

Open
The Next Generation

Eclipseで作るJSF / Spring / Hibernate プラグイン勉強会

オープンソースソフトウェア推進部

堅田 淳也 田中 雅俊 川島 徹

2005/6/17

NTTコムウェア株式会社

Agenda

- 13:30 – 13:45 Eclipse Japan Working Groupについて
NTTコムウェア 玉久保
- 13:45 – 16:00 サンプルアプリケーション概説
Integration層, Business層の開発
NTTコムウェア 田中 雅俊
- 16:15 – 17:00 Presentation層の開発
NTTコムウェア 川島 徹
堅田 淳也

サンプルアプリケーション概説

Agenda ~ Spring × Hibernate ~

- ✓ サンプルアプリケーションの紹介
- ✓ アーキテクチャ
- ✓ 開発方針

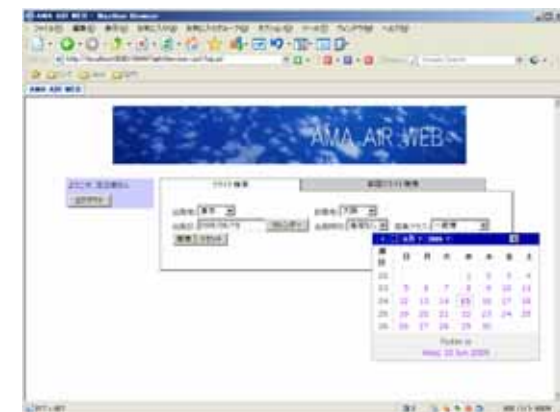
サンプルアプリケーションの紹介

✓ 概要

- ✓ AMA (All Makuhari Airline)
- ✓ 某航空会社のサイトを模倣して作成した、航空機チケット予約アプリケーション (Webアプリケーション)

✓ 機能

- ✓ フライトの検索
 - ✓ 出発日, 出発地, 目的地, 出発時間 (option) からフライトを検索する。
- ✓ 前回搭乗便の検索
 - ✓ 以前に搭乗したフライトを出発日から3回分まで検索する。
- ✓ フライトの予約
 - ✓ 検索したフライトを予約する。
- ✓ (ログイン)
 - ✓ 会員はeカードを所有している
 - ✓ eカード番号とパスワードでログインする



提供するアプリケーションについて

✓ Struts版

- ✓ Struts + Spring + Hibernate
- ✓ Presentation層をStrutsで構築
- ✓ オリジナル

✓ JSF版

- ✓ JSF + Spring + Hibernate
- ✓ Presentation層をJSFで構築
- ✓ Struts版のページフローや画面構成は変更しない

