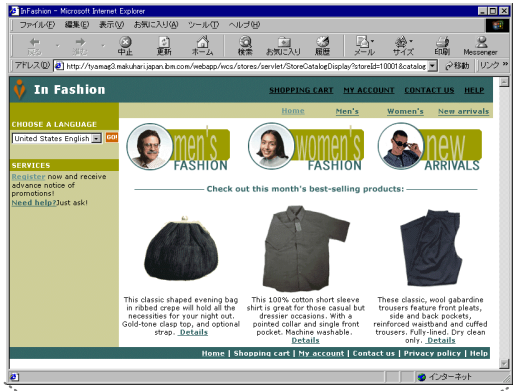
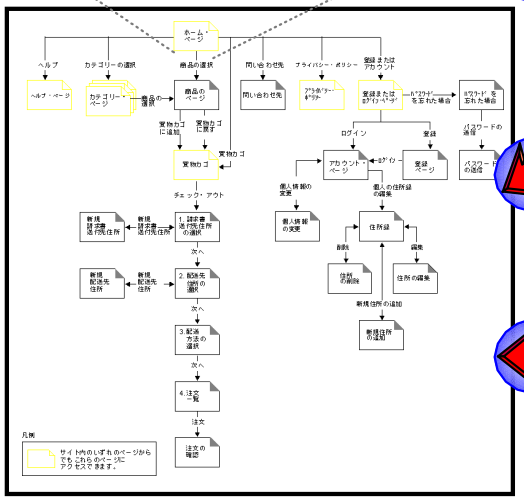
# カストマイズ

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# カストマイズのポイント



カストマイズポイント

- デザインを変えたい
  - ヘッダのデザインを変えたい
  - 表示色を変えたい
- ラベルを変えたい
  - Welcomeメッセージを変えたい
  - 『登録』を『アカウント』にしたい
- 表示項目を変えたい
  - 登録ページの登録項目を変えたい
  - 商品の詳細ページに関連商品を表示したい
- フローを変えたい
  - オーダーフローを短くしたい
  - 登録ページに強制的に行かせたい
- 新規の機能を追加したい
  - 配送日指定をしたい
  - オーダー確認画面を付けたい
- 現行機能を変えたい
  - オーダー時に追加テーブルにフラグを立てたい
  - 登録画面で郵便番号検索をやりたい

Presentation Logic
- HTML
- CSS
- Java Script
- JSP
- Property File

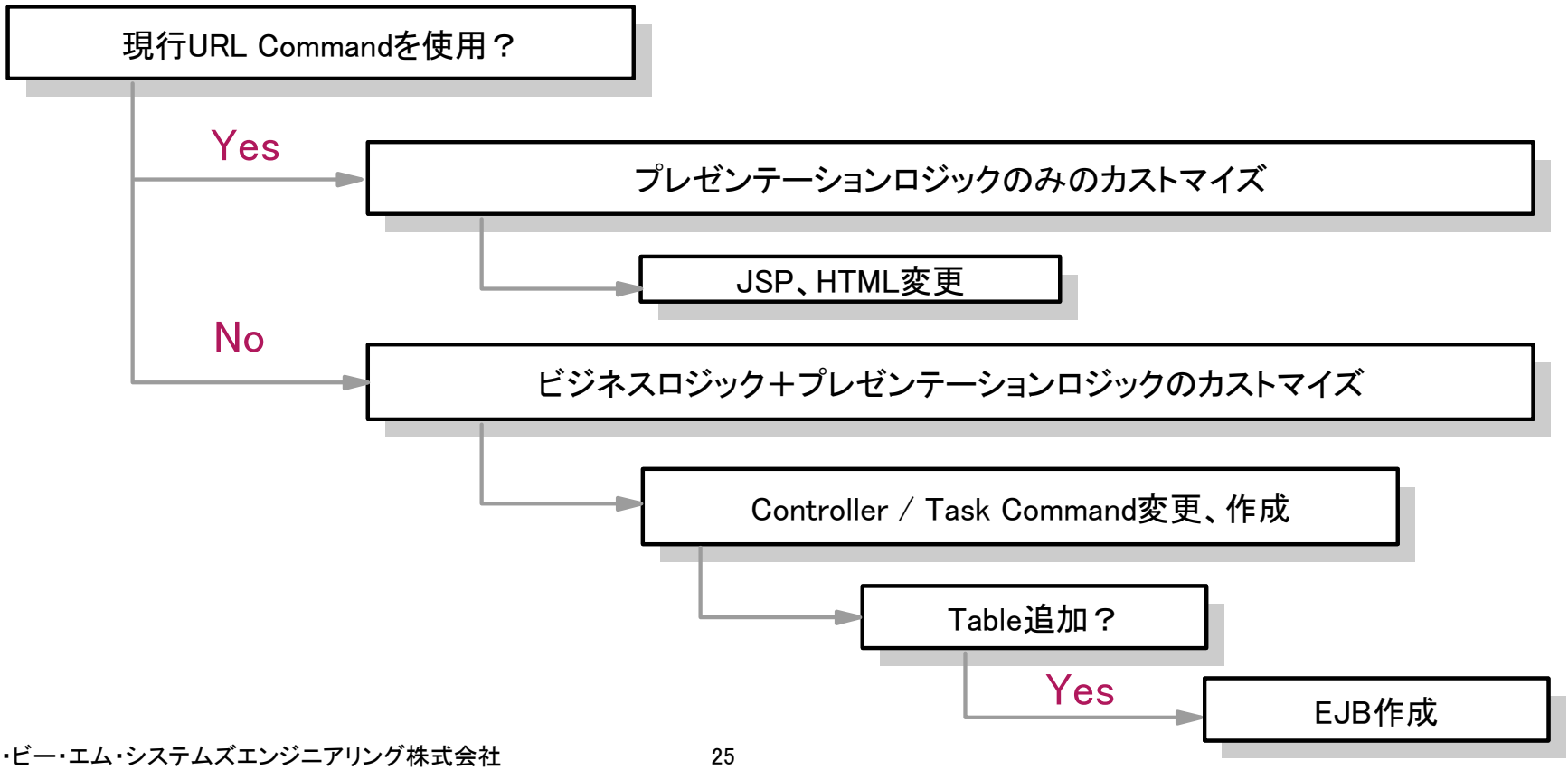
Business Logic
- Database Record
- URL Command
- Controller / Task Command
- AccessBean/DataBean
- EJB (Entity Bean)
- Database Table

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# カストマイズのポイント

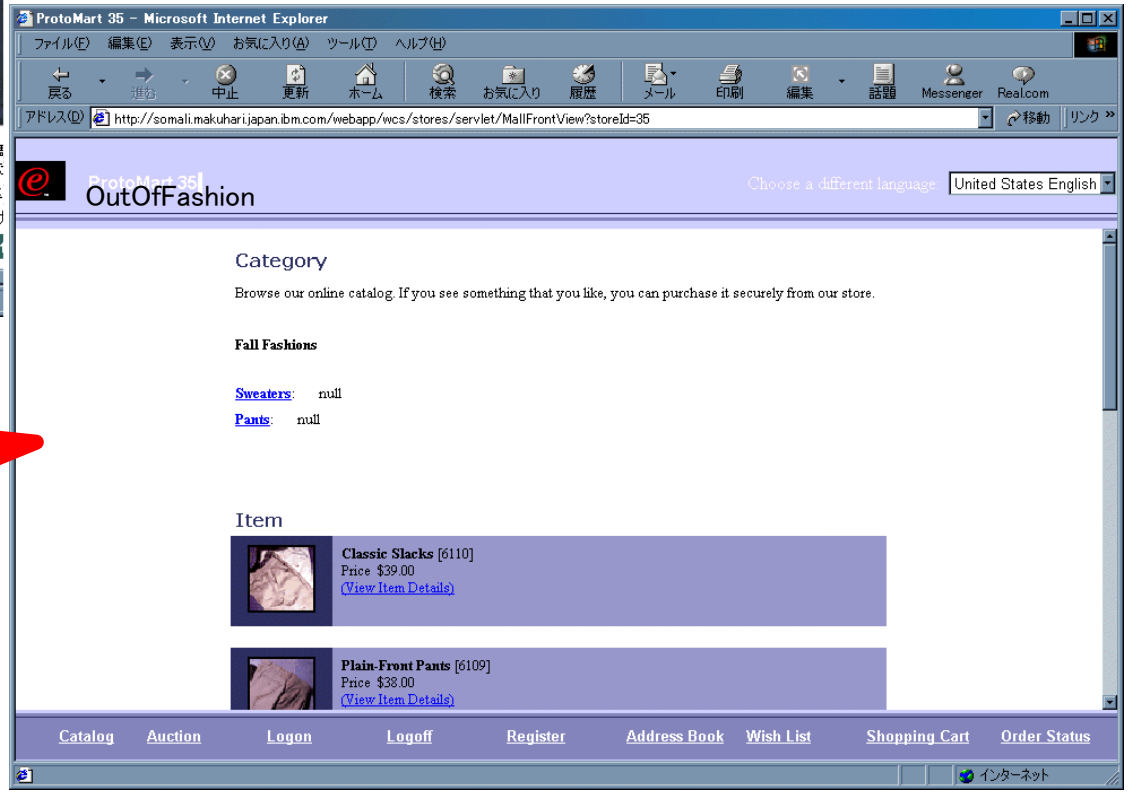
## 考慮点

- ポイントは多種存在するが大別すると2点、プレゼンテーションロジックとビジネスロジック
  - プレゼンテーションロジック
    - JSPに代表される表示部分であり、直接ユーザー(ショッパー)の目に触れる部分。
  - ビジネスロジック
    - 操作を行うことで実行されるアクションの変更。ControllerCommandsに代表されるJavaコマンド構造により実現される。
- 何をカストマイズすればよいかは以下を参照




(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# PresentationLogicのみのカストマイズ


(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# JSPのカストマイズ (1)

## ▶使用されているJSPファイルを特定する

- CommandRegistryより特定ページで使用されているJSPファイルを取得
  1. 使用するURL Commandをオンラインヘルプより取得
     - Topics → Reference → Commands → URL Commands
     - 使用コマンドのBehaviorの項で呼び出すViewNameをチェック
     - サンプルストアのJSPに関してはオンラインヘルプに詳細の説明あり
  2. VIEWREG表よりViewNameをキーにpropertiesからJSPファイル名を取得
  3. WAS WebPathから指定JSPファイルを選択
     - JSPファイル内でINCLUDEの形でヘッダーやフッターなどのファイルを呼んでいる場合もある
  4. さらにJSPによっては内部ロジックにより複数JSPへ分岐することもある
  5. 必要に応じ、新たなJSPファイルを作成／コピー
     - その場合VIEWREG表の定義を更新する
- 商品ページ、カテゴリーページなどは別の表にJSPとの関連を保管
  - DISPENTREL（商品）、DISPCGPRREL（カテゴリ）に保管
  - 通常コマンドよりもユーザーに応じた細かなJSPの対応が必要なため

