Seasar Conference
2006 Autumn

突撃！隣のSeasarプロジェクト

- 現場で役立つ実践Tipsを教えてもらおう -

2006.11.12

株式会社 ティーアンドエフカンパニー
出羽 健一（dewa@tafc.co.jp）

本セッションの概要

- 概要
  - Seasar2を採用した実プロジェクトを通じて
    獲得した実践Tipsを事例も交えてフィードバック
  - 小さな企業が短期間で大規模開発を成功させた
    全容に迫る
  - 今回の事例紹介により、Seasar2の導入障壁が
    少しでも低くなれば幸いです

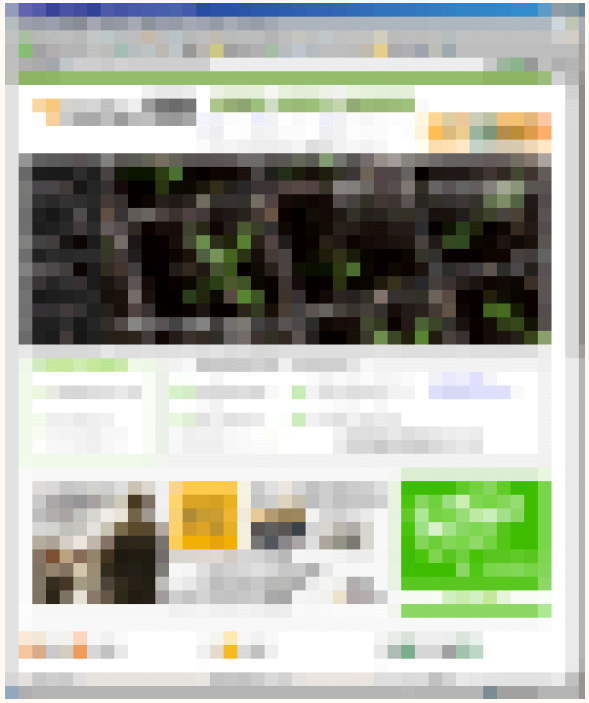
アジェンダ

- 講師紹介
- 事例紹介
- アーキテクチャ
- Tips紹介

講師紹介

- 「出羽 健一」の紹介
  - 株式会社 ティーアンドエフカンパニー
    - CTO
  - Seasar2のコミッター
    - 新米です
  - 金沢工業大学院　客員助教授
    - 実は去年から授業でSeasar2を教えていました
  - 出身地
    - 大阪府
  - 趣味
    - 格闘技 / ルービックキューブ / 手品

事例紹介①



- サイト内容
  - 学生と企業の就職マッチング支援サイト
  - 人事担当者 採用業務システム
- 開発元
  - 株式会社 ティーアンドエフカンパニー
- 規模
  - 約1000ページ（Webアプリ数: 6個）
- 期間
  - 約9ヶ月（実装期間: 5.5ヶ月）
- 使用S2プロダクト
  - S2Contaner 2.3 / S2JSF / S2Dao
- ミドルウェア
  - Tomcat 5.5 / Postgre SQL 8.1

