StrutsからSAStrutsへ

株式会社ティーアンドエフカンパニー

出羽 健一

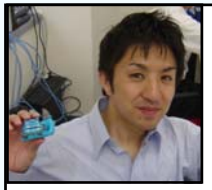
http://d.hatena.ne.jp/dewa/

アジェンダ

- 自己紹介・はじめに
- Strutsのおさらい
- SAStrutsの紹介
- 簡単なS2JDBCの紹介
- Todoリスト開発のデモ

自己紹介

プロフィール



出羽 健一

所属：㈱ ティーアンドエフカンパニー
チーフアーキテクトとして
Web+DBアプリの開発に従事。

日本初のコンピュータ専門大学で有名な会津大学出身。学生時代より実務プロジェクトに携わり、経験プロジェクト本数は50を超える。同大学生が中心となって作ったWebアプリ開発会社、㈱ ティーアンドエフカンパニーの設立メンバー。
平成14年、考案したソースコード自動生成ツールが経済産業省の技術研究開発補助金事業に採択される。平成16年～20年、金沢工業大学 大学院工学研究科 客員准教授。

Seaserプロジェクト

- Seasar Conference 2007 Autumn で登壇
  実践的なサンプルアプリをその場でコーディングします！
  （http://event.seasarfoundation.org/sc2007autumn/Session#c4）
- Seasar Conference 2007 Spring で登壇
  半歩先行くSeasar2の実践活用
  （http://event.seasarfoundation.org/sc2007spring/Session）
- Seasar Conference 2006 Autumn で登壇
  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～
  (http://event.seasar.org/sc2006autumn/Session)

執筆活動

- WEB+DB PRESS vol.41

＜特集２＞
つらいJavaから楽しいJavaへ
Seasar2サクサク開発
実践カリキュラム

- WEB+DB PRESS vol.38

＜特集１＞
無駄なコードを書かない技術

メディア紹介記事

- ITpro
  HTML画面をそのまま仕様書に
  5カ月で1000画面を構築した就職サイトPuffの高速開発手法
  (http://itpro.nikkeibp.co.jp/article/COLUMN/20070214/261859/)

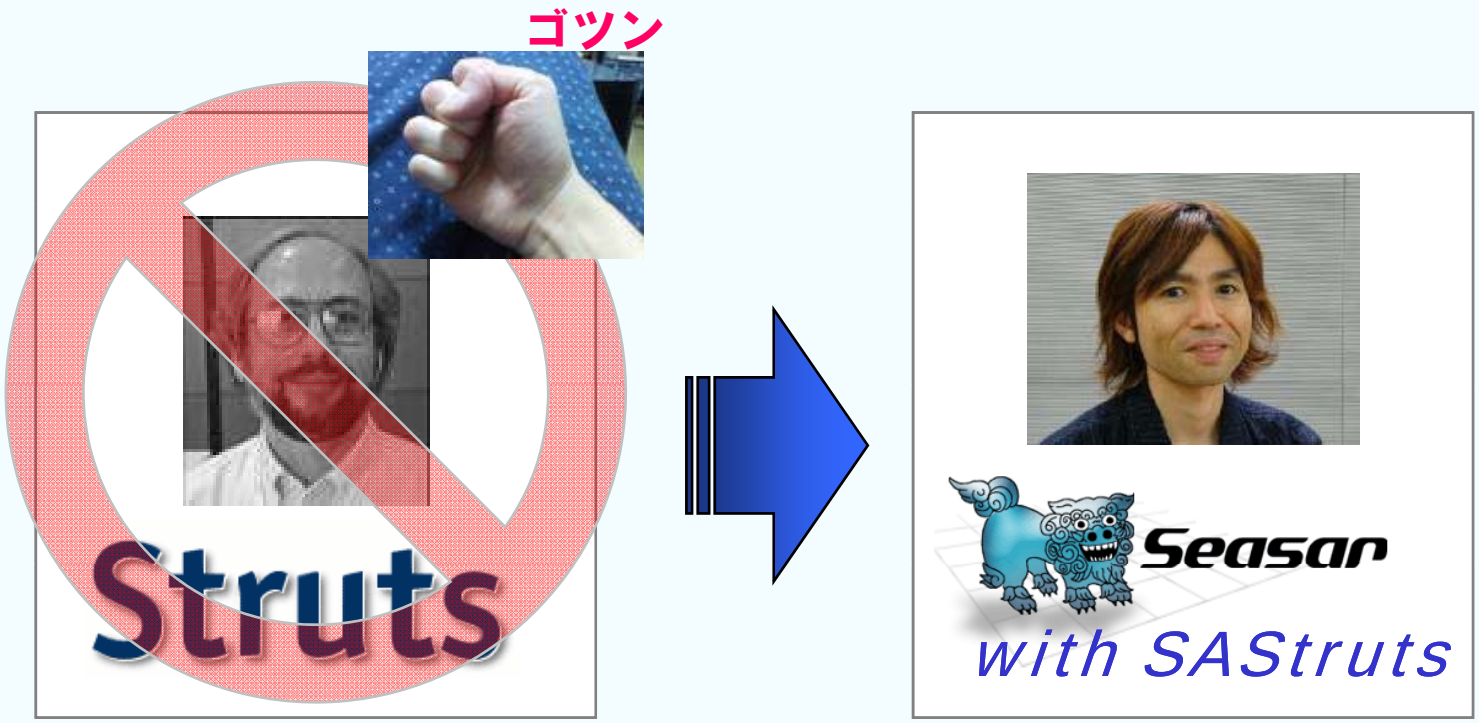
- コンポーネントスクウェア コラム記事
  Javaコンポーネント技術解説：Seasar2
  (http://www.c-sq.com/modules/cbdsqcont/index.php?category=8)

本日のタイトル

『Strutsから SAStrutsへ』

タイトルが紛らわしいのですが、、、

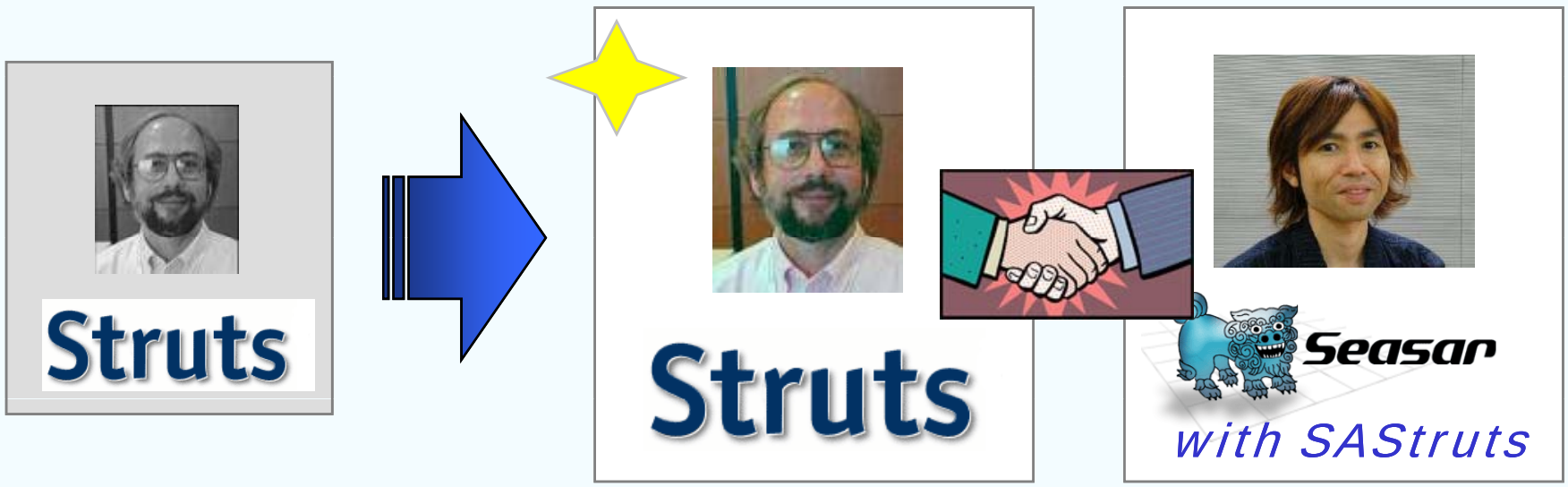
■こんなお話ではありません。



From **To**

タイトルが紛らわしいのですが、、、

■こんなお話です！



From **To**

Strutsをより使いやすく！

Strutsとは

- 良くも悪くもJava界隈で
  デファクトスタンダードな
  Webアプリケーションフレームワーク
  - 2001年のβ版リリース以来、
    急速に世の中に広まった

現場の開発者よりも、
マネージャーに人気にある

⇒ 無難だから…

Strutsのメリット

- Strutsリソースが膨大
  - 人　・・・ プログラマ、コミュニティ
  - モノ　・・・ ライブラリ、ツール、関連プロダクト
  - 金　・・・ ビジネス案件
  - 情報　・・・ 書籍、Webページ、コンサルサービス、教育サービス
- 拡張が容易
  - Strutsを独自拡張するプロジェクト、企業、製品が多い
- 大規模アプリケーションの十分な実績
  - 枯れている（ただし、バージョン　1.2.9）
  - 高い実行性能（パフォーマンス、安定性）

バージョン1.2.9

- なぜ、バージョン1．2．9なのか？
  - Strutsの全バージョンの中で最も枯れていて、大規模開発実績が多いから
  - 1.3系は大幅にコードが書き換えられていてまだ安定してない（らしい）
  - 2.x系はほとんど別物

