

GETTING STARTED WITH WINDOWS AND MAC DEVELOPMENT

Connecting to Data

E-Learning Series Course Book

Lesson 6

Embarcadero Technologies

© Copyright 2012 Embarcadero Technologies, Inc. All Rights Reserved.

Americas Headquarters
100 California Street, 12th Floor
San Francisco, California 94111

EMEA Headquarters
York House
18 York Road
Maidenhead, Berkshire
SL6 1SF, United Kingdom

Asia-Pacific Headquarters
L7. 313 La Trobe Street
Melbourne VIC 3000
Australia

Lesson 6 – Connecting to Data

Version: 0.3

Presented: June 6, 2012

Last Updated: June 11, 2012

Prepared by: David Intersimone “David I”, Embarcadero Technologies

© Copyright 2012 Embarcadero Technologies, Inc. All Rights Reserved.

davidi@embarcadero.com

<http://blogs.embarcadero.com/davidi/>

Contents

| | |
|--|-------------------------------------|
| Lesson 6 – Connecting to Data | 2 |
| Introduction..... | 3 |
| Databases..... | Error! Bookmark not defined. |
| Writing Database Applications..... | 4 |
| Database Components included in RAD Studio | 4 |
| dbExpress | 7 |
| Supported Databases and Versions for XE2 | 8 |
| Dbxdrivers.ini..... | 9 |
| Dbxconnections.ini..... | 9 |
| Database Explorer..... | 10 |
| LiveBindings..... | 13 |
| LiveBindings Overview | 13 |
| LiveBindings Usage | 14 |
| LiveBindings Visual Components | 14 |
| BindingsLists..... | 15 |
| LiveBinding Dialog Box..... | 16 |
| The Binding Expression Editor | 17 |
| The New DB Link Dialog Box | 18 |
| LiveBinding Methods Dialog Box | 19 |
| LiveBindings Output Converters Dialog Box..... | 20 |
| LiveBindings Columns Editor | 21 |
| Creating LiveBindings..... | 22 |
| Using LiveBinding to Create a FireMonkey HD Application Without Code | 22 |

E-Learning Series: Getting Started with Windows and Mac Development

| | |
|---|-------------------------------------|
| Building FireMonkey HD SQL Master-Detail Database Applications | Error! Bookmark not defined. |
| Using InterBase Express to Create a FireMonkey HD Application Without Code | Error! Bookmark not defined. |
| Deploying Database Applications..... | 27 |
| Summary, Looking Forward, To Do Items, Resources, Q&A and the Quiz | 29 |
| To Do Items | 29 |
| Links to Additional Resources..... | 30 |
| Q&A:..... | 30 |
| Self Check Quiz | 30 |
| Answers to the Self Check Quiz: | 31 |

Introduction

The world is awash in data. We are creating, storing, retrieving, indexing, searching and reporting with more data than ever before. Estimated

Almost every business application works with data. FireMonkey includes built-in component based, high performance native access to MS SQL Server, Oracle, IBM DB2, MySQL, Sybase, Informix, InterBase and Firebird databases. You can also work with local database tables (Paradox and dBase) and XML data.

RAD Studio includes the following database access technologies and components that can be used in your Windows and Mac applications:

- dbExpress
- InterBase Express (IBX)
- ClientDataSet

To connect your Windows and Mac User Interface (UI) to your data you can visualize live data using LiveBindings. LiveBindings allow you to connect any type of data source to any UI or graphic objects in HD or 3D. FireMonkey gives you unprecedented freedom to express, communicate and interact with data in powerful new ways only limited by your imagination.

In lesson 6 you will learn how to create Windows and Mac applications that can connect to databases, query and process the data. You'll also learn how you can use SQL and the LiveBindings technology to connect your FireMonkey applications to your data.



Connecting Your Windows and Mac Applications to Data

RAD Studio provides many ways to work with data and databases in your Windows and Mac applications. You can also work with XML files containing data. Let's take a look at the ways you can connect to data with RAD Studio.

Writing Database Applications

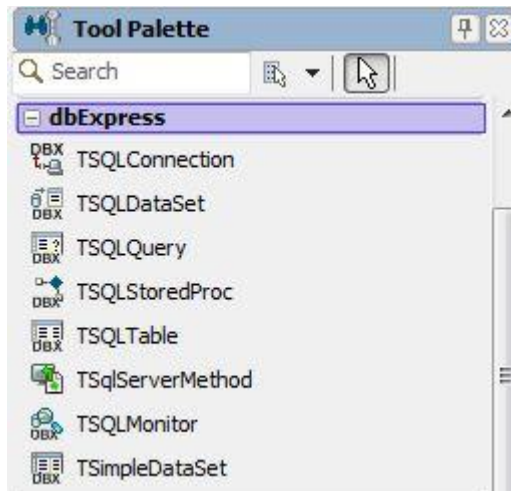
You can create advanced database applications using tools to connect to SQL servers and databases such as Oracle, Sybase, InterBase, MySQL, MS-SQL, FireBird, Informix, and DB2 while providing transparent data sharing between applications. RAD Studio includes database components for connecting, SQL operations, transaction processing, stored procedure execution and more. To build database applications you can visually connect to and query the data and bind the data to your Windows and Mac user interfaces without writing a line of code. You can also build database applications with a combination of components and business logic code. It's your choice which databases you want to work with and how your applications will create, read/retrieve, update and delete (CRUD) the data.

Database Components included in RAD Studio

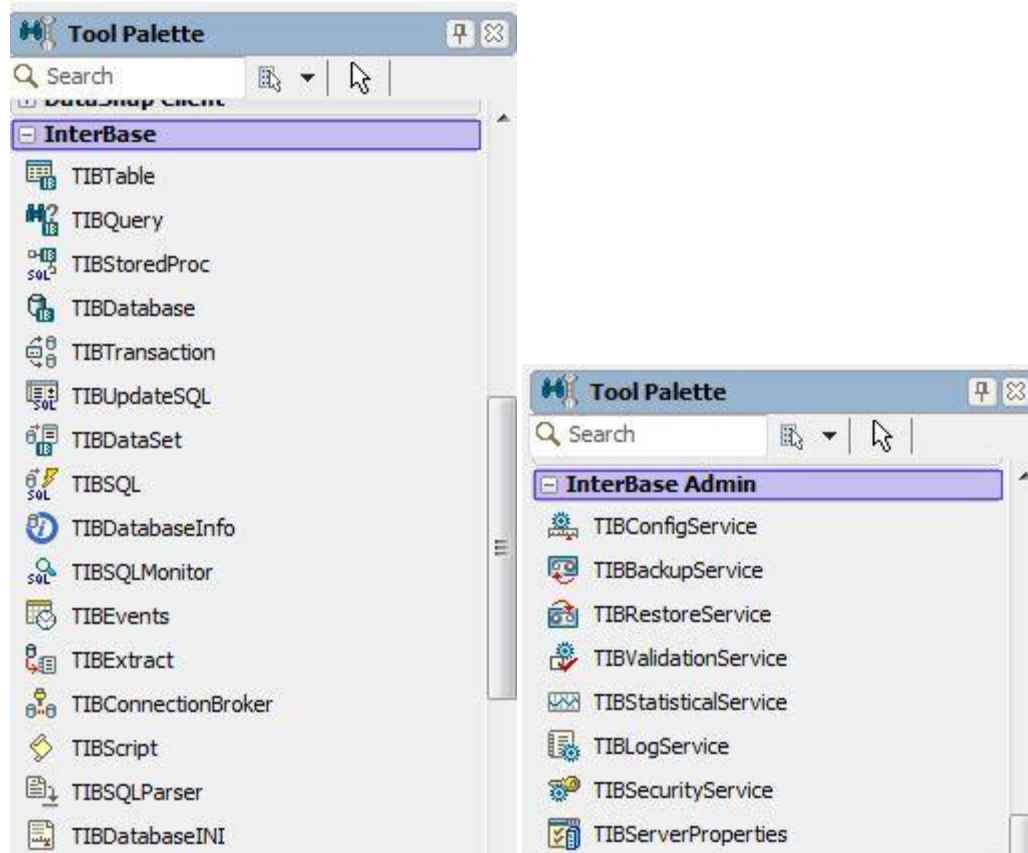
The Tool palette includes many components for accessing databases and representing the information they contain. The database components are grouped according to the data access mechanism and function. The following groups of components can be used in Windows and Mac applications.

E-Learning Series: Getting Started with Windows and Mac Development

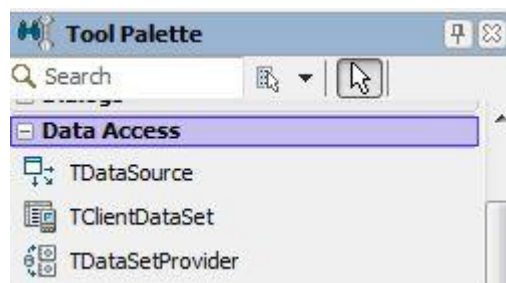
- dbExpress (DBX) – Cross-platform components that use dbExpress to access database information. dbExpress drivers provide fast access to databases. dbExpress' table, query and stored procedure components can return rows of data. To perform database updates you will need to also use the TClientDataSet and TDataSetProvider data access components.