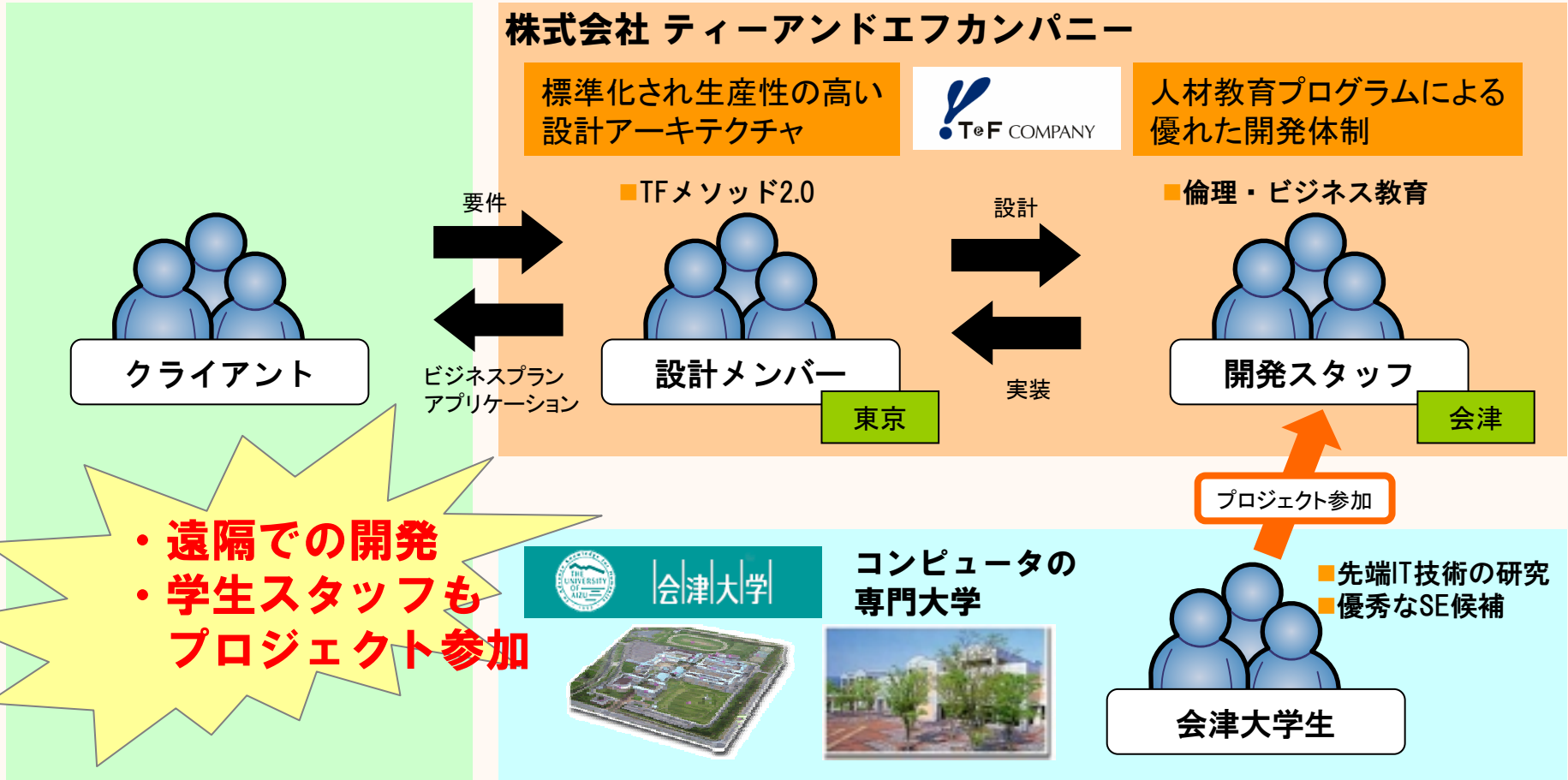
5.5ヶ月で
1000画面

しかも、同時進行で他に3件もSeasar2のプロジェクトをやってた（30~90画面程度）

事例紹介④

- どんな奴らが作ったのか？
  - **社員18人（開発者11人）＋ 学生スタッフ**

会社の話であって、
プロジェクトの話ではない



事例紹介⑤

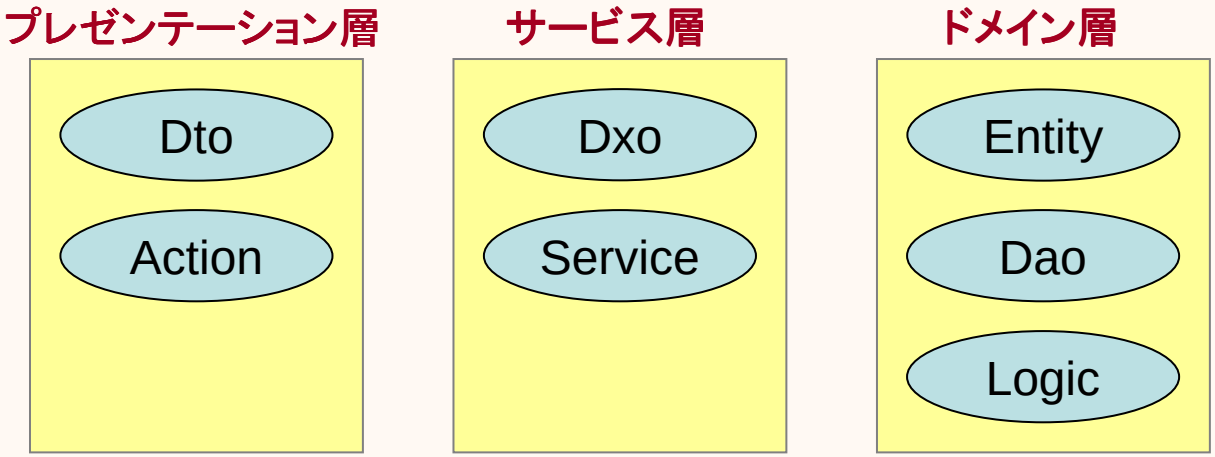
- 技術的側面の成功要因分析
  - Goyaベースのアーキテクチャ
    - ジェネレータとの相性が抜群！！
  - 独自開発のコードジェネレータ
  - 多くの非機能要件を宣言的・透過的に対処できた
    - DI / AOP / アノテーション / 名前規約 / ServletFilter
  - 「IDの導入」を採用したERD
  - S2Daoの生産性の高さ

