

Seasar Conference 2008 Autumn



Seasar.NETファミリーを使った開発

S2Container.NETコミッタ

藤井宏明



- S2Container.NET、S2Dao.NETコミッタ
- 某SierでSEをしています。
- 昼間は要件定義書、仕様書、議事録書き。
夜はプログラマの生活

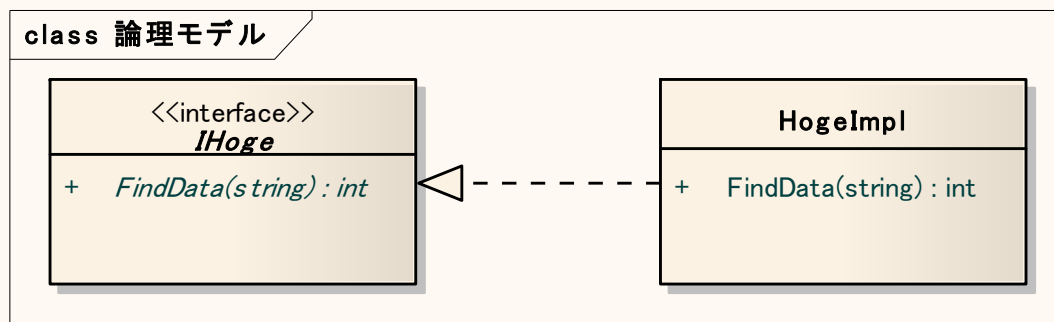


SEASAR.NETの紹介



Dependency Injectionとは

- 一般的に、各部分の結合を最小限にするように設計します。
⇒ セパレート・インターフェイス・パターン



<コードでの例>

```
IHoge hoge = new HogeImpl();
hoge.FindData("1");
```





Dependency Injectionとは

- 依存関係を外部から設定することを
Dependency Injectionと呼ぶ。

```
S2Container container = S2ContainerFactory.GetInstance("app.dicon");  
IHoge hoge = (IHoge) container.GetComponent(typeof(IHoge));  
hoge.FindData("1");
```

- 依存関係を外部から設定するモノが、
Dependency Injectionコンテナ (DIコンテナ)
 - S2Container.NET
 - Spring.NET
 - Unity



Dependency Injectionとは

- 主な使われ方:
 - フレームワークのベース
 - フレームワークが実装を用意して、Injectionする。
⇒ ユーティリティ的な使い方
 - アプリケーション・アーキテクチャーのベース
 - DIコンテナに登録した実装クラスをInjectionする。
⇒ 依存関係の処理に使う
 - ユニットテスト用



Seasar.NETファミリー (一部)