- InterBase Express (IBX) – Contains components that access InterBase databases directly thru the API, without going through a separate engine layer. InterBase Express components can be used on Windows and Mac. The components give you additional functionality beyond dbExpress's multiple database support. IBX contains both access and administration component groups.



- **Data Access** – Included in RAD Studio are components that can be used with any of the data access mechanisms. TClientDataSet allows you to work with local data files (binary format or XML format) on disk as well as with server data. TDataSetProvider provides data from a dataset and resolves updates to that dataset or its database server. TDataSource provides an interface between a dataset component and data-aware controls on a form.



- **Data Controls** – In FireMonkey, all UI controls can be connected to databases via the LiveBindings capability. See the section about XE2's LiveBindings below.

When designing a database application, you must decide which data access mechanism to use. Each data access mechanism differs in its range of functional support, the ease of deployment, and the availability of drivers to support different database servers.

dbExpress

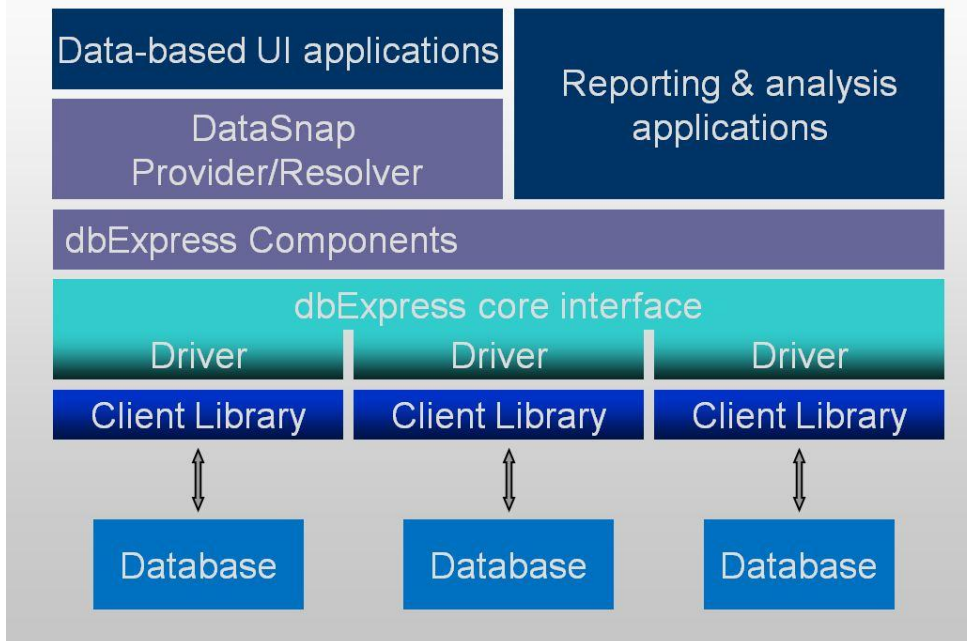
The dbExpress framework (DBX framework) is a set of abstract classes provided in the unit DBXCommon. Applications

Although many applications interface with dbExpress drivers via the dbExpress components, the DBX framework offers a convenient, lighter weight option to communicate with a database driver. You can also create a database driver for dbExpress by extending the framework's DBXCommon abstract base classes. The DBX framework provides most commonly needed database driver functionality for a "set" oriented database application, yet provides a simple interface.

Here are some of the key features of the DBX framework:

- The driver framework is written entirely in Delphi and allows drivers to be written in Delphi.
- It uses strongly typed data access instead of pointers. For instance, it uses String types rather than pointers to strings.
- The driver framework is single sourced. You can compile the source with the native Delphi compiler.
- The framework has only Abstract base classes that are used for drivers, connections, commands, readers, and so on.
- The framework uses exception based error handling rather than returning error codes.

dbExpress Architecture



Supported Databases and Versions for XE2

- InterBase XE, 2009, 2007, 7.5.1, 7.1*, 8.0*, 6.5* (All) (Driver dbxINT.dll, Client GDS32.DLL)
- Firebird 2.5, 2.1, 1.5 (Ent/Ult/Arch) (Driver dbxfb.dll, Client fbclient.dll)
- Sybase SQL Anywhere 12, 11, 10, 9, 8*, (Ent/Ult/Arch) (Driver dbxASA.dll, Client dbodbc*.dll)
- DB2 UDB 9.5, 9.1, 8.x*, 7.x* (Ent/Ult/Arch) (Driver dbxDB2.dll, Client db2cli.dll)
- Informix 9.x (Ent/Ult/Arch) (Driver dbxINF.dll, Client isqlb09a.dll)
- Microsoft SQL Server 2008, 2005, 2000 (Ent/Ult/Arch) (Driver dbxMSS.dll, Client sqlndi10.dll)
- MySQL 5.1, 5.0.27, 4.1* (Pro/Ent/Ult/Arch) (Driver dbxMYS.dll, Client libmysql.dll)
- Oracle 11g, 10g, 9.2.0*, 9.1.0* (Ent/Ult/Arch) (Driver dbxora.dll, Client OCI.DLL)
- Sybase ASE 12.5 (Ent/Ult/Arch) (Driver dbxASE.dll, Client libct.dll & libcs.dll)

* Driver not fully certified with this version of the database.

Note about MySQL driver support in dbExpress – you need to have the MySQL 5.1.x DLL (libmysql.dll) available (even if you are using later versions of MySQL – see readme note!) I got the DLL and put it in “C:\Program Files (x86)\Embarcadero\RAD Studio\9.0\bin” so that it would be found.

dbExpress Components

dbExpress is a set of lightweight database components that provide fast access to SQL database servers. For each supported database, dbExpress provides a driver framework that adapts the server-specific software to a set of uniform dbExpress interfaces. When you deploy a database application that uses dbExpress, you include a DLL (the server-specific driver) with the application files you build.

dbExpress lets you access databases using unidirectional datasets. Unidirectional datasets are designed for quick lightweight access to database information, with minimal overhead. Like other datasets, they can send an SQL command to the database server, and if the command returns a set of records, retrieve those records. Unidirectional datasets do not buffer data in memory, which makes them faster and less resource-intensive than other types of dataset. However, because there are no buffered records, unidirectional datasets are also less flexible than other datasets.

The dbExpress category of the Tool Palette contains components that use dbExpress to access database information. They are:

- TSQLConnection - Encapsulates a dbExpress connection to a database server
- TSQLDataSet - Represents any data available through dbExpress, or sends commands to a database accessed through dbExpress
- TSQLQuery - A query-type dataset that encapsulates an SQL statement and enables applications to access the resulting records, if any
- TSQLTable - A table-type dataset that represents all of the rows and columns of a single database table
- TSQLStoredProc - A stored procedure-type dataset that executes a stored procedure defined on a database server

- TSQLMonitor - Intercepts messages that pass between an SQL connection component and a database server and saves them in a string list
- TSimpleDataSet - A client dataset that uses an internal TSQLDataSet and TDataSetProvider for fetching data and applying updates

Dbxdrivers.ini

The dbxdrivers.ini file contains database driver information used by dbExpress. The file can be found in C:\Users\Public\Documents\RAD Studio\dbExpress\9.0.

Here is a sample of the dbxdrivers.ini file entry for the InterBase dbExpress driver. Additional database connection information is provided in the dbxconnections.ini file. You can also set dbExpress component properties and parameters in your code.

```
[Interbase]
DriverUnit=Data.DBXInterBase
DriverPackageLoader=TDBXDynalinkDriverLoader,DbxCommonDriver160.bpl
DriverAssemblyLoader=Borland.Data.TDBXDynalinkDriverLoader,
Borland.Data.DbxCCommonDriver,Version=16.0.0.0,Culture=neutral,PublicKeyToken=91d62ebb5b0d1b1b
MetaDataPackageLoader=TDBXInterbaseMetaDataCommandFactory,DbxInterBaseDriver160.bpl
MetaDataAssemblyLoader=Borland.Data.TDBXInterbaseMetaDataCommandFactory,Borland.Data.DbxInterBaseDriver,Version=16.0.0.0,Culture=neutral,PublicKeyToken=91d62ebb5b0d1b1b
GetDriverFunc=getSQLDriverINTERBASE
LibraryName=dbxint.dll
LibraryNameOsx=libsqlib.dylib
VendorLib=GDS32.DLL
VendorLibwin64=ibclient64.dll
VendorLibosx=libgds.dylib
BlobSize=-1
CommitRetain=False
Database=database.gdb
ErrorResourceFile=
LocaleCode=0000
Password=
RoleName=RoleName
ServerCharSet=
SQLDialect=3
IsolationLevel=ReadCommitted
User_Name=
waitOnLocks=True
Trim Char=False
```

