DataSnap - Business Display Demo.

Table of Contents

Introduction

Who should read this paper

The project is written to give a practical example of multi-tier software development with DataSnap in Delphi and C++ Builder and provide an overview of a number of language features and raises discussion about their usage with practical examples.

This paper is aimed at Software Developers, Development Team Leads and System Architects. The project has a range of isolated and integrated examples giving everyone from beginner to advanced professionals the chance to take something from this paper.

Background

The paper covers a specific DataSnap project written to enable the display of Business information with the idea of showing code in action and give the reader the ability to learn from it. The idea for the project came from a discussion with a partner who wanted to show live time data from multiple systems in a managed way. From that discussion a DataSnap demo was started and, well, its still growing.

Following initial discussion it was decided to make this demo flexible enough to accept a range of configurations and produce a variety of outputs. The example was initially refactored to make it core design appropriate for multiple different business practices - It is however, a working demo; there is a lot in here that you could take away and get working into your own applications very quickly, but there is nothing to stop you refactoring it further. It really is about getting you started with DataSnap and a number of other language features in Delphi.

Aim of this paper

The aim of this project was to help developers see a practical example of DataSnap in action and introduce multi-tier development using Delphi and C++ Builder. The project has been written to help developers of all levels develop their stills and thus uses a number of different language features to provide a basis for a number of supporting papers and videos.

The features are covered in the code are from Beginner & Intermediate level to Advanced level. As your ability improves, you should be able to dip in and out of the project and gain a deeper understanding of what it does and how it works in each section.

It is hoped that the aspects covered within this demo are not only relevant to this project, but also to your wider project code. Feel free to copy what you want from the project where you find it useful.

This paper and project is not designed to be a bible to good coding practice as there are examples where it could be coded in better way depending on the design decisions you have taken, and you are actively encouraged to play with the refactoring tools to improve your version, please remember that it is coded in a way to show example of language features were possible.

Setup and Configuration

System requirements

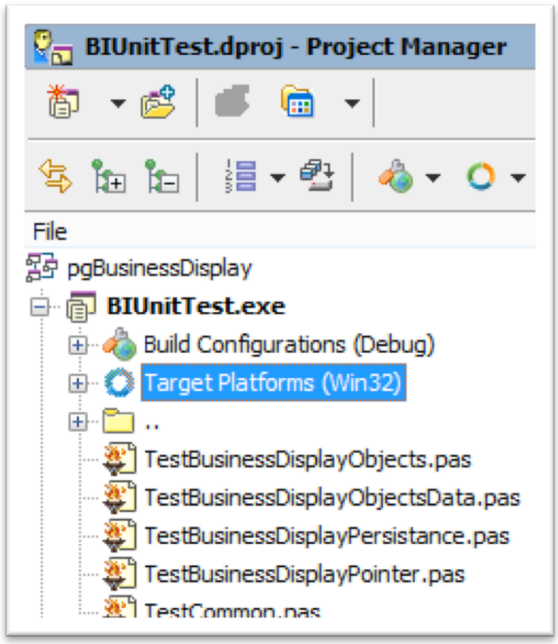
This demo is written for Delphi XE3 Enterprise version (or above)
As InterBase is included with the IDE, InterBase is used as the database, however if you wish to convert the data access to an alternative database engine, you can do this using the DBExpress components and updating the DB configuration.

Before you start

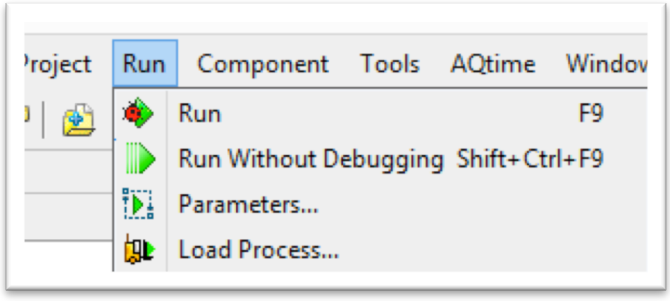
Please see the readme.txt for the default location of the database. Typically c:\data.

> *For more information on InterBase visit*
> *www.embarcadero.com/products/interbase*
> *www.embarcadero.com/database-in-action/InterBase-labs*

Once the database is in the correct location, Open the project group (File > Open Project) pgBusinessDisplay; double click on BIUnitTest.exe in the Project Manager (typically on the right hand side of the screen, or choose "View > Project manager") so it is highlighted and ready for compilation. Check the Target platform is selected to Win32 and Compile.