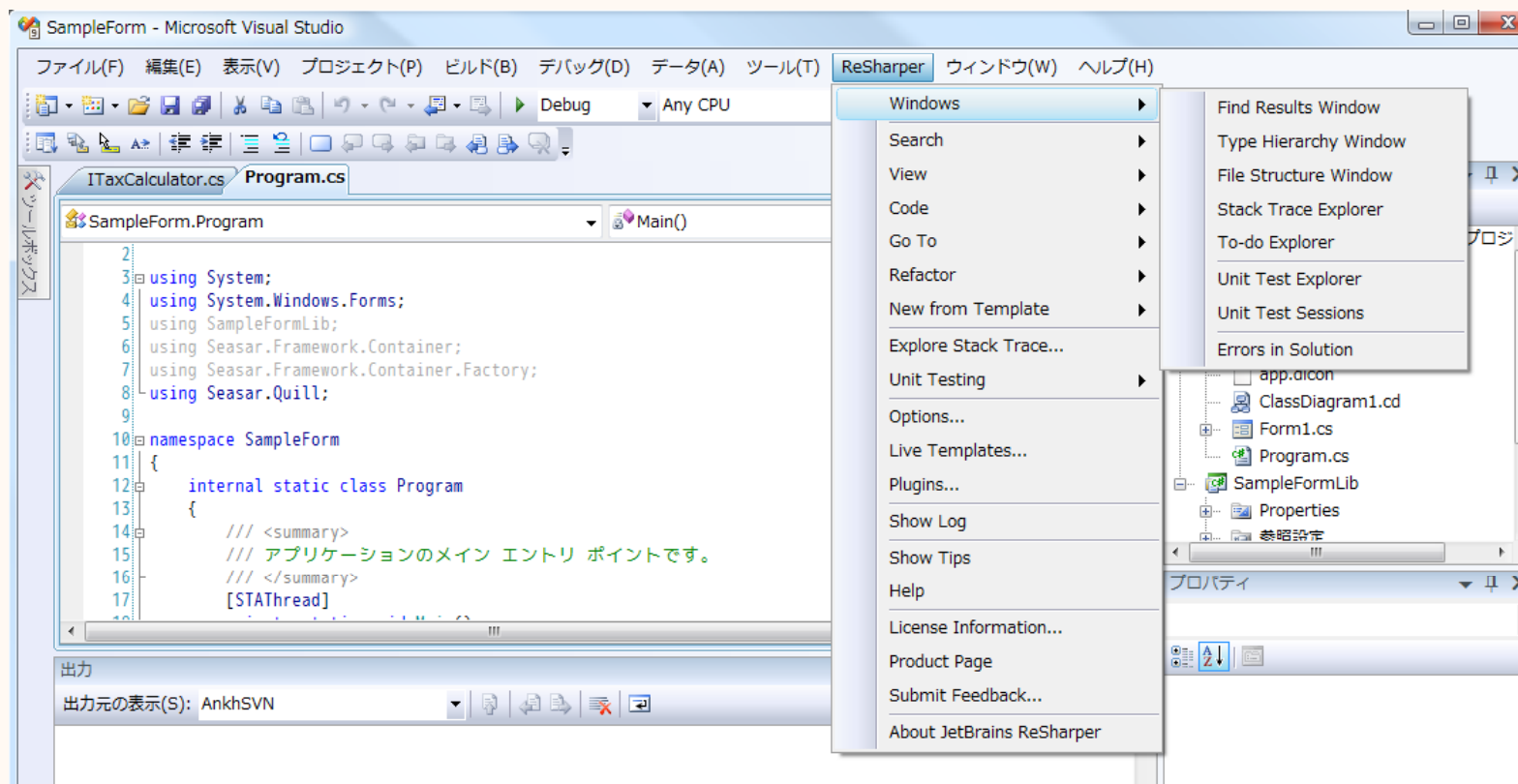
- S2Container.NET
 - S2Container.NETとQuill
- S2Dao.NET
- S2Unit.NET
- S2Windows.NET



余談1 : ReSharperのススめ

商用

- VisualStudioのリファクタリング用プラグイン





- Dependency Injection(DI)コンテナ
 - 使い方はJava版に近い。
 - 基本的に設定はdiconファイルという設定ファイルで指定する。
 - 登録するクラス、実装クラスの対応。
 - Aspect指向プログラム(AOP)の指定
 - データソースの指定
 - コンポーネントの自動登録設定
 - 色々な種類でオブジェクトを作成できる。



- 簡易版DIコンテナ

- Java版にはない機能。diconファイルを使わない。
- コンテナへの登録・指定はコードで行う。
- 基本はSingletonのみ。
- AOPの指定もコードで行う。
- データソースの指定はapp.configで行う
- プロバイダは基本は固定。



S2Container.NETとQuillの違い

- インスタンス生成のタイミングの違い
 - S2Container.NETは起動時にすべて初期化
 - Quillは初めて呼ばれたときに初期化
- コードへの埋め込みと柔軟性
 - S2Container.NETはdiconファイルの変更でよい
 - Quillは再ビルドを必要
 - パッケージのような柔軟な仕様の場合にはS2Container.NET
 - 請負開発のような場合にはQuill



- O/Rマッパー
 - 毎度決まった処理を楽にするために。
 - DbConnection作成 → DbCommand生成
 - DataSet(or DbDataReader)生成
 - 画面や帳票にセット
 - ⇒ 自動化します。
- 2Way仕様
 - SQL文の自動生成を行う。
 - SQL文を記述したテキストファイルを使用する。



S2Dao.NET – PONO(Value Object)

[Table("T_EMP")]

```
public class EmployeeDto  
{
```

```
    private string _code;  
    private DepartmentDto _department;
```

[Column("s_code")]

```
public string Code
```

```
{  
    get { return _code; }  
    set { _code = value; }  
}
```