Strutsのデメリット

- 設定ファイル地獄
- ソースコード修正時のサーバ再起動
- アクションクラス数の増加による管理が大変
- テストしづらい
- ・・・

いろいろあるが要は、

生産性が低い

なぜStrutsはデファクトになれたのか？

■ Strutsの開発者曰く、

Sun Microsystems, Inc.
シニア・スタッフ・エンジニア

Craig McClanahan 氏

•技術的にシンプル

であったこと

•既存技術からの移行方法を

明確に示したこと

•コミュニティの存在

•いろいろな開発ツールに

組み込まれたこと

参考：
http://sdc.sun.co.jp/news/2005/03/feature050301.html

Strutsは「技術的にシンプル」なのか？

- （生産性はともかく、）
  「提供している機能のカバー領域が
  本質的で無駄がない」という意味でシンプル

ダメなのは、設定ファイルを書かなければいけないこと。

参考：Strutsをなめんな
http://d.hatena.ne.jp/higayasuo/20080214

今のままだと、、、

- 生産性の低いStrutsで開発している
  デベロッパーが可哀想だ

開発者の告白

Strutsはつらいよ

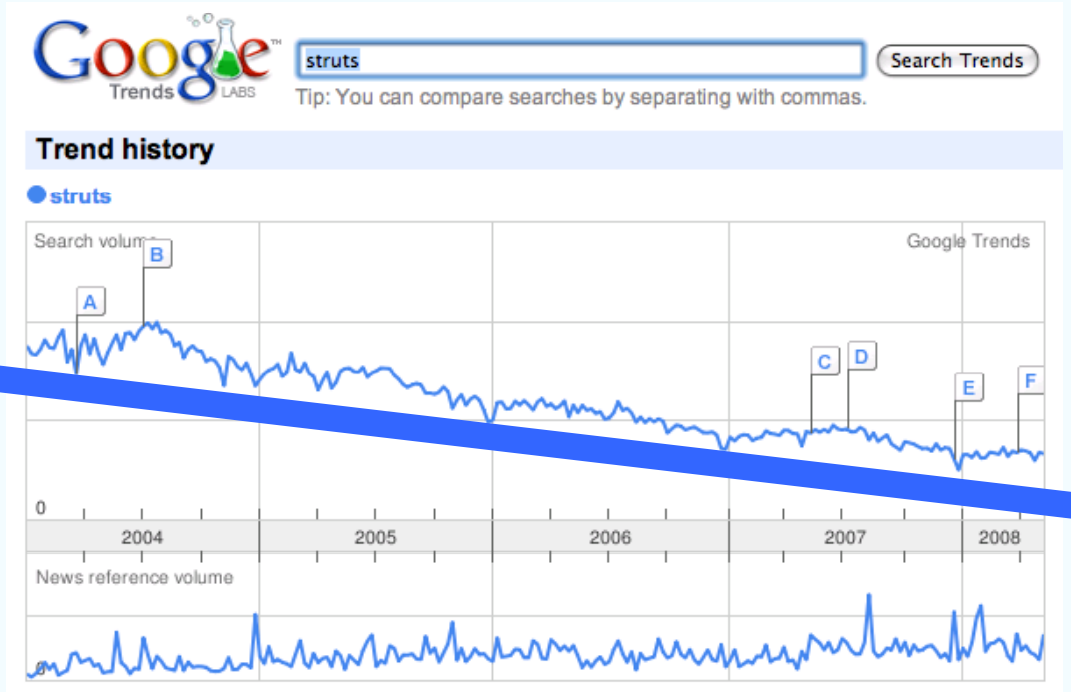
今のままだと、、、

- 無難だという理由でStrutsの採用決めた
  プロジェクトマネージャー

- 将来も今の開発予算・期間・品質で
  通用するだろうか？という不安感

Strutsの傾向分析

- Google Trends で “struts” を調べてみる



下降気味

http://www.google.com/trends?q=struts

Strutsはこのまま沈むのか？

救世主


遂に
登場

お待たせ
しました！

えすえ～
すとらっつぅ～

重要：ためらわずに『大山のぶ代』っぽく発声

Strutsの救世主

■ そこでStrutsの救世主としての

SAStruts ですよ！

– StrutsはSI業界で最も多く利用されている
Java言語のデファクトのフレームワークであり
背負うものが大きい

– それだけに、プログラマ、SE、PM、Java界隈、SI業界、
そして、お客様に「導入して良かった」と言って頂けると
嬉しいです

SAStrutsとは

- SAStrutsはStruts1.2.9ベースの
  シンプルで高い生産性を実現する
  より良いStrutsです

公式サイト
http://sastruts.seasar.org/
2008年1月に 1.0.0 リリース

メイン開発者

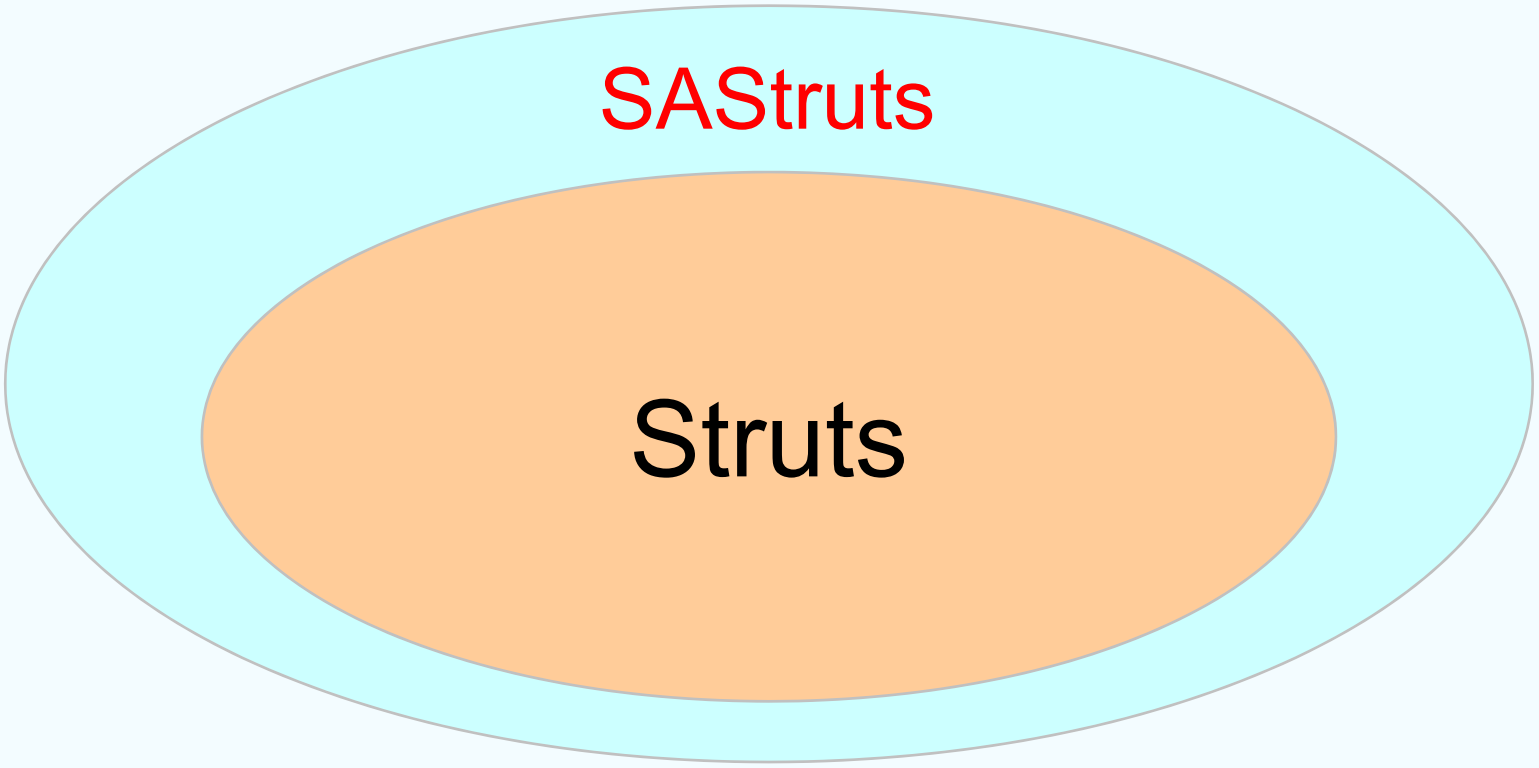
Seasarプロジェクト チーフコミッター
ひがやすを氏
（ http://d.hatena.ne.jp/higayasuo/ ）

※ SA = Super Agile

SAStrutsの基本構造

- 大規模プロジェクトでの実績あるプロダクトを内部的にそのまま採用して利点を活かす

–薄いラッパー



薄いラッパー

- どれくらい薄いのか？
  - SAStrutsのクラス数は約80個と少ない
    - さらに、ほぼロジックが含まれないannotationとexceptionのパッケージを除けば約40個

パッケージ	SAStruts (1.0.1)	Struts (1.2.9)
action	13	25
actions		17
config	5	27
plugins		2
taglib	2	130
util	14	16
tiles		44
upload		14
validator	4	13
filter	1	
interceptor	1	
customizer	1	
enums	1	
annotation	26	
exception	11	
合計	79	288

※ 複雑さを表す1つの指標はクラス数

薄いラッパー

■どのくらい薄い（小さい）のか？

struts-1.2.9.jar	536KB
sa-struts-1.0.1.jar	92KB

クラスが少なくて小さいと何が嬉しいのか？

- プロダクトとして早く枯れやすい
  - バグの混入余地が少ない
  - メンテナンス（バグ修正）が容易
- 内部構造の見通しが良くて把握しやすいため、各プロジェクト、各企業で独自拡張しやすい
  - Struts拡張の経験がある人は同じノウハウが使える