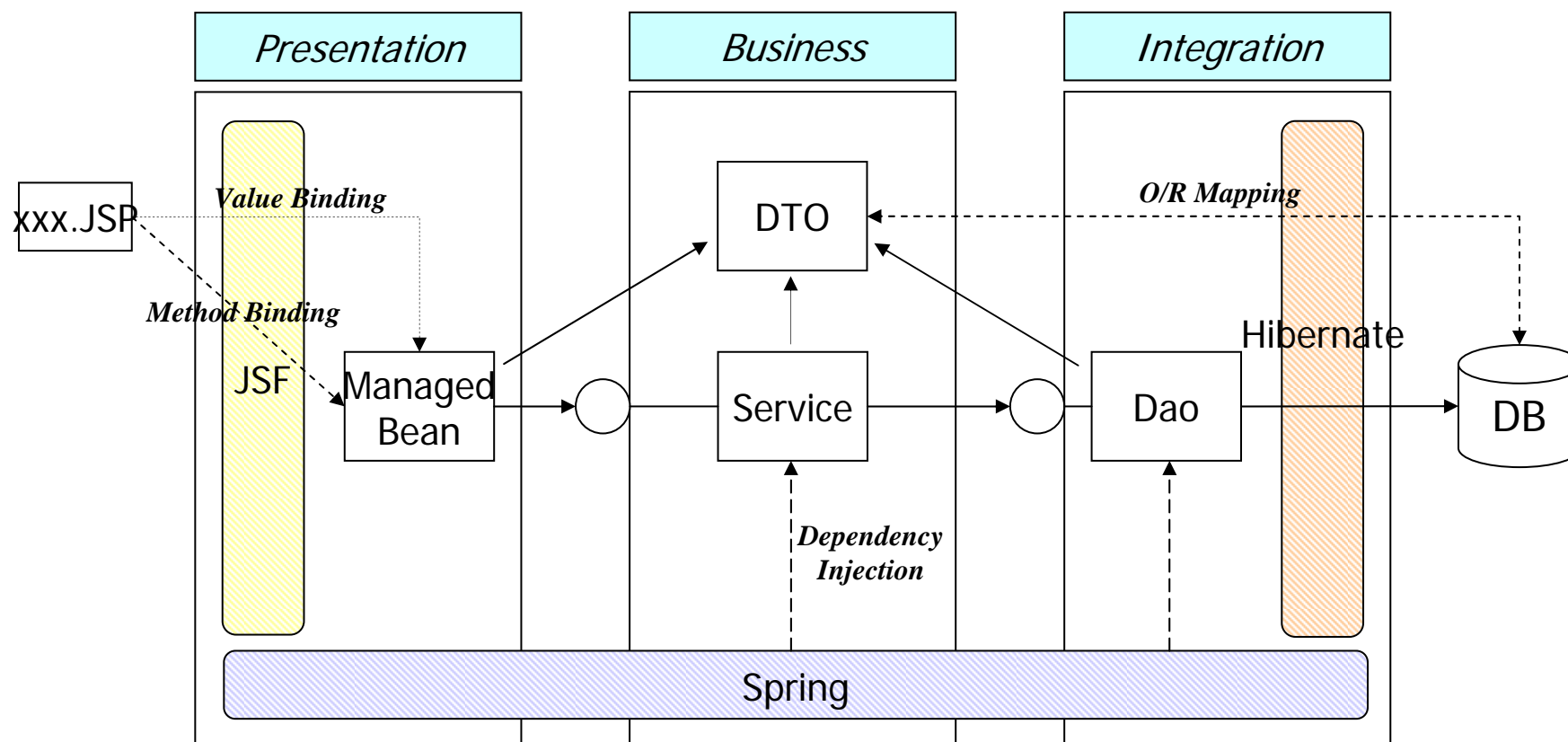
✓ xJSF版

- ✓ JSF + Spring + Hibernate
- ✓ Presentation層をJSFで構築
- ✓ MyFacesの拡張コンポーネントを使用(カレンダー, タブ, スタック etc...)



アーキテクチャ

- ✓ テーブルとDao, テーブルとDTOは1:1
- ✓ ServiceクラスとDaoクラスをSpringで管理
- ✓ トランザクションはB層のインターフェースから設定



Business層, Integration層の開発

Agenda

Open
The Next Generation

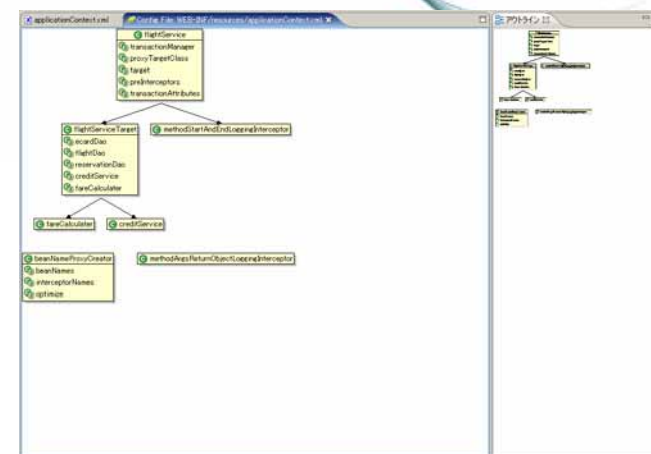
- ✓ Spring Framework
- ✓ Hibernate
- ✓ Demo Scenario

