Quill及びS2Form サンプル解説

S2Container.NET, S2Dao.NET コミッタ
藤井 宏明

概要

- S2Container.NETの機能QuillとS2Formを使ったWindowsFormサンプルの解説です。

- アプリケーション・アーキテクチャー
- ソリューション構成
- プロジェクト概要
- サンプル説明

Ⅰ．アプリケーション・アーキテクチャー

アプリケーション･アーキテクチャー

このサンプルにおけるアプリケーション構造は、
次のような特徴をしています。

- レイヤー･アプローチ
- 各画面の目的に応じた柔軟なアプローチ方法をベースにしたクラス設計

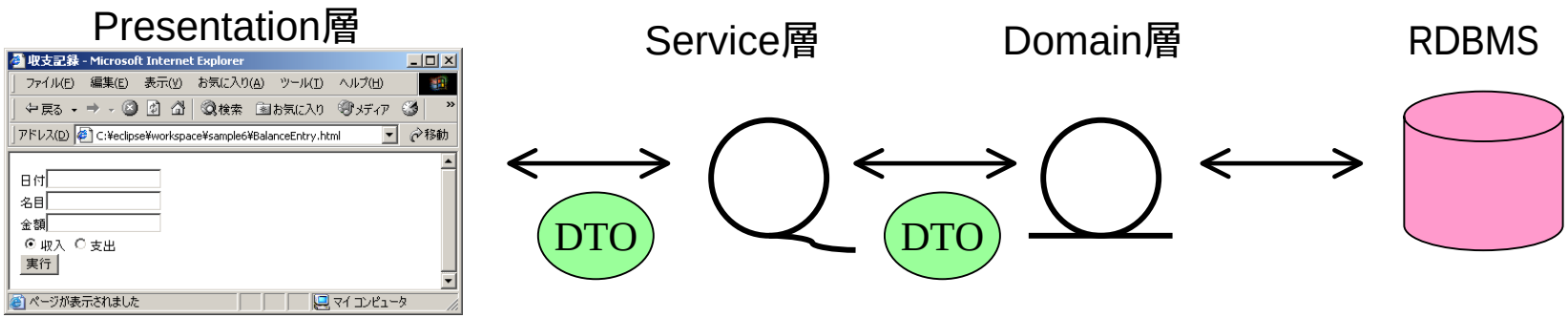
当然、DIコンテナ(=S2Container.NET）を
使っています。

レイヤーアプローチ

レイヤーは次のものを用意しています。

この層間をData Transfer Object(DTO)を使ってアクセスし、直接アクセスを禁止します。

- Presentation層
- Service層
- Domain層



Presentation層

主として、WindowsFormで構成される層です。

- Formクラスと画面遷移インターフェイス
  - Formクラスもdiconファイルに含めて、DIコンテナ管理にします。
  - FormクラスはS2Formを継承します。
- Programクラス(起動用）
  - S2Containerを初期化しています。これ自身はDIコンテナ管理外です。
  - Seasar.Windows.S2ApplicationContextを使用して、アプリケーションコンテクストによる起動にしています。
  - Seasar.Windows.S2ApplicationContextはdiconファイルで起動フォームを指定します。

```
<component name=“AppContext" class="Seasar.Windows.S2ApplicationContext">
      <arg>container</arg>
      <property name=“MainForm”>”FrmMenu"</property>
</component>
```

Service層

DAOやロジックをコントロールする層です。

次のインターフェイス・実装クラスは、Quillの仕様に合わせて作成します。

- IBaseServiceインターフェイス
  - Service層のインターフェイスの基底インターフェイス
- BaseServiceImplクラス
  - IBaseServiceの実装クラス
- Service層インターフェイス
  - IBaseServiceを継承する
- Service層インターフェイスの実装クラス
  - BaseServiceImplを継承する