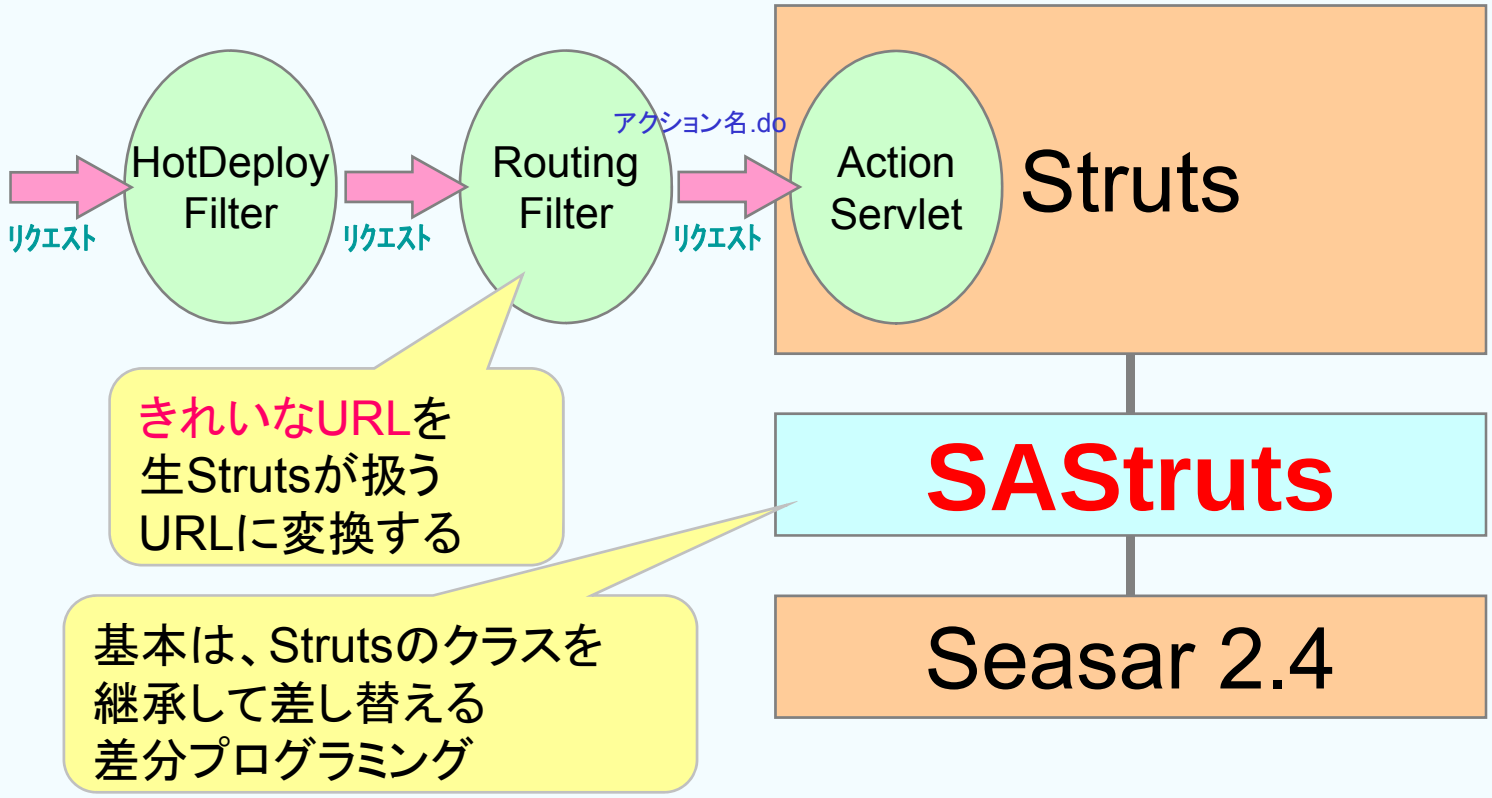
どのようにStrutsと共存しているのか？

きれいなURL

/blog/archive/2008/05 ⇒

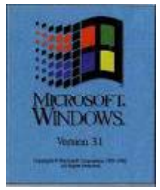
Struts用URL

/blog.do?command=archive&year=2008&month=05



Strutsの駄目な使い勝手を周りから改善するのがSAStrutsの役割。
コアなエンジンは、安定実績のあるStruts に委ねている。

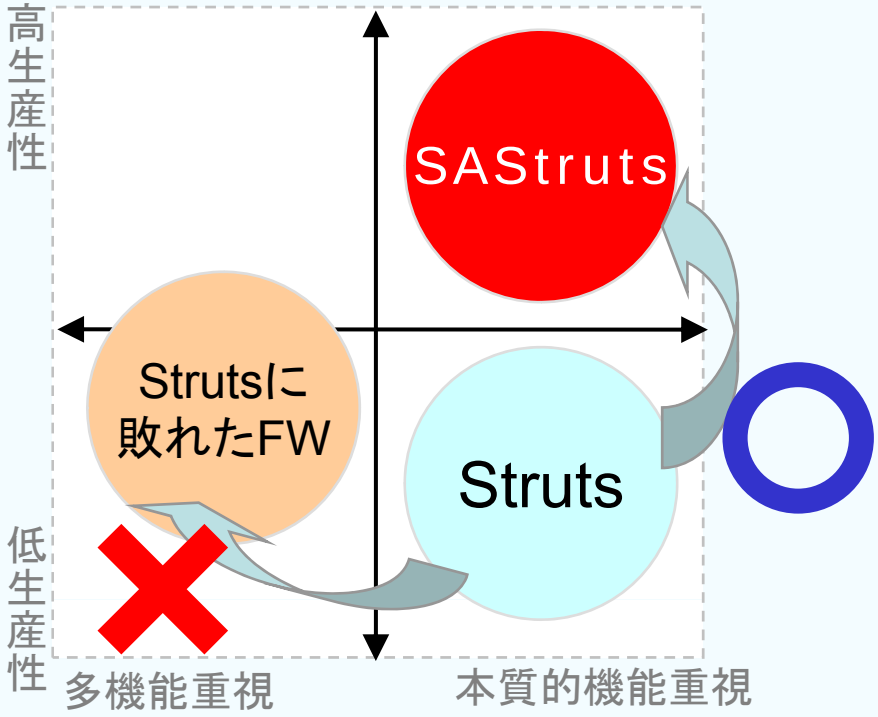
ラッパー戦略の成功事例に学ぶ

	中身	外側	成功要因
Windows 3.1 	MS-DOS ・当時のデファクトのPC用OS ・使い勝手が悪い	・使い勝手を大幅に改善するGUI	・中身の普及度・実績 ・中身を内包（いざとなったらそのまま使える） ・使い勝手の改善 ⇒ 新しい利用者を開拓
C++ 	C言語 ・当時のデファクトプログラミング言語 ・モジュール化の手法が貧弱	・より使い勝手を向上させるオブジェクト指向	
SAStruts 	Struts(1.2.9) ・デファクトFW ・使い勝手が悪い	・HOT deploy ・設定ファイルをほぼ書かない	これからだが成功プロダクトとの類似点多し

ポジショニング（その１）

- 多機能より本質的な機能を重視
  - 肥満化・複雑化しないように気を配る
  - 足らない機能はユーザーに委ねる
    （フレームワークが頑張り過ぎない）

薄い
フレームワーク



なぜ生産性がそんなにも重要なのか？

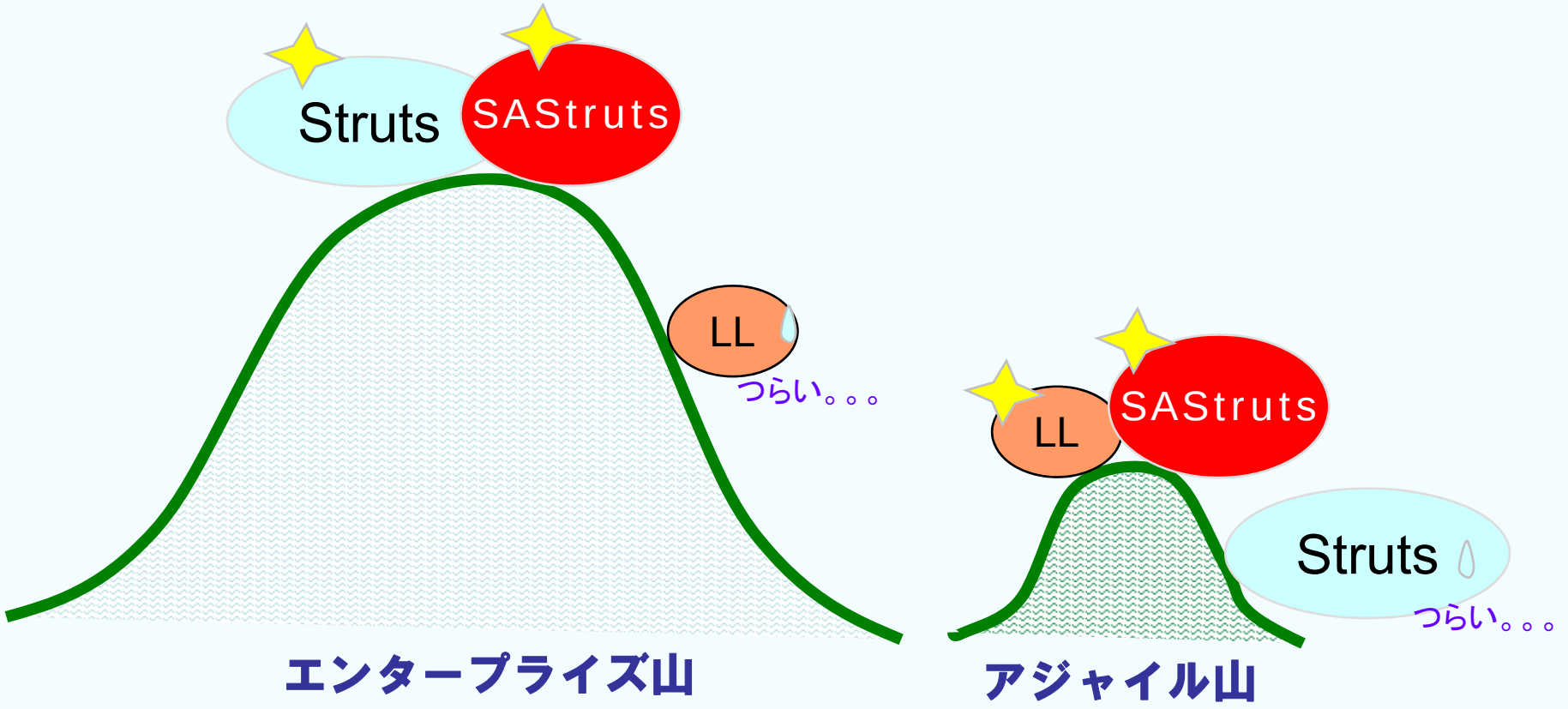
- 生産性が高くなれば、
  - 各プロジェクトで必要な開発者数を減らせる
    - コミュニケーションに割く労力を減らせる
    - マネジメントに割く労力を減らせる
  - 機能やパフォーマンス、品質の改善といったビジネス要求への取り組みに時間を割ける