Spring Framework

- ✓ Dependency Injection (依存性の注入) によるJ2EEフレームワーク
 - ✓ コンポーネントがコンテナから依存性を取得するのではなく...
 - ✓ コンテナがコンポーネントに依存性を注入する。
- ✓ Spring Frameworkのメリット
 - ✓ コードがコンテナに依存しない
 - ✓ インタフェース指向プログラミングの強制
 - ✓ コンポーネント単位での機能拡張や追加が可能
 - ✓ テストが容易になる
- ✓ 問題点
 - ✓ 依存性定義ファイル(applicationContext.xml)の作成が面倒
 - ✓ 依存性定義ファイルが誤っていた場合、デバッグが困難
 - ✓ デバッグ時のトレースが困難

Spring IDE

- ✓ Overview
 - ✓ Bean定義ファイルの文法チェック
 - ✓ Bean間の関連グラフの表示
- ✓ Plug-in
 - ✓ Spring IDE version 1.2 (2005/6/2現在)
 - ✓ Graphic Editing Framework (GEF) に依存
 - ✓ Apache License, Version 2.0
 - ✓ eclipse 3.0.x, eclipse 3.1.x
- ✓ Install
 - ✓ [ヘルプ] - [ソフトウェア更新] - [更新とインストール]
 - ✓ <http://springide.org/updatesite/> (eclipse 3.0.x)
- ✓ Uninstall
 - ✓ [ヘルプ] - [ソフトウェア更新] - [構成の管理]



Hibernate

- ✓ O/Rマッピングフレームワーク
 - ✓ オブジェクト指向とRDBの設計思想の違いを自動的にマッピング
- ✓ Hibernateのメリット
 - ✓ 煩雑なO/Rマッピング作業を開発者から隠蔽。開発者はオブジェクトのみを操作すればよい。
 - ✓ ハイパフォーマンス
 - ✓ 遅延ローディング, キャッシュアーキテクチャ
 - ✓ 柔軟性
 - ✓ HQL (Hibernate Query Language), Native SQL
 - ✓ JBossグループによるサポート
- ✓ 問題点
 - ✓ マッピングファイルの作成が面倒
 - ✓ マッピングファイルが間違っているとデバッグが困難
 - ✓ 定型的なDTOやDaoの作成が面倒

Hibernate Synchronizer

- ✓ Overview
 - ✓ 既存のテーブルからHibernateアプリケーションの開発を行うためのツール
 - ✓ テーブルからマッピングファイルやDTO, Daoを自動生成
 - ✓ テーブルからユーザ定義ファイルを自動生成
- ✓ Plug-in
 - ✓ Hibernate Synchronizer Version 2.3.1 (2005/6/2現在)
 - ✓ GNU General Public License (GPL)
- ✓ Install
 - ✓ [ヘルプ] - [ソフトウェア更新] - [更新とインストール]
 - ✓ <http://www.binamics.com/hibernatesync> (eclipse 3.0.x)
 - ✓ <http://www.binamics.com/hibernatesync/eclipse2.1> (eclipse 2.1.x)
- ✓ Uninstall
 - ✓ [ヘルプ] - [ソフトウェア更新] - [構成の管理]

フライト予約ユースケースの開発

前提条件:

- ✓ テーブルは既存のテーブルを利用
- ✓ Integration層のインタフェース, DTOは設計済み
- ✓ Serviceクラス, DTO, Daoのコード自体は設計モデルから実装可能

