Einfügen einer neuen Datenklasse in DiversityPhone

# Import der Zugehörigen Tabelle aus der Serverdatenbank

**DiversityService/Model**

Für jede Datenbanktabelle, die verwendet werden soll muss eine Datenklasse angelegt werden. Diese muss für die Verwendung durch Petapoco (ORM) geeignet sein.

Näheres: <http://www.toptensoftware.com/petapoco/>

# Einfügen der nötigen Zugriffsmethoden

**Im DiversityService Projekt**  
In das IDiversityService Interface

[OperationContract]

IEnumerable<Term> GetStandardVocabulary();

In die DiversityService Implementierung

IEnumerable<Term> GetStandardVocabulary();

# Anlegen der (reduzierten) Modellklasse

**DiversityPhone/Model**

Dies ist nur nötig für Klassen, die auch clientseitig verwendet werden. Das Datenmodell kann dahingehend modifiziert werden, dass nur eine Teilmenge der Felder verwendet werden muss. Werden zusätzliche Felder clientseitig benötigt, werden sie an dieser Stelle hinzugefügt. Die Anpassung der Datenmodelle wird von der Klasse DiversityService übernommen (siehe DiversityService (TODO)).

Für die Synchronisation Implementieren alle editierbaren Klassen das Interface IModifyable

public enum ModificationState

{

New,

Unmodified,

Modified

}

public interface IModifyable

{

/// <summary>

/// Tracks the persistance status of an object.

/// null - Newly created, not yet persisted

/// false - persisted remotely, local copy unchanged

/// true - persisted only locally OR the local copy has been changed

/// </summary>

ModificationState ModificationState { get; set; }

}

Wenn verschiedene Tabllen eine identische oder vergleichbare Struktur haben, können diese zu einer Klasse zusammengefasst werden z.B. Term.

Die Klassen müssen für Linq2SQL mit entsprechenden Attributen versehen werden.

Näheres: <http://msdn.microsoft.com/en-us/library/bb386971.aspx>

# Anlegen einer Tabelle im DatenKontext

**Im DiversityPhone Projekt,   
DiversityDataContext**

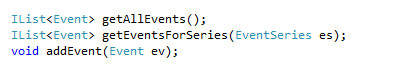
public Table<Event> Events;

# Einfügen der nötigen Zugriffsmethoden

**Im DiversityPhone Projekt,   
Ordner Services**

In das IOfflineStorage bzw. IOfflineFieldData Interface  
In die OfflineStorage Implementierung

Standard: Klassenweise Gruppierung



# ViewModels der gewünschten Seiten/Elemente anlegen

**Im DiversityPhone Projekt,   
Im Ordner ViewModels**

Für alle Seiten, die angezeigt werden sollen müssen ViewModels angelegt werden.

Es werden insgesamt 3 Klassen für jede Felddatenklasse des Typs EventSeries, Event, Specimen und IdentificationUnit angelegt, sowie 2 Klassen für alle weiteren editierbaren Klassen. Für Read-Only-Klassen werden keine eigenen Seiten zur Ansicht und somit auch keine ViewModels benötigt.

Darüber hinaus werden für folgende Prozesse ViewModels angelegt:

* UserProfil
* Kartenauswahl
* Kartenansicht
* Multimediaaufnahme
* MultimediaAnsicht
* OberflächenCustomizing

Basisklasse [Felddatenklasse]VM z.B.

public class EventVM : ElementVMBase<Event>

Enthält:

* Property Model mit Verweis auf das Objekt der zugrunde liegenden Klasse
* Property Description mit anzuzeigendem Text
* Property Icon für das Icon
* ReactiveCommands Edit und Select (falls sinnvoll, also falls entsprechende Seiten existieren)

Die ViewModels für Seiten leiten alle von PageViewModel ab.

Dieses enthält:

* StateObservable:   
  Observable, das die PageState Objekte enthält, die dem ViewModel beim aufruf seiner Seite übergeben werden. Aus diesem PageState nimmt das ViewModel die Informationen darüber, was angezeigt werden soll.
* CurrentState Property:  
  Letzter erhaltener PageState
* SaveState() Methode:  
  Diese kann von abgeleiteten ViewModels überschrieben werden, um beim Verlassen der Seite den Seitenzustand im PageState Objekt zu sichern.

Klasse zum Editieren Edit[Felddatenklasse]VM z.B.:

public class EditEVVM : PageViewModel

Enthält:

* Property Model mit Verweis auf das Objekt der zugrunde liegenden Klasse
* ReactiveCommands Edit (für Editiermodus), Save und Delete
* Editierbare Felder der zugrunde liegenden Klasse
* Regeln zur Freigabe des Speicherns (neu!) Verallgemeinerung von cansave
* Methode updateView([Feldatenklasse] f)
* Methode executeSave()

Klasse zur Auswahl von Kindklassen (nur ES, EV, CS , IU) View[Felddatenklasse]VM z.B.

public class ViewEVVM : PageViewModel

Enthält:

* Property Model mit Verweis auf das Objekt der zugrunde liegenden Klasse
* Liste von ViewModells der Kindklasse IList<KindklasseVM>
* Property Current mit ViewModel der aktuellen Klasse
* ReactiveCommand Add

# Seiten- ViewModels im ViewModelLocator registrieren

**Im DiversityPhone Projekt,   
Im Ordner ViewModels**

* IOC Constructor Lambda
* Zugriffs-Property:

public ViewEVVM ViewEV { get { return \_ioc.Resolve<ViewEVVM>(); } }

# View erstellen

**Im DiversityPhone Projekt,   
Im Ordner View**

PortraitPage Template

Template in der Templateklasse erstellen/wählen

🡪DataContext der Seite per Binding auf das ViewModel setzen:

DataContext="{Binding Source={StaticResource Locator}, Path=ViewEV}"

# Seite registrieren

**Im DiversityPhone Projekt,   
Im Ordner Services**

🡪Zum Page enum hinzufügen

🡪Im NavigationService eintragen (Navigate() und evtl. noch mehr Message Listener für Select oder Edit Nachrichten)

# Dokumentieren

Anpassungsschritte der neu eingefügten Klassen im Dokumentationsfile

.docx hinterlegen.<