[Relno(0), Relkeys("n_dept_id:n_id")]

```
public DepartmentDto Department
```

```
{  
    get { return _department; }  
    set { _department = value; }  
}
```

取得先テーブルを指定する属性

取得先フィールドを指定する属性

結合を設定する属性

Value Object は「エンタープライズ
アプリケーションアーキテクチャーパターン」から



Quillを使った場合のコードの例

< interface >

[Implementation(typeof (EmployeeEditServiceImpl))]

```
public interface IEmployeeEditService
{
    EmployeeEditPage GetData(int id);
}
```

実装クラスの指定を示す属性

< 実装クラス >

```
public class EmployeeEditServiceImpl : IEmployeeEditService
{
    protected IEmployeeDao dao;

    public EmployeeEditPage GetData(int id)
    {
        ...
    }
}
```

public、protectedなら自動的にinjection



Quillを使ったS2Dao.NET – Daoインターフェイス

• interfaceを作成するだけ

```
[S2Dao]
[Implementation]
[Bean(typeof(EmployeeDto))]
public interface IEmployeeDao
{
```

```
    [Query("order by t_emp.n_id")]
    IList<EmployeeDto> GetAll();
```

```
    [Sql("select n_id from t_emp where s_code = /*code*/'000001'")]
    int GetId(string code);
```

```
    IList<EmployeeCsvDto> GetAll();
}
```

S2Daoを示す属性とAOPを示す属性(Quill用)

自動生成するPONOクラスを示す属性

自動生成するSQL文のWhere句を示す属性

実行するSQL文を示す属性

この場合、IEmployeeDao_GetAllというファイルのSQL文を実行
注意:このSQL文のファイルを埋め込みリソースにする必要あり。



- .NET Framework3.5より導入

// DataContext で接続文字列を取得します。

```
DataContext db = new DataContext("c:¥¥projects¥¥Sample¥¥sample.mdf");
```

// クエリを実行するために、型指定されたテーブルを取得します。

```
Table< EmployeeDto> employees = db.GetTable<EmployeeDto>();
```

// クエリを指定

```
var q =
```

```
    from c in EmployeeDto
```

```
    where c.Code == "001"
```

```
    select c;
```

```
foreach (var cust in q)
```

```
    Console.WriteLine("Code = {0}, Name = {1}", cust.Code cust.Name);
```




- 仕様書との親和性が高い
 - Seasar.NETで設定することは、普通のソースの中に記述することになることが多い。
 - デザイナーやツールを必要としない。
 - 複雑なXMLファイルをほとんど必要としない。

仕様書からソースの自動生成ができる。

※もちろん、すべて生成できる訳ではない。



Linq To SQL他との違い

- Linq To SQL、他
 - デザイナを使ったり、コード中に記述する。
- S2Daoの場合
 - 属性ベースによるSQL文の自動作成
 - 他のDB管理ツールで作成したSQL文をそのまま使える。

**業務用アプリを開発し、保守する上ではS2Daoの
メリットは大きい。**



- Unitテスト用フレームワーク
 - S2Unit.NETが提供するクラスを継承して使う。
 - S2Container.NET用がS2TestCase。
 - Quill用がQuillTestCase。
 - MbUnitを使用する。
 - NUnitではない。
 - 使用するときにはReSharperと、MbUnitを連携させるMbUnit-ReSharperを使うと便利



S2Unit.NET (Quillの場合)

[TestFixture]

```
public class TestGender : QuillTestCase  
{
```

```
    protected IGenderDao dao;
```

[SetUp]

```
public void Setup()  
{
```

```
    // 初期セットアップ  
}
```

[Test, Quill(Tx.Rollback)]

```
public void TestDao()  
{
```

```
    IList<GenderDto> list = dao.GetAll();
```

```
    Assert.AreEqual(2, list.Count, "Count");  
}
```

```
}
```

テストクラスを示す属性

初期設定用メソッドを設定する属性

テストメソッドを示す属性

- PONOを自動的にバインディングする

```
[ControlModifier("txt", "")]  
public partial class FrmDepartmentEdit : S2Form  
{  
    ... ..  
    DepartmentEditPage data = service.GetData(_id.Value);  
    if (data != null)  
    {  
        this.DataSource = data;  
    }  
}
```