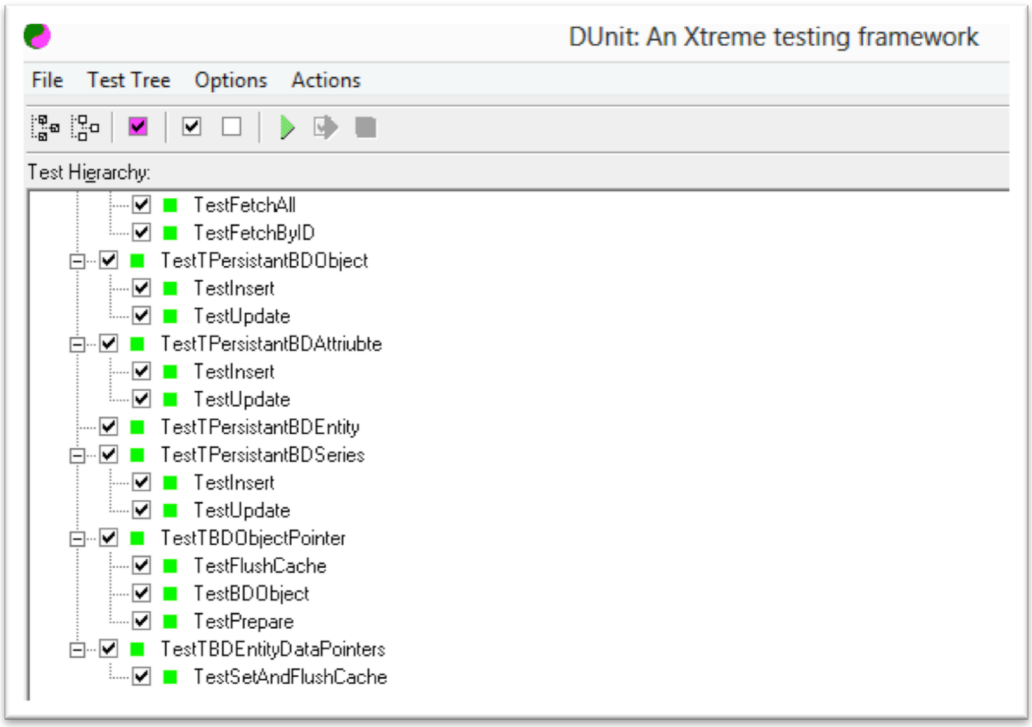


If you run the application (Run, Run without Debugging is quickest)



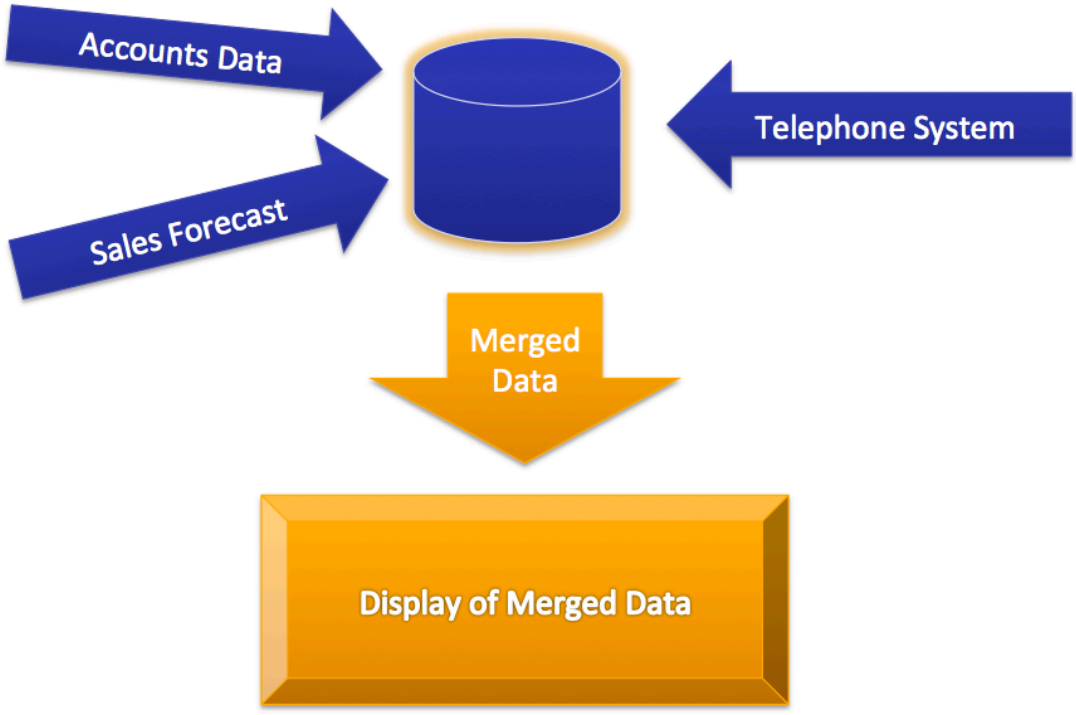
Once the application is open, you should be able to run the unit tests. (using the green run button).

If you scroll down you should be able to see the bottom tests (integration tests) passed. (These are the ones that mention TestPersistant / Test DB etc) These will show the database engine is up and running and ready.



Getting to know Business Display Demo (in short)

Use case examples



The picture above is a pictorial example of how the system works. Data from multiple sources is merged together and displayed. The initial requirements

were for sale, accounts and phone system data to be brought together to show number of calls and time on the phone per sales person each day and the value of sales.

