シンプルなDIコンテナ「*Quill*」の応用

S2Container.NETコミッタ
小谷　圭

自己紹介

- S2Container.NET/S2Dao.NETのコミッタ
- アドイン始めました（Koropokkur.NET)

発表内容

- Quillって？
- S2Containerと何が違うの？
- Quill応用
  - インジェクション
  - Quill+S2Dao.NET
  - 複数データソースの切り替え
- ***????***
- Koropokkur.NETの紹介

Quillって？

DIコンテナの一種。極論すると、

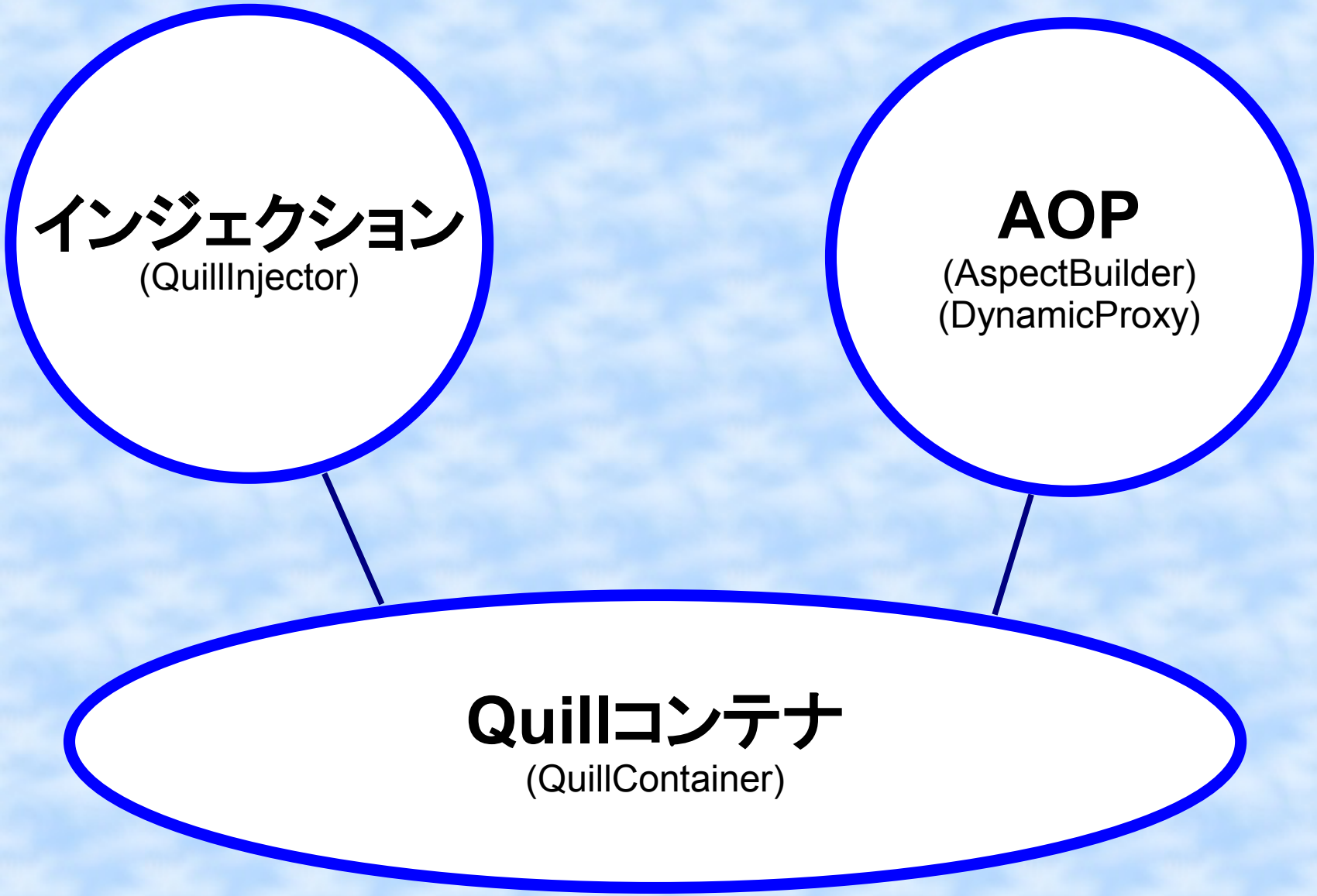
Dictionary

です。

```
public class QuillContainer : IDisposable
{
    /// <summary>
    /// ログ
    /// </summary>
    private readonly Logger _log = Logger.GetLogger(typeof(QuillContainer));

    // 作成済みにコンポーネントを格納する
    protected IDictionary<Type, QuillComponent> components =
        new Dictionary<Type, QuillComponent>();

    // Aspectを構築するBuilder
    protected AspectBuilder aspectBuilder;
```

