LastName FROM Employees">
      </asp:SqlDataSource>

      <asp:ListBox
```

```
            id="ListBox1"
            runat="server"
            DataTextField="LastName"
            DataSourceID="SqlDataSource1">
        </asp:ListBox>

    </form>
  </body>
</html>
```

If the database you are working with supports stored procedures, you can set the SelectCommand property to the name of the stored procedure and the SelectCommandType property StoredProcedure to indicate that the SelectCommand property refers to a stored procedure. The following example demonstrates a simple stored procedure that you can create in SQL Server:

```
CREATE PROCEDURE sp_GetAllEmployees AS
    SELECT * FROM Employees;
GO
```

To configure the SqlDataSource to use this stored procedure, set the SelectCommand text to "sp_GetAllEmployees" and the SelectCommandType property to StoredProcedure.

Most stored procedures use parameters. For more information about using stored procedures with parameters, see Using Parameters with the SqlDataSource Control.

At run time, SqlDataSource control submits the text in the SelectCommand property to the database, and the database returns the result of the query or stored procedure to the SqlDataSource control. Any Web controls that are bound to the data source control display the result set on your ASP.NET page.

Passing Parameters to SQL Statements

Users often interact with data based on parameters that can be resolved or evaluated only at run time. For example, data displayed on an ASP.NET Web page might represent a report for a specific date. If the user selects a different date, the data in the report might also change. Whether the date is changed explicitly by the user or programmatically by the Web application, the SQL query that you submit to the database can be made more flexible and more maintainable if it is a parameterized SQL query, in which elements of the SQL statement are bound to Web application variables and evaluated at run time.

The SqlDataSource control supports parameterized SQL queries by associating parameters you add to the SelectParameters collection with placeholders in the SelectCommand query. Parameter values can be read from another control on the page, from session state, from the user profile, and from other elements. For more information, see Using Parameters with the SqlDataSource Control.

The syntax used for the placeholders varies, depending on the type of your database. If you are working with SQL Server, the parameter name begins with the '@' character, and its name corresponds to the name of the Parameter object in the SelectParameters collection. If you are working with an ODBC or OLE DB database, parameters in a parameterized statement are not named and instead are specified with the placeholder character '?'.

The following example demonstrates how a parameterized SQL query retrieves all the orders in the SQL Server Northwind database, based on the employee ID of the currently logged-in employee.

```
SELECT * FROM Orders WHERE EmployeeID = @empid
```

In this example, the `@empid` expression is the parameter that is evaluated at run time.

The following code example demonstrates a parameterized SQL query which takes the parameter value from another control on the page:

C#

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
  <head runat="server">
    <title>ASP.NET Example</title>
</head>
<body>
    <form id="form1" runat="server">

      <p><asp:dropdownlist
          id="DropDownList1"
          runat="server"
          autopostback="True">
          <asp:listitem selected="True">Sales Representative</asp:listitem>
          <asp:listitem>Sales Manager</asp:listitem>
          <asp:listitem>Vice President, Sales</asp:listitem>
      </asp:dropdownlist></p>

      <asp:sqldatasource
          id="SqlDataSource1"
          runat="server"
          connectionstring="<%$ ConnectionStrings:MyNorthwind%>"
          selectcommand="SELECT LastName FROM Employees WHERE Title = @Title">
          <selectparameters>
              <asp:controlparameter name="Title" controlid="DropDownList1"
propertyname="SelectedValue"/>
          </selectparameters>
      </asp:sqldatasource>

      <p><asp:listbox
          id="ListBox1"
          runat="server"
          datasourceid="SqlDataSource1"
          datatextfield="LastName">
      </asp:listbox></p>

    </form>
  </body>
</html>
```

For more information about using parameters with the SqlDataSource control, see Using Parameters with the SqlDataSource Control. For more general information about using data source parameters, see Using Parameters with Data Source Controls for Filtering.

Specifying How Data Is Returned

The SqlDataSource control's DataSourceMode property determines how data is maintained by the SqlDataSource control. By default, the DataSourceMode property is set to DataSet, which means that the result set returned from the database is stored in server memory by the SqlDataSource control. When the SqlDataSource control retrieves data in DataSet mode, associated data-bound controls such as GridView and DetailsView can offer rich data display capabilities such as automatic sorting and paging.

Alternatively, you can set the DataSourceMode property to DataReader, which means that the result set is not stored in memory. For scenarios where you do not need to keep a result set in memory on the server, use DataReader mode.

The following code example demonstrates how to set the DataSourceMode property of the SqlDataSource control to DataReader for a scenario that requires no sorting, paging, or filtering.

C#

```
<%@ Page language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
  <head runat="server">
    <title>ASP.NET Example</title>
</head>
<body>
    <form id="form1" runat="server">
      <asp:SqlDataSource
          id="SqlDataSource1"
          runat="server"
          DataSourceMode="DataReader"
          ConnectionString="<%$ ConnectionStrings:MyNorthwind%>"
          SelectCommand="SELECT LastName FROM Employees">
      </asp:SqlDataSource>

      <asp:ListBox
          id="ListBox1"
          runat="server"
          DataTextField="LastName"
          DataSourceID="SqlDataSource1">
      </asp:ListBox>

    </form>
  </body>
</html>
```

Adding Custom Processing Using SqlDataSource Control Events

The SqlDataSource control exposes events that you can handle to run your own code before and after the control performs a data retrieval operation.

The SqlDataSource control raises the Selecting event before it calls the Select method to execute the SQL query set in the SelectCommand property. You can handle the Selecting event to examine the SQL query before it is run and to validate

parameters that are contained in the SelectParameters collection or to perform any additional work prior to retrieving data. For example, if you are using a FormParameter with the SqlDataSource control, you might handle the Selecting event to validate the value of the parameter before retrieving data. (The FormParameter takes the value posted in an HTML element and submits it to the database without any validation.) If the value is not acceptable, you can cancel the query by setting the Cancel property of the SqlDataSourceSelectingEventArgs object to **true**.

The SqlDataSource control raises the Selected event after the data has been retrieved. You can handle the Selected event to determine whether an exception was thrown during the database operation or to examine any values returned by the data operation.

Displaying the Data

To display data on an ASP.NET page, you use a data-bound control such as a GridView, DetailsView, or FormView control, or controls such as the ListBox or DropDownList controls. The data-bound control acts as a consumer of the data that the SqlDataSource control retrieves. Set the data-bound control's **DataSourceID** property to the ID of the SqlDataSource control. When the page is rendered, the SqlDataSource control retrieves the data and makes it available to the data-bound control, which in turn displays the data. For more information about data-bound controls and how to use them with data source controls to display data, see Data-Bound Web Server Controls.

The following code example demonstrates how to display the results of the query using a GridView control.

C#

```
<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
  <head runat="server">
    <title>ASP.NET Example</title>
</head>
<body>
    <form id="form1" runat="server">

      <asp:SqlDataSource
          id="SqlDataSource1"
          runat="server"
          DataSourceMode="DataReader"
          ConnectionString="<%$ ConnectionStrings:MyNorthwind%>"
          SelectCommand="SELECT FirstName, LastName, Title FROM Employees">
      </asp:SqlDataSource>

      <asp:GridView
          id="GridView1"
          runat="server"
          DataSourceID="SqlDataSource1">
      </asp:GridView>

    </form>
  </body>
</html>
```

See Also

Concepts

SqlDataSource Web Server Control Overview