Service層のメソッド設計

画面で提供するサービスが何か？を考えます。

- 本当は概念モデルよりサービスを考えていくのかもしれませんが、簡単に設計するためにそうは考えません。
  - 画面一つに対応するサービスのインターフェイスを一つ用意します。
  - 画面で発生するイベントに対応したメソッドをサービスに用意します。
  - 画面のコントロール初期化用の各画面共通メソッドはIBaseServiceに用意します。
  - 作成したサービスのインターフェイスはIBaseServiceを継承します。

画面のイベント　＝　サービスの提供するメソッド

Domain層

サービスが必要とするDomainへアクセスする層です。

- S2Dao.NETのインターフェイスやSQLファイル
  - S2Dao.NETで使用するインターフェイスやsqlファイル
- Domain層のインターフェイス
  - 必要とあれば、作成します。
- Domain層インターフェイスの実装クラス
  - 必要とあれば、作成します。

Domain層のメソッド設計

サービスが必要としているリソースが何か？を考えます。

- リソース・アクセスを提供するData Access Object(DAO)を作ります。
  - リソース（例えば、DBのテーブルや帳票)に1：1にDAOを用意します。
  - アクセスする種類がメソッドになります。

リソースアクセスの種類　＝　DAOの提供するメソッド

Service層とDomain層の間にある、DAOの組み合わせを再利用する層です。

- Logic層のインターフェイス
  - DAOインターフェイスの組み合わせパターンを規定するインターフェイス
- Logic層インターフェイスの実装クラス
  - DAOのインターフェイスの組み合わせを定義する実装クラス

実装はステートレスにする

Presntation層以外の各レイヤー（層）の実装クラスはステートレス、つまり状態を持たないようにします。

```
public class HogeServiceImpl : IHogeService
{
    private  Hoge  _state;
    private  int _arg1;
    // プロパティ、daoは省略

    public void GetData()
    {
        _state = _dao.GetData( _arg1 );
    }
}
```

```
public class HogeServiceImpl : IHogeService
{
    // ただし、daoは状態ではなく、DIする。

    public Hoge GetData(int arg1)
    {
        return (  _dao.GetData( arg1 ) );
    }
}
```

- 状態を持たないようにして、生成や呼び出しの順序などの依存性を低くします。

層間のデータのやり取りをするためのクラス

- PONO(Plain Old .NET Object)クラス
  - 普通のプライベート・フィールドとそのプロパティで構成されるクラス
  - S2Dao.NETとS2Formのコントロール用のクラス
  - S2Form用のクラスでは、プロパティ名をFormのコントロール名にあわせるか、FormのControl属性で関係付けを行います。

Ⅱ．Visual Studio ソリューション構成

ソリューションの構成

一つのソリューションは次のプロジェクトを保持します。

- Formsプロジェクト　　→Exe
  - WindowsFormを含んでおり、起動用構成になっている。
  - Presentation層に相当します。
  - ビルドイベントで、Testsプロジェクトの.diconファイルをコピーします。
- Logicsプロジェクト　　→DLL
  - Service層、Domain層、DTOを含んでいます。
- Testsプロジェクト　　→DLL
  - ユニットテスト用クラスと.diconファイルなど設定ファイルを含んでいます。

名前空間

プログラム、クラス、インターフェイス、リソースを所在を区別するために設定します。

- 名前空間を次のように設定します。
  - （会社名）.（製品名）.（サービス名）
- 基本的には、MSの指針に従います。
  - （サービス名）には、Forms, Logics, Tests, Utilsなどが入ります。

画面と画面処理用クラスを含んだメインプロジェクト

- WindowsFormクラス
- Programsクラス
  - アプリケーション起動用のクラス。
  - プロジェクトの起動用に設定されています。
- IFormDispatcherインターフェイス
  - 画面遷移用。
  - メソッドで画面遷移をコントロールします。

```
/// <summary>
/// 一覧フォームを表示する
/// </summary>
/// <returns>ダイアログ結果</returns>
[TargetForm(typeof (FrmEmployeeList), ModalType.Modal)]
DialogResult ShowFlowA();
```