実利	マインド
コストが安くなる	プログラマは同じ時間で いっぱいアプリが書けて嬉しい

参考：「Java から Ruby へ　マネージャのための実践移行ガイド」
Bruce A. Tate(著), 角谷 信太郎(翻訳) / オライリー・ジャパン

ポジショニング（その2）

■エンタープライズからアジャイルまで



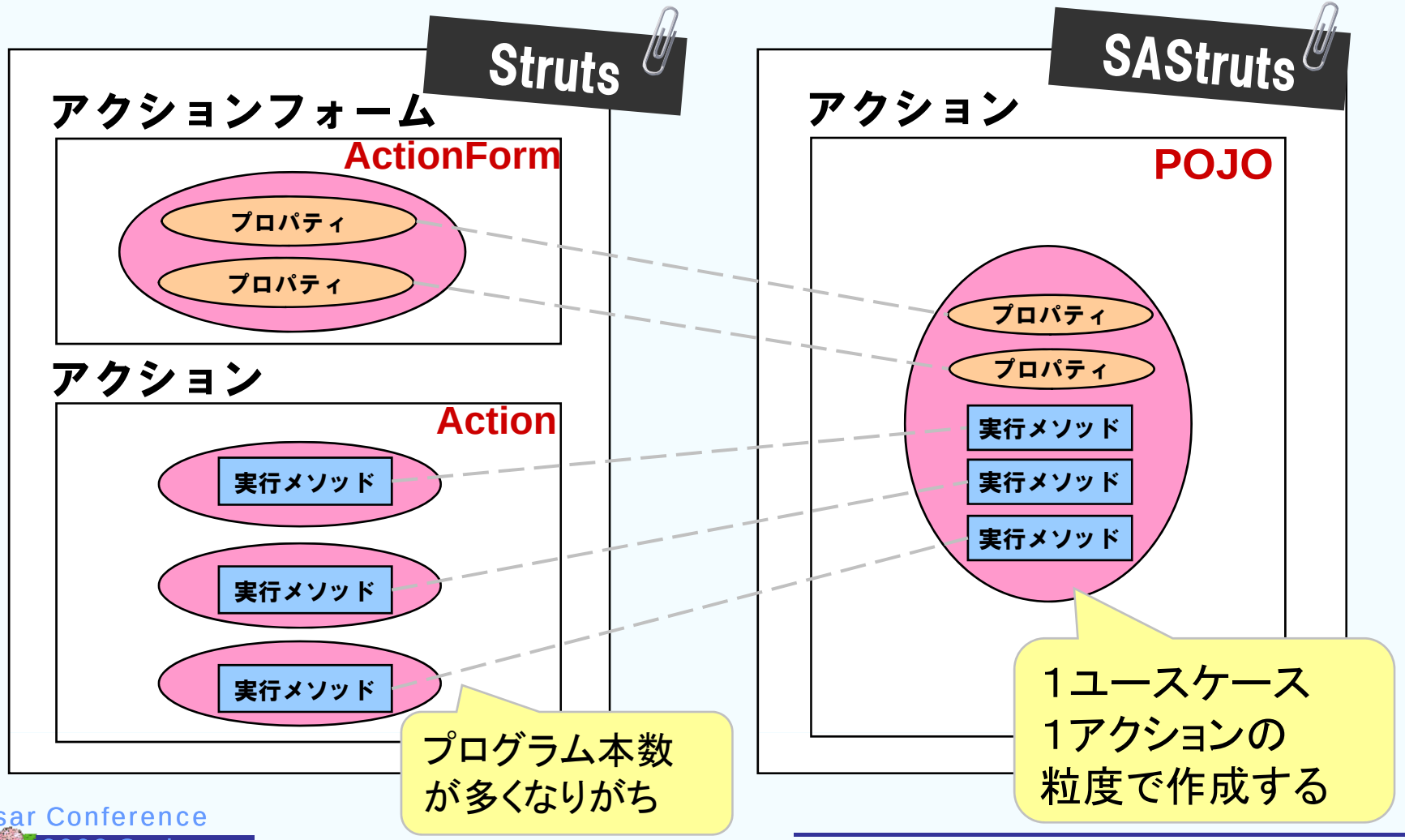
※ **LL** … PHP,Perl,Python,Rubyに代表されるLightweight Language。
ここではLLを使用したWebアプリケーションフレームワークの意。

アジャイルに必要な特性とその対策

特性	SAStrutsの対策
短時間でコーディングに着手できる環境が整う	・プロジェクト雛形自動生成（Dolteng ）
修正 ～ 反映までの時間が短い	・HOT Deploy / 設定ファイル不要 ・**SAStruts プラグイン（Eclipse）**
学習コストが低い	・本質的機能重視なので機能が少ない
コード記述量が少ない	・publicフィールド ・宣言的プログラミング(アノテーション)
プログラム本数が少ない	・**アクションのアーキテクチャを改善**

アクションのアーキテクチャを改善

■ 従来の複数のアクションをPOJOベースで
１アクションでまとめて記述できるようになった



Hello World

URL

http://localhost:8080/sa-struts-tutorial/helloWorld/hello

アプリ名　アクション名　メソッド名

アクション

HelloWorldAction.java

```
package tutorial.action;
public class HelloWorldAction {

    public String message;

    @Execute(validator = false)
    public String hello() {
        message = "Hello World!";
        return "hoge.jsp";
    }
}
```

戻り値に遷移先のパスをセット

JSP

hoge.jsp

```
${f:h(message)}
```

$で始まるEL式でプロパティを参照

生成されるHTML

ブラウザでの表示

Hello World!

View層について

- JSPも進化しています
  - JSP 2.0から導入されたEL式とJSTLの組み合わせはなかなか良い仕様
  - EL式(Expression Language)
    - **${～}** の形式で記述。
      オブジェクトのアクセスを簡略化して記述できる式言語
  - JSTL(Java Server Pages Standard Tag Library)
    - JSPのバージョン2.0で標準化されたタグの集まり
      例：日付データを指定フォーマットで出力する

```
<fmt:formatDate value="${date}" pattern="yyyy年MM月dd日" />
```

SAStrutsを使った人の感想

■体験者の声(ブログから抜粋)

Strutsを長くやっているため、
SAStrutsは非常にとっつきやすい

うすい機能で、
大規模で使いやすい
と思う。

SAStrutsとStrutsのアクションクラスのソースコードは
見た目が全然異なるだけに、Strutsの知識が活用でき
てしまうのは、不思議な感覚です

現場の事が良く分かっている人
が作っているので、
実務で使いやすいと思う。

小規模な開発だとRailsがJavaより
圧倒的に生産性が高いかというと、今のJavaには、HOT deployのできるSeasar2がある

さくさく開発できる感覚
はいい、Javaじゃない感じ

SAStrutsの利用状況

- 2008年1月に正式版がリリースされたばかりにもかかわらず、既に数社の大企業が採用を決定している
- ブログやMLを見るかぎり、既にいくつかの実案件で投入されている
- 「sastruts」をGoogle で検索すると66,500件
  - 「s2dao」が157,000 件であることを考慮すると勢いが感じられる

SAStrutと S2JDBCとの 連携デモの前に

S2JDBCのおさらい

S2JDBCとは？

- Seasar 2.4本体にバンドルされるようになった
  データアクセス用フレームワーク

- 「SQL書きたい派」と「SQL書きたくない派」の
  両方のニーズに対応

■特徴①

– 読みやすくて書きやすい
「流れるようなインターフェース」で
90%のSQL文を生成する

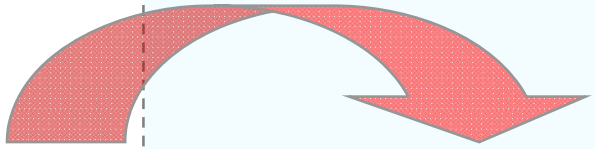
```
List<Employee> results = jdbcManager.from(Employee.class)
        .leftOuterJoin("department")
        .where("id in (? , ?)", 11, 22)
        .orderBy("name desc")
        .getResultList();
```