A second example of how to use the engine was written around motor racing. This looked at different races with different drivers with different data recorded per lap of the race.

Building the functionality

From this simple use case we ended up with 3 core concepts: Series, Entity and Attribute as the table below defines.

Core Definitions

	Series	Entity	Attribute
Sales Display	Days / Quarters etc	Sales People	Forecast, Target, Actual
Motor Racing	Races	Drivers	Top Speed End of Lap Position Lap Time Time in Pits

These three concepts enable the system to be setup ready for differing usage. The system follows this simple rule

Rules

A Series will contain a list of Entity and Attributes that will be recordable for that Entity. Attributes and Entities may appear in multiple series, but do not always have to appear.
So each Series will be associated with multiple Entity / Attributes.

Design decisions taken

This may not be extensive, but in summary:

The system will run with one configuration at a time. You can swap the database to run multiple data structures. (eg. Sales and Motor Racing would use two separate databases but would work via the same engine)

Behind the system storage will be in a Database. TDataSet will be used as a base point for data connection and it is assumed that any Query component will have properties and methods for SQL, Params, ParamByName etc accessable.

It is not envisaged that large scaled images will be passed via the application. It is therefore assumed that having the image as part of the base object used in the application will provide a minimal performance hit, but the benefit of less data

packets being sent and easier consumption is acceptable against the cost. If you want to make this more scalable, then the end user can add extra calls to the server to not transport the image if they so desire.

The client should not be aware of the data it exports, but should work within the rules to expose the data.

Code should be sharable between applications with only minimum reliance on other units.

Core units and objects

In-sort, the engine has been written to enable custom application to be written that can then insert data via the exposed API (written using DataSnap) The core file to look at on the server products is **unitSmBusinessDisplay.**

In this unit of code you can see the server side methods and the roles they are allocated to. You will also see the role based authentication implemented above each method as an attribute.

On Disk there are "Common" folders. Each common folder has files that you should be aware of. There are 2 core Common folders. One for all applications and the other for client applications.

The root common folder includes the core objects and also server side common objects.

The client/common folder has files that are appropriate for use in multiple client applications, including files connecting to the DataSnap server (the proxy is the file generated that has the contract between the server and client for remote connect) and also units that give simple calls for connecting to the server methods from the client.

Language features by project unit

.

<TBC> This will be expanded along with the coming videos. The project includes examples of:

- *Beginner*
  - Language Features, Loops, Classes, Inheritance, DBExpress, Interfaces, InterBase, Unit Testing, Implementing Bubble Sort, Documentation Insight
- *Intermediate*
  - Class Helpers, Generics (Collections, TDictionary), Custom Notification Events, Anonymous Methods, TDictionary, FireMonkey Controls, FireMonkey Styles, Caching
  - DataSnap Features (Security Roles + more to come)
- *Advanced*
  - DataSnap Object Marshaling, RTTI (Generics, TValue), Custom Attributes

DataSnap Resources

Recommended Supporting Articles

If you like Video and learning from practical examples, then check out DataSnap in Delphi-Labs videos by Pawel Głowacki

10+ lessons with short bite size videos to dip in and out of.

http://www.embarcadero.com/rad-in-action/delphi-labs

If you want some really good documented overview that then takes you step by step into the different capabilities of the components, then check out DocWiki:

The following are two great articles (with illustrations) to make it clear how DataSnap architecture works.

http://docwiki.embarcadero.com/RADStudio/XE3/en/Developing_DataSnap_Applications

http://docwiki.embarcadero.com/RADStudio/XE3/en/DataSnap_Overview_and_Architecture

InterBase is the database used in the demo. For more information on InterBase visit

http://www.embarcadero.com/products/interbase
http://www.embarcadero.com/database-in-action/InterBase-labs

https://downloads.embarcadero.com/free/interbase
http://docs.embarcadero.com/products/interbase/

Further Reading: DocWiki articles on DataSnap.

- DataSnap Overview and Architecture
- DataExplorer support for DataSnap
- Using a DataSnap Server with an Application (Tutorial)
- DataSnap Server Application
  - Creating a DataSnap Server
  - DataSnap Server Wizard
  - DataSnap REST Application Wizard
  - DataSnap WebBroker Application Wizard
  - Generating DataSnap Server Method Proxies
  - Exposing DataSnap Server Methods
  - Using Callbacks
  - Server Class LifeCycle
  - Monitoring and Controlling DataSnap TCP/IP Connections
- DataSnap Client Application
  - Connecting the Client to DataSnap Server
  - Filtering DataSnap Byte Stream
  - HTTPS Certificate Verification
- JSON
  - Serializing User Objects
- REST
  - Request Filters
  - JavaScript REST Proxy
  - DBX Parameter Caching
  - REST Heavyweight Callbacks
  - Authentication with a JavaScript Client
  - JavaScript client Sessions
  - DataSnap REST Messaging Protocol
- Authentication and Authorization
- Server Side Session Management
- Cloud Computing with DataSnap
- Deployment Manager
- DataSnap Connectors for Mobile Devices