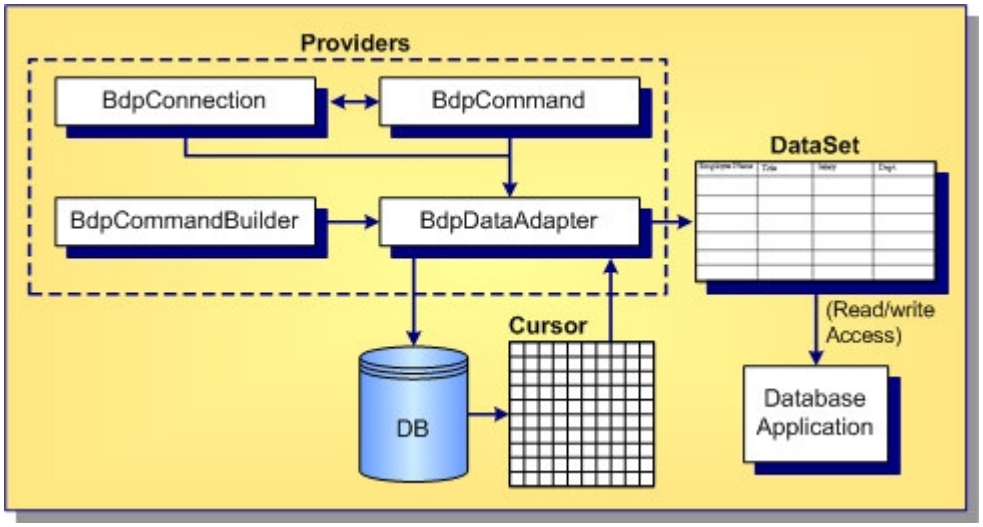
BDP.NET Component Designers

Almost all distributed applications revolve around reading and updating information in databases. Different applications you develop using ADO.NET have different requirements for working with data. For instance, you might develop a simple application that displays data on a form. Or, you might develop an application that provides a way to share data information with another company. In any case, you need to have an understanding of certain fundamental concepts about the data approach in ADO.NET.

Using these designers, you can work efficiently to access, expose, and edit data through database server-specific schema objects like tables, views, and indexes. These designers allow you to use these schema objects to connect to a variety of popular databases, and perform database operations in a consistent and reliable way.

This topic includes:

Component Designer Relationships



The major elements of the database component designers include:

- The **Connection Editor** to define a live connection to a data source
- The **Command Text Editor**to construct command text for command components
- The **Configure Data Adapter** to set up commands for a data adapter
- The **Stored Procedure Dialog box** to view and specify values for Input or InputOutput parameters for use with command components
- The **Generate Dataset** to build custom datasets
- The **Data Explorer** to browse database server-specific schema objects and use drag-and-drop techniques to automatically populate data from a data source to your Delphi for .NET project

Connections Editor

The **Connections Editor** manages connection strings and database-specific connection options. Using the **Connections Editor** you can add, remove, delete, rename, and test database connections. Changes to the connection information are saved into the `BdpConnections.xml` file, where they are accessed whenever you need to create a new connection object. Once you have chosen a particular connection, the **Connections Editor** generates the connection string and any connection options, then assigns them to the ConnectionString and ConnectionOptions properties, respectively.

Display the **Connections Editor** dialog box by dragging the BdpConnection component from the **Tool Palette** onto the form, and then clicking the component designer verb at the bottom of the **Object Inspector**.

Command Text Editor

The **Command Text Editor** can be used to construct the command text for command components that have a CommandText property. A multi-line editing control in the editor lets you manually edit the command or build the command text by selecting tables and columns. Display the **Command Text Editor** dialog box by dragging a BdpCommand component from the **Tool Palette** onto the form, and clicking the designer verb at the bottom of the **Object Inspector**.

The **Command Text Editor** is a simplified version of a SQL builder capable of generating SQL for a single table. The database objects are filtered by the SchemaName property set in the BdpCommand and only tables that are part of that schema are used. If there is no SchemaName listed, all of the available objects for the current login user are listed. The QuoteObjects setting for the ConnectionOptions property determines whether the objects are quoted with the database-specific quote character or not. This is important, for instance, when retrieving tables from databases that allow table names to include spaces.

To populate the Tables and Columns list boxes with items and build SQL statements, you must have defined a live BdpConnection . Otherwise, data cannot be retrieved. The **Command Text Editor** allows you to choose table and column names from a list of available tables and columns. Using this information, the editor generates a SQL statement. To generate the SQL, the editor uses an instance of the BdpCommandBuilder . When you request optimized SQL, the editor uses index information to generate the WHERE clause for SELECT, UPDATE, and DELETE statements; otherwise, non-BLOB columns and searchable columns form the WHERE clause.

When the SQL is generated, the BdpCommand . CommandText property is set to the generated SQL statement.

Stored Procedure Dialog Box

The **Stored Procedure** dialog box is used to view and enter Input and InputOutput parameters for a stored procedure and to execute the stored procedure. Display the **Stored Procedure** dialog box by dragging a BdpCommand component from the **Tool Palette** onto the form, setting the CommandType property for the BdpCommand component to **StoredProcedure**, and clicking the **Command Text Editor** designer verb at the bottom of the **Object Inspector**.