Dbxconnections.ini

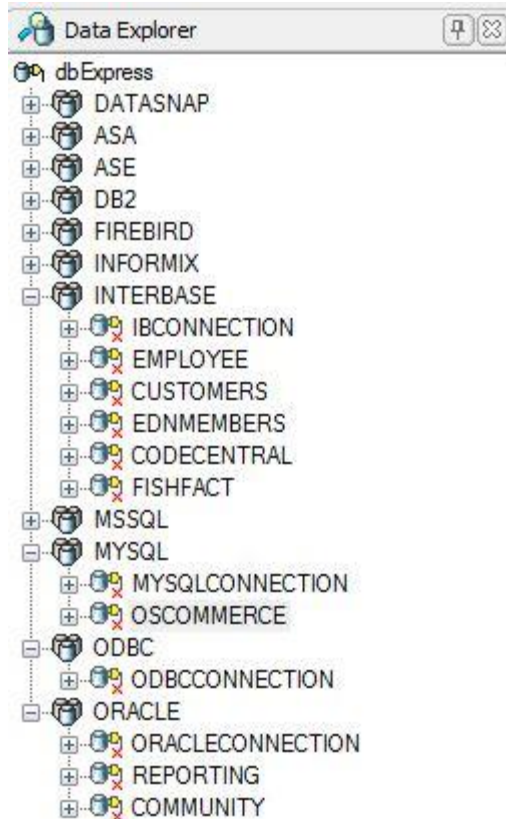
E-Learning Series: Getting Started with Windows and Mac Development

The dbxconnections.ini file contains database connection information. This file is found in the C:\Users\Public\Documents\RAD Studio\dbExpress\9.0 folder. When you want to use a TSQLConnection dbExpress component, you can use the Object Inspector to select a connection string name (the text in the [] below, to set the ConnectionName property.

```
[CUSTOMERS]
drivername=INTERBASE
librarynameosx=libsqlib.dylib
vendorlibwin64=ibclient64.dll
vendorlibosx=libgds.dylib
blobsize=-1
commitretain=False
Database=C:\Embarcadero\InterBase\examples\database\employee.gdb
localecode=0000
password=
rolename=RoleName
sqldialect=3
isolationlevel=ReadCommitted
user_name=
waitonlocks=True
trim char=False
```

Database Explorer

The Database Explorer is an IDE view that helps you create and explore dbExpress database driver and connection information including table metadata and other metadata found in the connected database.



InterBase Express (IBX)

InterBase Express (IBX) is a set of data access components that provides a means of accessing data from InterBase databases. The InterBase Administration Components, which require InterBase, are described after the InterBase data access components.

IBDatabase

Use a IBDatabase.TIBDatabase component to establish connections to databases, which can involve one or more concurrent transactions. Unlike BDE, IBX has a separate transaction component, which allows you to separate transactions and database connections.

Tip: You can store the user name and password in the IBDatabase component's Params property by setting the LoginPrompt property to False after logging in. For example, after logging in as the system administrator and setting the LoginPrompt property to False, you may see the following when editing the Params property:

```
user_name=sysdba  
password=masterkey
```

IBTransaction

IBX controls transactions with a separate component, `IBDatabase.TIBTransaction`. This powerful feature allows you to separate transactions and database connections, so you can take advantage of the InterBase two-phase commit functionality (transactions that span multiple connections) and multiple concurrent transactions using the same connection.

Use an `IBTransaction` component to handle transaction contexts, which might involve one or more database connections. In most cases, a simple one database/one transaction model are enough.

IBX dataset components

There are a variety of dataset components from which to choose with IBX, each having their own characteristics and task suitability:

- **IBTable** - Use an `IBTable.TIBTable` component to set up a live dataset on a table or view without having to enter any SQL statements.
- **IBQuery** - Use an `IBQuery.TIBQuery` component to execute any InterBase DSQL statement, restrict your result set to only particular columns and rows, use aggregate functions, and join multiple tables. `IBQuery` components provide a read-only dataset, and adapt well to the InterBase client/server environment.
- **IBDataSet** - Use an `IBCustomDataSet.TIBDataSet` component to execute any InterBase DSQL statement, restrict your result set to only particular columns and rows, use aggregate functions, and join multiple tables. `IBDataSet` components are similar to `IBQuery` components, except that they support live datasets without the need of an `IBUpdateSQL` component.
- **IBStoredProc** - Use `IBStoredProc.TIBStoredProc` for InterBase executable procedures: procedures that return, at most, one row of information. For stored procedures that return more than one row of data, or "Select" procedures, use either `IBQuery` or `IBDataSet` components.
- **IBSQL** - Use an `IBSQL.TIBSQL` component for data operations that need to be fast and lightweight. Operations such as data definition and pumping data from one database to another are suitable for `IBSQL` components.
- **IBUpdateSQL** - Use an `IBUpdateSQL.TIBUpdateSQL` component to update read-only datasets.
- **IBSQLMonitor** - Use an `IBSQLMonitor.TIBSQLMonitor` component to develop diagnostic tools to monitor the communications between your application and the InterBase server. When the `TraceFlags` properties of an `IBDatabase` component are turned on, active `TIBSQLMonitor` components can keep track of the connection's activity and send the output to a file or control. A good example would be to create a separate application that has a `TIBSQLMonitor` component and a Memo control. Run this secondary application, and on the primary application, activate the `TraceFlags` of the `IBDatabase` component. Interact with the primary application, and watch the memo control in the second application fill with data.
- **IBDatabaseInfo** - Use an `IBDatabaseInfo.TIBDatabaseInfo` component to retrieve information about a particular database, such as the sweep interval, ODS version, and the user names of those currently attached to this database.

- **IBEvents** - Use an IBEvents component to register interest in, and asynchronously handle, events posted by an InterBase server.

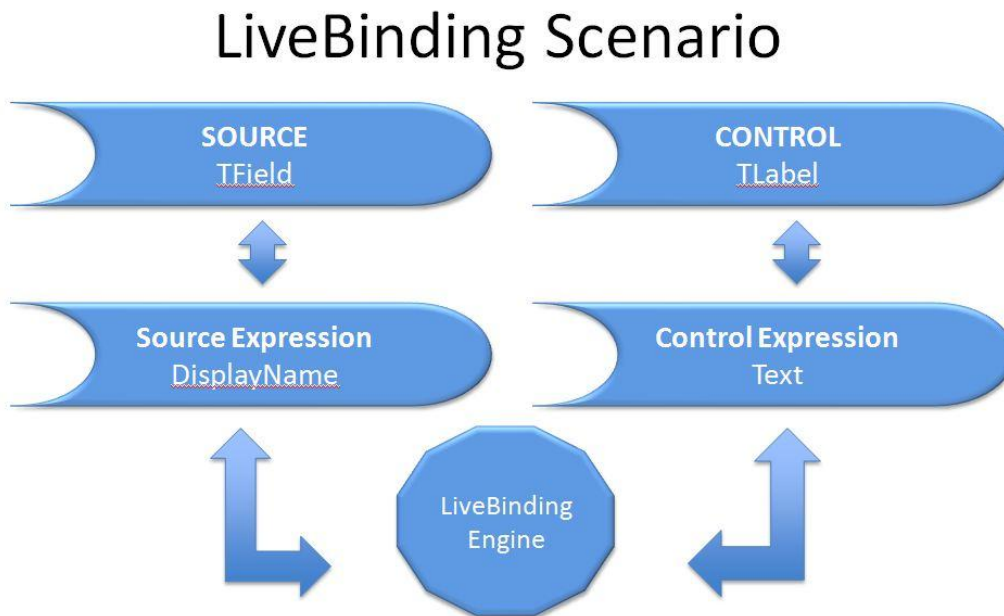
LiveBindings

LiveBindings is the data-binding feature supported by both the VCL and FireMonkey in RAD Studio. LiveBindings is expression-based, which means it uses expressions to bind objects to each other, by means of their properties.

LiveBindings Overview

LiveBindings is based on relational expressions, called binding expressions, that can be either unidirectional or bidirectional. LiveBindings is also about control objects and source objects. By means of binding expressions, any source object can be bound to itself (it becomes both source and control object) or to any other control object, simply by defining a binding expression involving one or more properties of the objects you want to bind together. For example, you can bind a TEdit control to a TLabel so that, when the text changes in the edit box, the caption of the label is automatically adjusted to the value evaluated by your binding expression. Another example is binding a tracker control to a progress bar so that the progress rises or lowers as you move the track bar.

In the same manner, you can connect to databases, alter one or more properties of different objects, and so on. Because LiveBindings propagates, you can even alter properties of objects that are connected to other objects that are bound to a control object.



LiveBindings Usage

You can use LiveBindings in your FireMonkey applications by means of the Object Inspector and the visual components used by LiveBindings.