事例紹介⑥

- ふりかえり
  - 悪い面も見えたが、全体としては、Seasar2の導入は
    SI企業としての開発力を確実に押し上げた
    - 正直、**Seasar2じゃなかったらやばかった**と思う、、、
  - 周りが使い始めて良さそうだったら、
    自分たちも使ってみようと思う人が多いはず
  - 今回の事例紹介により、Seasar2の導入障壁が
    少しでも低くなれば幸いです

アーキテクチャ①

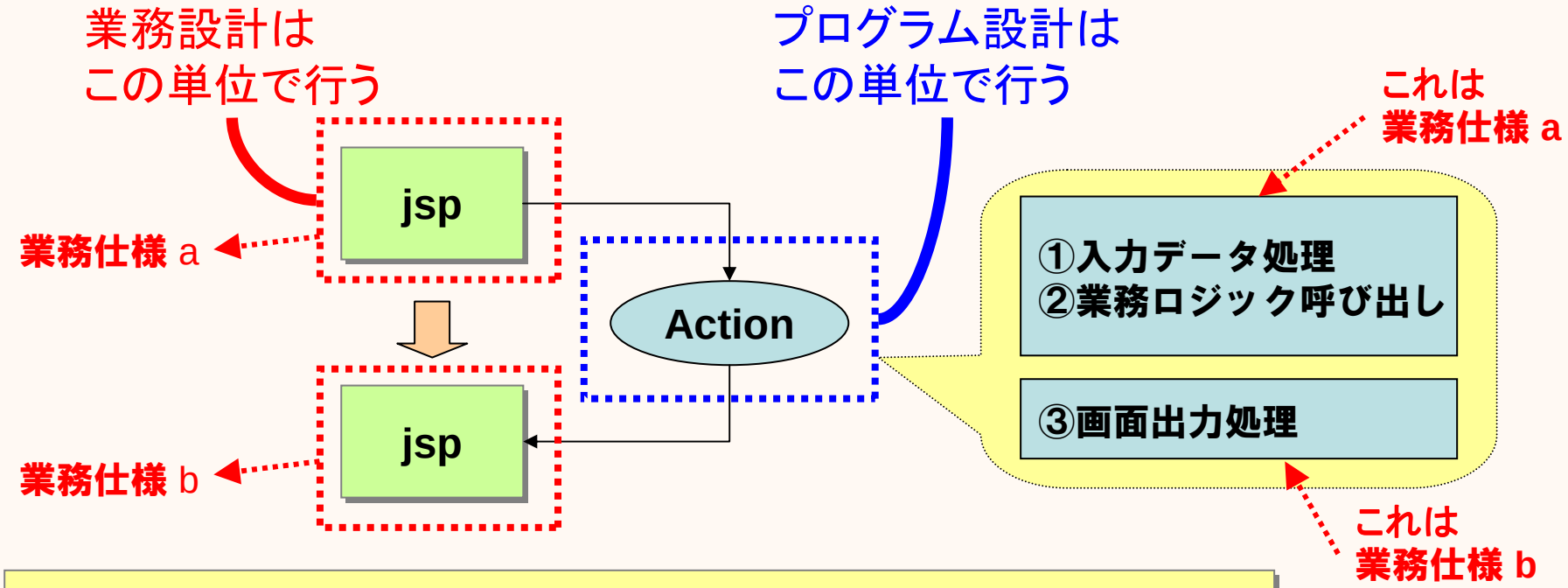
- ベースはGoya(ゴーヤ)
  - ひがやすを氏 と 羽生章洋氏 が
    共同で取りまとめている設計指針
  - 参考:「Web+DB Press vol.31 Seasar2 徹底攻略」