Quillって？

- Quillのメリット、特徴
  - 構造がわかりやすい
    - 機能をしぼりこんでシンプルに
    - 積極的にコメントをつけている
    - C#の基本的な文法を知っていればコードが読める
      - *QuillContainer,QuillComponent,QuillInjector,AspectBuilder* あたりを読めば大部分の動きがわかる（はず）！
  - 属性ベース
    - カスタマイズの余地を残しつつ設定は少なく
    - 設定ミスがあった場合に原因追求をしやすい
    - IDEの基本機能で入力補完


インジェクション
(QuillInjector)
AOP
(AspectBuilder)
(DynamicProxy)
Quillコンテナ
(QuillContainer)

S2Containerと何が違うの？

	Quill	S2Container
生い立ち	S2Container.NETオリジナルのDIコンテナ	従来のJava版から移植されたDIコンテナ
DI設定方法	属性	XML(diconファイル)
インスタンス生成	singleton	singleton, prototype, request, session, outer
DIの種類	フィールド・インジェクション	コンストラクタ・インジェクション プロパティ・インジェクション メソッド・インジェクション
AOPの種類	DynamicProxy(型の拡張)	DynamicProxy(型の拡張) RealProxy(透過プロキシ)
用途	・シンプルなDI+AOPを利用したい場合 ・Windows Formアプリ ・WPFアプリ	・Java版Seasar2の利用経験があり、同じように使いたい場合 ・細かくDIを制御したい場合 ・設定(dicon)ファイルを置き換えることで実装を切り替えたい場合 ・ASP.NETで開発する場合

参考

※*DI、AOP、S2Container.NET/S2Dao.NETについて*

・**前回Seasar Conferenceセッション資料**

http://event.seasarfoundation.org/sc2008autumn/Session#e4

・Seasar.NET

http://s2container.net.seasar.org/ja/seasarnet.html

インジェクション

2種類の方法

- Implementation属性
- InjectionMapクラス(ver.1.3.14～)

Implementation属性

・インターフェースを使う場合

```
[Implementation(typeof(HogeImpl))]
public interface IHoge
{
    void SayHelloWorld();
}


public class HogeImpl : IHoge
{
    public void SayHelloWorld()
    {
        Console.WriteLine("Hello, world.");
    }
}
```

Implementation属性

- クラスを直接指定する場合

```
[Implementation]
public class Hoge
{
    public void SayHelloWorld()
    {
        Console.WriteLine("Hello, world.");
    }
}
```

Implementation属性

```
var mainForm = new MainForm();
var injector = QuillInjector.GetInstance();

injector.Inject(mainForm);
```

Go !!

```
public class ImplementationSample
{
    protected IHoge _hogeByInterface;

    protected Hoge _hogeOnly;

    public void Execute()
    {
        _hogeByInterface.SayHelloWorld();
        _hogeOnly.SayHelloWorld();
    }
}
```

InjectionMap

```
var mainForm = new MainForm();
var injector = QuillInjector.GetInstance();

var injectionMap = InjectionMap.GetInstance();

injectionMap.Add(typeof(IHoge), typeof(HogeImpl));

injectionMap.Add(typeof(Hoge));

injector.InjectionMap = injectionMap;

injector.Inject(mainForm);
```

Go !!

```
public class ImplementationSample
{
    protected IHoge _hogeByInterface;

    protected Hoge _hogeOnly;

    public void Execute()
    {
        _hogeByInterface.SayHelloWorld();
        _hogeOnly.SayHelloWorld();
    }
}
```