Step1: DTOの開発

Step2: フライト予約Dao (ReservationDao) の開発

Step3: フライト予約Serviceクラス (FlightService) の開発

JSF & FacesIDE

Agenda

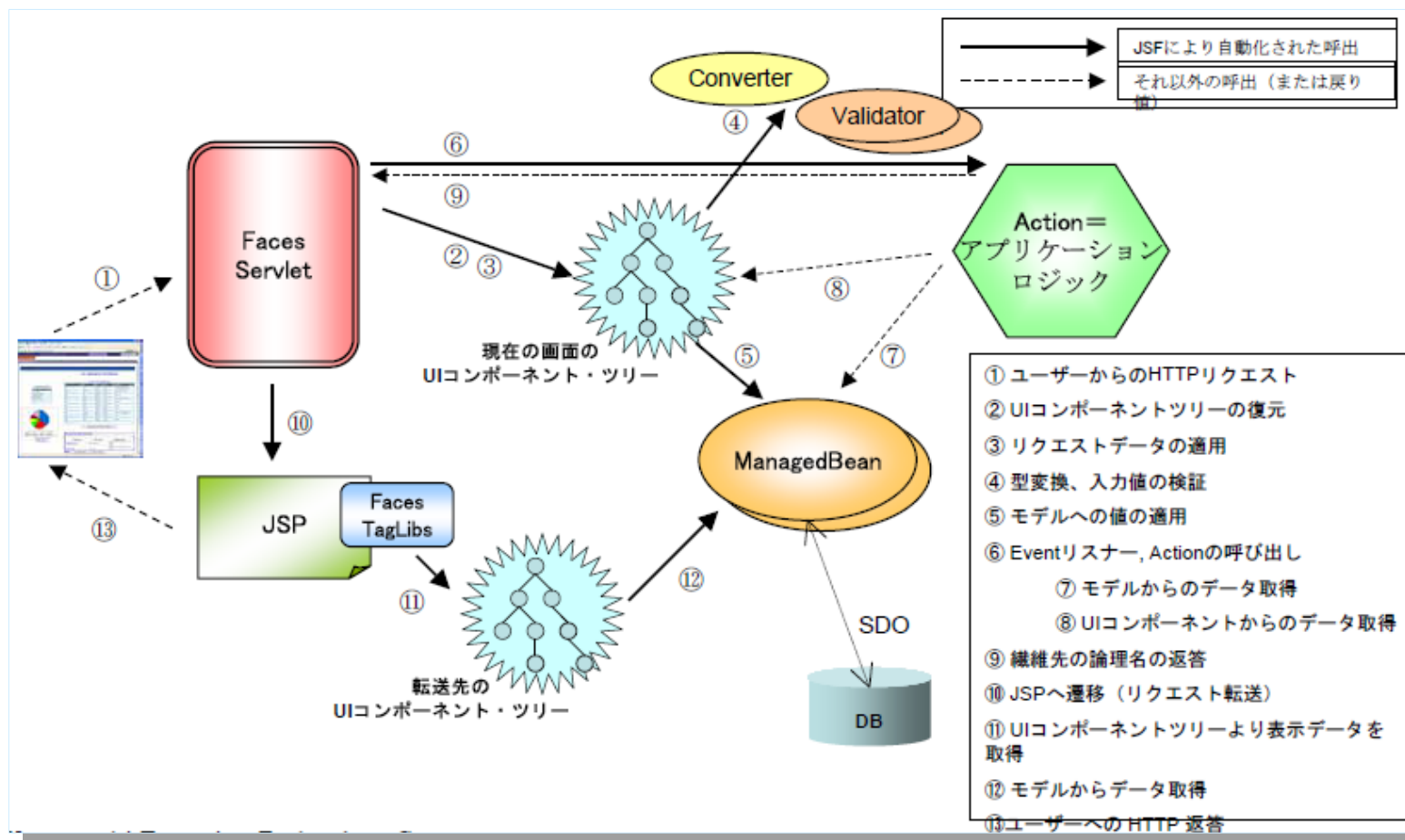
- ✓ JSF
- ✓ UI Evolution
- ✓ Motivation
- ✓ FacesIDE
- ✓ Sample Scenario
- ✓ AMA Metrics
- ✓ Advanced Topic: Custom Component
- ✓ References

JSF

- ✓ **標準サーバサイド GUI コンポーネントフレームワーク**
 - ✓ 基本は JSP-Servlet Model .. を簡単にする仕組み
- ✓ **たかが Component (Spec.)**
 - ✓ タグを書くと画面ができる
 - ✓ Level
 - ✓ 標準タグの利用
 - ✓ 拡張タグ (RI, IDE提供) の利用
 - ✓ Custom component の作成 & 利用
- ✓ **されど Framework**
 - ✓ ページフロー定義
 - ✓ サーバサイド Java Bean とのシームレスな連携 (Managed Bean)
 - ✓ Value Bind, Action, Event
 - ✓ 拡張ポイント
 - ✓ Converter, Validator, ...
- ✓ **なんでも Implementation**
 - ✓ ブラウザ上で実行できるものであればなんでも実装に使える
 - ✓ JavaScript, DHTML, CSS, Flash/Flex, Ajax

