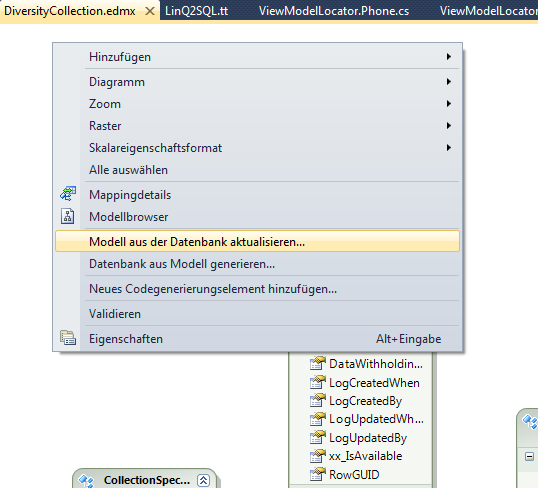
Einfügen einer neuen Datenklasse in DiversityPhone

# Import der Zugehörigen Tabelle aus der Serverdatenbank

**Projekt DiversityService  
Im Entity Framework Designer**

Es müssen alle Tabellen der Serverseite importiert werden, mit denen in irgendeiner Weise kommuniziert werden soll.

Das importierte Schema ist zu Datenbanken mit identischem Schema kompatibel.



# Anlegen der (reduzierten) Modellklasse

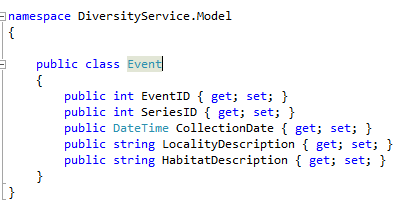
**Im DiversityService.Model Projekt**

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 erhalten alle editierbaren Klassen die Felder isModified und LogUpdatedWhen.

Readonly-Klassen erhalten nur das Feld isModified und es wird im Kommentar darauf hingewiesen, dass es sich um eine ReadOnly-Klasse handelt.

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



# Einfügen der nötigen Zugriffsmethoden

**Im DiversityService Projekt**  
In das IDiversityService Interface  
In die DiversityService Implementierung  
🡪Bilden von der Enitity Framework Modelklasse auf das reduzierte Datenmodel ab.

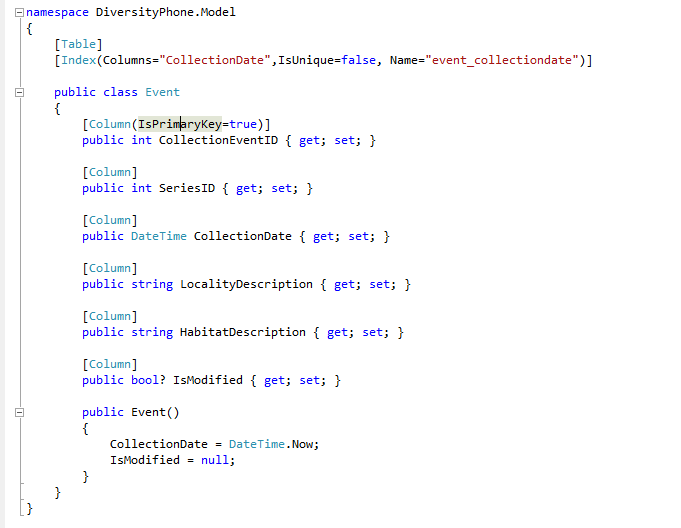
[OperationContract]

IEnumerable<Term> GetStandardVocabulary();

# Anlegen einer annotierten identischen Modellklasse

**Im DiversityPhone Projekt,   
Ordner Model**

Eventuell nur auf dem Mobilgerät verwendete Properties einfügen (IsModified z.B.)



# Anlegen einer Tabelle im DatenKontext

**Im DiversityPhone Projekt,   
DiversityDataContext**

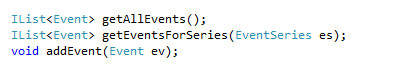
public Table<Event> Events;

# 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 VieModels 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

Allen Klassen ist gemein:

* Ableitung von ReactiveObject
* \_IMessageBus messenger zur Navigation
* IList<Disposable> \_subscriptions zur Steuerung der GarbageCollection

Basisklasse [Felddatenklasse]VM z.B.

public class EventVM : ReactiveObject

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

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

public class EditEVVM : ReactiveObject

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 updateModel()
* Methode updateView([Feldatenklasse] f)
* Methode executeSave()

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

public class ViewEVVM : ReactiveObject

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.<