典型的なGoyaのアーキテクチャの概要図

アーキテクチャ②

- Strutsベースのアーキテクチャの問題点
  - 業務設計とプログラム設計との間にズレが発生する

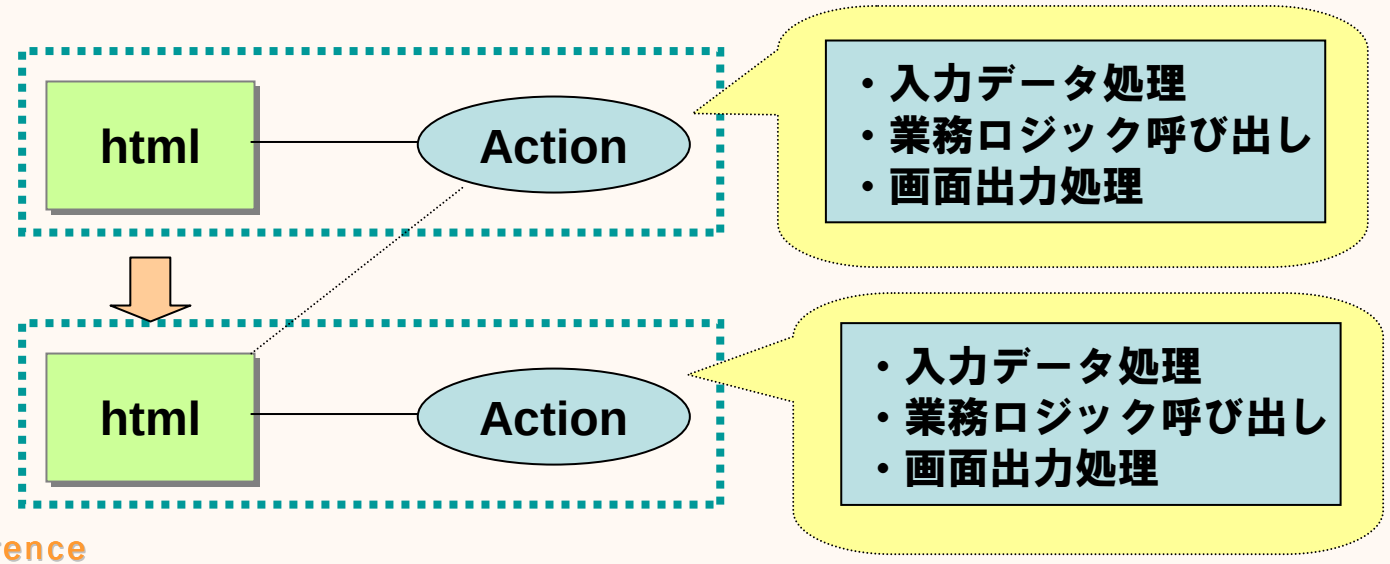


原因は「UI」と「サーバ上のプログラム」が一対一に対応していないこと

参照：.NETエンタープライズ Webアプリケーション 開発技術大全 Vol.2

アーキテクチャ③

- S2JSFの場合
  - 「業務設計」と「プログラム設計」の間にズレが生じない
    - 「UI」と「サーバ上のプログラム」が1対1に対応
  - Html名と画面遷移図からクラス設計は機械的に定まる
  - シンプル・疎結合・POJO
  - コードジェネレータとの相性が抜群に良い



アーキテクチャ⑤

- 「検索」画面遷移パターン
  - 1html = 1Dto + 1Action 方式で知らないと辛いパターン

・初期化イベント
選択項目の値などをDBから取得してSearchDtoにセット
・検索イベント
検索条件の詰め替え (SearchDto ⇒ ListDto)
“検索結果一覧”へ遷移

・初期化イベント
ListDto内の検索条件を使ってDBへ検索。
結果をListDtoにセット
・次へイベント / 前へイベント / 再検索イベント
ListDtoに検索条件をセット
“自画面”へ遷移

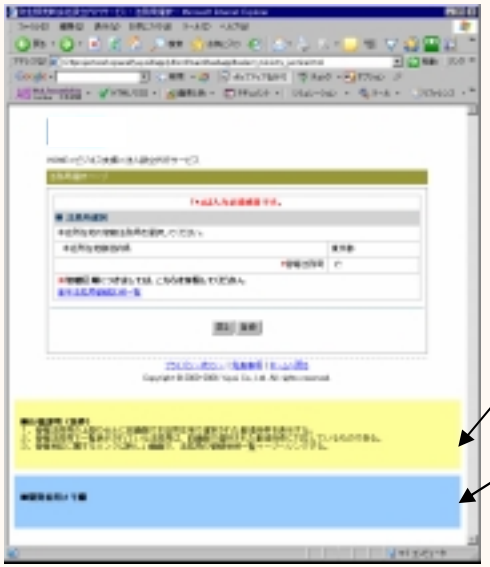
これよりTips紹介

Htmlテンプレート

- Htmlテンプレート
  - ビュー（JSP）の代わりにHtmlファイルを使う技術
  - プロトタイプとViewテンプレートは同じhtmlファイル
    - アプリサーバが無くても確認可能
    - 仕様変更時における2重メンテナンス不要
    - 何時でもお客様に最新版のプロトタイプを届けることができる

外部仕様 In Html

- HTMLプロトタイプに『仕様』と
  『開発者向けメモ』を書いてしまう



ココにお客様に見せる仕様を書く

ココに開発者向けのメモを書く

※ 全画面に対し、一気にCSSで表示・非表示を切り替えるようにしておく
※ Htmlテンプレート の採用、及び、「業務設計」と「プログラム設計」の間に
ズレが生じない 1html = 1Action + 1Dto 方式で効果を発揮