LiveBindings can be managed or unmanaged. Managed bindings are automatically evaluated when an object or a property in the input expression is modified. For example, if the input expression references the Text property of an edit box component called Edit1, the binding will be evaluated when the expression engine is informed of a change to Edit1. TBindingList.Notify is called to notify the expression engine. The first parameter to Notify() is the object (Edit1, in this case) and the second parameter is the affected property. The property parameter can be blank, in order to cause all bindings that reference the object to be evaluated.

RAD Studio does not have a notification system that informs the expression engine of any object or property change. In the sample applications, Notify is called from event handlers. Typically, the Sender parameter of the event handler is used to call BindingsList1.Notify(Sender, "");. In this way, you can use one event handler to handle notifications from many controls. The SynchControlsSampleProject sample application (that is also listed in the table below) exemplifies how the notifier is implemented.

TBindExpression and TBindExprItems are the LiveBindings components that support managed bindings. Set the Managed property. These two component also have another property that affects notifications. When the NotifyOutputs property is True, the expression engine will automatically send notification to an output object or property that is modified when a binding is evaluated.

None of the other binding components use managed bindings. They use unmanaged bindings, which are always explicitly evaluated.

The TBindLink, TBindListLink, and TBindGridLink components and the closely related FireMonkey TBindDB*Link components bind data such as a TDataSet field to controls such as TEdit. Rather than rely on the expression engines dependency graph to keep controls and data in sync, these components listen to notifications from the data source and notifications from the controls (using the new TObserver API).

There are two other components that use unmanaged bindings. TBindList and TBindGridList are used to populate a list or grid. It is possible for two binding components to be used together on the same control. So, for example, a TBindList and a TBindDBList component can be used together to fill a TListBox that can edit a TDataSet field.

LiveBindings Visual Components

The LiveBindings engine is a run-time feature, but a number of visual components are present in the component palette, to help you create live bindings between components at design time.

If you browse the component palette, in the LiveBindings tab, there are the following components that you can drag and drop to a form (either VCL or FMX):

- TBindingList for holding of binding lists

- TBindScope for bindings to components
- TBindScopeDB for bindings to databases

When you use the Object Inspector to create live bindings between components, the appropriate LiveBindings component (TBindingsList) is automatically placed on the form. This component holds a list of all bindings available in your Application.

When using C++Builder to develop console applications that make use of LiveBindings, you have to explicitly instruct the C++ compiler to generate RTTI information for the classes in your project. To do this, add the `#pragma explicit_rtti methods()` directive at the beginning of your project, as shown in the Using LiveBinding Programmatically tutorial. For more information regarding the `#pragma explicit_rtti methods()` directive, please refer to the Pragma explicit rtti help topic.

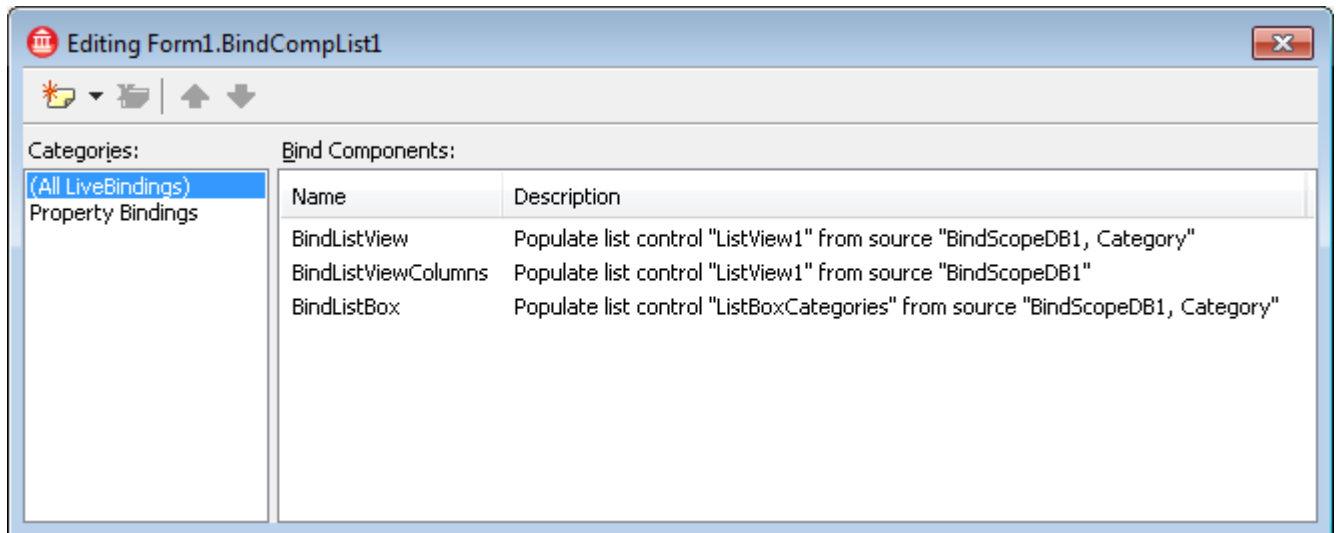
BindingsLists

Bindings Lists are collections of bindings. Every LiveBinding created from within your Application can be found in a Bindings List.

Whenever you create a new LiveBinding using the Object Inspector, the **TBindingsList** component is automatically placed on your form (be it a VCL form or an FMX form). The **TBindingsList** component exposes (publishes) five properties in the Object Inspector:

- **Methods** - Specifies a collection of available methods.
- **Name** - Specifies the name of the TBindingsList component.
- **OutputConverters** - Specifies a collection of available output converters.
- **Tag** - Specifies a 4-byte storage place for anything you want: pointers, reference to objects, numbers, and so on. This property is not LiveBinding-related, as it can be found on other components too.
- **UseAppManager** - Specifies whether the bindings are using the Application Manager or not. In other words, UseAppManager specifies whether the bindings are managed or unmanaged.

Double-click the component to gain access to the actual bindings (either binding expressions or property bindings) and further, to the list of binding expressions affiliated to their bindings. The following picture illustrates the way the bindings list editor looks like.



The bindings list editor lets you:

- View the binding expressions in your Application.
- View the property bindings in your Application.
- Create a new binding.
- Delete an existing binding.
- Move a binding up or down in the bindings list.

You can see that, for this example, a number of bindings are present in the list. To view the parameters for any of the listed bindings, double-click one of them, and the Binding Expression Editor appears on the screen.

Use this dialog box to visually edit the parameters of your binding expressions.

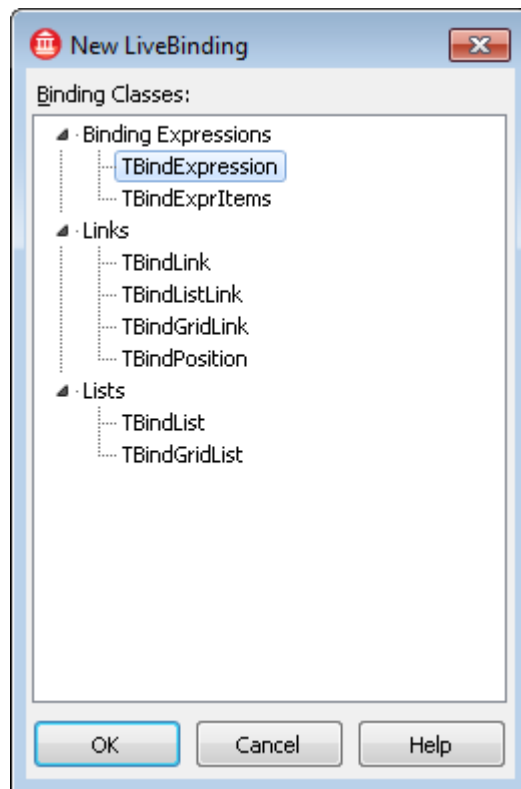
LiveBinding Dialog Box

The “New LiveBinding” dialog box appears when you attempt to create a new LiveBinding in your application. From within the New LiveBinding dialog box, you are prompted to select a binding class for your new LiveBinding. The available binding classes are:

- Binding Expressions:
 - TBindExpression creates a new binding expression.
 - TBindExprItems is used for synchronization of various controls.
- Links:
 - TBindLink is used when creating bindings to the columns in the grid of a dataset. In general, TBindLink is used for linking operations, for example, linking dataset fields to various lists.
 - TBindListLink is similar to TBindLink; use it to bind a control to a list.
 - TBindGridLink is similar to TBindLink; use it to bind a control to a grid.