■特徴②

– publicフィールド対応

```
@Entity
public class Employee {
    @Id
    private Integer id;

    private String name;

    private BigDecimal salary;

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public BigDecimal getSalary() {
        return salary;
    }

    public void setSalary(BigDecimal salary) {
        this.salary = salary;
    }
}
```

publicフィールド対応



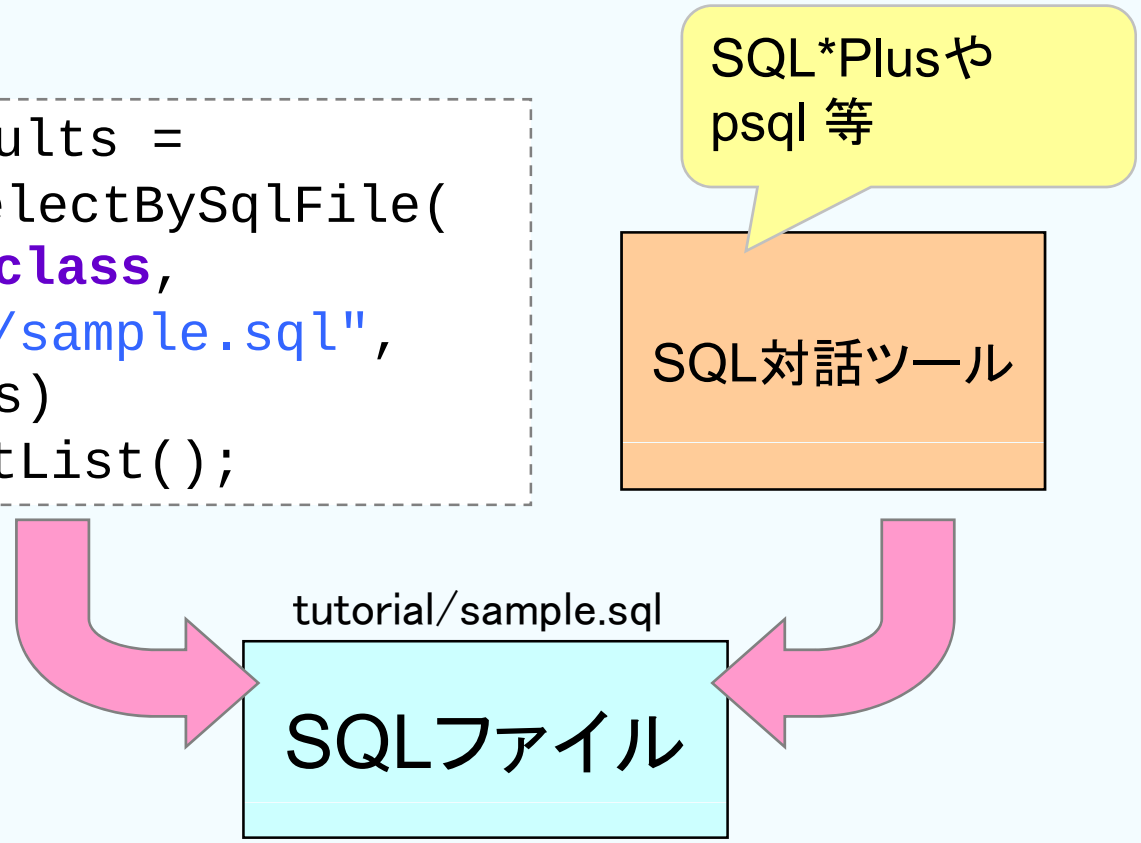
```
@Entity
public class Employee {
    @Id
    public Integer id;

    public String name;

    public BigDecimal salary;
}
```

可読性が高い

■特徴③

– S2Daoで大好評の2Way SQL をサポート

```
List<Employee> results =
    jdbcManager.selectBySqlFile(
        Employee.class,
        "tutorial/sample.sql",
        conditions)
        .getResultList();
```



「SQL対話ツール」と
「S2JDBC」の両方が
SQLファイルを読み込み可能

■特徴④

- 学習コストが低く、トラブりにくい
  - 公式ドキュメントが充実
    - これだけ読めば理解できる構成になっている
  - 強力だがトラブりやすい機能は意図的に排除
    - JPAやHibernateのLazy lodingやEager loadingなど

S2JDBCの公式ドキュメント

■どこにあるか分かりにくい。。。（笑）

Seasarのトップページ

http://www.seasar.org/

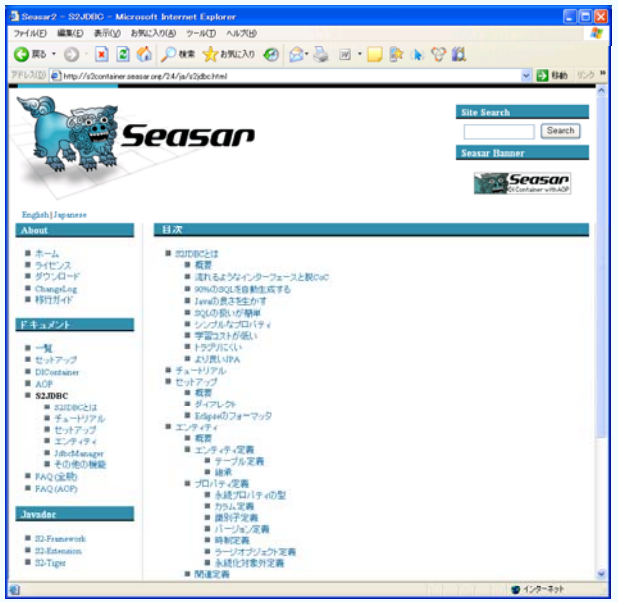
Seasar2
(S2Container)

