Table of Contents

[1 Overview 1](#_Toc418502604)

[2 Components 1](#_Toc418502605)

[2.1 Record Data Layer 1](#_Toc418502606)

[2.2 Application + View Models 2](#_Toc418502607)

[2.3 Windows Presentation Foundation UI 2](#_Toc418502608)

[2.4 Prism Infrastructure + Modules 3](#_Toc418502609)

# Overview

This document gives a technical overview of the components in the JosephM solution which are primarily for building a windows application with a WPF User Interface

The windows application framework is written with the MVVM pattern built upon .net framework 4.5. The framework has been designed and implemented to allow a pluggable (module based) application architecture

# Components

## Record Data Layer

The Myriad.Record project contains the data access layer

The main components are

* IRecordService – this is an interface to a data source with basic CRUD and other IO operations as well as provision of metadata
* IRecord – this is an interface to a late bound record with field/properties accessible by a field name exposed by the IRecordService

Concrete implementations of IRecordService will work with a specific source and implementation of IRecord for the specific types of record provided by that source

Examples are

1. The XrmRecordService implementation interfaces to a Dynamics CRM instance through its web services and exposes the XrmRecord implementation of IRecord. Originally XrmRecord was simply a wrapper to a CRM SDK Entity object however mapping of data to the UI had performance issues when converting types so the implementation was changed to convert the specific data types to those supported by the IRecord framework on load and save
2. The ObjectRecordService implementation is used for any type of .Net C# CLR object and exposes the ObjectRecord implementation of IRecord

The main purposes of this component is that it provides an abstract data source without knowledge of the specific data sources context

For example

1. The UI framework component only has knowledge of the Interfaces meaning that any implementation of IRecordService will work with the UI. For example The UI may be used to input/output crm records, application settings or general user input through an IRecordService instance
2. Implementations of IRecordService may be swapped out for testing

## Application + View Models

The Myriad.Record.Application is in effect the ViewModel layer as well as additional classes to support an application

1. ViewModels for binding to the user interface
   1. For IO related data (RecordEntryFormViewModel) these interface to and from IRecordService instances which will load specific data/view model types and perform CRUD operations on it
2. IApplicationController for
   1. Managing threads in an application
   2. User Popup Messages
   3. Various other pieces specific to an applications runtime context
3. Various other components to support the user interface and navigation in an application

The components purpose is the core objects for an application implementation to work with to at runtime. The idea is it has no knowledge of the specific application which is running and therefore supports

1. The Prism Windows Application Implementation
2. Test Scripts
3. Console Applications If Necessary

## Windows Presentation Foundation UI

The Myriad.Wpf project provides a User Interface (view) layer for the MVVM pattern

These UI components are used by the Myriad.Prism.Infrastructure application implementation for the user interface

## Prism Infrastructure + Modules

The Myriad.Prism.Infrastructure project contains the components for a windows application using the WPF UI to load modules and startup

The main components are

* PrismApplication – this is a base class for an application instance to load modules then run (startup) (see app.xaml.cs in a concrete implementation)
* PrismModuleBase – this is a base class for a plugin module to load into a PrismApplication instance. There are several extended base classes for
  + ServiceRequestModule – for a plugin where the application performs some task then completes
    1. User enters some data for the application to perform a task
    2. The application performs the task keeping the user informed of progress
    3. The task completes and the application provides the user with a summary of what happened during the tasks execution
  + SettingsModule – for a plugin where the user may edit and saves user-defined settings for the application. The settings may then be loaded at runtime to drive behavior of the application or services