WCS V5.1 GA TECH WAS

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# View Command

## View Commandとは

- クライアント要求に対し、Viewを返す
- 以下の三種類がある
  - Forward View Command
    - JSPなど別コンポーネントに要求を転送
    - ほとんどがこれを使用する
    - VIEWREG.PROPERTIESには'docname = *view_file_name*'の形で記載される
  - Redirect View Command
    - リダイレクトでViewをクライアントに戻す
    - 即ちクライアントから2回のURLリクエストが出ることになる
    - リロードされたとき、リダイレクト先のURLを実行するので間違いの場合ロジックを複数回実行することを防ぐ
    - 2回のリクエストと言うことはトランザクションは分割される
    - VIEWREG.PROPERTIESには'url = *Redirect_URL*'の形で記載される
  - Direct View Command
    - クライアントに直接、応答Viewを返す
    - VIEWREG.PROPERTIESには'textDocument = *Java.io.InputStream object*'（テキスト）もしくは'rawDocument = *Java.io.InputStream object*'（バイナリ）の形で記載される
- View Commandを単体で呼び出すことも可能

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# Command Registry(1) - VIEWREG

- ViewCommandと使用ViewCommand、JSPの関係を記載
  - ViewCommand名はCommandのリファレンスに記載
  - ストア、デバイスごとに指定JSPの変更が可能
    - 直接JSP名をControllerCommandから渡してあげることも可能だが、外出しにすることで変更対応がコードから分離される
  - 登録するためのGUIインターフェースは無い（SQLでINSERT/UPDATE）

Sample Data

VIEWREG

| | |
|---|---|
| VIEWNAME | StoreCatalogDisplayView |
| STOREENT_ID | 10051 |
| PROPERTIES | docname=StoreCatalogDisplay.jsp |
| INTERFACENAME | com.ibm.commerce.command.HttpForwardViewCommand |
| CLASSNAME | com.ibm.commerce.command.HttpForwardViewCommandImpl |
| DEVICEFMT_ID | -1 |
| HTTPS | 0 |

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# JSPのカストマイズ (2)

## JSPファイルを修正する－Overview

- デザイン（HTML）の修正
  - 各ページごとにベースデザインに基づきHTML部分の修正を行う
    - INCLUDEページもあわせて修正
  - 提供Sample（InFashion）はStyleSheet（css）を使用しているのでCSSの変更も含む
- 入力、表示項目の修正
  - サンプルなどを参考に、URL Commandに応じた処理を盛り込む
  - 結果表示画面であれば、必要データのDataBeanを取り込むことで表示項目をカストマイズ
  - 入力画面であれば使用コマンドが受け付けうるパラメーターなどをHTMLに追加
- その他
  - 条件分岐
    - 条件によって呼び出すJSPファイルやURLCommandを変えるといった若干のプログラムロジックがあるのでそれらの修正を行う
  - JavaScriptなどの処理変更
  - Cache機能によりサーバー側に特定ページはキャッシュされる
    - 変更の反映が無い場合はキャッシュを消去する
    - 開発時は構成マネージャーよりキャッシュ機能をOffにしておく
- シンプルな機能JSPで開発、テストを行い、最終的にデザインとJSPコード部分をマージする

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# JSPのカストマイズ (3)

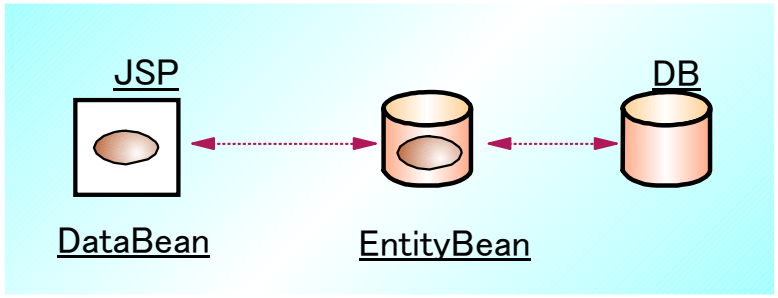
- 主な操作
  - リクエストパラメーターを受領する
    - HTML FORM、URLなどで渡されるパラメーターを読むために
      - HTTPServletRequest.getParameter()
      - HTTPServletRequest.getParameterValues()
  - DataBeanをactivateする
    - DB参照レコードはDataBean経由でなされる
      - com.ibm.commerce.benas.DataBeanManager.activate(DataBean, HTTPServletRequest)
  - DataBeanメソッドを実行し、レコードを取得

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# DataBeanとは

## EntityBeanの単純化

- EntityBeanにアクセスするための複雑な手順を簡便化
- JSPの中から使われる
  - JSPを作成する開発者はEntityを知る必要は無く、DataBeanから取得できるデータに専念できる
- 2種類のDataBean
  - Smart DataBean
    - AssociateされたEntityBeanのうち必要なデータのみを取得
    - Lazy Fetchを使用
    - ほぼ全てのDataBeanはこちらを使用
  - Command DataBean
    - AssociateされたEntityBeanの全データが取得される分パフォーマンスが悪い
    - 下位互換用(?)
- 参照
  - オンラインヘルプ→トピック→参照→DataBeanおよびEntityBean
    - 以下のパッケージがDataBean
    - com.ibm.commerce.<subsystem>.beans



(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# (参考)InFashionトップページJSP

```
<%@ page language="java" %>

<% // All JSPs requires the first 4 packages for getResource.jsp which is used for
multi language support %>
<%@ page import="java.io.*" %>
<%@ page import="java.util.*" %>
<%@ page import="com.ibm.commerce.server.*" %>
<%@ page import="com.ibm.commerce.command.*" %>

<%@ page import="com.ibm.commerce.catalog.beans.*" %>
<%@ page import="com.ibm.commerce.catalog.objects.*" %>
<%@ page import="com.ibm.commerce.beans.*" %>
<%@ page import="com.ibm.commerce.common.beans.*" %>

<%@ include file="getResource.jsp"%>
<% response.setContentType(infashiontext.getString("ENCODESTATEMENT")); %>

<%
String catalogId = request.getParameter("catalogId");
String storeId = request.getParameter("storeId");
String languageId = request.getParameter("langId");
%>
<jsp:useBean id="catalog"
class="com.ibm.commerce.catalog.beans.CatalogDataBean" scope="page">
<%
catalog.setCatalogId(catalogId);
com.ibm.commerce.beans.DataBeanManager.activate(catalog, request);
%>
</jsp:useBean>
<%
CategoryDataBean topCategories[] = catalog.getTopCategories();
CategoryDataBean category;
%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"DTD/xhtml1-transitional.dtd">

<head><title><%=infashiontext.getString("INDEX_TITLE")%></title>
<link rel=stylesheet href="<%=storeDir%>/fashionfair.css" type="text/css">
</head>
```

必要なパッケージをimport(上の4つは必須)

URLパラメーターよりgetParameter()にてパラメーター取得

id="catalog"にてCatalogDataBeanのオブジェクトリファレンスを取得

setCatalogIdにて検索条件を取得し
DataBeanManager.activate

CatalogDataBeanのgetTopCategories()メソッドで指定カタログのトップカテゴリーを取得し、配列に入れ込む

```
CATALOG_ID           IDENTIFIER
-------------------- --------------------
               10001 Mens Fahions
               10001 Womens Fahions
               10001 Whats Hot
               10001 HOMEPAGE_PROMO
```

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# (参考)InFashionトップページJSP

```
<body marginheight="0" marginwidth="0">

<%
String incfile;
incfile = "/" + storeDir + "/header.jsp";
%>
<jsp:include page="<%=incfile%>"/>
<%
incfile = "/" + storeDir + "/sidebar.jsp";
%>
<jsp:include page="<%=incfile%>"/>
<td bgcolor="#FFFFFF" width="600" rowspan="6" valign="top">

<table cellpadding="5" cellspacing="0" width="600" border="0">
<tr>
<%
String promo_category = null;
for (int i = 0; i < topCategories.length; ++i)
{
        category = topCategories[i];
        if (category.getIdentifier().trim().equals("HOMEPAGE_PROMO")) {
                promo_category = category.getCategoryId();
        }
        else {
%>
<td width="207" valign="top" align="middle">
<a
href="CategoryDisplay?catalogId=<%=catalogId%>&storeId=<%=storeId%>&categoryId=<%=category.getCategoryId()%>&langId=<%=languageId%>&top=Y"><img
src="<%=storeDir%>/<%=category.getDescription().getThumbNail()%>" alt="<%=category.getDescription().getShortDescription()%>" width="190"
height="81" border="0"></a><br>
</td>
<%
        }
}
%>
</tr><tr>

<td align="center" valign="bottom" colspan="3">
<img src="<%=storeDir%>/<%=locale.toString()%>/images/bar.gif" alt="<%=infashiontext.getString("IMAGE_ALT1")%>" width="550" height="25">
</td>

</tr><tr>
```

header.jsp、sidebar.jspをinclude

```
CATALOG_ID           IDENTIFIER
-------------------- --------------------
                10001 Mens Fahions
                10001 Womens Fahions
                10001 Whats Hot
                10001 HOMEPAGE_PROMO
```

topCategories[]の配列より、propertyファイルの
HOMEPAGE_PROMOパラメーターとIDENTIFIER名を比較し
一致するものを取得、promo_categoryに入れ込む

一致しないものはトップカテゴリーとして
CategoryDisplayに
リンクを張ったサムネールを表示
category.getDescription().getThumbNail()