JSF Architecture

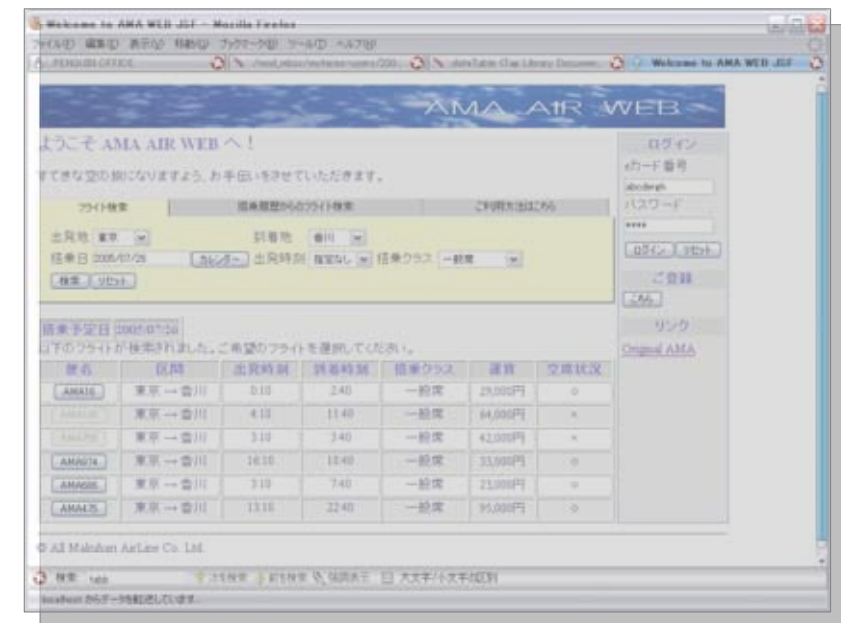
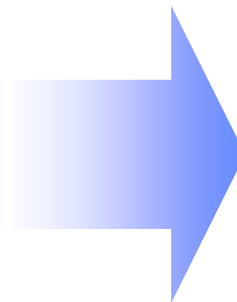
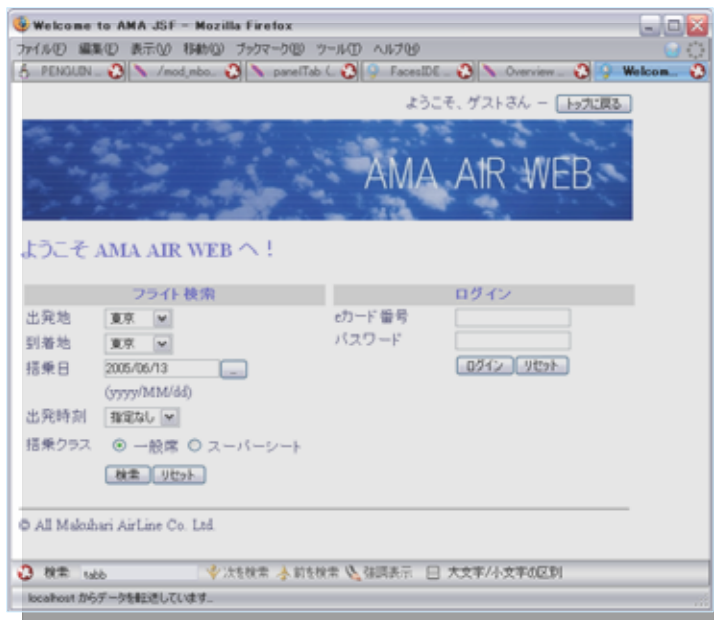
- ✓ UI コンポーネントツリーを構成
- ✓ サーバサイドで処理



出展：米持先進技塾 © 2004 Copyright IBM Corp.

UI Evolution

- ✓ UI 再考: UI はシステムの顔
 - ✓ 使い手のためのものであって
 - ✓ 作り手のためのものであってはならない
- ✓ Web システムとして敬遠してきた、もしくは苦勞してきた UI を比較的容易に実現できる(ことがある)
 - ✓ コンポーネントの充実がキー
 - ✓ Rich User Interface
 - ✓ ページ遷移 状態遷移へ
- ✓ 例えば...




