SqlDataSource Web Server Control Overview

The SqlDataSource control enables you to use a Web server control to access data that is located in a relational database. This can include Microsoft SQL Server and Oracle databases, as well as OLE DB and ODBC data sources. You can use the SqlDataSource control with data-bound controls such as the GridView, FormView, and DetailsView controls to display and manipulate data on an ASP.NET Web page, using little or no code.

For information about how to configure the SqlDataSource control by using code instead of markup, see the documentation for the SqlDataSource class and for its methods, properties, and events.

This topic contains:

- Background
- Code Examples
- Class Reference

Background

The SqlDataSource control uses ADO.NET classes to interact with any database supported by ADO.NET. This includes Microsoft SQL Server (using the System.Data.SqlClient provider), System.Data.OleDb, System.Data.Odbc, and Oracle (using the System.Data.OracleClient provider). Using a SqlDataSource control allows you to access and manipulate data in an ASP.NET page without using ADO.NET classes directly. You provide a connection string to connect to your database and define the SQL statements or stored procedures that work with your data. At run time, the SqlDataSource control automatically opens the database connection, executes the SQL statement or stored procedure, returns the selected data (if any), and then closes the connection.

Connecting the SqlDataSource Control to a Data Source

When you configure a SqlDataSource control, you set the ProviderName property to the type of database (the default is System.Data.SqlClient) and the ConnectionString property to a connection string that includes information required to connect to the database. The contents of a connection string differ depending on what type of database the data source control is accessing. For example, the SqlDataSource control requires a server name, database (catalog) name, and information about how to authenticate the user when connecting to a SQL Server. For information on valid connection strings, see the **ConnectionString** property topics for the SqlConnection, OracleConnection, OleDbConnection, and OdbcConnection classes.

Instead of setting connection strings at design time as property settings in the SqlDataSource control, you can store them centrally as part of your application's configuration settings using the connectionStrings configuration element. This enables you to manage connection strings independently of your ASP.NET code, including encrypting them using Protected Configuration. The following example shows a connection to the SQL Server Northwind sample database using a connection string stored in the **connectionStrings** configuration element named MyNorthwind.

C# VB

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

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

<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>
```

Issuing Data Commands with the SqlDataSource Control

You can specify up to four commands (SQL queries) for the SqlDataSource control: a SelectCommand, UpdateCommand, DeleteCommand, and an InsertCommand. Each command is a separate property of the data source control. For each command property, you specify a SQL statement for the data source control to execute. If the data source control connects to a database that supports stored procedures, you can specify the name of a stored procedure in place of the SQL statement. If you use an asterisk (*) in the Select command to select all columns, and if you use automatic code generation to perform update or delete operations, make sure that no columns have spaces in their names.

You can create parameterized commands that include placeholders for values to be supplied at run time. The following example shows a typical parameterized SQL Select command:

```
Select CustomerID, CompanyName From Customers Where City = @city
```

You can create parameter objects that specify where the command should get parameter values at run time, such as from another control, from a query string, and so on. Alternatively, you can specify parameter values programmatically. For more information, see Using Parameters with the SqlDataSource Control.

The data source control executes the commands when its corresponding Select, Update, Delete, or Insert method is called. The Select method is called automatically when you call the DataBind method of the page or of a control bound to the data source control. You can also call any of the four methods explicitly when you want the data source control to execute a command. Some controls, such as the GridView control, can call the methods automatically, without requiring that you call the methods or that you explicitly call the DataBind method. For more information, see Selecting Data Using the SqlDataSource Control and Modifying Data using the SqlDataSource Control.

Note

By default, if one of the parameters is **null** when you execute a **Select** command, no data will be returned and no exception will be thrown. You can change this behavior by setting the CancelSelectOnNullParameter property to **false**.

Returning DataSet or DataReader Objects

The SqlDataSource control can return data in two forms: as a DataSet object or as an ADO.NET data reader. You can specify which form to return by setting the data source control's DataSourceMode property. A DataSet object contains all the data in server memory, allowing you to manipulate the data in various ways after retrieving it. A data reader provides a read-only cursor that can fetch individual records. As a rule, you choose to return a dataset if you want to filter, sort, or page through data after retrieving it or if you want to maintain a cache. In contrast, you use a data reader when you simply want to return the data and are using a control on the page to display that data. For example, using a data reader is ideal for returning data that you want to display in a ListBox, DropDownList, or GridView control where a list of results is displayed in a read-only format.

Caching with the SqlDataSource Control

The SqlDataSource control can cache data that it has retrieved, which can enhance the performance of your applications by avoiding expensive queries. Caching is practical in almost any situation where the data is not highly volatile and the cached results are small enough to avoid utilizing too much system memory.

Caching is not enabled by default. You can enable it by setting EnableCaching to **true**. The caching mechanism is based on time; you can set the CacheDuration property to the number of seconds to cache data. The data source control maintains a separate cache entry for each combination of connection, select command, select parameters, and cache settings.

The SqlDataSource control can also take advantage of the cache dependency feature of SQL Server (if available in your version of SQL Server). This feature allows you to specify that the data in the cache is maintained until SQL Server reports a change in the specified table. This type of caching allows you to improve the performance of data access in your Web applications, because you can minimize data retrieval to only those times when it is necessary to get refreshed data.

For more information, see Caching Data with the SqlDataSource Control.

Filtering with the SqlDataSource Control

If you have enabled caching for the SqlDataSource control and have specified a dataset as the format for data returned by a Select query, you can also filter the data without re-running the query. The SqlDataSource control supports a FilterExpression property that allows you to specify selection criteria that are applied to the data maintained by the data source control. You can also parameterize the filter expression by creating special FilterParameters objects that provide values at run time to the filter expression.

Sorting with the SqlDataSource Control

The SqlDataSource control supports sort requests from the bound control when the DataSourceMode is set to DataSet.

Back to top

Code Examples

Selecting Data Using the SqlDataSource Control

Using Parameters with the SqlDataSource Control

Modifying Data using the SqlDataSource Control

Caching Data with the SqlDataSource Control

How to: Enable Filtering for the SqlDataSource Control

Back to top

Class Reference

The following table lists the key classes that relate to the SqlDataSource control.

Member	Description
SqlDataSource	The main class for the control.

Back to top

See Also

Concepts

Data Source Web Server Controls

Other Resources

LinqDataSource Web Server Control Overview

ObjectDataSource Web Server Control Overview

Is this page helpful?

Yes No

Dev centers

Windows

Office

Visual Studio

Microsoft Azure

More...

Learning resources

Microsoft Virtual Academy

Channel 9

MSDN Magazine

Community

Forums

Blogs

Codeplex

Support

Self support

Programs

BizSpark (for startups)

Microsoft Imagine (for students)

United States (English)

NewsletterPrivacy & cookiesTerms of use Trademarks