(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# (参考)InFashionトップページJSP

```
<%
if (promo_category != null ) {
        CategoryDataBean subCategories[];
        %>
<jsp:useBean id="parentCategory" class="com.ibm.commerce.catalog.beans.CategoryDataBean" scope="page">
<%
parentCategory.setCategoryId(promo_category);
parentCategory.setCatalogId(catalogId);
com.ibm.commerce.beans.DataBeanManager.activate(parentCategory, request);
%>
</jsp:useBean>
        <%
        subCategories= parentCategory.getSubCategories();
        ProductDataBean promo_products[] = parentCategory.getProducts();
        ProductDataBean promo_product;
        for (int k = 0; k < promo_products.length; ++k) {
                promo_product = promo_products[k];
%>
<td align="middle" valign="top" width="200">
<a href="ProductDisplay?catalogId=<%=catalogId%>&storeId=<%= storeId%>&productId=<%=
promo_product.getProductID()%>&langId=<%=languageId%>&parent_category_rn=<%=promo_category%>"><img src="<%=storeDir%>/<%=
promo_product.getDescription().getFullImage() %>" alt="<%= promo_product.getDescription().getShortDescription()%>" width="150" height="150"
border="0"></a><br>
<font class="text"><%= promo_product.getDescription().getLongDescription()%> <a href="ProductDisplay?catalogId=<%=catalogId%>&storeId=<%=
storeId%>&productId=<%= promo_product.getProductID()%>&langId=<%=languageId%>&parent_category_rn=<%=promo_category%>">
<%=infashiontext.getString("DETAILS")%></a></font></td>
<%
        }
}
%>
</td></tr></table>
</td>

</tr></table>
<%
incfile = "/" + storeDir + "/footer.jsp";
%>
<jsp:include page="<%=incfile%>"/>


</body></html>
```

id="parentCategory"にてCategoryDataBeanの
オブジェクトリファレンスを取得、promo_categoryとcatalogIdを
キーにactivate

getProducts()でPROMO_CATEGORY下の商品を取得

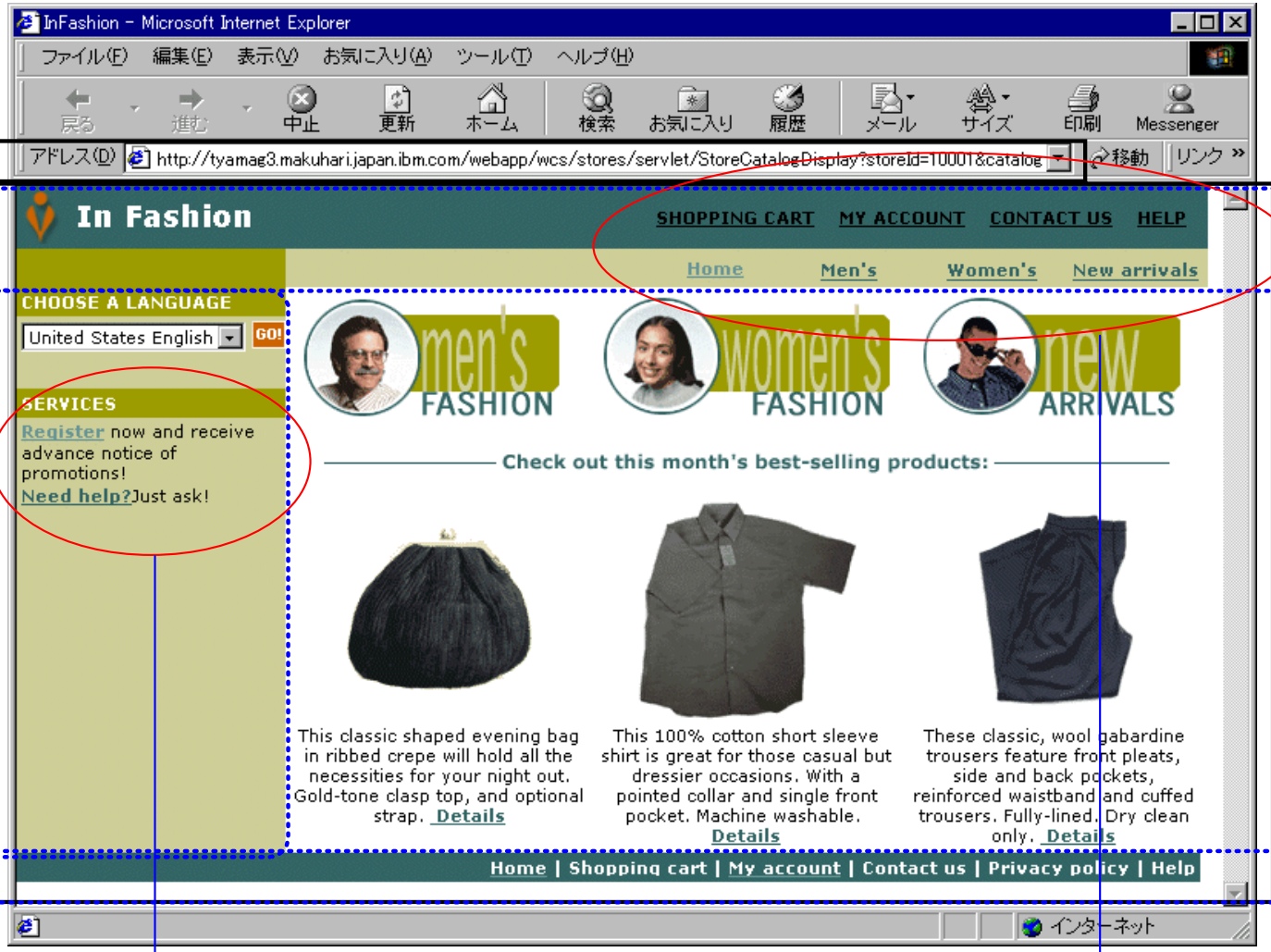
商品のサムネール、詳細説明をそれぞれリンク付きで表示

footer.jspをInclude

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# （参考）InFashionページ構造

http://<hostname>/webapp/wcs/stores/servlet/StoreCatalogDisplay?...




Include : header.jsp

Include : sidebar.jsp

JSP File : StoreCatalogDisplay.jsp

Cascade Style Sheet : fashionfair.css

Property File: infashion_ja_JP.properties

Include : footer.jsp

(c) Copyright 日本アイビーエム システムズ エンジニアリング株式会社

# （参考）InFashionページ構造

## ▶トップページ解析

- URL CommandとしてStoreCatalogDisplayが呼ばれることで、以下のファイルが参照される
- ファイルはデフォルトで以下に配置
  - JSP　　¥WebSphere¥WCS¥stores¥web¥<store_name>¥
  - property file　　¥WebSphere¥WCS¥stores¥properties¥<store_name>¥

### JSP

- StoreCatalogDisplay.jsp
  - メインのJSP、以下の3ファイルをinclude

1. header.jsp　　ヘッダーJSP
2. sidebar.jsp　　サイド・バーJSP
3. footer.jsp　　フッターJSP

### CSS

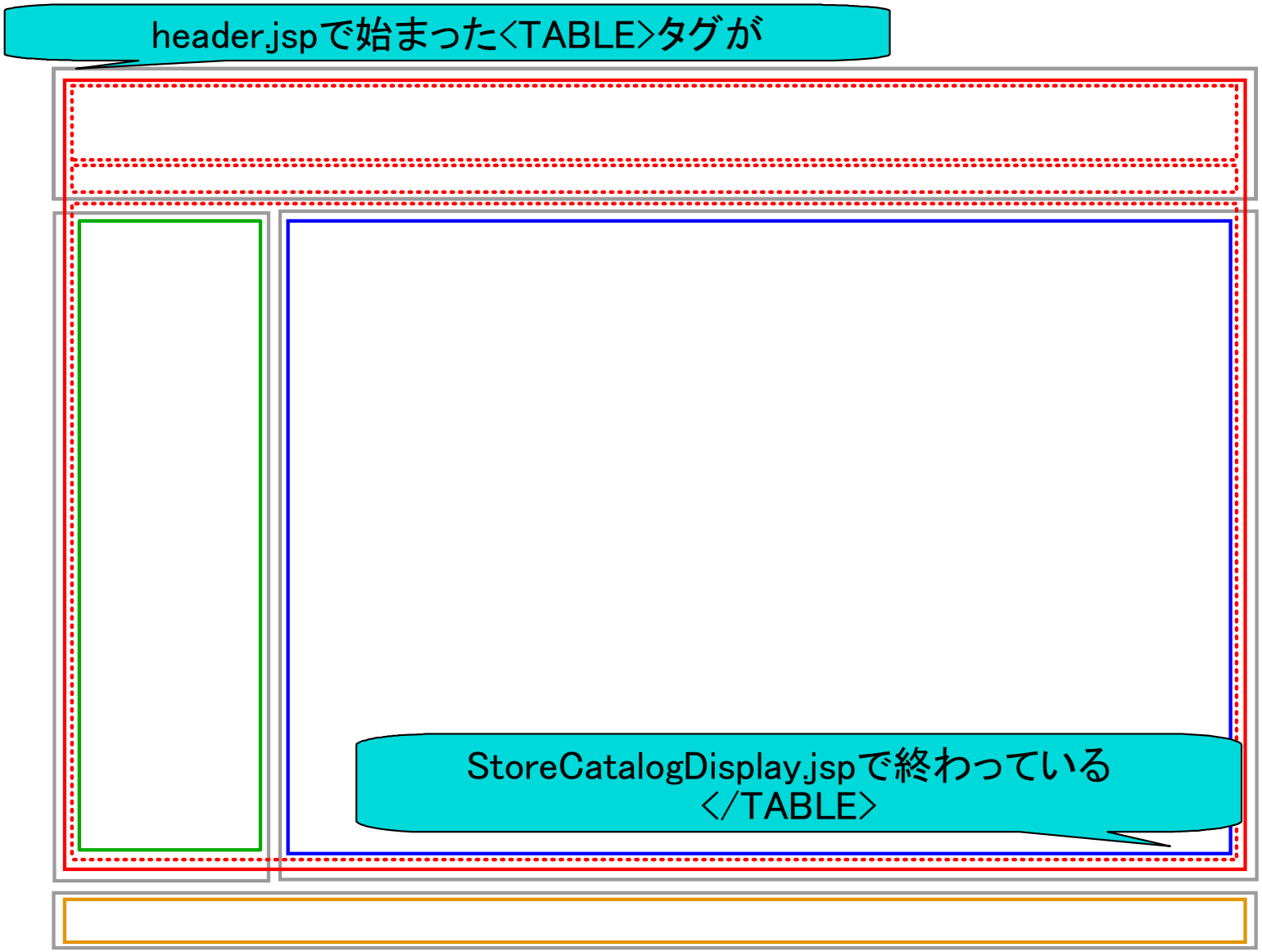
- fashionfair.css
  - カスケードスタイルシートファイルとしてタグの属性情報を保管