- App.Config設定ファイル。ロードするDLLを設定します。
- Testsの.diconファイルを、ビルドイベントでコピーします。

Service層やDomain層などを構成するプロジェクト

- インターフェイスだけで構成します。
- 実装クラスはその下のImplフォルダに配置します。
  - Serviceフォルダ
    - Service層インターフェイスを格納
  - Daoフォルダ
    - Domain層インターフェイスを格納
    - S2Dao.NET用SQL文ファイルも一緒に格納
  - Logicsフォルダ
    - Logics層インターフェイスを格納
  - Dtoフォルダ
    - Domain層で使用するDTO用のPONOクラスを格納
  - Pageフォルダ
    - Formで使用するDTO用のPONOクラスを格納

ユニットテスト用クラスと.diconファイルを格納するプロジェクト

- ユニットテストにはS2Unit.NETを使用します。
- S2Container.NET用設定ファイルに三種類使用する。
  - Ex.dicon →DB接続用のファイル。
  - Example.dicon →DIするFormを含んだファイル。
  - ExampleLogic.dicon →Logics.Implのクラスを含んだファイル。(S2Unit.NETで使用する)
- App.Config
  - ロードするDLLを記述。

アプリケーションの起動

Programsクラスを使って実行します。

このクラスは名前空間とdiconファイル名を変更して、そのまま使います。

サンプルのExamples.diconの中で次のように設定しています。

```
<components>

  <component name="AppContext" class="Seasar.Windows.S2ApplicationContext" >
      <arg>container</arg>
      <!-- MainFormを初期起動フォームに変更する -->
      <property name="MainForm">FrmMainMenu</property>
  </component>

  <component name="FrmMainMenu" class="Seasar. S2FormExample.Forms.FrmMainMenu" />

</components>
```

画面遷移

画面遷移はIFormDispatcherインターフェイスを使ってコントロールし、フォームにDIする。

```
[Implementation]
[Aspect(“FormInterceptorr”)]
public interface IFormDispatcher
{
        /// <summary>
        /// フローAのフォームを表示する
        /// </summary>
        /// <returns>ダイアログ結果</returns>
        [TargetForm(typeof (FrmEmployeeList), ModalType.Modal)]
        DialogResult ShowFlowA();
}
```

```
<component name="FormInterceptorr" class="Seasar.Windows.AOP.Interceptors.FormInterceptor" />
```

WindowsFormの作り方１

- コンストラクタで画面の初期化をしない。FormのLoadイベントメソッドで行います。
  - 基本的にDIコンテナがFormを管理し、SingltonなオブジェクトのためOnLoadイベントメソッドで行います。
- 画面に値を渡すときや戻り値は、プロパティを作り、同名の引数をIFormDispatcher（画面遷移インターフェイス）の呼び出しメソッドに持たせます。
- Service層のインターフェイスのフィールドをprotected以上でFormに作成すれば、Quillがオブジェクトを自動的にセットしてくれます。ただし、そのオブジェクトはSingletonです。

WindowsFormの作り方2

- S2FormのDataSourceプロパティを使えば、コントロールに値のセットや取得が楽に出来ます。設定の仕方はControl属性を使って指定します。
- コントロールの値を更新して、PONOに反映するにはFormのValidateメソッドを実行するか、Control属性を利用します。

Service層、Domain層の作り方

- Domain層のインターフェイスのフィールドをService層の実装クラスにprotected以上で作成すれば、Quillがオブジェクトを自動的にセットしてくれます。
- Quillの仕様で作成します。トランザクションはService層の実装クラスの行わせるメソッドに、Aspect属性を使って指定します。

Ⅲ．サンプルの説明

サンプルで使用するテーブル1

T_EMP(従業員テーブル)