InjectionMapの用途

- Quill用に作られていない
  他のライブラリを利用したいとき

- テストと本番でコードや
  設定ファイルを変更することなく
  コンポーネントを切り替えたいとき

Quill+S2Dao.NET

これも属性を使って指定

```
[S2Dao]
[Implementation]
public interface HogeDao
{
    Hoge GetHoge(int hogeId);
    int Update(Hoge entity);
    int Insert(Hoge entity);
    int Delete(Hoge entity);
}
```

```
[Transaction]
[Implementation]
public class HogeFacade
{
    protected HogeDao _dao;

    public int UpdateHoge(Hoge entity)
    {
        if(entity.HogeId.HasValue)
        {
            return _dao.Update(entity);
        }

        return _dao.Insert(entity);
    }
}
```

Quill+S2Dao.NET

属性引数なし→デフォルト設定

```
[S2Dao]
[Implementation]
public interface HogeDao
```

```
[Transaction]
[Implementation]
public class HogeFacade
{
```

- S2Dao
  - *S2DaoInterceptor*
  - *DaoMetaDataImpl*
  - etc.
- *TypicalS2DaoSetting*

  で設定

- トランザクション
  - TransactionInterceptor
  - TransactionContext
  - LocalRequiredTx
  - ReadCommited
- *TypicalTransactionSetting*

  で設定

Quill+S2Dao.NET

カスタマイズしたいときは？

- IS2DaoSetting
- ITransactionSetting

```
public class CustomS2DaoSetting : IDaoSetting
{
```

```
[S2Dao(typeof(CustomS2DaoSetting))]
[Implementation]
public interface CustomizeHogeDao
{
```

```
public class CustomTransactionSetting : ITransactionSetting
{
```

```
[Transaction(typeof(CustomTransactionSetting))]
[Implementation]
public class CustomizeHogeFacade
{
```

Quill+S2Dao

- どんなことがカスタム可能？
  - S2Dao
    - 適用するInterceptor
    - 使いたいデータソース名
    - (DaoMetaDataFactoryなど)
  - トランザクション
    - 適用するInterceptor
    - 使いたいデータソース名
    - (TransactionContextなど)

Quill+S2Dao.NET

- カスタムした設定をあちこちに書くのは面倒！

設定ファイルに書けば全体に適用！

```
<configuration>
  <quill>
    <!-- S2Dao設定 -->
    <daoSetting>CustomizeDaoSetting</daoSetting>
    <!-- トランザクション設定 -->
    <transactionSetting>CustomizeTransactionSetting</transactionSetting>
```

