



DI+AOP
Seasar2を使ってみて



EY-Office 吉田

概要

DI+AOP (Seasar2を使って)

DIって何？

クラスの依存関係を
祖にできる

何が嬉しいの？

テストし易くなる

アプリとフレームワークの
依存関係を減らせる

AOPは便利

横断的な処理を
簡単に記述できる

何が嬉しいの？

宣言的トランザクション

AOPといえばログ・トレース

アプリに共通な処理の括り出し

Seasar2

開発者に優しい

少ないコンフィグファイル記述

Railsの影響も

現実的な仕様

S2Dao

S2JSF

S2DaoTestCase

日本語ドキュメント

成功してるOSS

プロジェクト運営が上手い

カリスマ開発者

表の顔

裏番町

DIって何？



- インターフェースと実装を分離し、コンポーネント(クラス)の依存関係をなくす。

今までの問題:

```
OsigotoIF osigoto = new OsigotoImpl();
```

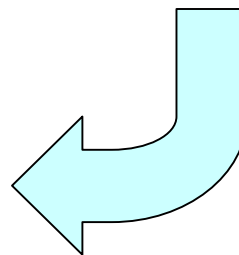
インターフェースを書き、実装と分離しても結局 new するのは実装クラスなので結局実装クラスに依存したコードになってしまう。

DIコンテナーがあると

- インターフェースと実装の関連を定義ファイル等に追い出し、DIコンテナーがコンポーネントの生成を管理してくれる

```
private OsigotoIF osigoto;  
  
public void hoge() {  
    osigoto.start();  
}  
  
public void setOsigoto(OsigotoIF oisigoto) {  
    this.osigoto = oisigoto;  
}
```

```
<components>  
  <component class="OsigotoImpl"/>  
</components>
```



DIコンテナーが実装クラスを生成し注入してくれる



それで何が便利なの？

- (ユニット)テストしやすくなる
- フレームワークとの依存関係を下げれる



ユニットテストしやすくなる

- **テスト重要！**
- **モックアップを使ってユニットテストする際にコードを書き換えなくてもよい**

フレームワークとの依存関係を下げれる

- フレームワークを使うのだからフレームワークに依存するのは当たり前のようにだが、コードの大部にフレームワークのクラスが入り込み…
 - 他のフレームワークでの再利用が困難
 - テストが難しくなる
 - 単純なアプリのコードを書く開発者も当然フレームワークの知識が必要になる
- やっぱり、フレームワークとの依存関係は低い方が良い！

S2Struts(StrutsにSeasar2を組み込んだもの) の例

Struts

```
public class LoginAction extends Action {
    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {
        LoginForm loginForm = (LoginForm)form;

        // ログイン認証
        if (equals(loginForm.getName(),loginForm.getPassword())) {
            // 成功
            return mapping.findForward("success");
        } else {
            // 失敗
            return mapping.findForward("error");
        }
    }
}
```

S2Struts

```
public class LoginActionImpl implements LoginAction {
    private LoginForm loginForm;
    public authorize() {
        // ログイン認証
        if (equals(loginForm.getName(),loginForm.getPassword())) {
            // 成功
            return success";
        } else {
            // 失敗
            return error";
        }
    }
    public void setLoginForm(LoginForm loginForm) {
        this.loginForm = loginForm;
    }
}
```

AOPは便利



- 色々なクラスに記述しなければならない同じようなコードを一ヶ所で定義し、それをクラスに適用できる。

なにが便利なの

- トランザクション

- begin/commit/rollbackをコードに書かなくてもメソッド単位にトランザクション制御を行ってくれる

- ログ/トレース

- AOPといえはロギング ^);
- ログやトレースのコードを各処理に書かなくてもロギングできる

- アプリで共通な処理(例: 認証済みチェック)の一元化

Seasar2



- 開発者に優しい DI+AOPコンテナ
 - コンフィグファイルの記述量が少ない
 - 実用的な仕様・プロダクト
- 成功している OSS プロジェクト

コンフィグファイルの記述が簡単

- ここがSpringとの設計思想の違い
- 自動バインディング
- 式記述にOGNLを使うことでシンプルで強力
- Convention over Configuration (Railsの影響?)
- アノテーション

実用的な仕様・プロダクト

- S2Dao
 - コードレス、XMLレスなO/Rマッピング
 - SQLレスもSQL記述もサポート
- S2JSF (JSF + S2 +)
 - JSPではなくHTMLベースのテンプレートをサポート
 - JSFの欠点の補完
- S2DaoTestCase
 - テスト用DBデータ(初期値/期待値)をExcelで書け、`assetEquals()` で簡単にチェックできる
- Kijimuna
 - S2設定ファイル(DICON)支援EclipsePlugin

成功しているOSS

- SeasarFoundationは開発以外の部分も充実しているように思う
 - ひが やすを
 - 理事 - - メインコミッター
 - カリスマ開発者
 - 栗原 傑享
 - 理事長 - - 表の顔
 - (株)グルージェント 代表取締役
 - 羽生 章洋
 - 理事 - - 裏番町
 - (株)スターロジック 代表取締役

Seasar2を使った感想

- S2Dao+S2DaoTestCaseは開発効率高い！
 - 自由にSQLを書ける
 - テストが楽チン
- 開発スタイルを統一しやすい、したがって効率も上がる
- テストし易い！
- バグフィックス・機能拡張は頻繁に行われている
- ドキュメントはOSSにはあると思う
- 慣れるまでは、戸惑う現象にあうことがある(コードだけでは全てが見えない)
- 一度使うと止められない 😊



ご静聴、ありがとうございました