カラム名	論理名	型
N_ID	社員ID	オートナンバー
S_CODE	社員コード	テキスト型
S_NAME	社員名	テキスト型
N_GENDER	性別ID	数値型
D_ENTRY	入社日	日付/時刻型
N_DEPT_ID	部門ID	数値型

サンプルで使用するテーブル2

T_DEPT(部門テーブル)

カラム名	論理名	型
N_ID	部門ID	オートナンバー
S_CODE	部門コード	テキスト型
S_NAME	部門名	テキスト型
N_SHOW_ORDER	表示順番	数値型

サンプルで使用するテーブル3

T_GENDER(性別テーブル)

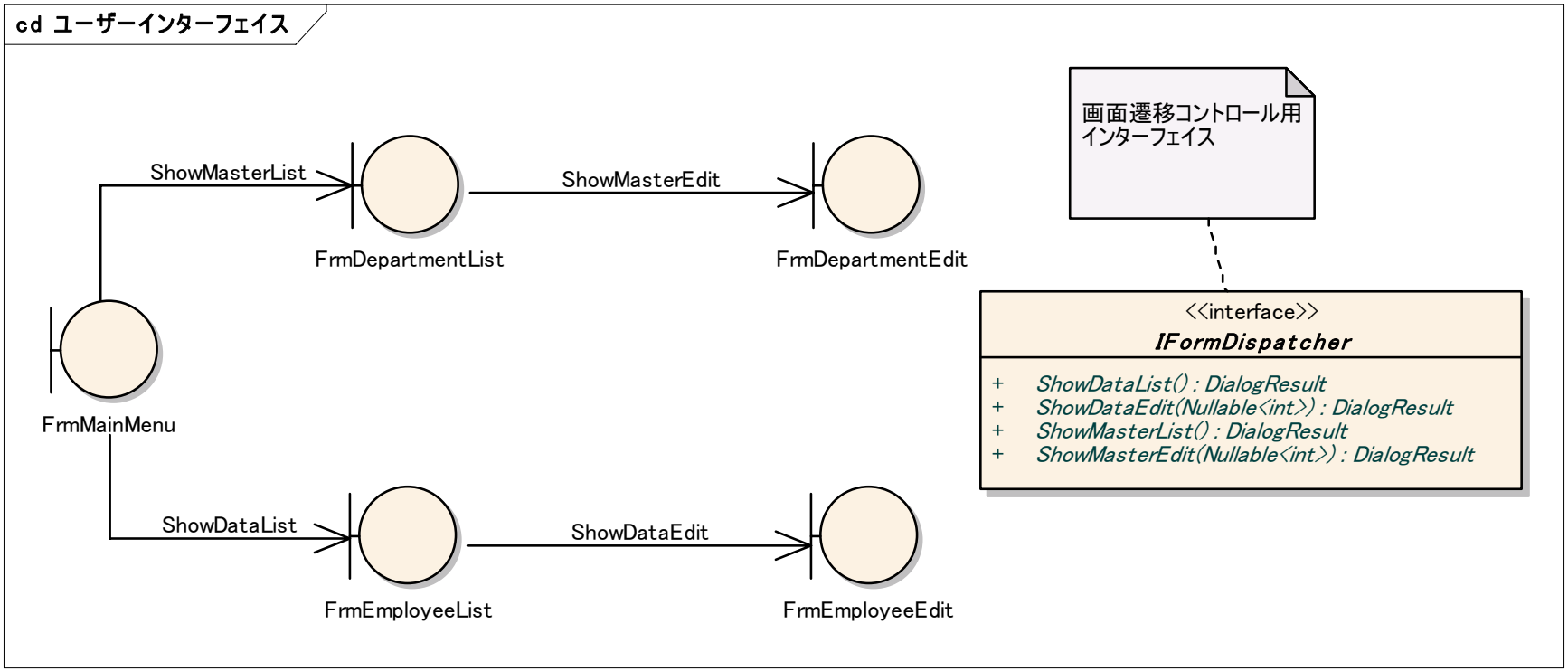
カラム名	論理名	型
N_ID	性別ID	オートナンバー
S_NAME	性別名	テキスト型