E-Learning Series: Getting Started with Windows and Mac Development

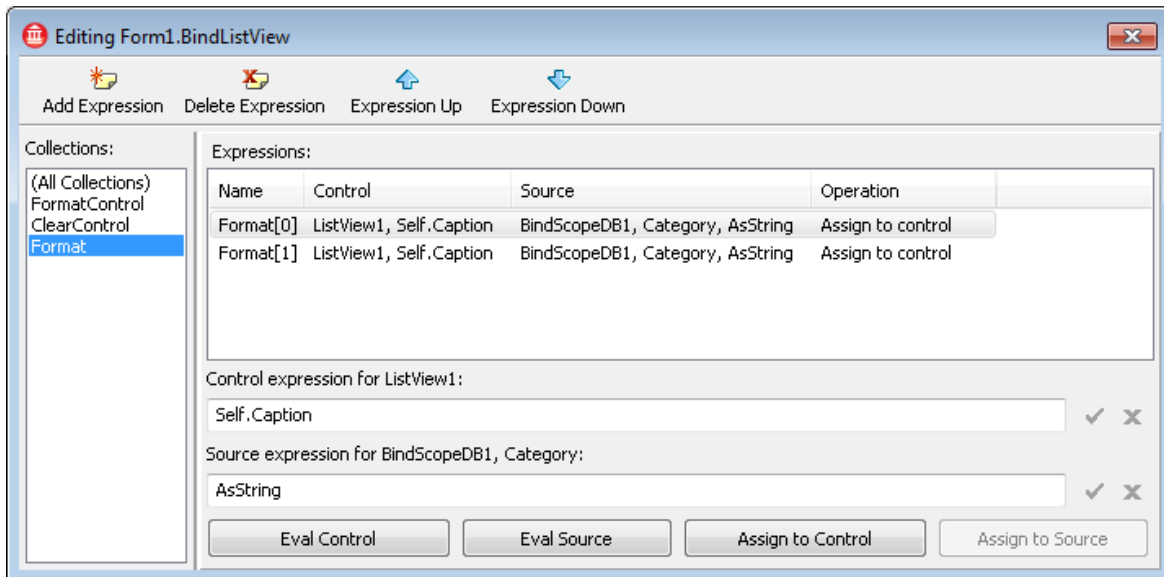
- TBindPosition creates a new LiveBinding for position-related controls (such as a track bar).
- Lists:
 - TBindLink is used when binding components to lists. You can populate a list with multiple data returned by a binding expression.
 - TBindGridList is similar to TBindList, but works on grids.



To validate your selection, click OK. At this point, your new LiveBinding should appear in the Object Inspector as well as in the bindings list present on your form.

The Binding Expression Editor


The "Binding Expression Editor" is available when double-clicking a binding expression that is readily available in a Bindings List. The Binding Expression Editor dialog box is pictured below.



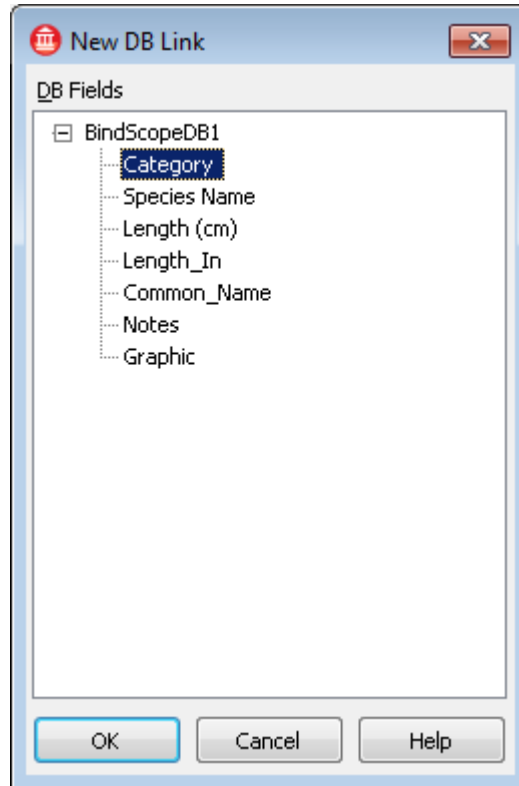
The Binding Expression Editor lets you:

- Add a new binding expression.
- Delete an existing binding expression.
- Move a binding expression up or down in the list.
- View the existing binding expressions.
- Edit the control expression for the control component.
- Edit the source expression for the source component.
- Evaluate the control expression.
- Evaluate the source expression.
- Assign the expression to the control or to the source component.

The New DB Link Dialog Box

The “New DB Link” dialog box appears when you attempt to create a LiveBinding to the fields of a database. In the Object Inspector, click the  in the LiveBindings value field and select **Link to DB Field**.

The “New DB Link” dialog box shows a list of available fields from a database. You can bind, for example, a label to a dataset column called “Category” so that, each time you navigate through the dataset, the caption of the label changes.



To validate your selection, click **OK**.

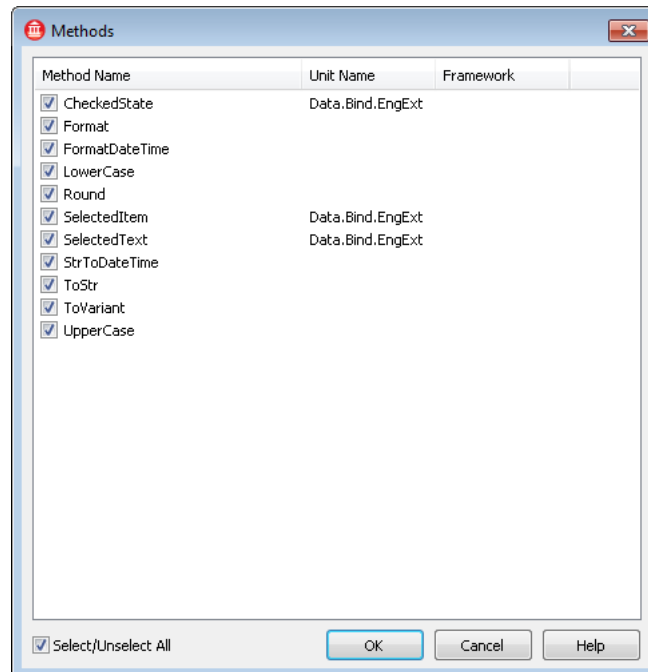
LiveBinding Methods Dialog Box

The “Methods” dialog box is available if you select a Binding List component and then double-click the Methods property published in the Object Inspector.

The “Methods” dialog box contains a collection of available methods to use in your binding expressions from a specific Binding List. These methods, if checked, can be used in any binding expression contained in the binding list, for example, if you have an edit box and you want to use its Text property where all lowercase characters are required. In this case, the source binding expression should look like: `LowerCase(Text)`.

The following items are available in the “Methods” dialog box”

- Methods List: Method Name - Lists the unique names of the available methods.
- Methods List: Unit Name - Displays the unit where the specific method is implemented. If the unit name is not shown, then the method is internal, implemented in the LiveBindings engine.
- Methods List: Framework - Displays the framework the method is intended for.
- Select/Unselect All check box - Allows for selection/unselection of all the available methods at once.



To validate the selection of available methods, click **OK**.

LiveBindings Output Converters Dialog Box

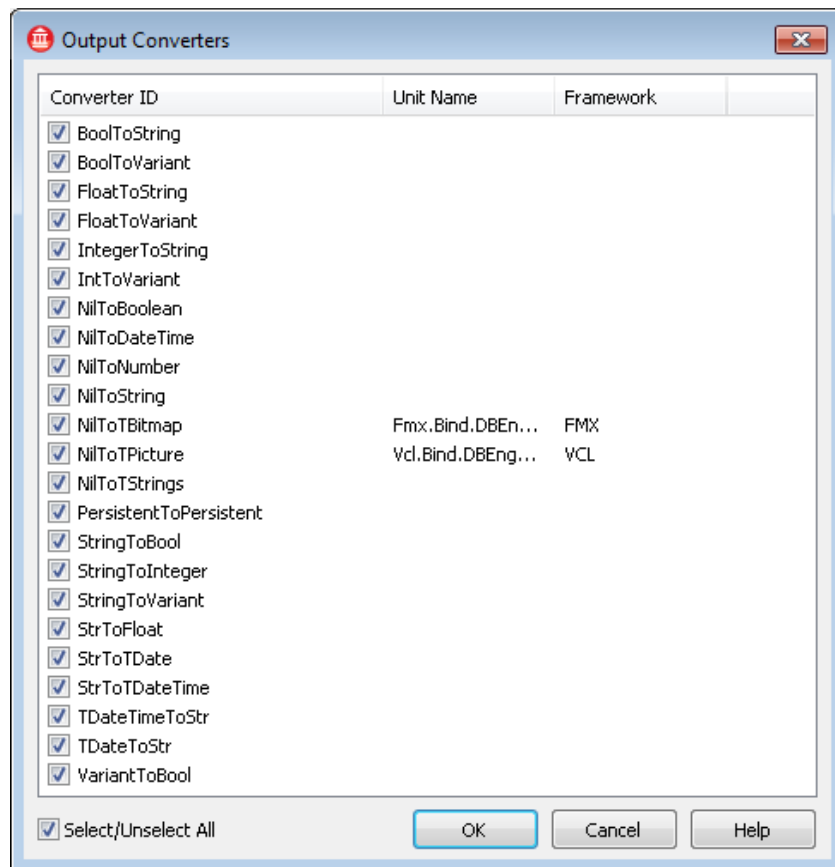
The “Output Converters” dialog box is available if you select a “Binding List” component and then double-click the Output Converters property published in the Object Inspector.