Entity Tips

- EntityクラスのtoStringメソッドはcommonsのToStringBuilderを使ってオーバーライド
  - 関連エンティティのプロパティまでも『変数名 = 値』の形式で表示されるのでとっても便利です！

toStringメソッドの実装例：

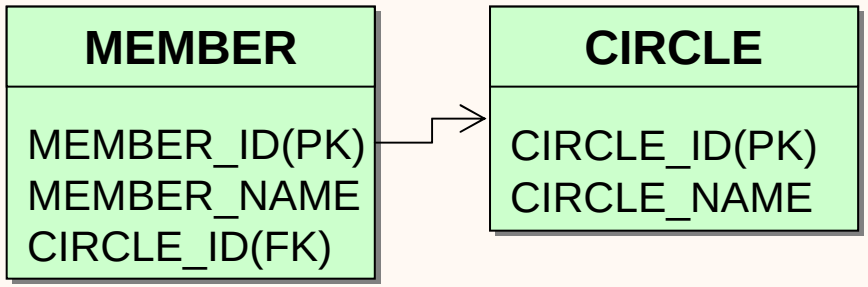
```
import org.apache.commons.lang.builder.ToStringBuilder;
import org.apache.commons.lang.builder.ToStringStyle;

public String toString() {
    return ToStringBuilder
        .reflectionToString(this, ToStringStyle.MULTI_LINE_STYLE)
        .toString();
}
```

使用例 ⇒

```
System.out.println(hogeEntity);
```

S2Dao Tips

- S2DaoのQueryアノテーションにおいて、
  『キャメル文字列のテーブル名 + “Entity“』
  の形式で、JOIN句を書かずにFK先のテーブルにアクセスできる
- 前提
  - EntityでN:1マッピングの指定が行われていること
- 例

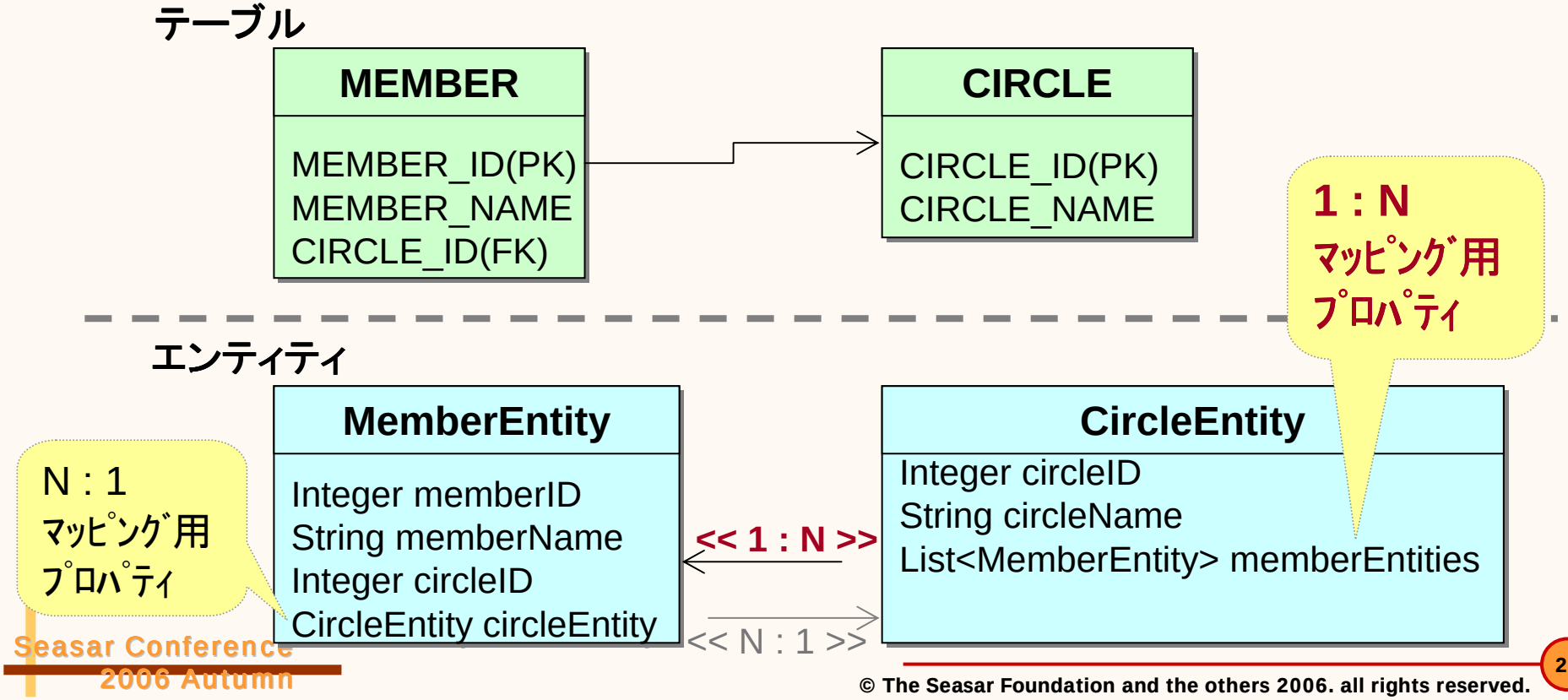


～ MemberDao.java にて下記のようなメソッドが定義可能 ～

```
@Query(“circleEntity.circleName = ?”)
public findByCircleName(String circleName);
```