### Property File

- infashion_ja_JP.properties
  - 『ショップカート』といったラベル、『ようこそInFashionへ…』といったメッセージ情報などを言語ごとに保管

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# （参考）InFashion <TABLE> タグ構造




(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# Cascade Style Sheet

- CSS
  - 必然ではないが．．．
  - 対応ブラウザーごとに作成する場合も多い

例 ¥stores¥Web¥<store>¥fashionfair.css

```
body {
        margin-top: 0px;
        margin-left: 0px;
        padding-top: 0px;
}
A:link {
        text-decoration: underline;
        color: #336666;
        font-weight: bold;
}
A:visited {
        text-decoration: underline;
        color: #669999;
}
A:active {text-decoration: underline}

a:hover {
        text-decoration: none;
        color: #CC6600;
}
font.subheader {
        font-family: Verdana, Arial, Helvetica, sans-serif;
        font-size: 8pt;
        font-weight: bold;
        color: White;
        padding: 0px;
        margin: 0px;
        text-decoration: none;
}

td.buttonbar {
        text-align: right;
        width: 620px;
        background-color: #336666;
        padding-right: 0px;
        margin-right: 0px;
        padding-top: 0px;
        padding-bottom: 0px;
}

td.banner {
        background-color: #336666;
}

td.navigation {
        text-align: right;
        width: 620px;
        background-color: #CCCC99;
        padding-right: 0px;
        margin-right: 0px;
}

h2.navoff {
        font-family: Verdana, Arial, Helvetica, sans-serif;
        font-size: 8pt;
        font-weight: bold;
        color: #336666;
        padding: 0px;
        margin: 0px;
}

h2.navon {
        font-family: Verdana, Arial, Helvetica, sans-serif;
        font-size: 8pt;
        font-weight: bold;
        color: #000000;
        padding: 0px;
        margin: 0px;
}
```

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# Property File

- Propertiesファイル
  - マルチカルチャーを諦めれば使用する必然性は無い
  - 各ページでの文言統一？
  - UTF Encoded
    - JDKのnative2asciiを使用してエンコード可能
    - 使用したい文字列をファイルに保管し、native2ascii <filename>で出力が得られる

```
# infashion store translation text

ENCODESTATEMENT = text/html; charset=Shift_JIS

# header.jsp
SHOPPING_CART = ¥u30b7¥u30e7¥u30c3¥u30d4¥u30f3¥u30b0¥u30fb¥u30ab¥u30fc¥u30c8
MY_ACCOUNT = ¥u30a2¥u30ab¥u30a6¥u30f3¥u30c8
CONTACT_US = ¥u554f¥u3044¥u5408¥u308f¥u305b¥u5148
HELP = ¥u30d8¥u30eb¥u30d7
SEARCH = ¥u691c¥u7d22
HOME = ¥u30db¥u30fc¥u30e0

#footer.jsp
SHOPPING_CART2 = ¥u30b7¥u30e7¥u30c3¥u30d4¥u30f3¥u30b0¥u30fb¥u30ab¥u30fc¥u30c8
MY_ACCOUNT2 = ¥u30a2¥u30ab¥u30a6¥u30f3¥u30c8
CONTACT_US2 = ¥u554f¥u3044¥u5408¥u308f¥u305b¥u5148
HELP2 = ¥u30d8¥u30eb¥u30d7
PRIVACY_POLICY = ¥u30d7¥u30e9¥u30a4¥u30d0¥u30b7¥u30fc¥u30fb¥u30dd¥u30ea¥u30b7¥u30fc
```

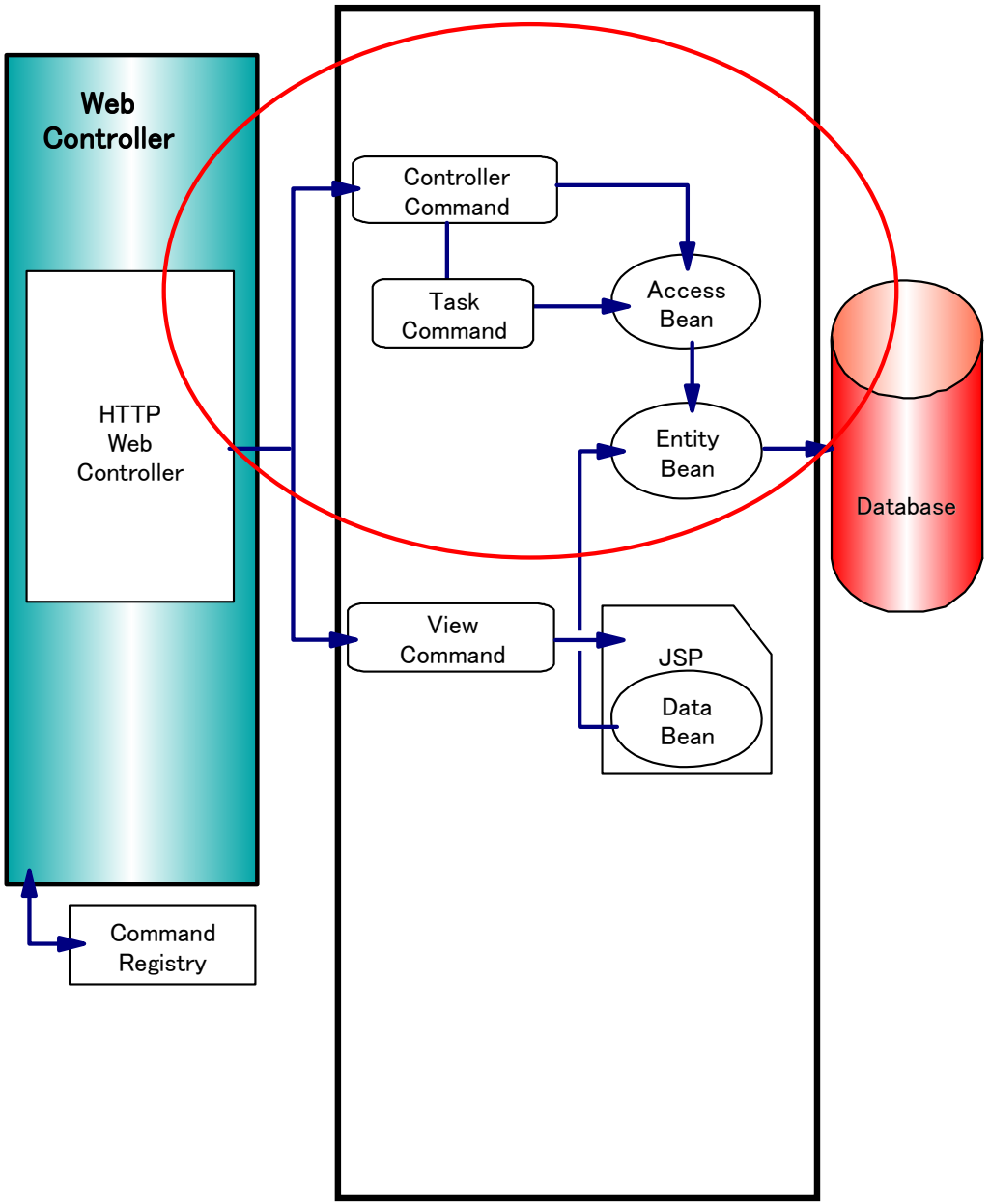
:

例 ¥stores¥propeerties¥<store>¥infashiontext_ja_JP.properties

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# ビジネスロジックの開発

- ビジネスロジック開発
  - Controller Command開発
  - Task Command開発
- テーブルの追加
  - Entity Bean開発

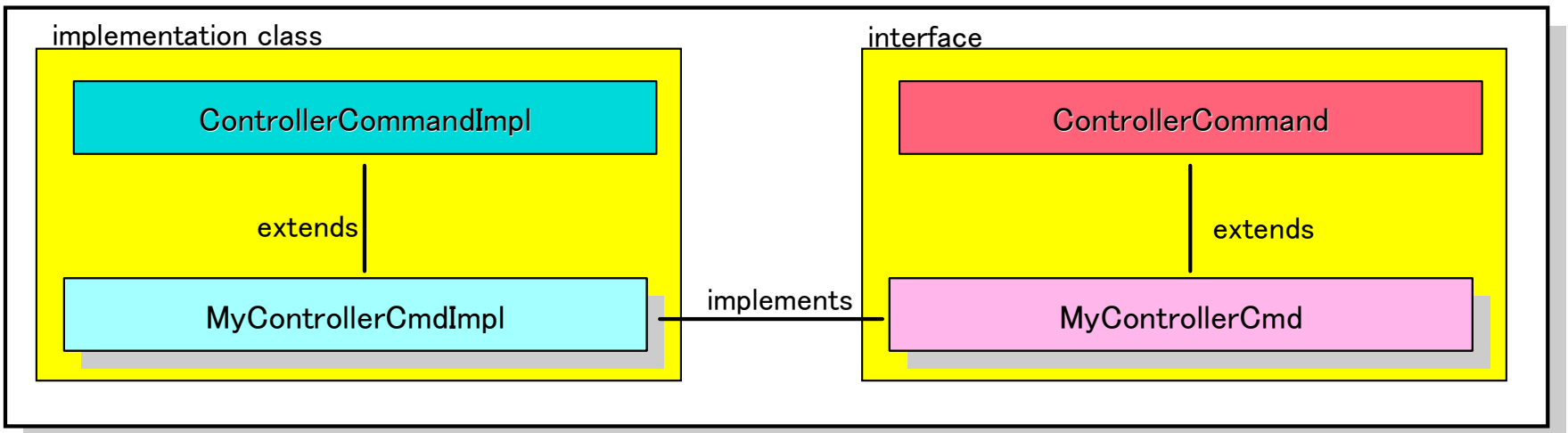



(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# Controller Command

## Controller Commandとは

- ビジネスロジックをカプセル化したJavaクラス
- WebControllerから呼ばれるため、クライアント依存しない
- URL Commandとマップされる
  - URLREG/CMDREG
- データベース操作をAccessBeanを介して行う
  - 主な処理はDB操作となる
- 必要に応じ、Task Commandを呼び出す
- VIEWNAMEをコマンド実行終了とともにWebControllerに戻す
- 関連図
  - インプリメンテーションクラスはcom.ibm.commerce.command.ControllerCommandImplをextend
  - インターフェースはcom.ibm.commerce.command.ControllerCommandをextend




(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# Controller Command Methods