バインドするコントロールの接頭語を示す属性

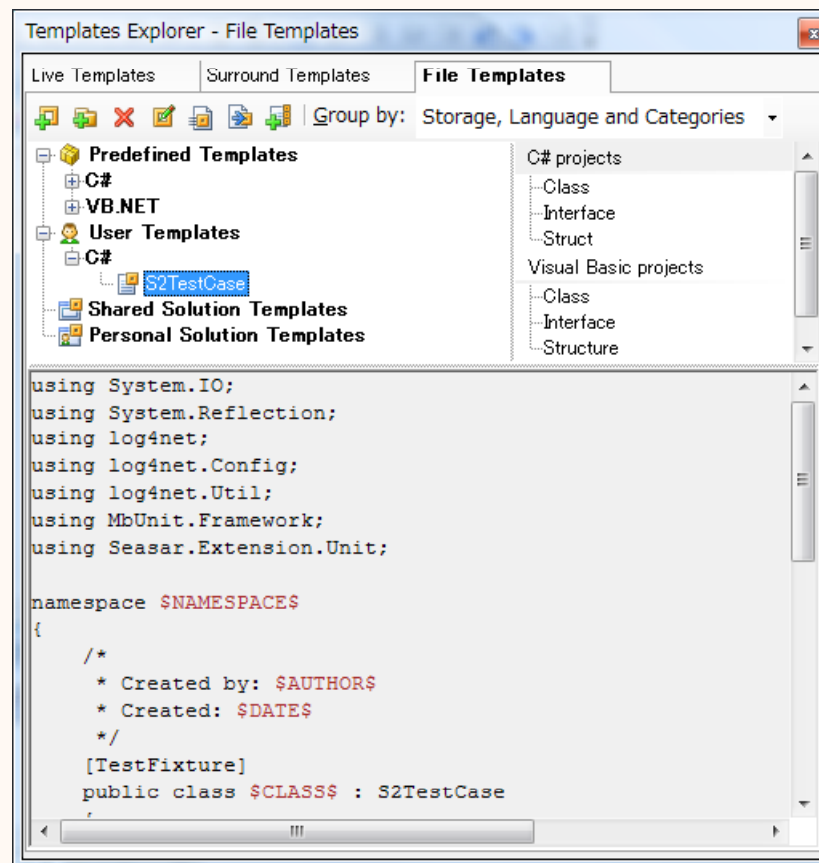
このクラスを継承する

PONOをバインドするプロパティ



余談2 : ReSharperとテンプレート

- そのReSharperの便利な機能にテンプレートがあります。





設計について



インターフェイスとトランザクション

- S2Container.NET (Quill) では、対象となるオブジェクトのメソッドに設定します。
 - S2Container.NET:diconファイル
 - Quill:[Transaction]属性

[Transaction]

```
public virtual int ExecUpdate(EmployeeEditPage data)
{
    if (data == null)
        throw new ArgumentNullException("data");

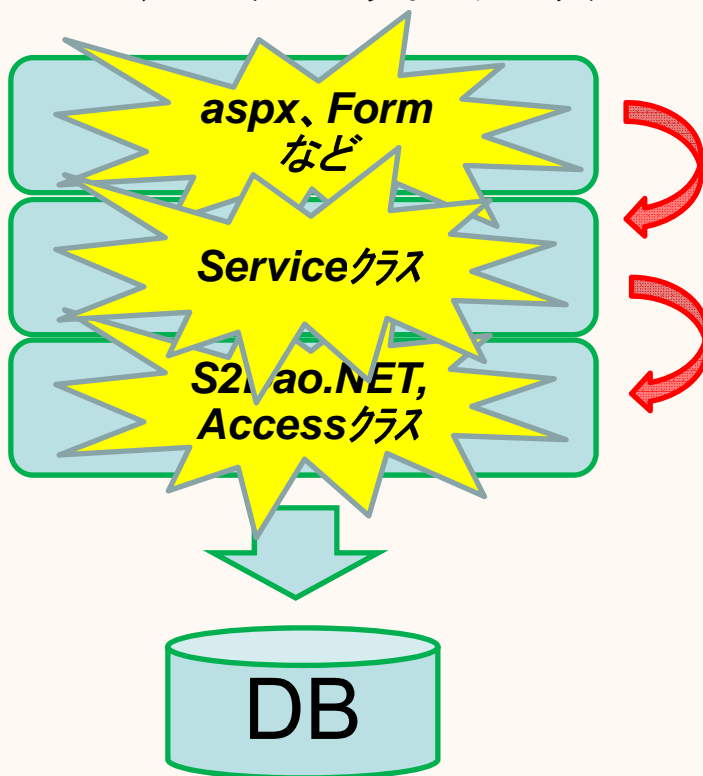
    ... ..
}
```