Motivation

- ✓ Presentation 層開発
 - ✓ JSP、faces-config.xml、Managed Bean

- ✓ JSF を使って開発しようとする...
 - ✓ フロー定義による画面遷移は概念的にはわかりやすいが...
 - ✓ 設定が面倒
 - ✓ フローがあっているか確認できない
 - ✓ Managed Bean は扱いやすいが...
 - ✓ どんな Bean があるのか、一覧性が悪い
 - ✓ 登録している Bean と実際のクラス(.java ファイル)の対応がとれない
 - ✓ フローや Managed Bean を定義しているコンフィグファイル(XML)に間違いがあると、デバッグが大変(みつからない)
 - ✓ タグベース開発でさくさくできそうなのだが...
 - ✓ そもそもやりたいことがどのタグで実現できるのかわからない
 - ✓ タグごとに属性もあるので覚えきれない
 - ✓ タグの対応も取れているのかどうか
 - ✓ JSP 書いては見たけど見栄えが意図したとおりになっているかどうか確認したい

FacesIDE

FacesIDE

- ✓ Overview
 - ✓ JSF アプリケーションを作成するための Eclipse Plug-in
 - ✓ 画面遷移、マネージドビーンのビジュアル編集
 - ✓ JSP ファイルの編集、コードアシスト、プレビュー
 - ✓ Plug-in
 - ✓ FacesIDE v 0.1.5
 - ✓ RI : MyFaces
 - ✓ CPL 1.0 準拠
- POWERED BY:
MyFaces 
- ✓ Software requirements
 - ✓ Eclipse SDK 3.0.x (JDT 含)
 - ✓ GEF
 - ✓ EclipseHTMLEditor
 - ✓ Sysdeo Tomcat Plugin (推奨)
 - ✓ Install & Uninstall
 - ✓ Install: ダウンロードした zip を解凍し、Eclipse Plug-in フォルダにコピー
 - ✓ Uninstall: Plug-in フォルダから以下を削除 (x.x.x はバージョン番号)
 - ✓ tk.eclipse.plugin.htmleditor_x.x.x
 - ✓ tk.eclipse.plugin.jsf_x.x.x

FacesIDE View

- ✓ Perspective
 - ✓ 設定 > HTML/JSP/XMLエディタプラグイン
 - ✓ パースpekティブのカスタマイズ
 - ✓ プロジェクト > プロパティ
- ✓ Editors
 - ✓ HTML/JSP/XML Editor
 - ✓ faces-config.xml Editor
 - ✓ Flow
 - ✓ Managed Bean
 - ✓ Source (XML)

The screenshot illustrates the FacesIDE perspective in Eclipse. The main editor shows a navigation diagram for 'login.jsp' with pages like 'top', 'search', and 'select'. A 'Managed Beans' dialog is open, showing details for a bean named 'flightList' with class 'jp.contcom.ama.view.helper.FlightList' and scope 'session'. A 'faces-config.xml' editor is also open, showing XML code for navigation rules and managed beans.

```

1 <?xml version="1.0"?>
2 <!DOCTYPE faces-config PUBLIC "-//Sun Microsystems, Inc.//DTD JavaServer Faces
3 <faces-config>
4
5 <navigation-rule>
6   <from-view-id>/pages/top.jsp</from-view-id>
7   <navigation-case>
8     <from-outcome>search</from-outcome>
9     <to-view-id>/pages/searchresult.jsp</to-view-id>
10  </navigation-case>
11 </nav
12 </nav>
13 <description
14 </navi
15 << from-view-id
16 << icon
  
```

Sample Scenario

✓ P層開発の前提

- ✓ 画面遷移、JSP 画面そのものは画面仕様書、画面遷移図などから実装可能
- ✓ Value Binding, Method Binding の対象となる Business 層 IF や DTO は設計済
 - ✓ 実装済みである必要はない
 - ✓ P 層の試験は Mock Object を利用

```

<<Interface>>
FlightService

+ searchFlight(searchCondition : SearchCondition) : List
+ login(eCardNum : String, password : String) : ECard
+ reserveTicket(eCard : ECard, departureDate : Date, boardingClass : String, flightName : String) : void
+ searchFlightFromBoardingRecord(startDate : Date, eCard : ECard) : List
  