The “Output Converters” dialog box contains a collection of available output converters to use in your binding expressions from a specific Binding List. These output converters, if checked, can be used in any binding expression contained in the bindings list.

These output converters are useful for making conversions between different data types. These converters, if selected, are automatically called when there is a need to match the data type of the destination property to the output of the binding expression affecting that very property. For instance, if you have a spin edit control bound to a label's caption, then the IntegerToString output converter is automatically called, internally, by the LiveBindings engine, to convert the numerical output of the binding expression to the required text (string) data type.

The following items are available in the “Output Converters” dialog box:

- Converters List: Converter ID - Lists the unique names (ID) of the available output converters.
- Converters List: Unit Name - Displays the unit where the specific output converter is implemented. If the unit name is not shown, then the converter is internal, implemented in the LiveBindings engine.
- Converters List:Framework - Displays the framework the output converter is intended for.
- Select/Unselect Allcheckbox - Allows for selection/unselection of all output converters at once.



To validate the selection of available output converters, click **OK**.

LiveBindings Columns Editor

With the “Columns Editor” dialog box you can edit the columns of an associated string grid (TStringGrid). There are two ways to open the “Columns Editor” dialog box:

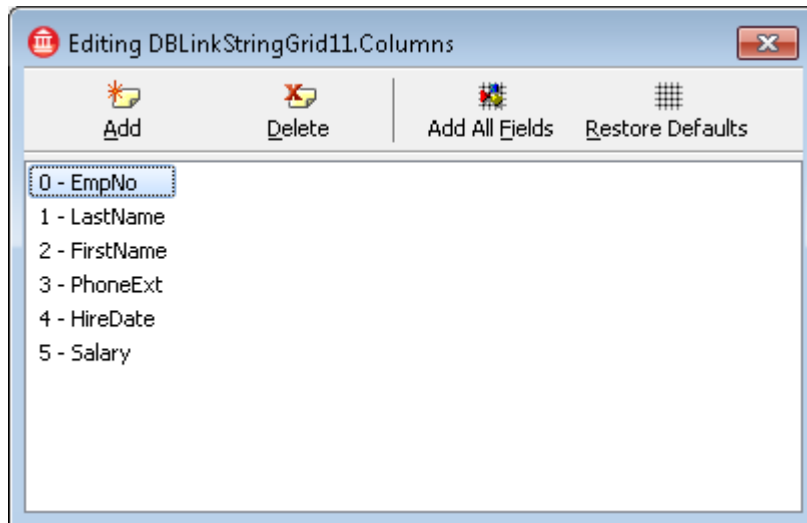
- Double-click the Bindings List component to see all the bindings, then locate the TBindDBGridLink associated with your string grid. Select it and use the Columns property to show the “Columns Editor” dialog.
- Select the string grid component and, in the Object Inspector, go to the created LiveBinding and select the Columns property. Double-click it to see the dialog box.

The “Columns Editor” allows you to add, delete, restore defaults, or add all the fields as columns of a string grid.

- Add - Adds a new column to the list.
- Delete - Deletes an existing column from the list.

E-Learning Series: Getting Started with Windows and Mac Development

- Add All Fields - Adds all fields reported by the data source. Clicking this button prompts you whether to remove all existing columns first.
- Restore Defaults - Switches all columns to their defaults.



Creating LiveBindings

The easiest way to create a LiveBinding is using the Object Inspector. Use the following steps to create such a LiveBinding.

- Locate the object you want to be the control object, for instance, a FireMonkey TEdit box.
- Locate the object you want to be the source object, for instance, a FireMonkey TSpinBox.
- Identify the LiveBinding functionality you want, then create your binding expression accordingly. For instance, get the value in the TSpinBox, convert to a string, and then put it in the edit box.
- Click to select the control object.
- In the Object Inspector, locate the LiveBindings property. Select it, and then add a new LiveBinding of type TBindExpression.
- Now look into the list of properties of the newly created LiveBinding and type in the source object.
- At this point, specify the control expression and the source expression. For example, the control expression can be Text and the source expression can be ToString(Value), supposing that you need to convert the Integer value to a string.
- Select the source object and, from the Object Inspector, go to the Events tab. In the OnChange event, issue a spin edit box value change notify command to the LiveBinding engine: `TBindings.Notify(SpinEdit1, 'Value');`

Using LiveBinding to Create a FireMonkey HD Application Without Code

E-Learning Series: Getting Started with Windows and Mac Development

LiveBinding is a data binding technology introduced in the XE2 release that helps you develop database applications. This tutorial demonstrates how to build a small yet powerful database application using LiveBinding. You can do almost everything here with a few simple mouse clicks; no coding is required.

This tutorial uses the “biolife.cds” database, typically installed in the Samples directory at Start | Programs | Embarcadero RAD Studio | Samples and also available with the Samples on SourceForge at: RAD Studio Demo Code. A code example that uses a similar database (DBDEMOS) is located at FishFact (Delphi) example.

Follow these steps to build a database application using LiveBinding:

1) Create a new project by choosing **File > New > FireMonkey HD Application – C++Builder** or **File > New > FireMonkey HD Application - Delphi**.

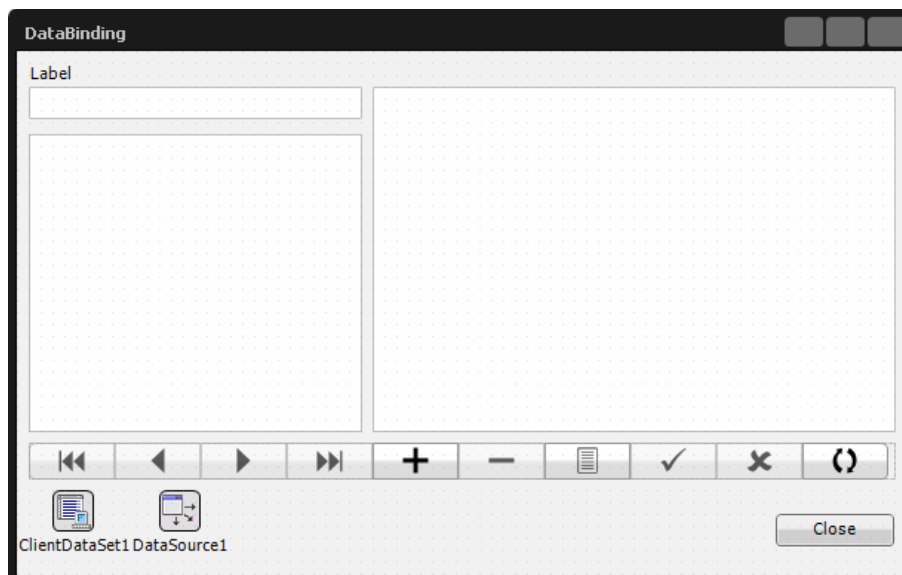
2) Set the Caption property on the form to “DataBinding”.

3) Drag and drop a DataSource component and a ClientDataSet component onto the Form Designer.

4) In the Object Inspector, connect the DataSource component to the ClientDataSet using the DataSource.DataSet property.

5) Add the following controls to the form: a label, an edit box, a memo, and an image control component. To be able to cycle through the records of the dataset you chose to open, you need an additional **BindNavigator** control. Optionally, you can add a Close button.

At this point, your application should look like the following image.

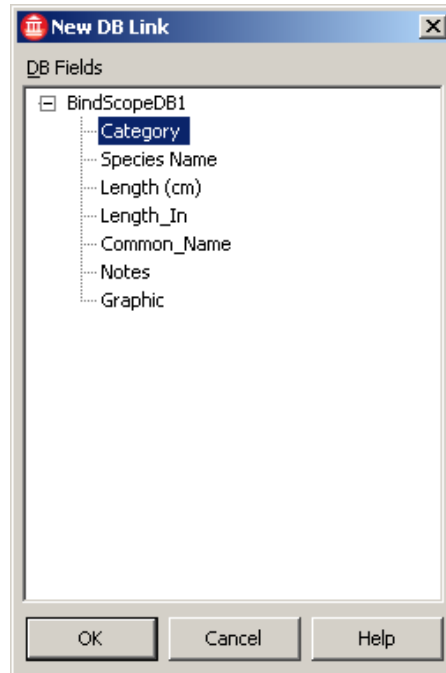


6) Before using LiveBinding, you need to specify a database file for the ClientDataSet component. Use the FileName property to specify a ClientDataSet (*.cds) file, and select the biolife.cds file.

7) Set the Active property to True.

8) Now you can simply bind each graphical control on the form to their appropriate field in the database.