- 以下のメソッドがインプリされる必要がある
  - isGeneric()
    - 管理者でない一般ユーザーがコマンドを使用可能かを指定
    - booleanが戻る（デフォルトfalse）
  - isRetriable()
    - トランザクションロールバック時にリトライするかを指定
    - booleanが戻る（デフォルトfalse）
    - Retriableの例：ProductDisplay
    - Nob-Retriableの例：OrderProcess
  - setRequestProperties()
    - 入力パラメーターをWebControllerからControllerCommandに渡す
  - checkParameters()
    - パラメーターチェック
    - 必要に応じてperformExecute()から呼ばれる
  - performExecute()
    - 実際のビジネスロジックが記載される部分
    - 受領したパラメーターに基づきAccessBeanのメソッドを呼び出し、データ操作を実行
    - 必要に応じCommandFactoryからTaskCommandを呼び出す
    - 最後にViewNameを必ず戻す

WCS5.1 GA TECH W/S

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# Command Registry(2) - URLREG

- Controller Command InterfaceとURLのマップ
  - URLはインターフェース名にマップされる
  - ストア毎に定義を切り替えられる
  - コマンドがSSLを要求するか(HTTPS)、ユーザーのログオンを要求するか(AUTHENTICATED)などを定義できる
  - これもSQLで更新

Sample Data

URLREG

| URL | StoreCatalogDisplay |
|---|---|
| STOREENT_ID | 0 |
| INTERFACENAME | com.ibm.commerce.catalog.commands.StoreCatalogDisplayCmd |
| HTTPS | 0 |
| AUTHENTICATED | - |

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# Command Registry(3) - CMDREG

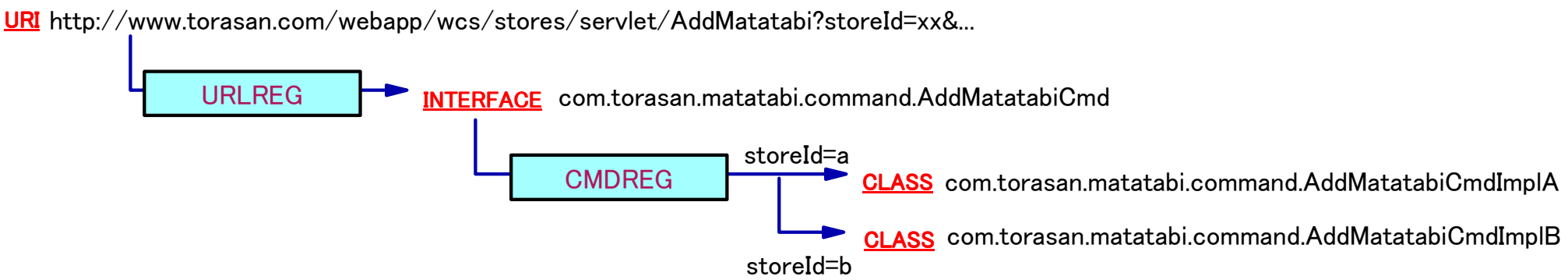
## Controller Command InterfaceとClassのマップ

- 1インターフェースが複数のインプリメンテーションクラスを持つ場合、CMDREGに定義される
  - クラスマップする必要が無い場合はINTERFACEにデフォルトのクラスをdefaultCommandClassName属性で指定
- プロパティーフィールドはデフォルトのパラメーターを渡す際に使用される

Sample Data

CMDREG

| | |
|---|---|
| INTERFACENAME | com.ibm.commerce.catalog.commands.StoreCatalogDisplayCmd |
| STOREENT_ID | 0 |
| CLASSNAME | com.ibm.commerce.catalog.commands.StoreCatalogDisplayCmdImpl |
| PROPERTIES | - |




(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# Controller Commandの作成

## 新規ロジックの作成

- ロジックの要望に応じて、ビジネスロジックを実行する
  - AccessBeanに対するメソッド実行が中心
    - getXX
    - setXX
- テーブルの拡張＝EJBの新規作成が入るケースが多い
  - その場合、要件に応じてEntityBean/AccessBeanをモディファイ

## 既存コマンドのextend

- 現行コマンドをextendして新規コマンドを作成
- クラス、インターフェースをそれぞれ元クラスをextendして宣言
  - interface CustomOrderItemUpdateCmd **extends** com.ibm.commerce.orderitems.commands.OrderItemUpdateCmd
  - class CustomOrderItemUpdateCmdImpl **extends** com.ibm.commerce.orderitems.commands.OrderItemUpdateCmdImpl **implements** CustomOrderItemUpdateCmd
- 各メソッドでスーパークラスのメソッドを参照
  - super.performExecute()
- 独自ロジックの追加

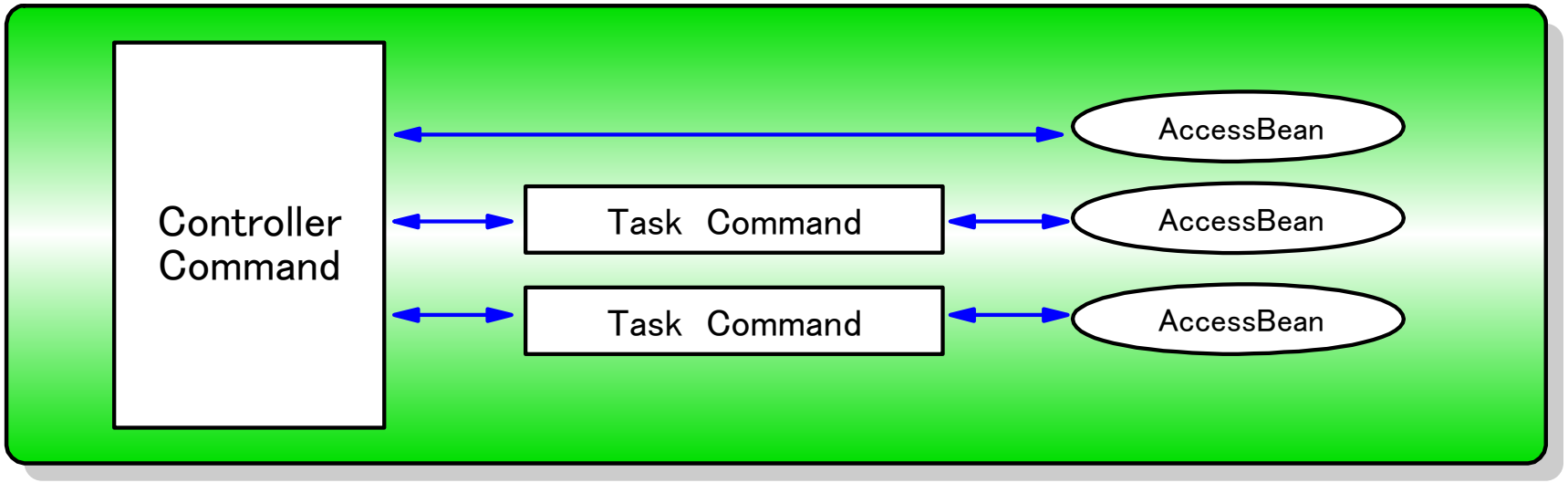
(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# Task Command

## Task Commandとは

- ControllerCommandから任意で呼ばれるビジネスロジック
- Command Factory経由で呼ばれる
  - CommandFactory.createCommand(interfaceName, storeId)
  - パラメーターセット後、execute()でperformExecuteメソッドが実行される

```
cmd = CommandFactory.createCommand(InterfaceName,storeId);
cmd1.setA();
cmd1.execute();
cmd1.getB()
```

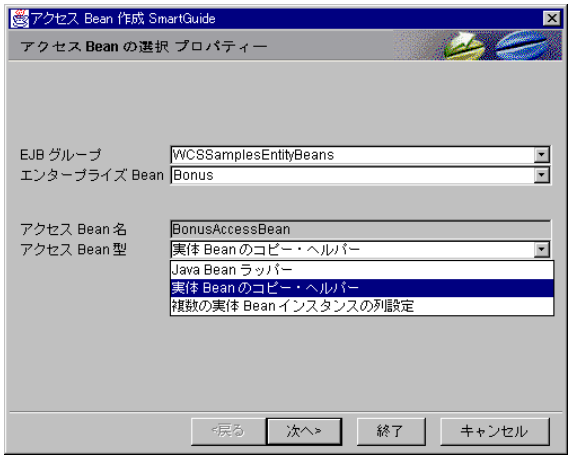


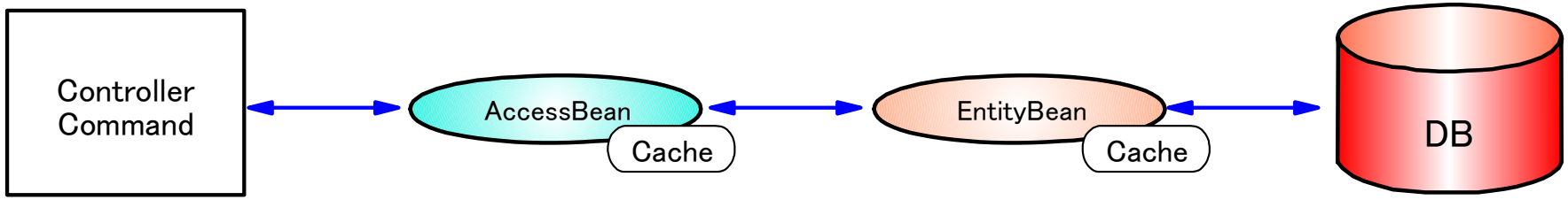

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# AccessBean

## AccessBeanとは

- これもまたEntityBeanの単純化
  - EJB特有のお作法が不要になり、EJBをJavaBeanとして簡単に利用できる
  - EnterpriseBeanのHomeオブジェクトをキャッシュするので、Lookupが軽減される
- Controller CommandからのDatabaseAccessはこのAccessBean経由で行われる
- AccessBeanの三種類
  - Beanified Wrapper
    - EJBnの単純ラップでEntity/SessionBean両方に対し作成可能
  - Copy Helper
    - プロパティーキャッシュ機能を持つ
    - EntityBeanに対してのみ作成可能
    - WCSが使用しているのはこのAccessBean
  - Rowset
    - CopyHelperの集合を作成するためのもの
- AccessBeanはVAJのツールで作成可能