S2FormExampleプロジェクト

メニュー、社員、部門管理を含んだメインプロジェクト

- 画面（Frm～)クラス
  - すべてExample.diconでS2Containerに登録されています。
  - Example.diconで、FrmMainMenuを起動クラスに設定しています。
- Programsクラス
  - アプリケーション起動用のクラスで、定数でdiconファイルを指定しています。
  - プロジェクトの起動用に設定されています。
  - 二重起動を防止しています。
- IFormDispatcherインターフェイス
  - 画面遷移コントロール用。
- その他
  - App.Config設定ファイル。ロードするDLLを設定します。
  - Testsの.diconファイルとMDBファイルを、ビルドイベントでコピーしてます。

画面の遷移は次のようになります。



.diconファイル

- Ex.dicon (DB接続設定)
  - Providerやデータソース設定など

- Examples.dicon (フォーム用）
  - 起動フォームの設定、画面遷移のインターフェイスの設定、次のようにフォームを登録します。Ex.diconをincludeしています。

```
<component name=“FrmMain" class=“Seasar.WindowsExample.Forms.FrmMainMenu" />
<component class=“Seasar.WindowsExample.Forms.FrmDepartmentEdit”/>
<component class=“Seasar.WindowsExample.Forms.FrmEmployeeList” />
```

- ExampleLogic.dicon (S2Unit.NET用)

```
<component name=“departmentDao" class=“Seasar.WindowsExample.Logics.Dao.IDepartmentDao">
    <aspect >Ex.DaoInterceptor</aspect>
</component>
<component class="Seasar.Windows.Logics.Service.Impl.DepartmentEditServiceImpl" >
    <aspect pointcut=“ExecUpdate”>Ex.LocalRequiredTx</aspect>
</component>
```

