How to create data-aware components

This document describes minimal steps necessary to create a data-aware browsing component that displays data for a single field. The example component is a *TPanel* descendant with *DataSource* and *DataField* properties similar to the *TDBText* component. &quout;Making a Control Data-Aware" in the Component Writer's Guide in the Delphi help file for further examples.

Basic steps to create a data-browsing component

Create or derive a component that allows the display, but not the entry of data. For instance, you could use a *TMemo* with *ReadOnly* set to true. In the example outlined in this document, we'll use a *TCustomPanel*. This will allow display, but not data entry.

- Add a data-link object to your component. This object manages communication between the component and the database table.
- Add *DataField* and *DataSource* properties to the component.
- Add methods to get and set the *DataField* and *DataSource*.
- Add a *DataChange* method the component to handle the data-link object's *OnDataChange* event.
- Override the component constructor to create the datalink and hook up the *DataChange* method.
- Override the component destructor to cleanup the datalink.

Creating the TDBPanel

Create or derive a component that allows the display, but not the entry of data. We'll be using a *TCustomPanel* as a starting point for this example.

In Delphi, choose the appropriate menu option to create a new component (this will vary between editions of Delphi), and specify *TDBPanel* as the class name, and *TCustomPanel* as the ancestor type. You may specify any palette page.

Add *DB* and *DBTables* to your **uses** clause.

Add a data-link object to the component's private section. This example will display data for a single field, so we will use a *TFieldDataLink* to provide the connection between our new component and a data-source. Name the new data-link object *FDataLink*. Example:

```
private
  FDataLink: TFieldDataLink;
```

Add *DataField* and *DataSource* properties to the component. We will add supporting code for the get and set methods in following steps. Note that our new component will have *DataField* and *DataSource* properties and *FDataLink* will also have its own *DataField* and *Datasource* properties. Example:

```
published
  property DataField: string read GetDataField write SetDataField;
  property DataSource: TDataSource read GetDataSource write SetDataSource;
```

Add private methods to get and set the *DataField* and *DataSource* property values to and from the *DataField* and *DataSource* for *FDataLink*. Example:

```
...
private
  FDataLink: TFieldDataLink;
  function GetDataField: String;
  function GetDataSource: TDataSource;
  procedure SetDataField(Const Value: string);
  procedure SetDataSource(Value: TDataSource);
...

implementation

function TDBPanel.GetDataField: String;
```

```
begin
  Result := FDataLink.FieldName;
end;

function TDBPanel.GetDataSource: TDataSource;
begin
  Result := FDataLink.DataSource;
end;

procedure TDBPanel.SetDataField(Const Value: string);
begin
  FDataLink.FieldName := Value;
end;

procedure TDBPanel.SetDataSource(Value: TDataSource);
begin
  FDataLink.DataSource := Value;
end;
```

Add a private *DataChange* method to be assigned to the data-link's *OnDataChange* event. In the *DataChange* method add code to display actual database field data provided by the data-link object. In this example, we assign *FDataLink*'s field value to the panel's caption. Example:

```
...
private
  procedure DataChange(Sender: TObject);
...

implementation

procedure TDBPanel.DataChange(Sender: TObject);
begin
  if FDataLink.Field = nil then
    Caption := '';
  else
    Caption := FDataLink.Field.AsString;
end;
```

Override the component constructor *Create* method. In the implementation of *Create*, create the *FDataLink* object, and assign the private *DataChange* method to *FDataLink*'s *OnDataChange* event. Example:

```
...
public
  constructor Create(AOwner: TComponent); override;
...

implementation

constructor TMyDBPanel.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  FDataLink := TFieldDataLink.Create;
  FDataLink.OnDataChange := DataChange;
end;
```

Override the component destructor *Destroy* method. In the implementation of *Destroy*, set *OnDataChange* to nil (avoids a GPF), and free *FDatalink*. Example:

```
...
public
  destructor Destroy; override;
...

implementation

destructor TDBPanel.Destroy;
begin
  FDataLink.OnDataChange := nil;
  FDataLink.Free;
  inherited Destroy;
end;
```

Save the unit and install the component (see the Users Guide, and the Component Writers Guide in Delphi Help for more on saving units and installing components).

To test the functionality of the component, add a *TTable*, *TDatasource*, *TDBNavigator* and *TDBPanel* to a form. Set the *TTable DatabaseName* and *TableName* to 'DBDemos' and 'BioLife' respectively, and the *Active* property to *True*. Set the *TDatasource.Dataset* property to *Table1*. Set the *TDBNavigator* and *TDBPanel.DataSource* property to *Datasource1*. The *TDBPanel.DataField* name should be set as 'Common_Name'. Run the application and use the navigator to move between records to demonstrate the *TDBPanel*'s ability to detect the change in data and display the appropriate field value.

Full source listing

```
unit Mydbp;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics,
  Controls, Forms, Dialogs, ExtCtrls, DB, DBTables;

type
  TDBPanel = class(TCustomPanel)
  private
    FDataLink: TFieldDataLink;
    function GetDataField: String;
    function GetDataSource: TDataSource;
    procedure SetDataField(Const Value: string);
    procedure SetDataSource(Value: TDataSource);
    procedure DataChange(Sender: TObject);
public
  constructor Create(AOwner: TComponent); override;
  destructor Destroy; override;
published
  property DataField: string read GetDataField write SetDataField;
  property DataSource: TdataSource read GetDataSource write SetDataSource;
end;

procedure Register;

implementation

procedure Register;
begin
  RegisterComponents('Samples', [TDBPanel]);
end;

function TDBPanel.GetDataField: String;
begin
  Result := FDataLink.FieldName;
end;

function TDBPanel.GetDataSource: TDataSource;
begin
  Result := FDataLink.DataSource;
end;

procedure TDBPanel.SetDataField(Const Value: string);
begin
  FDataLink.FieldName := Value;
end;

procedure TDBPanel.SetDataSource(Value: TDataSource);
begin
  FDataLink.DataSource := Value;
end;

procedure TDBPanel.DataChange(Sender: TObject);
begin
  if FDataLink.Field = nil then
    Caption := ''
```

```
  else
    Caption := FDataLink.Field.AsString;
end;

constructor TDBPanel.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  FDataLink := TFieldDataLink.Create;
  FDataLink.OnDataChange := DataChange;
end;

destructor TDBPanel.Destroy;
begin
  FDataLink.Free;
  FDataLink.OnDataChange := nil;
  inherited Destroy;
end;

end.
```

Original resource:	The Delphi Pool
Author:	Unknown
Added:	2012-07-08
Last updated:	2012-07-08