(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# EJBの新規開発

## EJBを追加する

- WCSの表に関してはEntityBean、AccessBean、DataBeanなどはあらかじめ提供されているので追加の必要は無い
- カストマイズ上、新規表を追加した際にCommand構造からアクセスさせるためにEntityBean、AccessBean、DataBeanなどを作成する必要がある
- EJBのカストマイズを行う場合、原則としてVAJ **Enterprise Edition**が必要

### アプローチ

1. EnityBeanを作成し、VAJにより表作成を行う
2. 表を作成し、VAJによりEntityBean作成を行う
3. 表とEntityBean両者を作成し、VAJ上でマップさせる

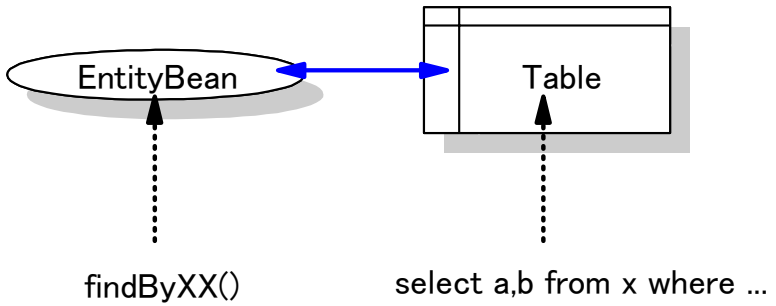
- 推奨は3のケース

### テーブルの作成

- アプリケーション要件に基づき、テーブルをデザイン、WCS DB上に作成

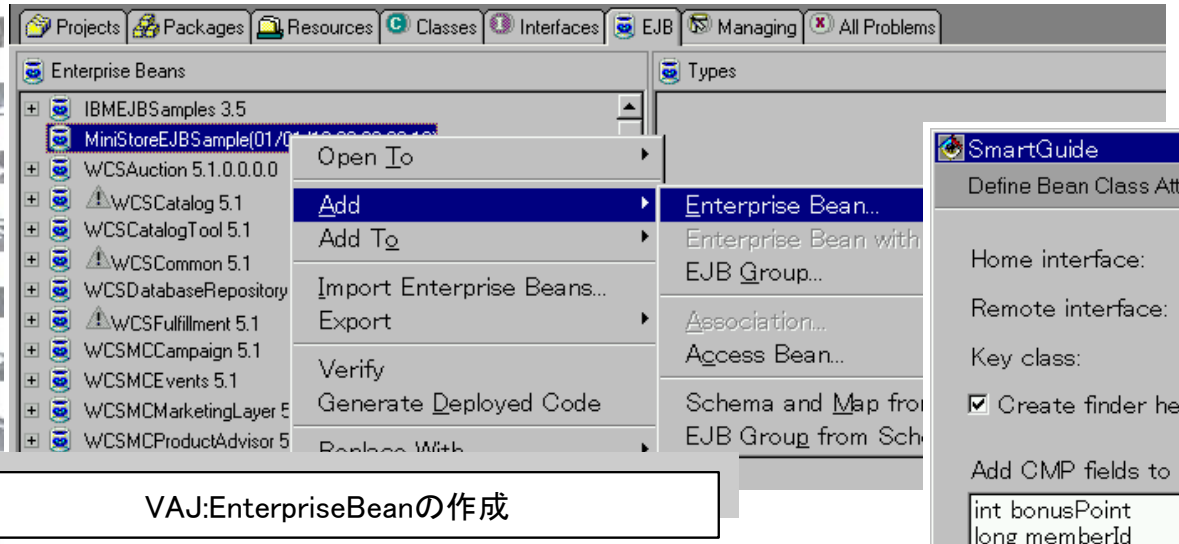
### EntityBeanの作成

- VAJ Smart GuideからAdd → EntityBeanの作成を選択
- Bean TypeにCMP EntityBeanを選択し、CMP Fieldsにカラムに対応するFieldを追加する
- 一部コードを修正
  - finderHelper, Homeインターフェースなどを必要に応じ修正
- 表とEntityBeanのマップ
  - VAJ Schema BrowserおよびMap Browserを使用
- AccessBeanの作成
  - VAJからEntityBeanを選択、Add → AccessBean

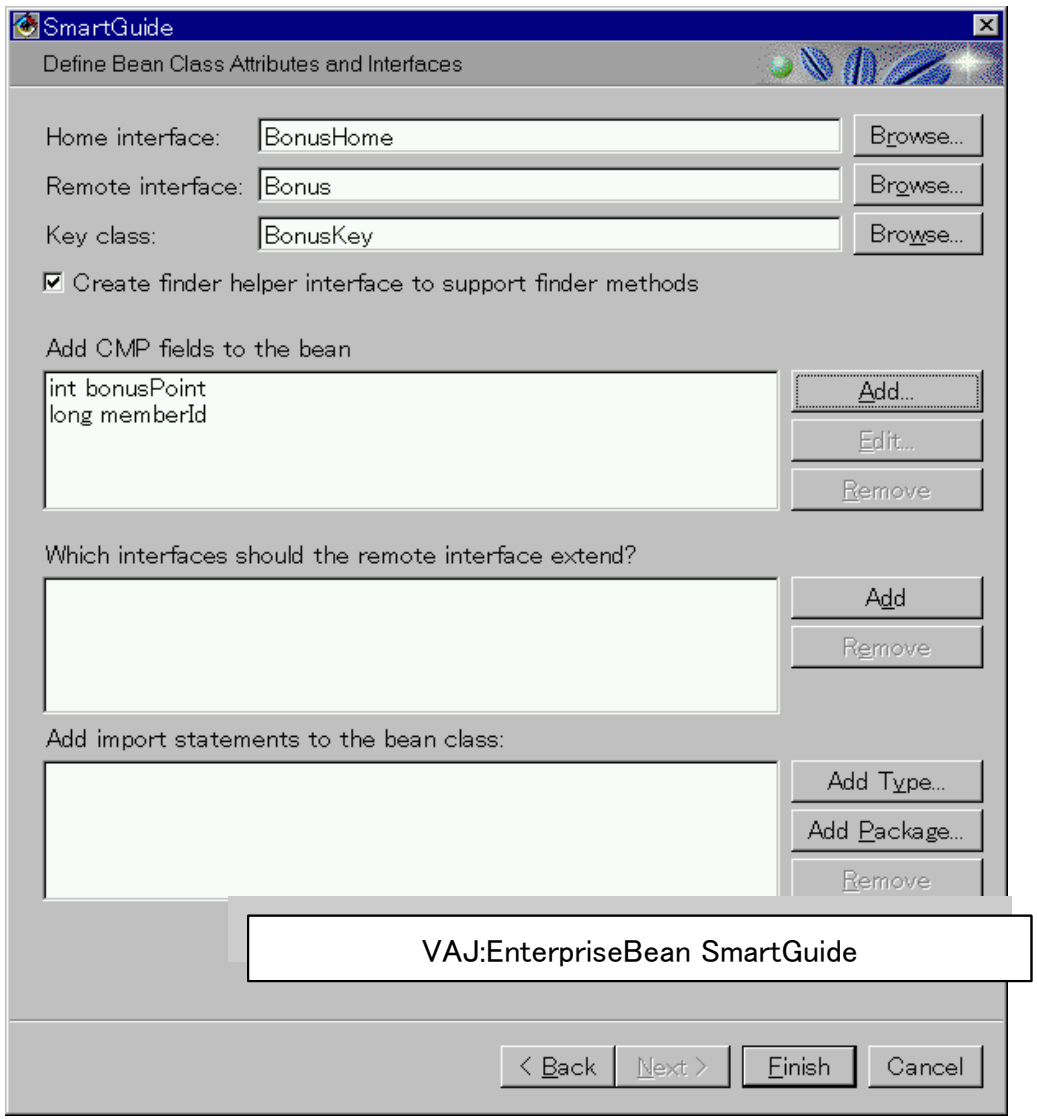



(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# (参考)VAJでのEntityBeanの作成



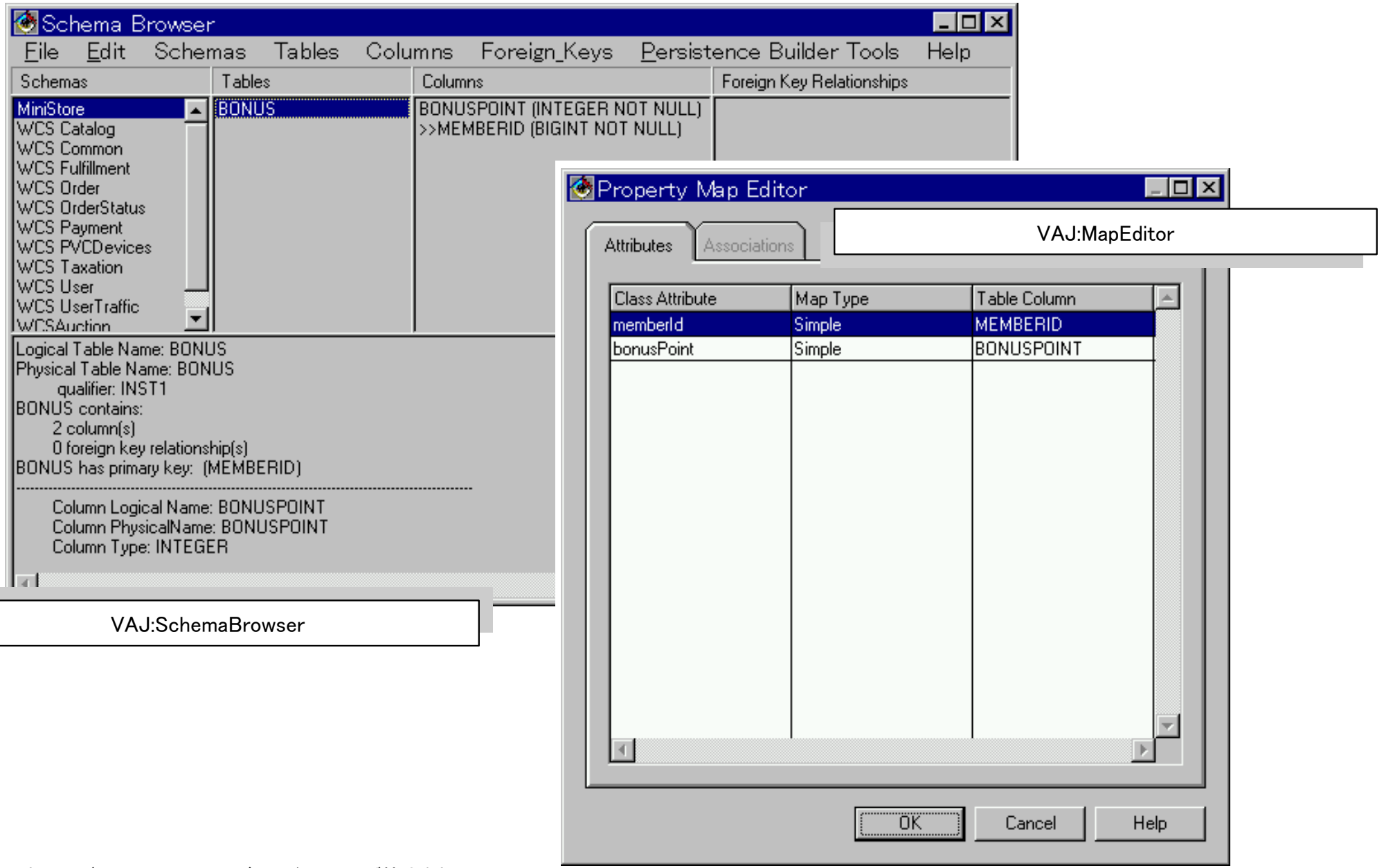

VAJ:EnterpriseBeanの作成




VAJ:EnterpriseBean SmartGuide

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# (参考)VAJでのEntityBeanとDBのマップ