Seasar サービスinterfaceと実装クラスについて

・レイヤー化

(「エンタープライズアプリケーションアーキテクチャパターン」より)



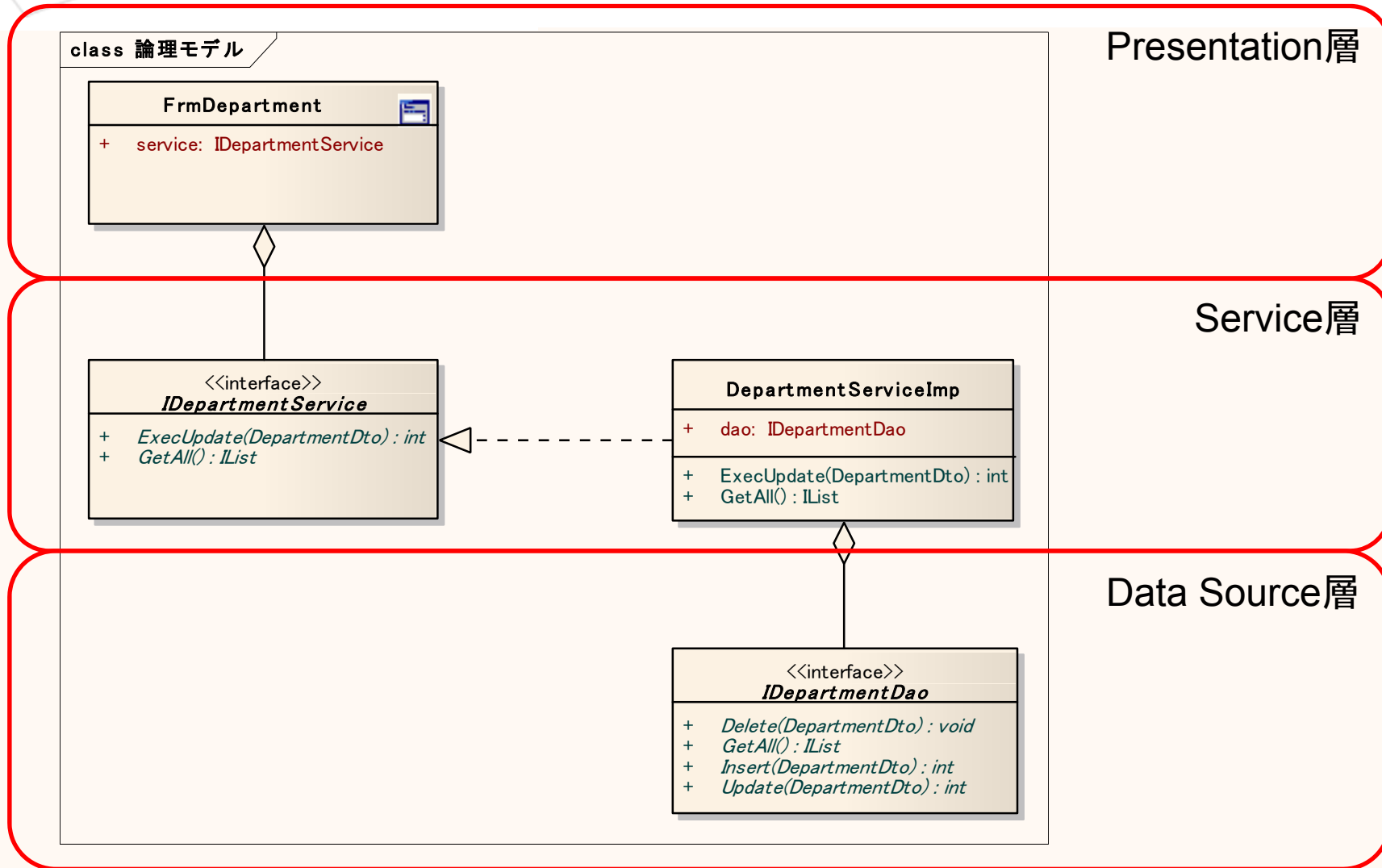
… 画面、帳票など、ユーザの入力部分

… **トランザクションはここで設定**

… S2DaoのメソッドはSQL文単位

Domain層以下をカプセル化し、
トランザクションを制御することを
サービスレイヤーという

実際のレイヤー化した設計





レイヤー化の一例

<Presentation層 → Domain層>

```
private void btnUpdate_Click(object sender, EventArgs e)
{
    DepartmentEditPage data = (DepartmentEditPage) DataSource;
    data.Id = _id;
    if (service.ExecUpdate(data) > 0)
        .....
}
```

層へ渡すデータ(Value Object)を取得する

Domain層へアクセス

<Domain層 → DataSource層>

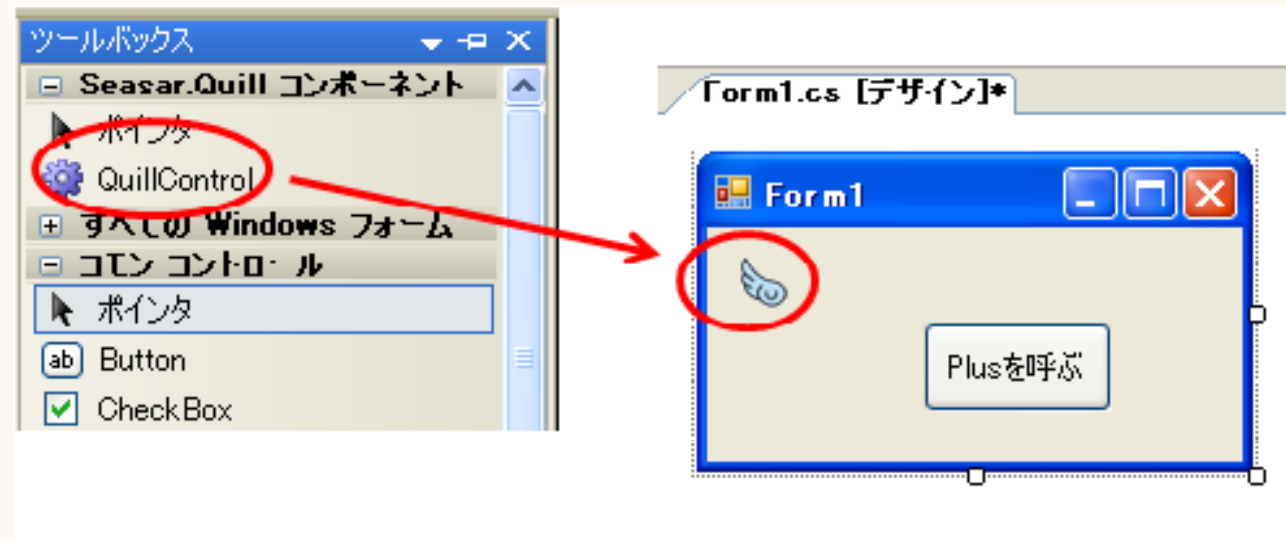
```
[Transaction]
public virtual int ExecUpdate(DepartmentEditPage dto)
{
    if (dto.Id.HasValue)
    {
        DepartmentDto departmentDto = dao.GetData(dto.Id.Value);
        if (departmentDto != null)
            return (dao.UpdateData(data));
        else
            return (dao.InsertData(data));
    }
}
```

DataSource層へアクセス

トランザクションはメソッド全体にかける

ビジネスロジック

- Formのデザイナー画面でQuillは使えます。



- Serviceクラスをpublicかprotectedなフィールドにすると自動バインドします。



- 件数が多い表のメタ情報を取得するとき時間がかかる。
 - 全件取得してからメタ情報を取得してる？
 - Oracle Clientの仕様？
- 対応策
 - Table属性で指定するテーブルを注意する。
 - テーブル定義情報のキャッシュを作成する。
(詳細は、ドキュメント参照)



- S2Container.NETサイト

<http://s2container.net.seasar.org/ja/>

- S2DAO.NETサイト

<http://s2dao.net.seasar.org/ja/>

- Seasar.NETサンプル

<http://s2container.net.seasar.org/ja/download.html>