Seasar2

http://s2container.seasar.org/2.4/ja/

S2JDBC

Seasar2 - S2JDBC -

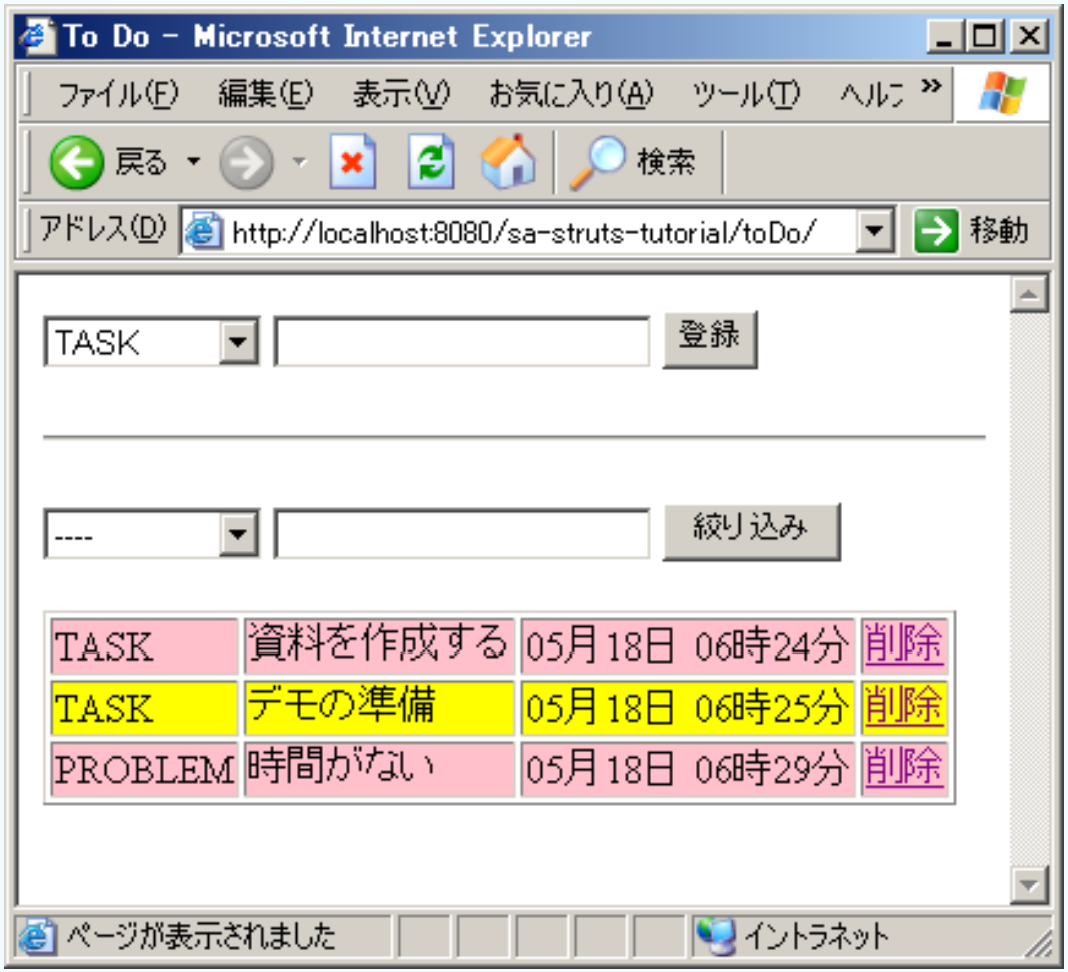


http://s2container.seasar.org/2.4/ja/s2jdbc.html

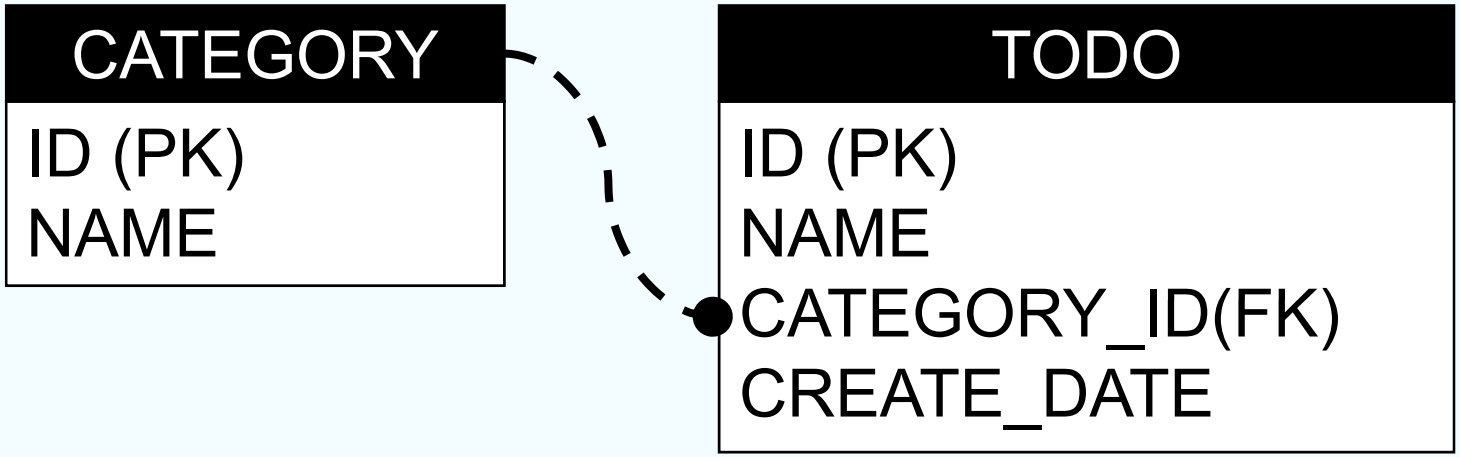
後半は
ライブ
コーディング

簡単なTodoリストを作ります

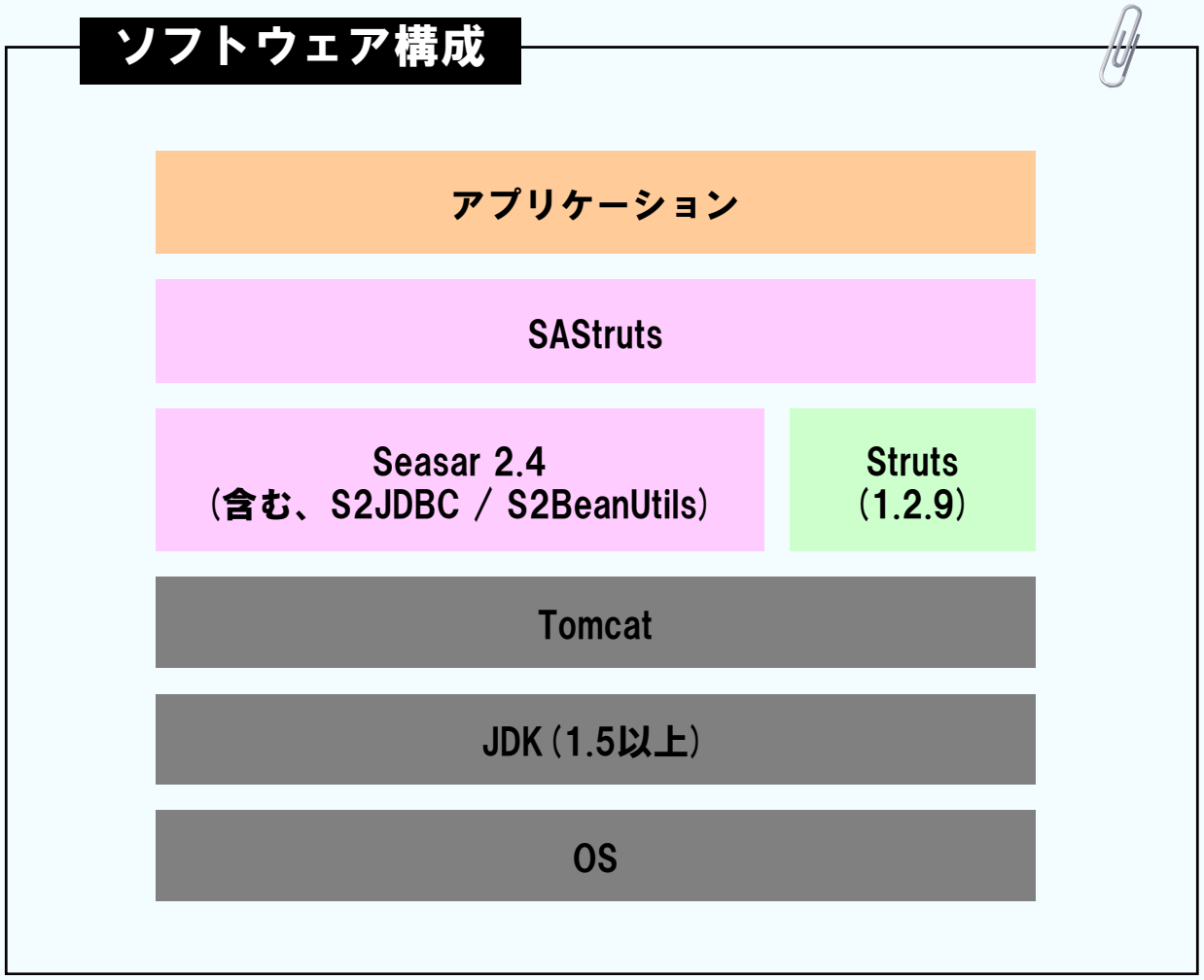
Todoリストアプリケーションの完成イメージ



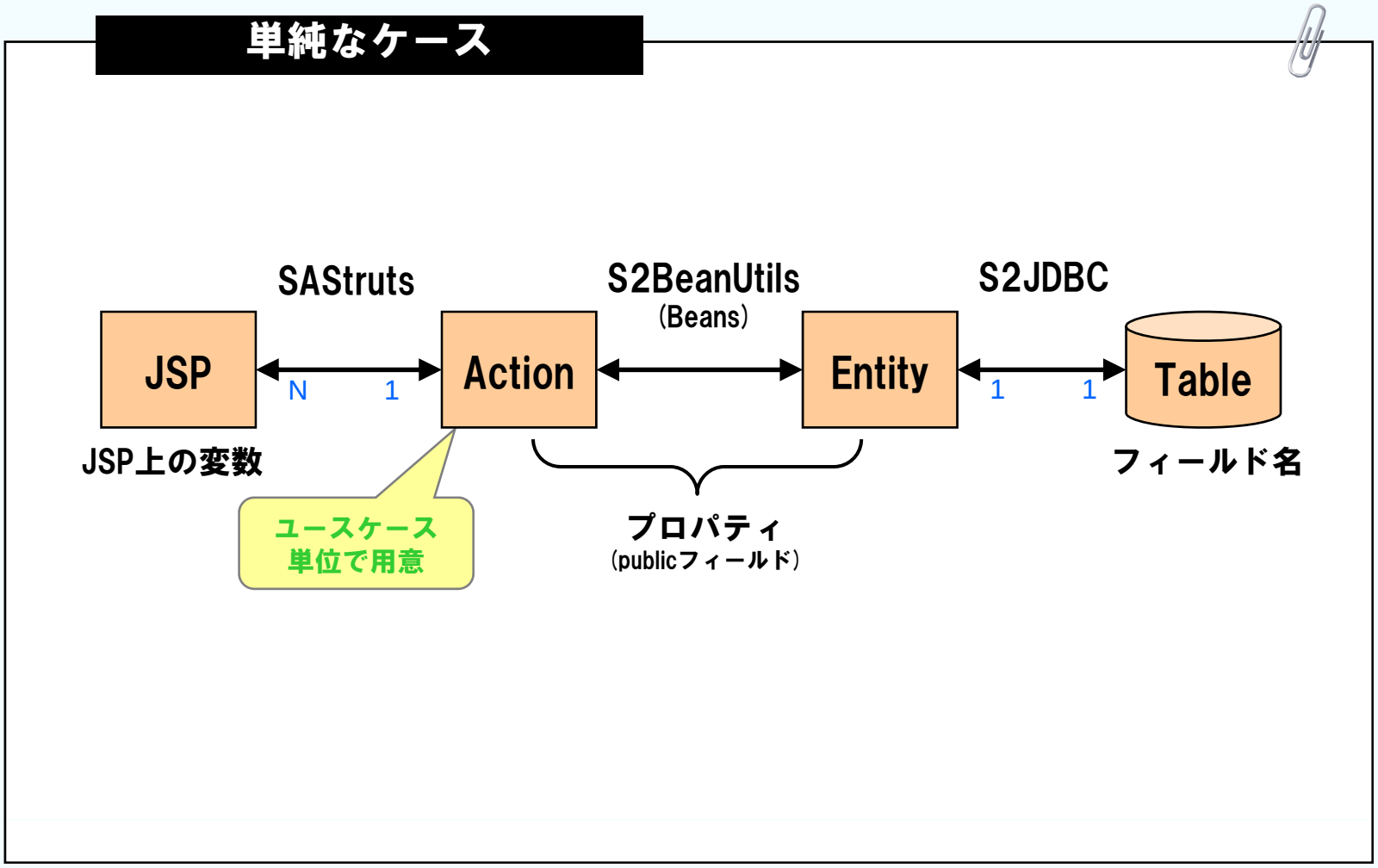
TodoリストのERD



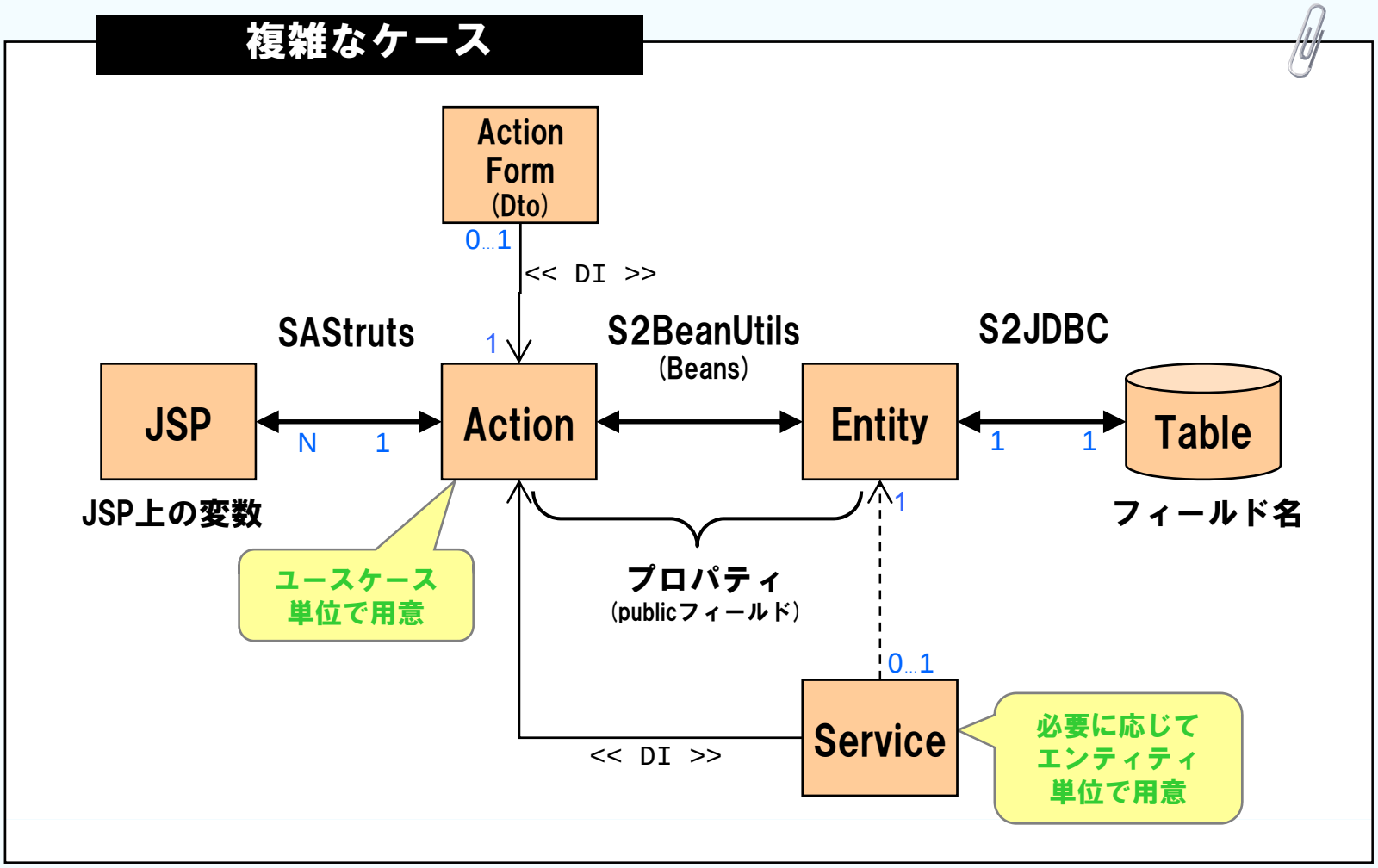
ソフトウェア構成



SAStruts + S2JDBCのアーキテクチャ（その１）



SAStruts + S2JDBCのアーキテクチャ（その２）



デモ開始

ぜひ、実際にダウンロードしていろいろ試してみて下さい！

Getting Start!

1. SAStrutsの公式チュートリアルを読む
   - http://sastruts.seasar.org/tutorial.html
2. プログラム環境を整える
   - チュートリアルのセットアップ
     http://sastruts.seasar.org/setup.html
3. チュートリアルを動かしたり、ソースを読む

■忘れっぽいエンジニアの Jakarta Strutsリファレンス

SAStrutsと相性バッチリの
Struts1.2.9, JDK1.5

http://struts.wasureppoi.com/

Appendix
～ その1 ～

- デモで用いた
  Todoアプリケーションの
  ソースコード

SQLスクリプト（HSQLDB用）

```
CREATE TABLE category (
  id integer generated by default as identity,
  name varchar(50) not null)

CREATE TABLE todo (
  id integer generated by default as identity,
  name varchar(255) not null,
  category_id integer,
  create_date timestamp,
  constraint fk_category FOREIGN KEY(category_id)
    REFERENCES category(id))

INSERT INTO category VALUES(1,'TASK')
INSERT INTO category VALUES(2,'PROBLEM')

INSERT INTO todo VALUES(1,'TASK 1',1, '2008-01-01')
INSERT INTO todo VALUES(2,'TASK 2',1, '2008-01-02')
INSERT INTO todo VALUES(3,'PROBLEM 1',2, '2008-01-03')
INSERT INTO todo VALUES(4,'PROBLEM 2',2, '2008-01-04')
```

作成ファイル

- Model (エンティティ)
  - Category.java
  - Todo.java
- View (JSP)
  - index.jsp
- Controller (アクション)
  - TodoAction.java

Category

Category.java

```
@Entity
public class Category {

    @Id
    @GeneratedValue
    public Integer id;

    public String name;

}
```

エンティティ

Todo

Todo.java

```
@Entity
public class Todo {

    @Id
    @GeneratedValue
    public Integer id;

    public String name;

    public Integer categoryId;

    @ManyToOne
    public Category category;

    @Temporal(TemporalType.TIMESTAMP)
    public Date createDate;

}
```

一覧表示(絞り込み無し)

■画面イメージ

TASK	資料を作成する	05月18日 06時24分	削除
TASK	デモの準備	05月18日 06時25分	削除
PROBLEM	時間がない	05月18日 06時29分	削除

■JSP

index.jsp

```
<table border="1">
  <c:forEach var="todo" varStatus="s" items="${todoList}">