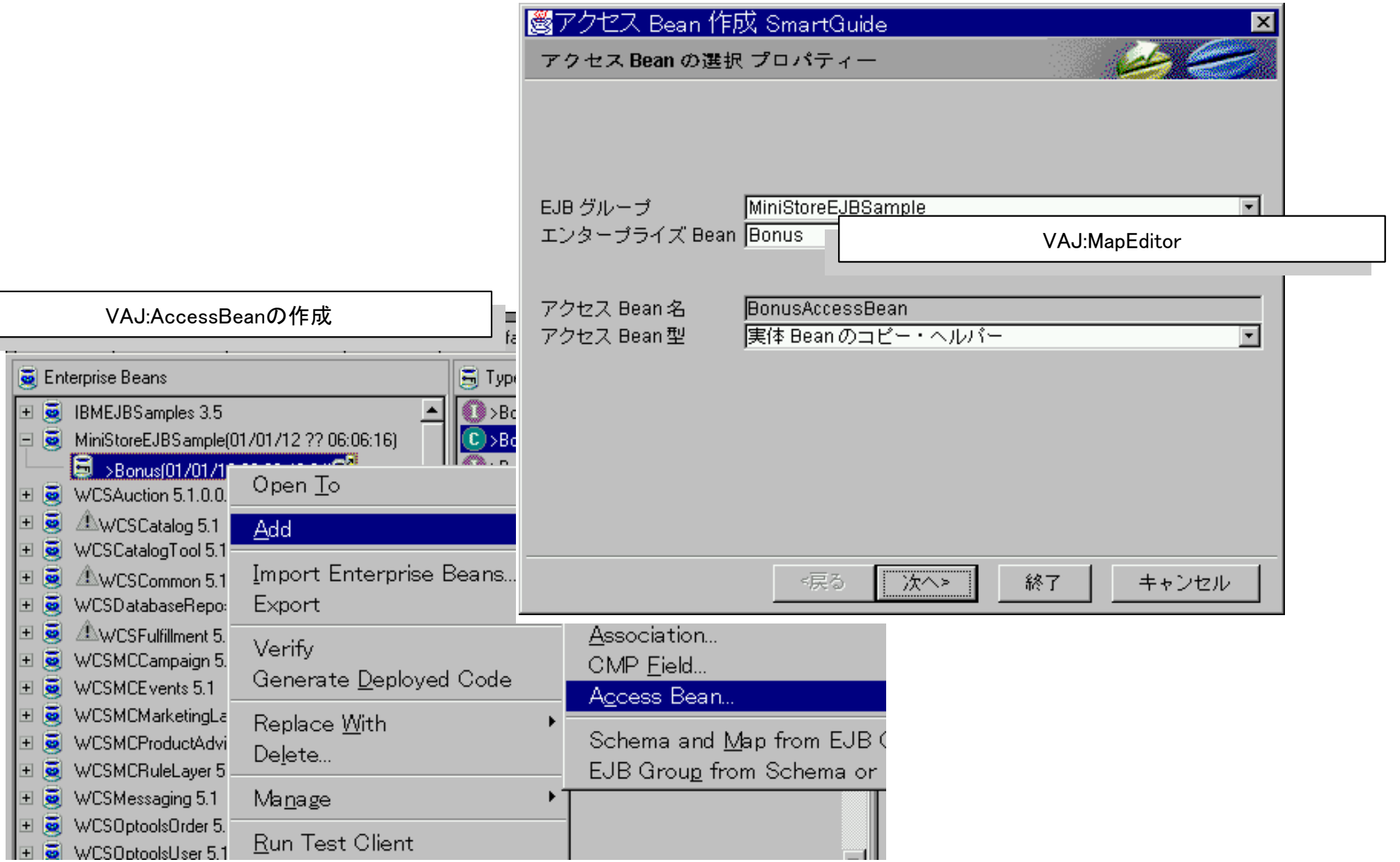



(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# (参考)VAJでのAccessBeanの作成

VAJ:AccessBeanの作成

VAJ:MapEditor

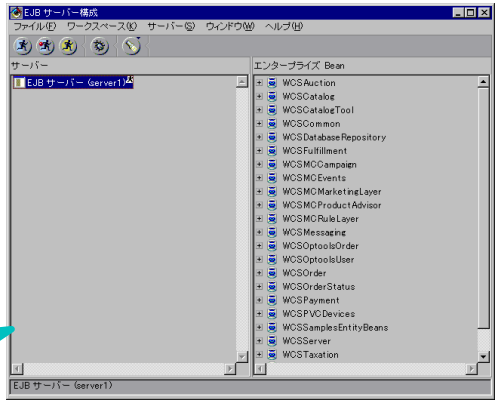



(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

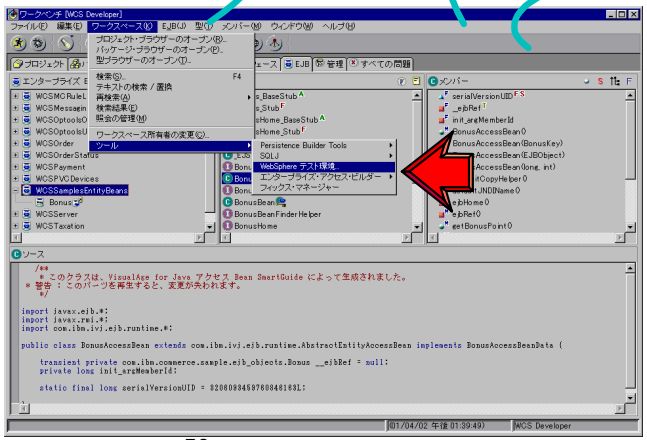
# テスト(単体)

WebSphereテスト環境コントロールセンター

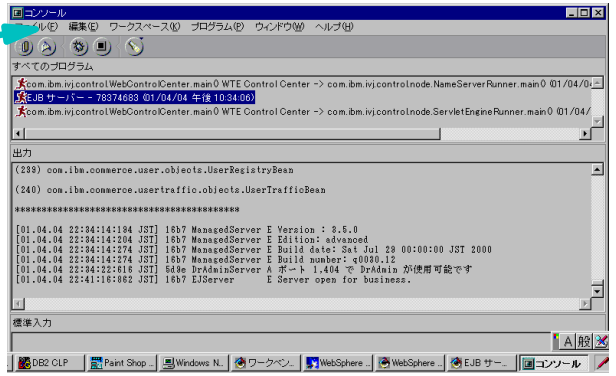

EJBサーバー




ワークベンチ




コンソール




ブラウザーからlocalhost:8080でアクセス

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

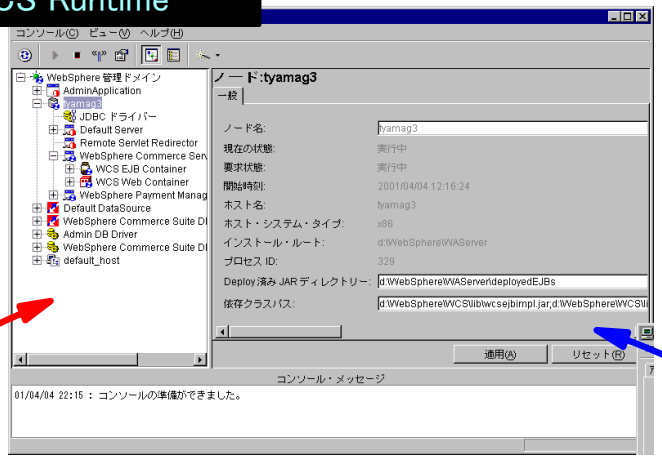
# テスト(単体)

- VAJ WebSphere Test Environment
  - Javaコマンド、JSPなどはWebSphere Test Environmentでテストが可能
  - Publish/Exportが必要ないため作成したコードを即時にテスト出来る
  - EJBサーバーも稼動する
  - コマンドフロー一連の流れ(ControllerCommand、TaskCommand、AccessBean、EntityBean、JSP、DataBean)を検証できる
- VAJ EJB Test Client
  - VAJ上のEJBサーバーで稼動するEnterprise Beanのテストが可能