S2Dao で擬似1:Nマッピングを実現①

- ４種類のリレーションのうちで、S2Daoは N : 1マッピングのみサポート
- 擬似的な 1 : N マッピングの実現方法を紹介



S2Dao で擬似1:Nマッピングを実現②

- やり方
  - FKを持っていない側のエンティティに以下のメソッドやフィールドを追加する
    1. Listや配列など、複数のエンティティが格納できる型のフィールド、及び、Setter/Getterメソッドを追加する
    2. equalsメソッドをオーバーライドする
       （必須では無いがコードが簡潔になる）
    3. 1：N マッピングをセットアップするメソッドを定義する

次ページ以降のサンプルコードをコピーして
少し修正するだけで、手軽に実現できます

S2Dao で擬似1:Nマッピングを実現③

- サンプルコード (CircleEntity.javaの抜粋)

```
import org.apache.commons.lang.builder.EqualsBuilder;

private List<MemberEntity> memberEntities;
public void setMemberEntities(List<MemberEntity> entities) {memberEntities = entities;}
public List<MemberEntity> getMemberEntities() {return memberEntities;}

public setupMemberRelation(List<MemberEntity> entities) {
    memberEntities = new ArrayList<MemberEntity>();
    for (entity : entities) {
        if (getCircleId().equeals(entity.getCircleId())) {
            getMemberEntites().add(entity);
        }
}

public boolean equals(Object other) {
    if(!(other instanceof CircleEntity)) return false;
    CircleEntity castOther = (CircleEntity) other;
    return new EqualsBuilder()
        .append(this.getAdminUserId(), castOther.getAdminUserId()).isEquals();
}
```

1：N ﾏｯﾋﾟﾝｸﾞ用のEntityをセットアップするメソッド

全てのEntityクラスでequalsメソッドのオーバーライドを推奨

S2Dao で擬似1:Nマッピングを実現④

- サンプルコード (CircleEntity.javaを呼び出す側)

```
/* 1 : N マッピングの候補となるエンティティのリストを取得する
   パフォーマンスを考慮してEntity数を絞り込んでも良いが、
   バグの温床になるので慎重に！ */
memberEntities = memberDao.findAll();

/* 1 : N マッピングの値をセットアップ */
circleEntity.setupMemerRelation(memberEntities);
```

ERDの勘所

- IDの導入
  - 業務的な意図とは関係の無い
    「レコードが存在している」ということだけを
    表現するフィールドを各テーブルに定義する
  - メリット
    - SQL文がシンプルになる
    - S2Daoと相性が良い
    - 複合キーに比べて、画面⇔DB間のパラメータ渡しのコストが少なく
      コーディングが楽になる
    - テーブル間の関係が疎結合になり変更が柔軟

参考

楽々ERDレッスン
㈱スターロジック 羽生 章洋

Seasar 2.3時代の再起動テクニック

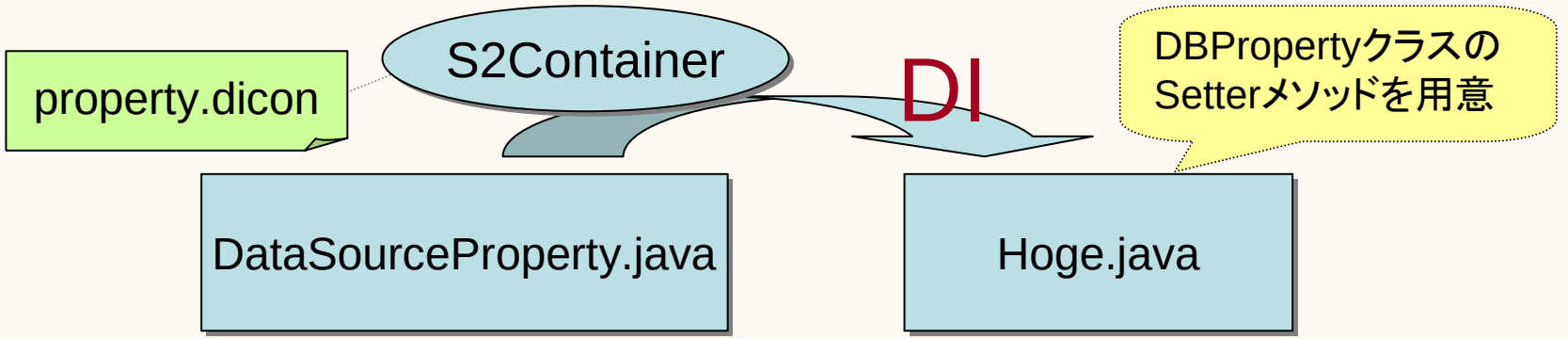
- Javaクラス修正に伴うアプリサーバ再起動は開発効率が悪い
- クラスファイルの型情報が不変ならアプリサーバ再起動は不要
  （ブラウザのリロードだけで、変更箇所が反映される）
  - 先にメソッドのインターフェースを定義してから、
    後で内部のコードを記述するやり方で再起動時間を激減
- 再起動必要
  - メソッド名、戻り値、引数の変更　/　メソッド名の追加・削除
  - クラス名の変更、クラスの追加・削除
- 再起動不要
  - メソッド内のコード変更
- Seasar 2.4系のホットデプロイを使えば考慮しなくて良い