    <tr style="background-color:${s.index % 2 == 0 ? 'pink' : 'yellow'}">
      <td>${f:h(todo.category.name)}</td>
      <td>${f:h(todo.name)}</td>
      <td><fmt:formatDate value="${todo.createDate}"
      pattern="MM月dd日 hh時mm分"/></td>
      <td><a href="delete/${f:u(todo.id)}">削除</a></td>
    </tr>
  </c:forEach>
</table>
```

一覧表示(絞り込み無し)

■アクション

todoAction.java

```
public JdbcManager jdbcManager;

public List<Todo> todoList;

@Execute(validator = false)
public String index() {
    todoList = jdbcManager.from(Todo.class)
        .leftOuterJoin("category")
        .orderBy("createDate")
        .getResultList();

    return "index.jsp";
}
```

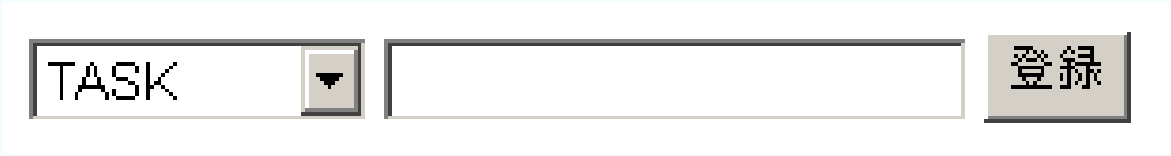
一覧表示(絞り込み無し)

■ 生成されるHTML

```
<table border="1">
    <tr style="background-color:pink">
      <td>TASK</td>
      <td>資料を作成する</td>
      <td>05月18日 06時24分</td>
      <td><a href="delete/21">削除</a></td>
    </tr>

    <tr style="background-color:yellow">
      <td>TASK</td>
      <td>デモの準備</td>
      <td>05月18日 06時25分</td>
      <td><a href="delete/22">削除</a></td>
    </tr>

    <tr style="background-color:pink">
      <td>PROBLEM</td>
      <td>時間がない</td>
      <td>05月18日 06時29分</td>
      <td><a href="delete/25">削除</a></td>
    </tr>
</table>
```

登録用画面部品の初期化 & 登録処理

■画面イメージ



■JSP

index.jsp

```
<html:errors/>
<s:form action="/todo">
  <html:select property="categoryId" >
    <html:options collection="categoryItems"
      property="id" labelProperty="name"/>
  </html:select>
  <html:text property="name" />
  <input type="submit" name="register" value="登録" />
</s:form>
```

登録用画面部品の初期化 & 登録処理

■アクション

– 登録用画面部品の初期化

todoAction.java

```
public JdbcManager jdbcManager;

public List<Category> categoryItems;

public Integer categoryId;
public String name;

@Execute(validator = false)
public String index() {
    categoryItems = jdbcManager.from(Category.class).getResultList();
    name = "";

    ...

    return "index.jsp";
}
```

登録用画面部品の初期化 & 登録処理

生成されるHTML

```
<form name="todoActionForm" method="post"
  action="/sa-struts-tutorial/todo/">
  <select name="categoryId">
    <option value="1">TASK</option>
    <option value="2">PROBLEM</option>
  </select>
  <input type="text" name="name" value="">
  <input type="submit" name="register" value="登録" />
</form>
```