複数データソースの切り替え

- 手順2

IDaoSetting/ITransactionSettingを書く

実装クラスを作ってデータソース名を返す！

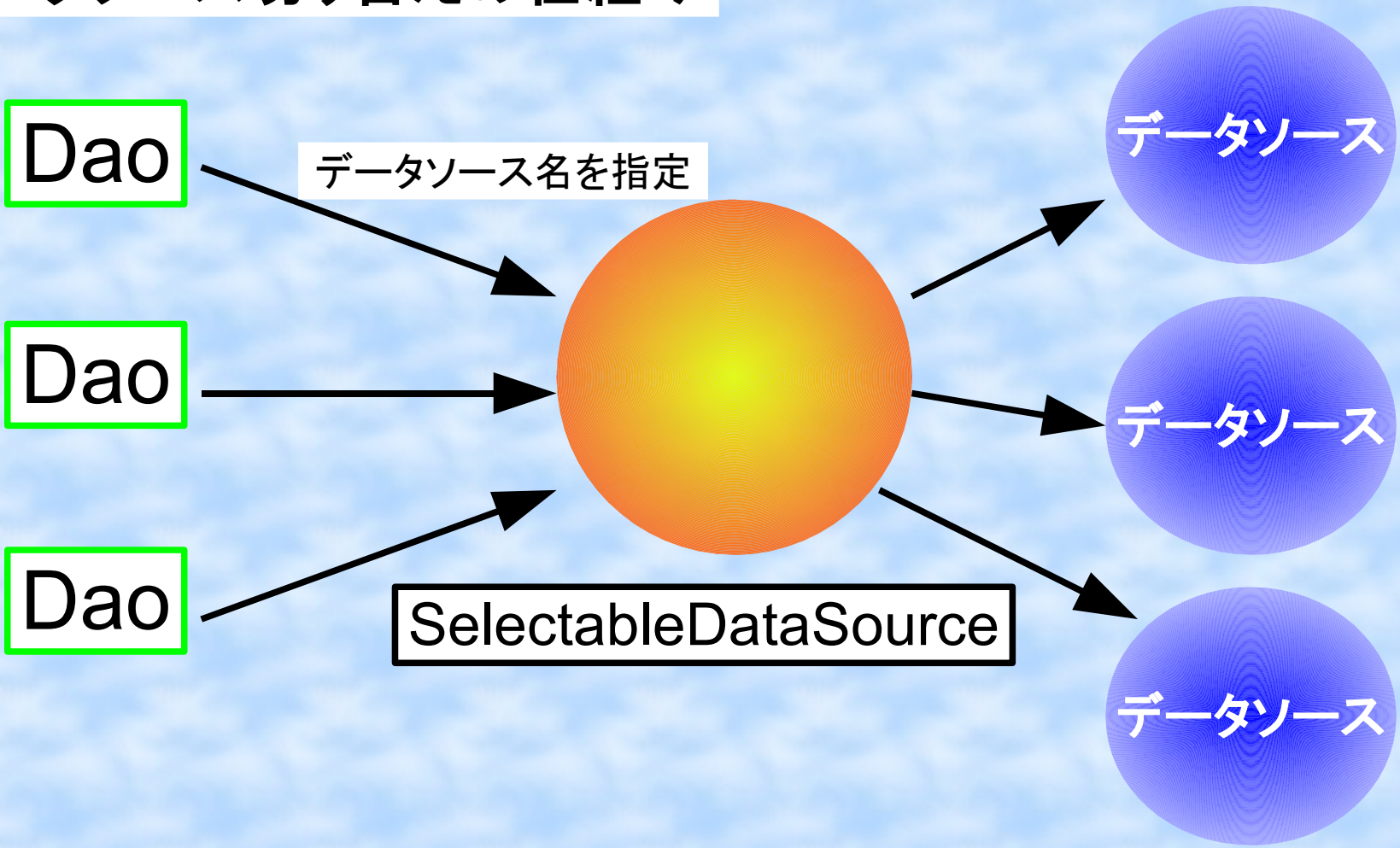
```
<!-- データソース設定 -->
<dataSources>
  <!-- データソース１  -->
  <dataSource name="Eclipse">
    <provider>SqlServer</provider>
    <connectionString>"Server=Eclipse;databas
    <class>Seasar.Extension.Tx.Impl.TxDataSou
  </dataSource>

  <!-- データソース２  -->
  <dataSource name="NetBeans">
    <provider>SqlServer</provider>
    <connectionString>"Server=NetBeans;databa
    <class>Seasar.Extension.Tx.Impl.TxDataSou
  </dataSource>
</dataSources>
```

```
public class S2DaoSettingA : IDaoSetting
{
    public string DataSourceName
    {
        get { return "Eclipse"; }
    }
```

```
public class S2DaoSettingB : IDaoSetting
{
    public string DataSourceName
    {
        get { return "NetBeans"; }
    }
```

データソースの切り替え

データソース切り替えの仕組み



Koropokkur.NETの紹介

VisualStudioでの開発をささやかに
支援することを目的としたアドイン集です。

Koropokkur.NETの紹介

- *VSArrange*
  - プロジェクト内の項目を整理するアドイン
    - 削除ファイル、フォルダをプロジェクトから除外
    - 設定で指定した条件に該当するファイル、フォルダも除外
    - 上記以外でプロジェクト未登録のものを登録
  - DBFluteなどの自動生成系ツールを使うときに効果を発揮！？
- *CopyMethodGen*（予定）
  - 文字通りコピー系メソッドを自動生成
  - Dxoを利用したくなる主な理由（なことが多い!?）

    「プロパティ一つ一つ「A = B;」と書くのが面倒」を解決！

今後の予定

- ***Quill / S2Container.NET / S2Dao.NET***
  - ドキュメント整備を中心に安定させていく方向で
- ***Koropokkur.NET***
  - 「なくてもそれほど気にならないけど
    あればとちょっとだけ楽になるものを」
    引き続き開発予定
  - 簡単なコード生成系
  - VisualStudioの機能不足（R#のすきまをぬうように）分のサポート