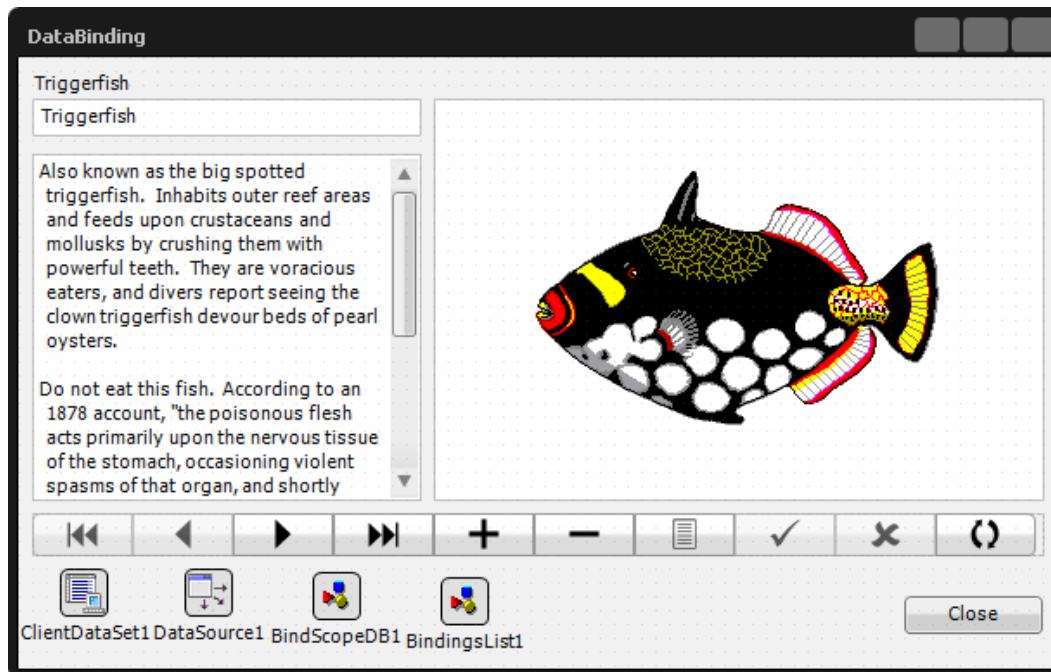
- A. Select the Label component and then seek the LiveBindings property in the Object Inspector. Click the down-arrow in the value field, and select Link to DB Field.
- B. On the New DB Link dialog box, select Category.



- C. Then select the edit box component and also bind it to the Category item in the list.
- D. Link the memo component to the Notes item.
- E. Bind the image control component to the Graphic item.
- F. Finally, link the bind navigator to the binding scope by assigning the BindScope property to BindScopeDB1.

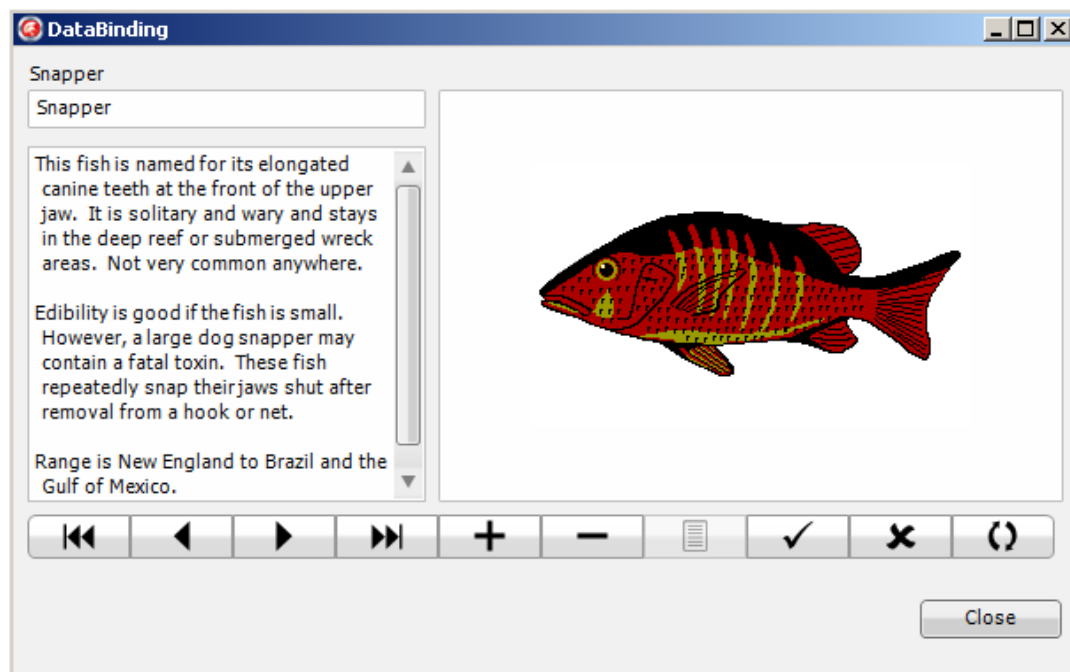
You can see that two new components, BindScopeDB1 and BindingsList1, have been added to the form. These components are not visual and are part of the LiveBinding engine. (To manually alter the bindings in the BindingsList1 component, double-click it. But that is not the purpose of this tutorial.)

Now your application should look like the following picture:



At this point, you can run the application (either choose Run > Run or press F9).

After the application is running, you can cycle through the database records, add a record, delete a record, or update the database.



Using InterBase Express (IBX) to Create an Application Without Code

See the Embarcadero DocWiki for the steps at

[http://docwiki.embarcadero.com/RADStudio/en/Tutorial: Using InterBase Express to Create an Application Without Code](http://docwiki.embarcadero.com/RADStudio/en/Tutorial:_Using_InterBase_Express_to_Create_an_Application_Without_Code). Note: this section is for creating a VCL application. I will update it shortly to create a FireMonkey HD application.

Connecting a Windows and Mac FireMonkey Application to the MySQL OSCommerce Database

One of the example databases that is publicly available for MySQL is the OSCommerce database. To create a Windows and Mac application that uses dbExpress, data access components and LiveBindings to display the OSCommerce Countries table, use the following steps:

1) Install and configure your MySQL server and OSCommerce database

- a) Install MySQL 5.1 or higher (<http://dev.mysql.com/doc/refman/5.1/en/windows-installation.html> and <http://dev.mysql.com/doc/refman/5.0/en/macosx-installation.html>).
- b) Install the OSCommerce MySQL database (http://www.siteground.com/tutorials/oscommerce/oscommerce_manual_installation.htm)
- c) Use the Data Explorer in the IDE to create a new connection to the MySQL:
- d) Right Mouse Click on the MYSQL node and choose "Add New Connection". In the dialog box, set the connection name to OSCOMMERCE.
- e) Right Mouse Click on the just added OSCOMMERCE connection name and choose "Modify Connection" from the popup menu.
- f) Set the Server Name to the name or IP address of the computer where you installed your MySQL Server.
- g) Set the Database Name to "oscommerce"
- h) Set the User Name and Password for your database.
- i) Click on the **Test Connection** button to make sure you've set the connection information correctly.
- j) If all is working, click the OK button to save the connection info to your dbxconnections.ini file.

2) Create a new project using either **File > New > FireMonkey HD Application – C++Builder** or **File > New > FireMonkey HD Application – Delphi**.

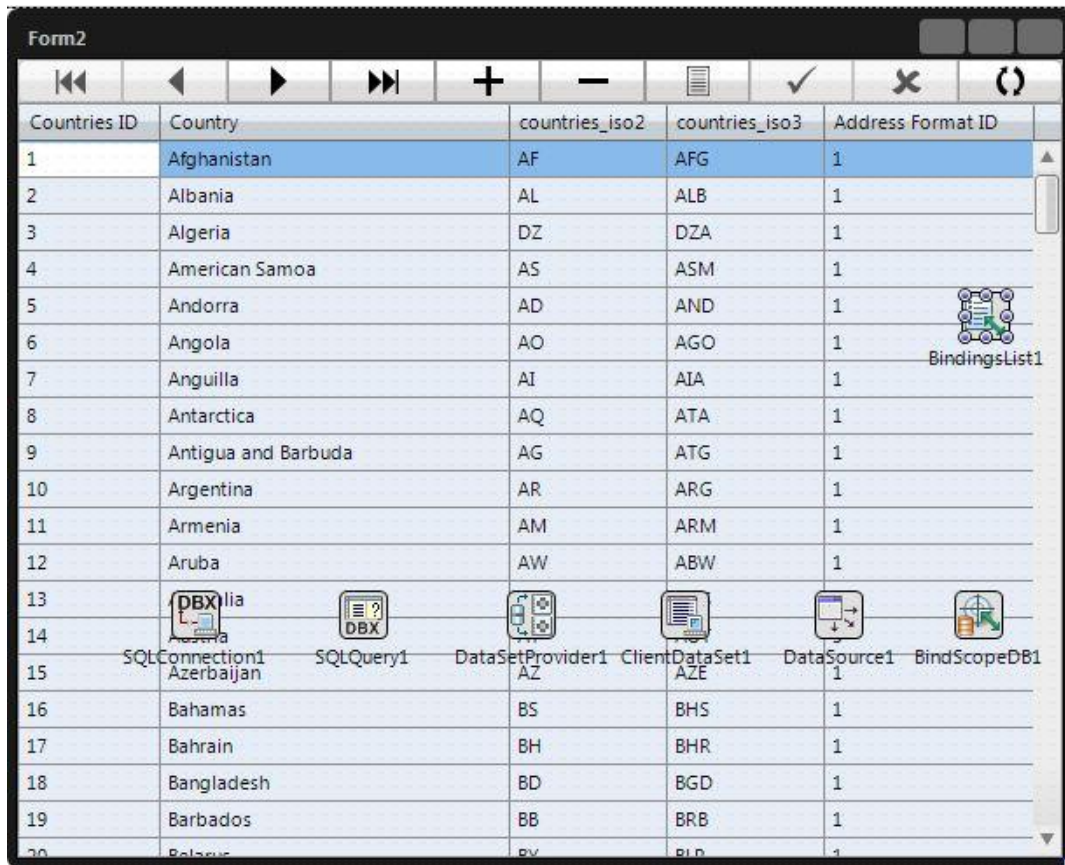
3) Add the following dbExpress and data access components to your FireMonkey form and set a few of their properties:

- a) SQLConnection – Set the ConnectionName property to OSCOMMERCE. Set LoginPrompt property to False. Set Connected property to True.
- b) SQLQuery – Set SQLConnection property to SQLConnection1. Set SQL property to "Select * from Countries". Set the Active to True.
- c) DataSetProvider – Set the DataSet property to SQLQuery1.
- d) ClientDataSet – Set the ProviderName to DataSetProvider1. Set the Active property to True.
- e) DataSource – Set the DataSet property to ClientDataSet1.
- f) BindScopeDB- Set the DataSource property to DataSource1.

4) Add the following UI components to your form and set a few of their properties:

- BindNavigator - Set the Align property to "alMostTop". Set the BindScope property to BindScopeDB1.
- StringGrid – Set the Align property to "alClient". In the LiveBindings property editor choose "Link to DB DataSource..." and choose the BindScopeDB1 data source. (you should see the Countries table entries appear in the StringGrid).
- Right Mouse Click on the StringGrid and choose "Columns Editor..." from the local menu. Select each of the columns in the Columns Editor and using the Object Inspector edit the Header property contents to have a more readable string. Set the Width property so that the header and data rows fit in the column width (for example: I used 200 for the countries_names column).

Your form should look something like the following:



Compile and Run your application.

Deploying Database Applications

Applications that access databases involve special installation considerations beyond copying the application's executable file onto the host computer. Database access is most often handled by a

E-Learning Series: Getting Started with Windows and Mac Development

separate database engine, whose files cannot be linked into the application's executable file. The data files, when not created beforehand, must be made available to the application. Multi-tier database applications require additional handling on installation, because the files that make up the application are typically located on multiple computers.

Since several different database technologies (dbExpress, and InterBase Express) are supported, deployment requirements differ for each. Regardless of which you are using, you need to make sure that the client-side software is installed on the system where you plan to run the database application. dbExpress and InterBase Express (IBX) also require drivers to interact with the client-side software of the database.

Specific information on how to deploy dbExpress and multi-tiered database applications is given in the following topics:

- Deploying dbExpress Database Applications - http://docwiki.embarcadero.com/RADStudio/en/Deploying_dbExpress_Database_Applications
- Overview of a Three-tiered Application (will be covered in Lesson 9) - http://docwiki.embarcadero.com/RADStudio/en/Overview_of_a_Three-tiered_Application

Database applications that use client datasets such as TClientDataSet or dataset providers require you to include midas.dcu (for Delphi) or midas.hpp (for C++) (for static linking when providing a standalone executable); if you are packaging your application (with the executable and any needed DLLs), you need to include Midas.dll.

When deploying database applications that use IBX, you need to be sure that the InterBase client is installed on the system where you plan to run the application. InterBase requires gds32.dll and interbase.msg to be located in an accessible directory. No other deployment steps are required. IBX components communicate directly with the InterBase Client API and do not require additional drivers.

If deploying database applications that use MySQL, you need to ensure that MySQL 5.1 is installed on the system where you plan to run the application.

MS SQL 2008 is now fully supported in RAD Studio Enterprise. All data type mappings are properly supported between dbExpress and MS SQL Server 2008. However, in order to use these data type mappings, you need to install a native SQL client, that is you need to make sure you have a copy of the SQLNCLI10.DLL file. MS SQL Server 2008 Native Client can be downloaded from this link. RAD Studio Enterprise also provides full support for MS SQL Server 2005, in case your operating system does not support the MS SQL 2008 version. In this case, you need to assure you have a copy of the SQLNCLI.DLL file that can be accessed by your application.

If deploying database applications that use Firebird, you need to install a Firebird client, that is you need to make sure you have a copy of the FBCLIENT.DLL file on the system where you plan to run the application.

If deploying database applications that use Oracle 11g, install the Oracle Instant Client on the system where you run the application.

In addition to the technologies described here, you can also use third-party database engines to provide database access. Consult the documentation or vendor for the database engine regarding redistribution rights, installation, and configuration.

If the library corresponding to the used technology is not found, an error message is displayed and the application may not run correctly.

Using Project > Deployment to add the dbExpress and MIDAS Library Feature Files

As mentioned in the IDE lesson, in order to build, deploy and test your FireMonkey dbExpress application on remote Windows and Mac target computers, you'll need to make sure the dbExpress driver DLL/Dylib and MIDAS DLL/Dylib files are sent to the target machine.

To do this, use the Project > Deployment menu item and click on the "Add Feature Files" speed button in the Deployment files window. Check the dbExpress Driver check box for the database you are using. Check the MIDAS Library check box if you are using ClientDataSet. Click the OK button.

This will ensure that the required files will be deployed with your application so that you can run and debug your dbExpress application on remote Windows and Mac computers. Depending on where you told dbExpress your database is, you will also have to have installed the database server and database data file on those target computers.

Summary, Looking Forward, To Do Items, Resources, Q&A and the Quiz

In Lesson 6 you learned about the new LiveBindings capabilities in RAD Studio. You also learned how to use LiveBindings to build FireMonkey applications that connect to your databases. We will cover data bases again in Lesson 9 when you learn how to build Multi-Client, Multi-Tier, Multi-Platform Windows and Mac applications.

In Lesson 7, you'll learn how to create your first 3D FireMonkey applications.

In the meantime, here are some things to do, articles to read and videos to watch to enhance what you learned in Lesson 6 and to prepare you for lesson 7.

To Do Items

Watch Jim Tierney's CodeRage 6 LiveBindings replay videos (use the links below). Create your own FireMonkey HD application using LiveBindigns: Put a TEdit and TTrackBar and connect the TTrackBar Value property to the TEdit Text property using LiveBindings. Follow the steps to create the FireMonkey HD application with no lines of code example using the BioLife ClientDataSet data file.

Links to Additional Resources

- Getting Started Course landing page - <http://www.embarcadero.com/firemonkey/firemonkey-e-learning-series>
- Jim Tierney's LiveBindings CodeRage 6 video replays (Jim Tierney is a Principal Engineer at Embarcadero Technologies. Since Delphi 3, Jim has helped develop many Delphi and C++Builder features):
 - Introduction to LiveBindings - This session introduces LiveBindings. LiveBindings are the new data binding system for VCL and FireMonkey. This session introduces key concepts including the binding engine, scopes, converters, methods, binding components, and custom components. - <http://cc.embarcadero.com/Item/28544>
 - LiveBindings: Practical Examples - This session demonstrates how to use LiveBindings in practice. Examples include populating lists, TDataSet editing, control synchronization, using custom data. Both VCL and Fire Monkey applications are demonstrated. - <http://cc.embarcadero.com/Item/28545>
- Learn to use LiveBindings in RAD Studio XE2 by Cary Jensen: OnDemand Webinar replay, White Paper and short tutorial videos - <http://www.embarcadero.com/rad-in-action/livebindings>
- Using LiveBindings to Create a FireMonkey database application with no lines of code - [http://docwiki.embarcadero.com/RADStudio/en/Tutorial: Using LiveBinding to Create an Application Without Code](http://docwiki.embarcadero.com/RADStudio/en/Tutorial:_Using_LiveBinding_to_Create_an_Application_Without_Code)
-

Q&A:

Here are some of the answers for the questions I've received (so far) for this lesson. I will continue to update this Course Book during and after course.

Q:

A:

If you have any additional questions – send me an email - davidi@embarcadero.com

Self Check Quiz

1. Which of the following RAD Studio database components can be used to build Windows and Mac applications?

- a) BDE
- b) dbExpress
- c) dbGO
- d) ADO
- e) XML

2. What is the name of the new RAD Studio XE2 technology that lets you bind data to FireMonkey user interface controls?

- a) Data Access
- b) SQL
- c) LiveBindings
- d) Data Bound Controls

Answers to the Self Check Quiz:

1b, 2c