さらばプロパティファイル①

- 慣れ親しんできたプロパティファイルの代わりにdiconファイルを使う
- 利点
  - String以外の型も扱える
  - 設定ファイルの構文ミスを実行前にKijimuna が検出
  - IDEを使えば呼び出しの影響範囲がすぐに分かる
  - やり方次第で環境依存変数の問題を解決できる
- イメージ
  - 従来のResourceBundleクラス ⇒ POJOクラス
  - 従来のproperties ファイル ⇒ diconファイル

さらばプロパティファイル②

- やり方
  1. SetterとGetterから構成されるPOJOベースのPropertyクラスを作成し、DIコンテナに登録する
  2. Propertyクラスへセットする値を定義するdiconファイルを作成する
  3. Setterを用意してPropertyクラスをDIして利用する



さらばプロパティファイル③

- サンプルコード:
  DataSourceProperty.java

```
public class DataSourceProperty {
    private String url;
    private String driverClassName;
    private String user;
    private String password;
    ～ SetterとGetterは省略 ～
}
```

さらばプロパティファイル④

- サンプル:
  property.dicon

```
<component name="dataSourceProperty"
        class=“sample.property.DataSourceProperty" instance="prototype" >
    <property name="url">"jdbc:mysql://xxx.xxx.xxx.xxx/hoge”</property>
    <property name="driverClassName">"com.mysql.jdbc.Driver"</property>
    <property name="user">”hoge”</property>
    <property name="password">”fuga”</property>
</component>
```

さらばプロパティファイル⑤

- サンプル

  Hoge.java

  ①Hoge.java が S2コンテナで管理されている場合

  以下のようなSetterメソッドを定義しておけば

```
private DataSourceProperty dataSourceProperty;
public void setDataSourceProperty(DataSourceProperty property) {
    this.dataSourceProperty = property;
}
```

  DIされたインスタンスを使ってプロパティ値を取得する

```
String user = dataSourceProperty.getUser();
```

さらばプロパティファイル⑥

- サンプル

  Hoge.java

  ②Hoge.java が S2コンテナの管理下にでない場合

  S2コンテナ経由でPropertyクラスを取得する

```
S2Container container = SingletonS2ContainerFactory.getContainer();
DataSourceProperty dataSourceProperty = (DataSourceProperty) container
                                        .getComponent(DataSourceProperty.class);

String user = dataSourceProperty.getUser();
```

※ 実務では、うまくユーティリティクラスを作りましょう

例：Java5 のジェネリックを使えば、キャスト不要でいい感じになります

```
DataSourceProperty property = S2Util.getComponent(DataSourceProperty.class);
```

さらばプロパティファイル⑦

- 環境依存問題に対処しよう
  - 環境が変わるたびにあちこちのdiconファイルの値を修正するのは避けたい！
- Seasar 2.4でスマートな機能が登場したのと、説明に時間が掛かるので詳細は割愛
  - 興味のある方は、気軽に私に尋ねて下さい！
  - 一度、設定を仕込めば、以下のような property.dicon ファイルでコメントを切り替えるだけで対応できます

```
<!-- ##### 環境依存ファイル (いずれか１つをインクルードする) ##### -->
<!-- <include path="env.dicon"/> --> <!-- 本番環境 -->
<!-- <include path="env_it.dicon"/> --> <!-- 結合テスト環境 -->
<include path="env_ut.dicon"/> <!-- ユニットテスト環境 -->
```