登録用画面部品の初期化 & 登録処理

■アクション

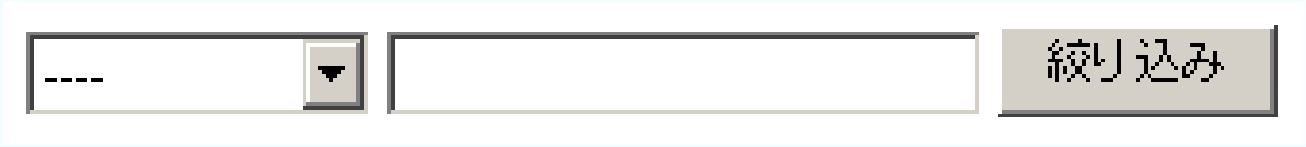
– 登録処理

todoAction.java

```
public JdbcManager jdbcManager;

public Integer categoryId;
public String name;

@Execute(input = "index")
public String register() {
    Todo todo = Beans.createAndCopy(Todo.class, this).execute();
    todo.createDate = new Date();

    jdbcManager.insert(todo).execute();

    return "/todo?redirect=true";
}
```

絞り込み機能

画面イメージ



JSP

```
<s:form action="/todo">
  <html:select property="cond_categoryId_EQ" >
    <option value="">----</option>
    <html:options collection="categoryItems"
      property="id" labelProperty="name"/>
  </html:select>
  <html:text property="cond_name_STARTS" />
  <input type="submit" name="search" value="絞り込み" />
</s:form>
```

絞り込み機能

アクション

todoAction.java

```
public JdbcManager jdbcManager;

public List<Category> categoryItems;

public Integer cond_categoryId_EQ;
public String cond_name_STARTS;

@Execute(validator = false)
public String index() {
    categoryItems = jdbcManager.from(Category.class).getResultList();
    name = "";

    ...

    return "index.jsp";
}

@Execute(input = "index")
public String search() {
    return index();
}
```

絞り込み機能

生成されるHTML

```
<form name="todoActionForm" method="post"
  action="/sa-struts-tutorial/todo/">
  <select name="cond_categoryId_EQ">
    <option value="">----</option>
    <option value="1">TASK</option>
    <option value="2">PROBLEM</option>
  </select>
  <input type="text" name="cond_name_STARTS" value="">
  <input type="submit" name="search" value="絞り込み" />
</form>
```

一覧表示(絞り込み有り)

■画面イメージ

TASK	資料を作成する	05月18日 06時24分	削除
TASK	デモの準備	05月18日 06時25分	削除
PROBLEM	時間がない	05月18日 06時29分	削除

■JSP

index.jsp

```
<table border="1">
  <c:forEach var="todo" varStatus="s" items="${todoList}">
    <tr style="background-color:${s.index % 2 == 0 ? 'pink' : 'yellow'}">
      <td>${f:h(todo.category.name)}</td>
      <td>${f:h(todo.name)}</td>
      <td><fmt:formatDate value="${todo.createDate}"
            pattern="MM月dd日 hh時mm分"/></td>
      <td><a href="delete/${f:u(todo.id)}">削除</a></td>
    </tr>
  </c:forEach>
</table>
```

一覧表示(絞り込み有り)

■アクション

todoAction.java

```
public JdbcManager jdbcManager;
public List<Todo> todoList;

public Integer cond_categoryId_EQ;
public String cond_name_STARTS;

@Execute(validator = false)
public String index() {
    ...

    todoList = jdbcManager.from(Todo.class).leftOuterJoin("category")
            .where(conditions())
            .orderBy("createDate")
            .getResultList();

    return "index.jsp";
}

private BeanMap conditions() {
    return Beans.createAndCopy(BeanMap.class, this)
            .prefix("cond_").execute();
}
```

一覧表示(絞り込み有り)

■ 生成されるHTML

```
<table border="1">
    <tr style="background-color:pink">
      <td>TASK</td>
      <td>資料を作成する</td>
      <td>05月18日 06時24分</td>
      <td><a href="delete/21">削除</a></td>
    </tr>

    <tr style="background-color:yellow">
      <td>TASK</td>
      <td>デモの準備</td>
      <td>05月18日 06時25分</td>
      <td><a href="delete/22">削除</a></td>
    </tr>

    <tr style="background-color:pink">
      <td>PROBLEM</td>
      <td>時間がない</td>
      <td>05月18日 06時29分</td>
      <td><a href="delete/25">削除</a></td>
    </tr>
</table>
```

削除用リンクの作成 & 削除処理

■画面イメージ

TASK	資料を作成する	05月18日 06時24分	削除
TASK	デモの準備	05月18日 06時25分	削除
PROBLEM	時間がない	05月18日 06時29分	削除

■JSP

–削除用リンクの作成

index.jsp

```
<c:forEach var="todo" varStatus="s" items="${todoList}">
  ...
  <td><a href="delete/${f:u(todo.id)}">削除</a></td>
  ...
</c:forEach>
```

削除用リンクの作成 & 削除処理

生成されるHTML

```
...
<td><a href="delete/21">削除</a></td>
...
<td><a href="delete/22">削除</a></td>
...
<td><a href="delete/23">削除</a></td>
...
```

削除用リンクの作成 & 削除処理

■アクション

– 削除処理

todoAction.java

```
@Execute(input = "index", urlPattern = "delete/{id}")
public String delete() {
    Todo todo = new Todo();
    todo.id = id;
    jdbcManager.delete(todo).execute();

    return "/todo?redirect=true";
}
```

Appendix
～ その2 ～

- 開発者のブログから読み解く設計思想
  - SAStrutsとS2JDBCの開発の背景
  - SAStrutsの特徴
  - SAStrutsの基本方針
  - Strutsをなめんな
  - Seasar2のロードマップについて
  - 薄いフレームワーク構想

SAStrutsとS2JDBCの開発の背景

Appendix

大規模プロジェクトをささえるには、
どういうフレームワークが必要なのか、
この半年以上ずっと考えてきて、
その答えが、
Super Agile Strutsと
新O/R Mapper(S2JDBC)です。

2007-10-17 - ひがやすを blog

「みんなが理解している
枯れた技術のだめな20%の部分を
改良する」
という発想で作られているのが、
S2JDBCとSAStrutsです。

2007-12-31 - ひがやすを blog

SAStrutsの特徴

Appendix

SAStrutsの特徴のひとつは、ドキュメントの充実。
いろんなサイトを見る必要はなく、
SAStrutsのサイトのドキュメントを一通り読めば、
開発できるようになっています。

また、Seasar2(DIやAOP)の知識も不要です。
Strutsの知識があれば、理解できるように書いています。

Railsが好きなんだけど、パフォーマンスに不安があるという方は、
ぜひSAStrutsを検討してみてください。

Railsのような生産性で、Javaのパフォーマンスを得ることができます。

from Java to Rubyは過去(Struts, Spring, Hibernate時代)の話だということも
わかってもらえるでしょう。

2008-01-08 - ひがやすを blog

SAStrutsの基本方針

Appendix

SAStrutsの基本方針は下記のとおりです。

- フレームワークは押し付けがましくなく、
  開発者が自由にコードを書ける余地をできるだけたくさん残すこと。

- ただし、ルーチンワークは徹底的に省略できるようにすること。

フレームワークでがちがちに固めると
弊害のほうが多いというのが最近の考え。

プログラマをもっと解放しようよ。

逆にがちがちに固めることも、もちろん許容しているので、
がちがちが必要な場合は、上にフレームワークをかぶせればいいんです。

2008-01-10 - ひがやすを blog

Strutsをなめんな

Appendix

Webフレームワークのやってることを超簡単に説明すると次のようになります。

* リクエストが飛んできたときに、URLに関連付けられているコントローラオブジェクトを見つける。
* リクエストのパラメータを何らかのオブジェクトにつめる。
* 入力値のバリデーションを行なう。
    - NGなら特定のURLに遷移する。
* コントローラオブジェクトにリクエストのパラメータをつめたオブジェクトを設定する。
* コントローラの特定のメソッドを呼び出す。
* 特定のURLに遷移する。

Strutsは、上記のことを淡々とやってくれる薄いフレームワークなのです。Strutsのだめなところは、上記のことをすべて設定ファイルに書かなければいけないことで、それ以外は、無駄なことはほとんどしていない見通しの良いフレームワークです。

Strutsは、もっと見直されていい。設定ファイルを書く必要をなくして、HOT deployをサポートすることで、弱点は解消され、非常に使いやすいフレームワークに生まれ変わります。そういう発想で作られたのが、SAStrutsです。

2008-02-14 - ひがやすを blog

Seasar2のロードマップについて

Appendix

Seasar2は、2.5でやろうとしていたS2PersistenceとS2Presentationを
S2JDBC、SAStrutsという形で、2.4ベースで実装してしまったため、
2.5を出すことはおそらくないでしょう。

また、コンテナに手が入ることももうないと思います。
やるべきことはやったと思うので。

S2JDBCで、データベースの定義からEntityを自動生成する機能と、
Entityからデータベースを再構築する機能を今後追加予定ですが、
それで、機能追加の予定は終わりです。
Seasar2.4は、安定バージョンとして、
ずっと使い続けられることになると思います。

Seasar2.4に対する追加要望があれば、もちろん検討します。
ただし、大きな変更や追加はもうないでしょう。

2008-01-29 - ひがやすを blog

薄いフレームワーク構想

Appendix

Seamが成功すると、フレームワークの肥大化には、
しばらく歯止めがかからなくなるでしょうね。
機能豊富であることが、勝ち残るための最も重要な要因だと
みんなが認めたということなので。

私の個人的な意見としては、**薄いフレームワークのほうが
世の中に好まれるんじゃないか**と思っています。
Strutsプラスアルファで、学習コストが低く、
ソースコードもさらっと読めるフレームワーク。

開発者全員が、フレームワークのことを把握していて、
技術的に見通しがいいフレームワーク。

この薄いフレームワーク構想を実現するために作ったのが、
SAStruts, S2JDBCです。

2008-02-13 - ひがやすを blog

May the SAStruts be with you.

(SAStrutsと共に！)

ご清聴
ありがとう
ございました