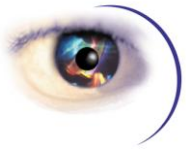

A Common Problem in Using Rich Internet Application Services

by Andrew Dallas, CTO, Full Spectrum Software

Overview

This highly technical article describes how to resolve the issue of data not being committed to a back-end database when using the intuitive approach of using data bound controls in a Rich Internet Application in either Windows Presentation Foundation (WPF) or Silverlight. We've found the online resources a bit misleading or unclear and we think many engineers will find that when using RIA Services in a WPF or Silverlight application, the Visual Studio Wizards can generate code that will not always work as expected.

The Usual Approach

The standard practice in using RIA Services is to first create an ADO.NET Entity Data Model to model the back end data store. The next step is to create a Domain Service Class and configure it to use whatever resources are needed from the Entity Data Model to support that service. For instance, in a medical application you may be creating a Domain Service for managing patient information. You might first create a "PatientService" Domain Service Class and select the relevant patient related tables, views and stored procedures. If you want to support editing the data in the database, you would make sure to check the Enable Editing checkbox in the wizard to generate the required methods. This creates a Domain Context class named "PatientContext" for your use in the code.

So far, so good. All of this works as intended to provide easy access to data in the back end database. The next step is to provide a view for this data in your user interface (UI). As an example, you might choose a DataGrid to display the data from a table in the database. A relatively easy approach is to drag the appropriate resource from the Data Sources window in Visual Studio to your design surface. Dragging a table resource will, by default, create a DataGrid in your UI page or window. If you examine the generated XAML code, you will see a DomainDataSource object declared just above the new DataGrid. This DomainDataSource represents your PatientContext object and the DataGrid's ItemSource is bound to this object.

Then You Test It

If you build and run your project you will see a table displayed with all the data from the associated table in your database. All is good or so it seems. Let's say you now want to allow the user to edit the data contained in the table. The DataGrid allows in-cell editing by default unless of course you set the IsReadOnly property to true. You select a cell in the table and click again to enter edit mode. You change the data and tab to another cell

or press Enter to commit the data. The data is in fact changed on the screen. However, if you reload or restart the application, you will notice the old data is displayed. The update was never sent to the database.

It appears that two-way binding is not working correctly when the DomainDataSource is used to connect your Domain Context to the DataGrid. This presents a challenging debugging situation unless you know the secret.

The Solution

Applying the Model View View-Model (MVVM) design pattern provides a way to correct the problem.

You can instantiate your Domain Context object in code in your ViewModel object instead of declaring it in XAML. You then provide an IEnumerable<EntityClass> property in the ViewModel that represents the collection of EntityClass objects in the database table. The EntityClass in this case would be the specific class that was generated by the Domain Service Class for the specified table such as "Patient".

After instantiating the Domain Context, you set the IEnumerable property to the appropriate Entity Set property of the Domain Context. Finally, you declare the ViewModel object in the XAML instead of the DomainDataSource object and you bind the DataGrid's ItemSource property to the IEnumerable property of the ViewModel. After building and running this code, you should now find that updated data in the UI is now persisted in the database.

Hopefully this highly technical article will save many people scads of time debugging and searching for solutions. Contact us if you have a better solution!

About Full Spectrum Software

Full Spectrum Software is an ISO 13485 certified, 15 year old consulting firm specializing in the development of embedded and applications software for the medical, life sciences and scientific industries. The company has delivered over 100 commercial products to market.

About the Author

Andrew Dallas, the firm's CTO, is widely considered one of the leading authorities on best practices in FDA and ISO 13485 controlled software projects Andrew serves on the Editorial Advisory Board of Medical Device and Diagnostic Industry magazine and he has published extensively in major trade and peer reviewed technical publications.

Contact Cindy Larkin, ClientServices@FullSpectrumSoftware.com
(508) 620-6400 • 225 Turnpike Road • Southborough, MA 01772 •
www.FullSpectrumSoftware.com