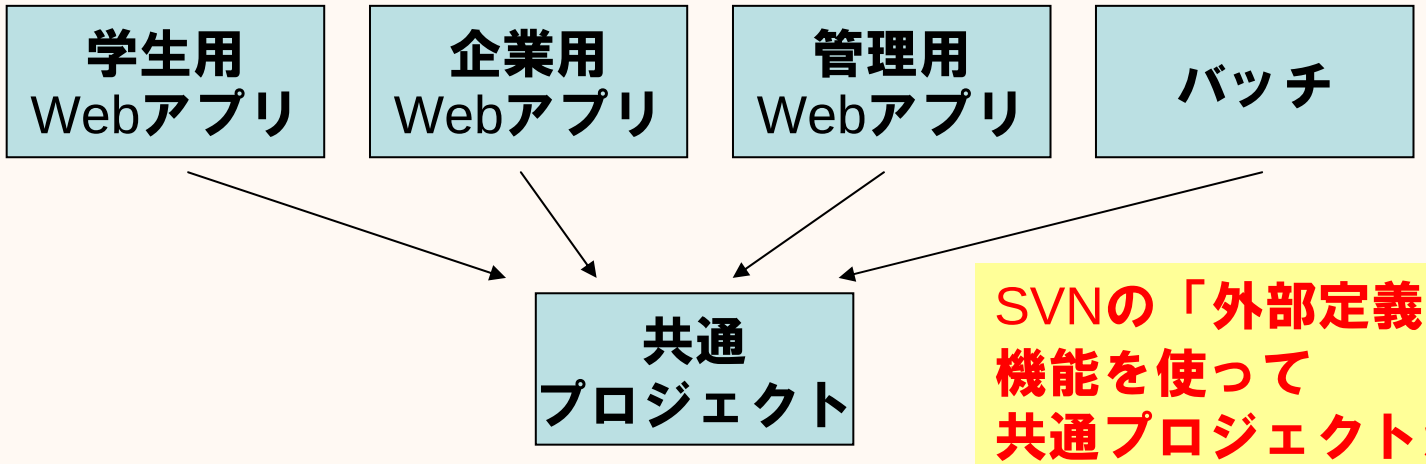
複数のWebアプリをどうやって統括するか？①

- 例えば、マッチングサイトで、
「学生Webアプリ」と「企業Webアプリ」、
「管理Webアプリ」、「バッチ」を作る場合

- EntityやDao, Utlity, diconファイルなど各Webアプリでの共通モジュールをどう管理するか？
  - 各Webアプリでの重複は絶対に避けたい
  - 更新頻度を考えると共通部分のjar化もイマイチ

複数のWebアプリをどうやって統括するか？②

- Subversionの「外部定義」機能はとっても素敵
  - 一度ディレクトリのマウント設定をすれば、個別アプリのディレクトリーツリーに共通プロジェクトのサブディレクトリを組み込めます



SVNの「外部定義」機能を使って共通プロジェクトから必要なディレクトリをマウントする

複数のWebアプリをどうやって統括するか？③

- 手順
  - 共通プロジェクトを作り、共通部分をここに集める
  - common-app.diconのような共通のdiconファイルを作成
    - Dao, Entity など共通箇所の設定をここに移す
    - app.dicon から common-app.dicon をインクルードさせる
  - Subversionの「外部定義」機能を使って、共通プロジェクトのディレクトリを個別プロジェクトにマウントする
    - svn:externals属性
    - Eclipseのsubversionプラグインを使えば比較的簡単に実現できる
    - 例：
      ある個別アプリのあるSVN上のディレクトリに共通プロジェクトのDaoディレクトリをマウントする際のSVNプロパティの設定値

      svn:externals dao https://svn.xxx.co.jp/hoge/trunk/common/src/main/java/jp/co/tafc/sample/dao

Interceptor Tips

- Interceptor内でHttpServletRequest を取得する方法
  - Interceptorクラス内でHttpServletRequestの setterメソッドを定義してもDIされない
  - 自身よりライフサイクルが短いものはDI不可
    - Interceptorのインスタンス属性は singleton
    - HttpServletRequestのインスタンス属性は request
    - singleton は request よりも長い
  - HttpServletRequestはS2コンテナ経由で取得可能

```
S2Container container = SingletonS2ContainerFactory.getContainer();
HttpServletRequest request = (HttpServletRequest) container
                                        .getComponent(HttpServletRequest.class);
```

リファラー

- 一部のセキュリティ対策ソフトでは、
  デフォルトでリンク元（リファラー情報）を
  送信しない設定になっている
  - 例：シマンテック社のノートンインターネットセキュリティ

- リファラー情報
  （HttpServletRequest#getHeader(“Referer”)）
  を使うモジュールを使用・開発する際は要注意

非機能要件対策

- 下記の非機能要件は
「AOP」、「アノテーション」、「名前規約」、「ServletFilter」などを駆使して、
直接的に個別のソースコードを記述することなく、
宣言的・透過的に処理することができる

項目	説明
戻るボタン対策	登録完了後に戻るボタンを押して再登録されるのを防止する
ダブルクリック対策	ダブルクリックによる2重送信を防止する
排他制御	多人数での同時利用環境で、同じデータが更新・削除
存在チェック	既に他のユーザーによりデータを削除されていないかをチェック
URL直打ち対策	URL直打ちによるエラーを防止する
ログイン認証	ログインしていないユーザーへのアクセスを禁止する
権限チェック	権限の無いユーザーには機能を利用させなくする処理
タイムアウト処理	セッションタイムアウト時にログインページ等へ強制転送する
トレースログ	障害発生時の原因究明作業を考慮して、ログインIDなども仕込む
例外処理	メソッド引数、リクエスト、セッション情報をログに書き込み、メール送信

スピーカーブースで
お待ちしています
気軽にお越しください

ご清聴
ありがとう
ございました