3rd party software

* MVVMLight Framework  
  “The MIT License”  
  <http://www.galasoft.ch/mvvm/getstarted/>
  + Messenger
  + ViewModelBase
  + ViewModelLocator
  + Code Snippets
* Unity Framework (IOC Container) – Microsoft Patterns & Practices  
  Microsoft Public License (Ms-Pl)  
  <http://unity.codeplex.com/>
  + Inversion Of Control Container
  + Dependency Injection
* xUnit (Unit Tests)   
  Ms-Pl  
  <http://xunit.codeplex.com/>
* Moq (Interface Mocking)   
  New BSD License (BSD)  
  <http://moq.codeplex.com/>
* WPF Localization  
  The Code Project Open License (CPOL) 1.02  
  <http://www.codeproject.com/KB/WPF/WPF_Localization.aspx>
* WPF Toolkit  
  MS-PL  
  <http://wpf.codeplex.com/>
* Log4net  
  Apache License, Version 2.0  
  <http://logging.apache.org/log4net/index.html>

# Architektur

MVVM Pattern Tutorials

* <http://blog.lab49.com/archives/2650> (video)
* <http://blogs.msdn.com/dancre/archive/2006/10/11/datamodel-view-viewmodel-pattern-series.aspx>   
  (DM-V-VM ähnlich zu MVVM)
* <http://msdn.microsoft.com/en-us/magazine/dd419663.aspx>

Das Projekt ist auf folgende Assemblies aufgeteilt:

* MVVMDiversity  
  View Klassen (GUI)
  + Nahezu keine Logik
  + Nur Oberflächennahes Code Behind
  + Per Databinding:  
    Zeigt Daten aus dem ViewModel an,  
    Leitet Aktionen an das ViewModel weiter (ICommand Interface)
  + App Klasse (Bootstrapper)
* MVVMDiversity.Common
  + Service Interfaces  
    Abstrahieren den Zugriff auf die Datenbank
  + Messages  
    Verständigung zwischen den ViewModels  
    Verständigung von VM zu View (Dialoge)  
    Globale Mitteilungen bei Statusänderungen
* MVVMDiversity.ViewModel  
  ViewModels  
  Aufbereitung der Daten zur Anzeige  
  Weiterleitung der Aktionen an die Services
* MVVMDiversity.Model  
  Reine Datenklassen
* MVVMDiversity.Services  
  Release Service Implementierungen
* MVVMDiversity.DesignServices  
  Debugging/Design Service Implementierungen  
  (Beispielfunktionalität/-daten)

# View

* **MainWindow**Zeigt die Pages eingebettet an und zeigt deren Eigenschaften an (Title, Description, Next, Previous)   
  Stellt die Buttons zur Seitennavigation und die einheitliche Menüleiste zur Verfügung.
* **StartWindow**Splash Screen
* **Controls NameSpace**
  + DateRangeControl   
    Ermöglicht die Eingabe einer von-bis Einschränkung
  + NameValueControl  
    Ermöglicht die Eingabe einer Text-Einschränkung
  + InterfacingPasswordBox  
    Ermöglicht die sichere Weitergabe eines Passworts an das ViewModel
* **Dialogs Namespace**
  + **YesNoWindow, MessageBoxWindow**Wiederverwendbare MessageBox Ersatz Fenster
  + **AboutView, OptionsView, TaxonView**Aufgabenspezifische Dialoge
* **Pages Namespace**Usercontrols, die als „Seiten“ im Hauptfenster eingebettet werden.

# ViewModel

* **ViewModelLocator**
  + Klasse zur zentralen Erstellung der ViewModels.
  + Einzige Instanz als Resource (Key: „Locator“) im Application Objekt
  + Die Views beziehen ihre jeweiligen ViewModels per Databinding davon.
* **PageViewModel**gemeinsame Basisklasse für SeitenViewModels.
* **xyzViewModel** 
  + ViewModel für die jeweilige View  
    (Spezifische Dialoge, Seiten, Hauptfenster)
  + Jedes ViewModel stellt   
    die anzuzeigenden Daten nahezu 1:1 als Properties   
    die möglichen Aktionen als ICommand-Implementierungen  
    bereit
  + Alle Methoden der öffentlichen Schnittstelle eines VM werden ausschließlich auf dem UIThread von WPF aufgerufen.   
    Alle aufwändigen Aktionen (Aufrufe an Services etc.) müssen auf Hintergrund Threads ausgeführt werden.

# Interfaces

* IConnectionManagementService  
  Verwaltet die Verbindungen
* IConnectionProvider  
  Bietet Zugriff auf die Serializer
* IConnectionProfilesService  
  Ruft die verfügbaren Verbindungsprofile ab (Server, IP,Port, Catalogs)
* …

# Messages

* ConnectionStateChanged, SyncStateChanged, SettingsChanged  
  Benachrichtigen alle Interessenten (VMs), dass sich etwas geändert hat.
* CustomDialog  
  Veranlasst, dass das Hauptfenster den angeforderten Spezifischen Dialog anzeigt.  
  (Per mit versandtem *Dialog* Wert)
* NavigateToPage  
  Veranlasst, dass das Hauptfenster die angeforderte Seite anzeigt.  
  (Per mit versandter *IPageViewModel* Instanz)

# Localization

Mit der Loc Extension werden die Lokalisierten Ressourcen anhand eines Key strings aus der Resources.resx Datei im MVVMDiversity Projekt abgerufen.   
Statische Ressourcen werden in XAML mit {Loc *key*} gesetzt.   
Dynamische über die LocBinding Erweiterung mit {LocBinding }.

# Dependency Injection

Alle Klassen mit Service Abhängigkeiten sollten über den IOC Container „aufgelöst“ werden.  
Unity kann Abhängigkeiten u. A. über Constructor Injektion oder Property Injektion erfüllen.

*class Dependant   
{*

*public Dependant(IService1 Dependee)  
 {}  
  
[Dependency]  
public IService2 Dependee{get; set;}*

*}*

*…*

*var ioccontainer = new UnityContainer();*

*ioccontainer.Register<IService1>(new Service1());*

*ioccontainer.Register<IService2>(new Service2());*

*Dependant d = ioccontainer.Resolve<Dependant>();*

*…*

# 