The **Stored Procedure** dialog box lets you select a stored procedure from a list of available stored procedures, which is determined by the BdpConnection specified in the Connection property for the BdpCommand component. When you select a stored procedure, the dialog box displays the parameters associated with the stored procedure, and the parameter metadata for the selected parameter. You can specify values for Input or InputOutput parameters and execute the stored procedure. If the stored procedure returns results, such as Output parameters, InputOutput parameters, return values, cursor(s) returned, they are all populated into a DataGrid in the bottom of the dialog box when the stored procedure is executed. After the CommandText , Parameters , and ParameterCount properties are are all set for the BdpCommand , the stored procedure can be executed at runtime by making a single call to ExecuteReader or ExecuteNonQuery .

Generate DataSets

The **Generate Dataset** designer is used to build a DataSet. Using this tool results in strong typing, cleaner code, and the ability to use code completion. A DataSet is first derived from the base DataSet class and then uses information in an XML Schema file (an `.xsd` file) to generate a new class. Information from the schema (tables, columns, and so on) is generated and compiled into this new dataset class as a set of first-class objects and properties. Display this dialog box by dragging a BdpDataAdapter component from the **Tool Palette** onto the form, and clicking the component designer verb at the bottom of the **Object Inspector**. If this component is not displayed, choose **Component** ▸ **Installed .NET Components** to add it to the **Tool Palette**.

Configure Data Adapter

The **Configure Data Adapter** designer is used to generate SELECT, INSERT, UPDATE, and DELETE SQL statements. After successful SQL generation, the **Configure Data Adapter** designer creates new BdpCommand objects and adds them to the BdpDataAdapter SelectCommand , DeleteCommand , InsertCommand , and UpdateCommand properties.

After successful SQL SELECT generation, you can preview data and generate a new DataSet. You can also use an existing DataSet to populate a new DataTable. If you create a new DataSet, it will be added automatically to the designer host. You can also generate Typed DataSets.

Data Adapters are an integral part of the ADO.NET managed providers. Essentially, Adapters are used to exchange data between a data source and a dataset. This means reading data from a database into a DataSet, and then writing changed data from the DataSet back to the database. A Data Adapter can move data between any source and a DataSet. Display the **Configure Data Adapter** dialog box by dragging a BdpDataAdapter component from the **Tool Palette** onto the form, and clicking the component designer verb at the bottom of the **Object Inspector**.

Data Explorer

The **Data Explorer** is a hierarchical database browser and editing tool. The **Data Explorer** is integrated into the IDE and can also be run as a standalone executable. To access the **Data Explorer** within the IDE, choose **View** ▸ **Data Explorer** . Use the context menus in the **Data Explorer** to perform the following tasks:

- Manage database connections—add a new connection, modify, delete, or rename your existing connections
- Browse database structure and data—expand and open provider nodes to browse database server-specific schema objects including tables, views, stored procedure definitions, and indexes
- Add and modify tables—specify the data structure for a new table, or add or remove columns, and alter column information for an existing table
- View and test stored procedure parameters—specify values for Input or InputOutput parameters and execute the selected stored procedure
- Migrate data—migrate table schema and data of one or more tables from one provider to another
- Drag-and-drop schema objects onto forms to simplify application development—drag tables or stored procedures onto your application form for the .NET Framework to add connection components and automatically generate connection strings

The **Data Explorer** provides connectivity to several industry-standard databases, and can be extended to connect to other popular databases. The **Data Explorer** uses the ISQLDataSource interface to get a list of available providers, database connections, and schema objects that are supported by different providers. The list of available providers is persisted in the `BdpDataSources.xml` file, and the available connections are persisted in the `BdpConnections.xml` file. Once you have chosen a provider the ISQLMetadata interface is used to retrieve metadata and display a read-only tree view of database objects. The current implementation provides a list of tables, views, and stored procedures for all BDP.NET-supported databases.

The **Data Explorer** lets you create new tables, alter or drop existing tables, migrate data from multiple tables from one provider to another, and copy and paste individual tables across BDP-supported databases. For all these operations, the **Data Explorer** calls into the ISQLSchemaCreate implementation of the provider.

Additionally, the **Data Explorer** can be used to drag data from a data source to any Delphi 2005 project for

the .NET framework. Dragging a table onto a form adds BdpConnection and BdpDataAdapter components to your application and automatically configures the BdpDataAdapter for the given table. Dragging a stored procedure onto a form adds BdpConnection and BdpCommand components to your application, and sets the CommandType property of the BdpCommand object to StoredProcedure.

Related Information

- ADO.NET Overview
- Borland Data Provider for .NET
- BDP.NET Data Types
- Building a Windows Forms Database Application
- Using the Command Text Designer
- Using the Connection Editor Designer
- Using the Data Adapter Designer
- Using the Data Adapter Preview
- Using the Generate Dataset Designer
- Migrating Data Between Databases
- Creating Table Mappings