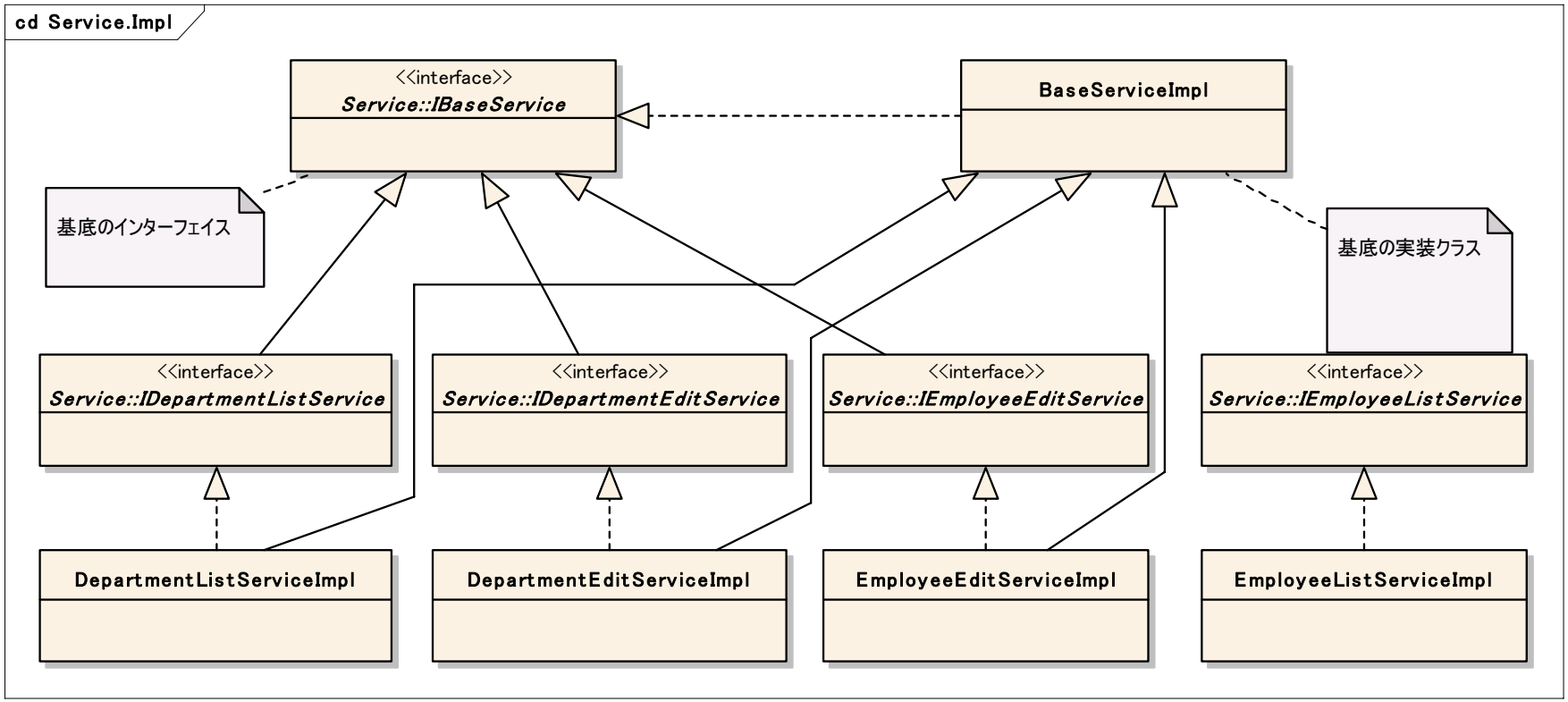
S2FormExamples.Logicsプロジェクト

Service層やDomain層などを構成するプロジェクト

- Serviceフォルダ
  - 基底サービス　…部門一覧、性別一覧を取得する。
  - 部門一覧、部門修正、社員一覧、社員修正サービス
    …一覧を取得する、更新する、削除する、CSVで出力する。
    - Implフォルダに実装クラスを配置
- Daoフォルダ
  - S2Dao.NET用インターフェイスとSQL文ファイル
  - CSV出力用インターフェイス
    - Implフォルダに出力用DAO実装クラスを配置
- Dtoフォルダ
    - S2Dao.NET用クラスに社員、部門、性別クラス
    - CSV出力用クラス