```

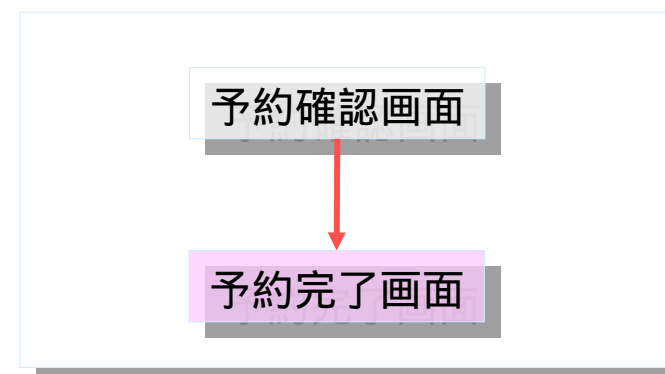
✓ 予約完了画面を追加してみましょう

- ✓ 予約確認画面で予約ボタンを押して予約が成功したならば、お客様に感謝の気持ちを伝えます

✓ Try It !

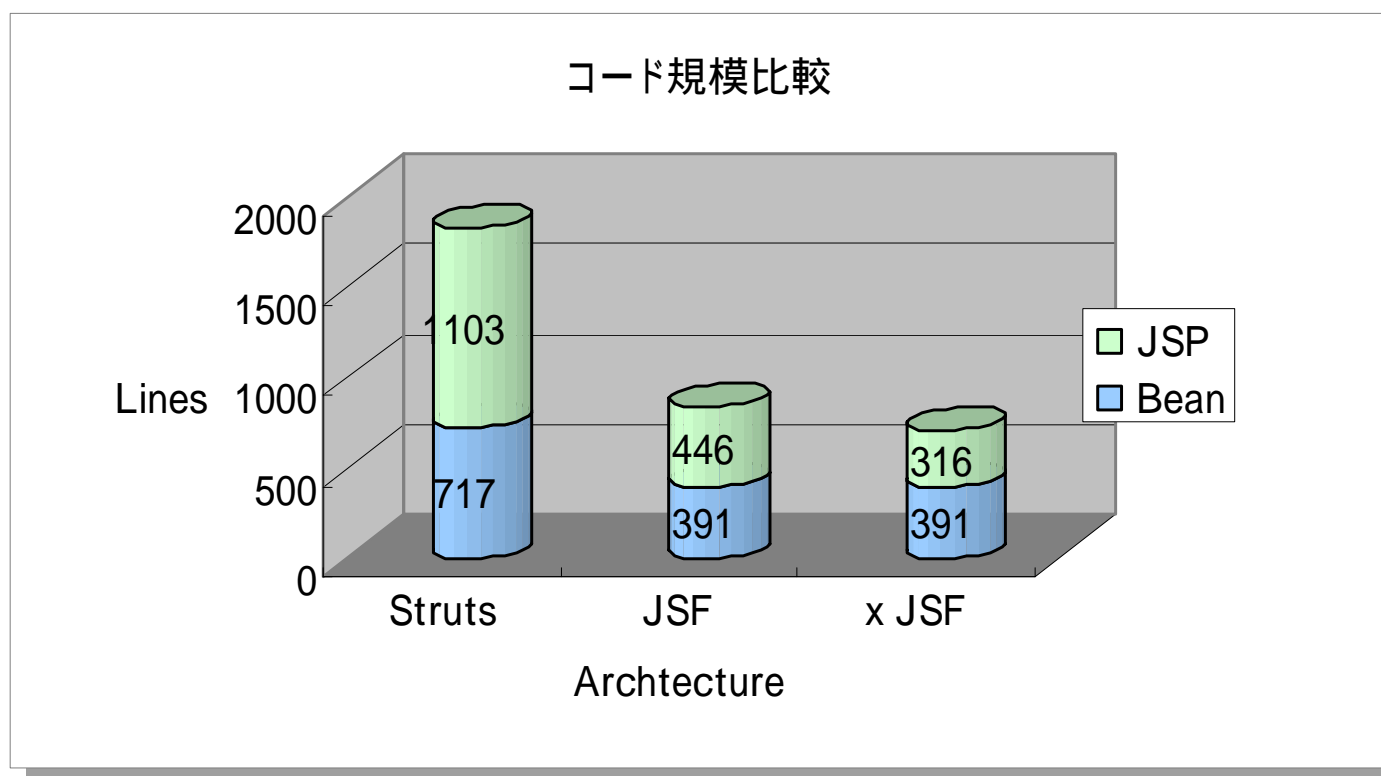
- ✓ Creating Navigation Rules
- ✓ Creating & Configuring Managed Beans
- ✓ Creating JSP Pages

DEMO



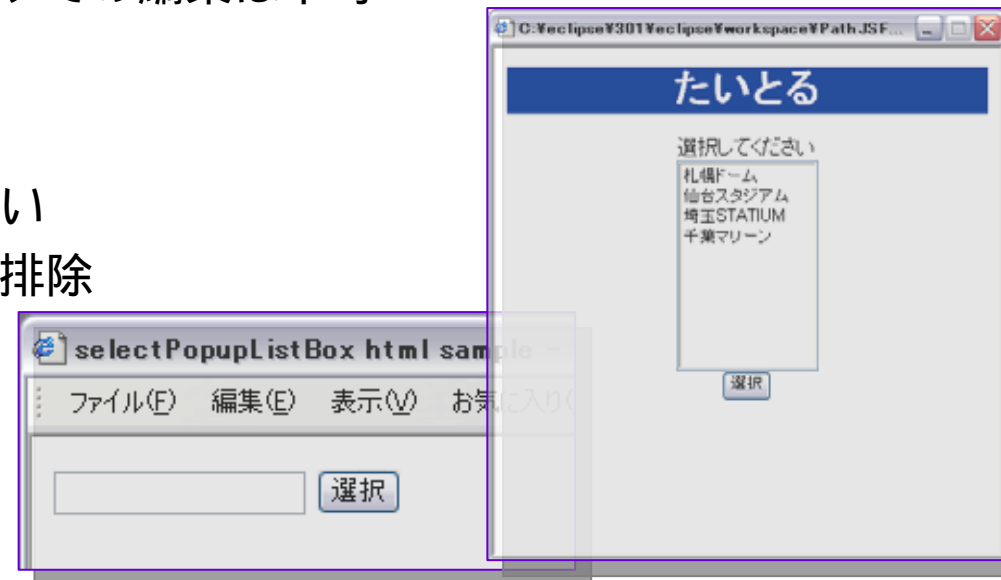
AMA Metrics

- ✓ AMA のケース
 - ✓ Struts: オリジナル
 - ✓ JSF: ページフローの基本はおなじ
 - ✓ x JSF: MyFaces 拡張コンポーネントを利用 (カレンダー、タブ、スタック、…)



Advanced Topic: Custom Component

- ✓ selectPopupListBox
 - ✓ システム全般にわたって同じ画面仕様を定義
 - ✓ DB のマスタデータを選択肢とし、ポップアップ画面のリストから選択
 - ✓ 選択結果を表示するが、text エリアでの編集は不可
 - ✓ Implementation: JavaScript
 - ✓ 効果
 - ✓ 選択肢ごとに JSP を作らなくてよい
 - ✓ JavaScript のコピー & ペーストの排除
 - ✓ UI 仕様の確立



```
<pt:selectPopupListBox id="name" value="#{bean}"
    title="選択してください" width="400" height="300">
    <f:selectItems value="#{selectable.list}" />
</pt:selectPopupListBox>
```

References

- ✓ FacesIDE
 - ✓ <http://amateras.sourceforge.jp/cgi-bin/fswiki/wiki.cgi?page=FacesIDE>
- ✓ マニュアル(英語)
 - ✓ <http://amateras.sourceforge.jp/docs/FacesIDE/>
- ✓ java.net
 - ✓ <https://amateraside.dev.java.net/>
- ✓ EclipseHTMLEditor
 - ✓ http://amateras.sourceforge.jp/cgi-bin/fswiki_en/wiki.cgi?page=EclipseHTMLEditor
- ✓ Apache MyFaces
 - ✓ <http://myfaces.apache.org/>
- ✓ JSF 本家
 - ✓ <http://java.sun.com/j2ee/javaserverfaces/index.jsp>
- ✓ @IT JSF を理解する
 - ✓ <http://www.atmarkit.co.jp/fjava/special/jsf01/jsf01.html>