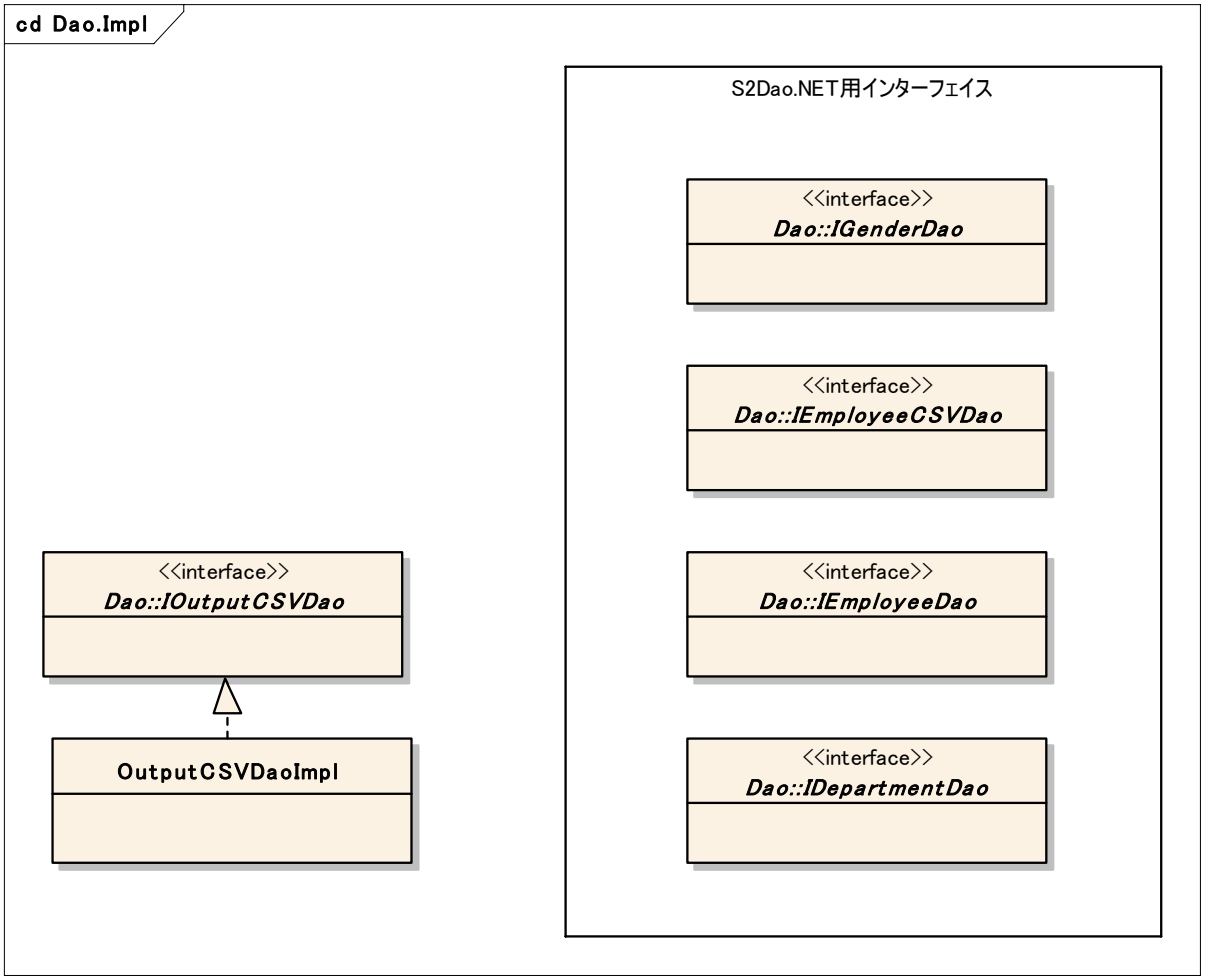
Service層クラス図

インターフェイスとその実装クラスです

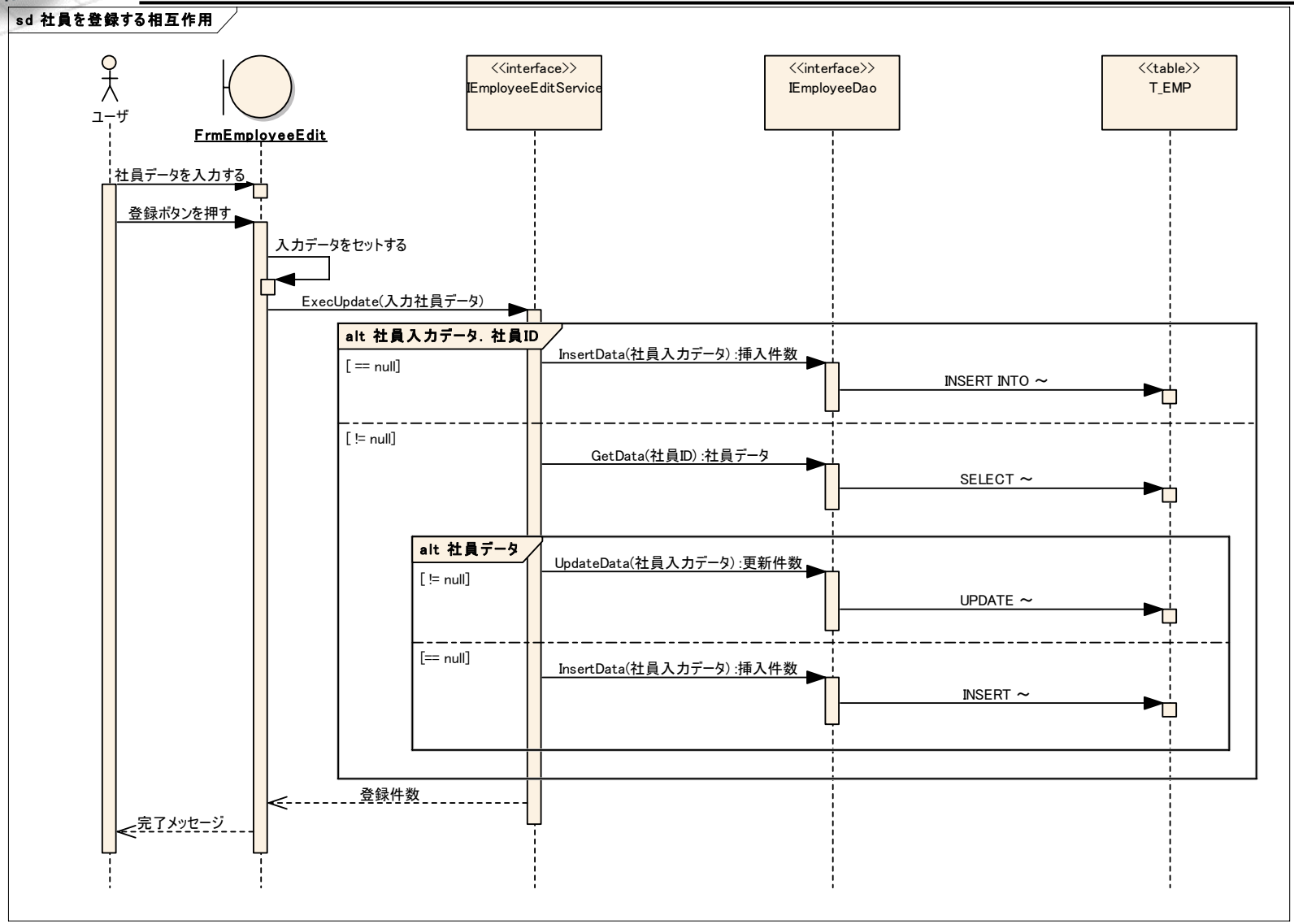


Domain層クラス図

S2DAO.NET用インターフェイスとCSV出力用クラス



例:社員データを登録するシーケンス図



- 簡単に表示するために正確なUMLではありません。

S2FormExample.Testsプロジェクト

ユニットテスト用クラスとdiconファイルを格納するプロジェクト

- テストクラスはS2Unit.NETを使うように作成
- .diconファイルは三種類作成し、ビルド後イベントでコピー
  - Ex.dicon →DB接続用のファイル。
  - Example.dicon →DIするフォームを含んだファイル。
  - ExampleLogic.dicon →Logics.Implのクラスを含んだファイル。（S2Unit.NETで使用する)
- データベース
  - Access MDBを使用し、ビルド後イベントでコピーしています。

Seasar.S2FormExample

- Formsプロジェクト。
  - Seasar.S2FormExample.Forms
- Logicsプロジェクト
  - Seasar.S2FormExample.Logics
- Testsプロジェクト
  - Seasar.S2FormExample.Test
- サンプルを再利用するときには既定の名前空間とフォルダ名、アセンブリ名を変更してください。

おわり

- S2Container.NETのページ
  - http://s2container.net.seasar.org/
- S2Dao.NETのページ
  - http://s2dao.net.seasar.org/
- S2Windows.NETのページ
  - http://s2container.net.seasar.org/ja/s2windows.html