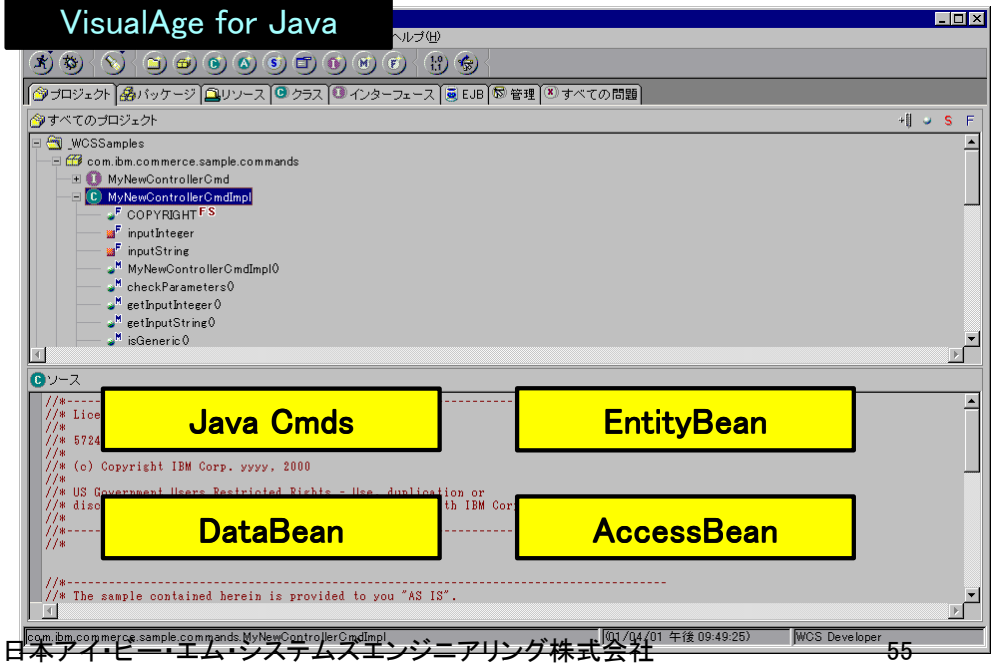
(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# Deployment

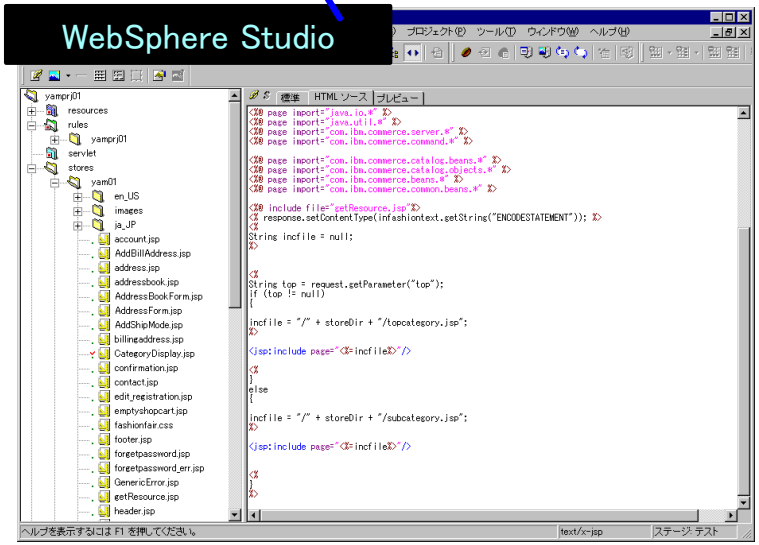
WCS Runtime



VisualAge for Java




WebSphere Studio



(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# Deployment

## 開発環境からサーバー環境への移設

- コードを実稼動状態にする

## From VAJ

- Java Command、DataBean
  - プロジェクト単位にJARでEXPORT
  - WAS AdminConsoleよりApplication Server停止
  - WAS AdminConsoleより上記JARの絶対パスをクラスパスに追加
  - コマンドレジストリにコマンド登録(URLREG、CMDREG、VIEWREG)
    - テスト環境でINSERT/UPDATEするSQLをファイルに保管しておく
- EntityBean、AccessBean
  - Deployed JAR
    - EJBタブよりEJBグループを選択、Export→Deployed JARで[WCS]¥libに保管
  - インプリメンテーションJAR
    - プロジェクトタブよりプロジェクトを選択、Export→JARで任意の場所に保管
  - Admin Consoleで[インプリメンテーションJAR]→[Deployed JAR]順で以下のクラスパス先頭に追加
    - ノードの依存クラスパス
    - Webアプリケーションのクラスパス
  - WASトポロジーへの登録
    - AdminConsole or XMLConfig

## From Studio

- JSP、HTML
  - "Publish"

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# WASアプリケーション開発との違い

- WAS　－新規開発
  - 開発負荷は大きい反面、自由度が効く
    - Webアプリケーション開発に必要な機能は提供される
      - セッション管理
      - DB接続プール
  - データベーススキーマを一から構築の必要性
- WCS　－出来ているArchitectureに乗る
  - 流用できるパーツが豊富にある
    - 他社開発のコンポーネント提供、別プロジェクトの成果物流用
    - 製品なので足りない機能は今後のバージョンで提供される可能性
  - コマンド構造が確立されているので、ビジネスロジックの開発に専念できる
    - 逆を言えば、なるべく今ある動きに合わせて開発を行う事が時間/コスト軽減につながる
    - 現行サイトの動きを理解しないと先に進めない
- 選択基準
  - B2Cサイト構築はまずWCSで検討
  - 要件に合わせた仕様変更の負荷を検討
  - 要件と仕様に大きな隔たりがある場合、WASでの新規開発がありうる

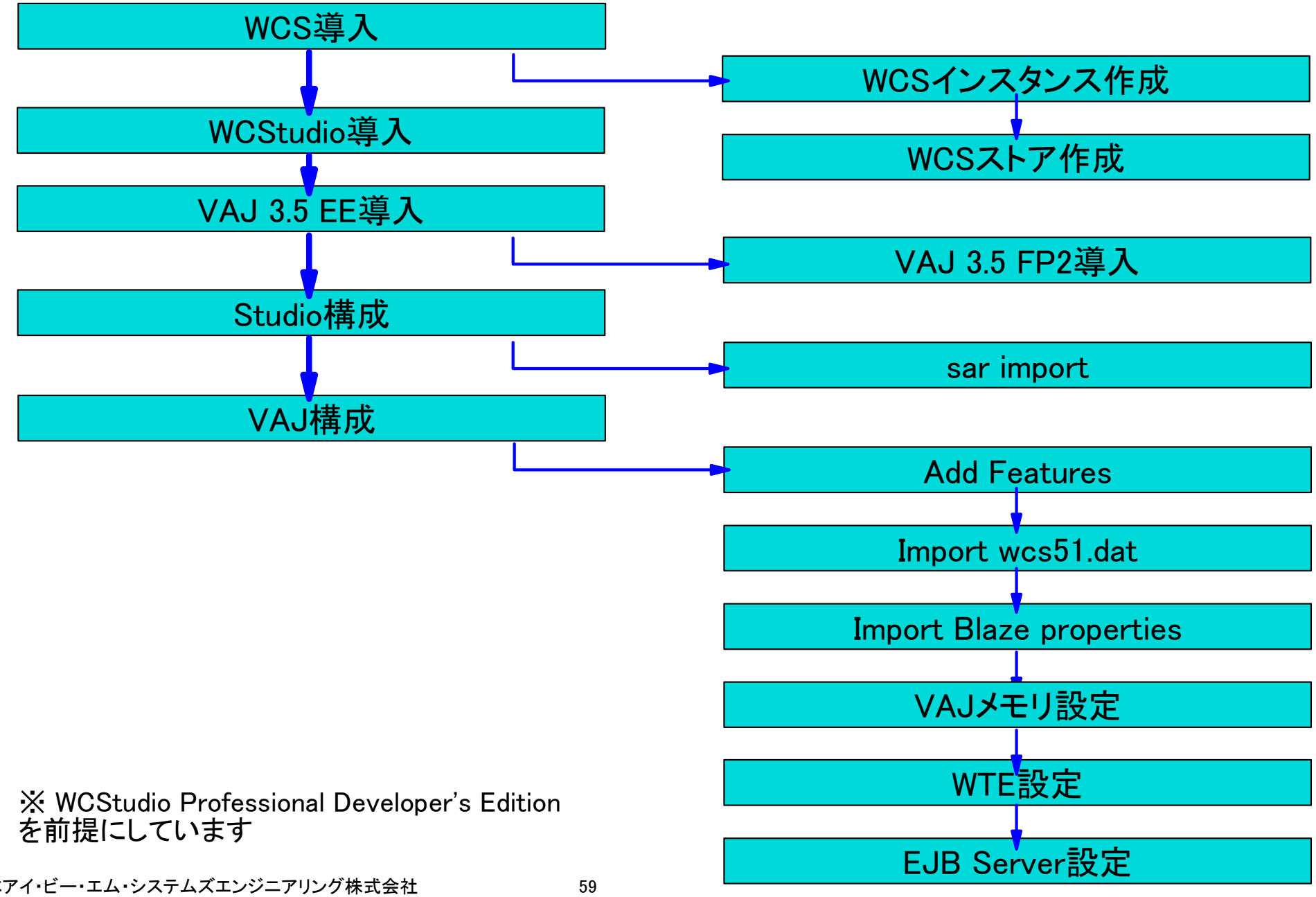
WCS5.1 GA TECH WAS

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# 参考－(1)
# 開発環境セットアップ手順

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# WCS開発/テスト環境セットアップ手順




※ WCStudio Professional Developer's Edition
を前提にしています

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# セットアップ手順(1)

## ▶参考資料

- WebSphere Commerce Studioインストールの手引き
- WebSphere Commerce Suite Readme.htm
  - WCStudioでなく、WCSのReadmeを参照
  - WCSのCDはStudioに同梱(テスト環境使用ライセンス)

## ▶WCS導入

- 導入、インスタンス作成、ストア作成まで完了させる
  - WCSインストールマニュアルおよび『導入』資料参照

## ▶WCStudio、VAJ導入

- WCStudioインストールマニュアル参照
- VAJを自動的に導入する
  - 30分以上程度かかるので、途中で止まったかのように見えてもしばし寛容に接する

## ▶VAJ FP2導入

- [WCStudio_CD]¥VAJFIX¥setup.exe

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# セットアップ手順(2)

## ▶VAJ構成

- Add Feature
  - IBM EJBDevelopment Environment 3.5
  - ~~IBM Common Connector Framework 3.5~~
  - ~~IBM Java Record Library 3.5.0.1~~
    - 下の2つはwcs51.datのインポート時に重複と警告されるので、EJB Development EnvのみAdd
  - ワークスペース所有者変更
    - "Administrator"に変更
- IBM WebSphere Test Environment Project
  - 右クリック → manage → Create Open Edition
- WAS 3.5.0 efixインポート
  - ファイル→インポート
    - [WCS_CD]¥wasfix¥wasefix¥was 3.5.0 efixes¥85699_350.jar
    - [WCS_CD]¥wasfix¥wasefix¥was 3.5.0 efixes¥88452_350.jar
- wcs51.datインポート
  - [WCS_CD]¥repositry¥wcs51.dat
  - 以下のエラーが発せられるが無視(readmeより)
    - 『メタデータのロード・エラー： 記憶クラスにメタデータがありません：＊＊＊＊』
  - ワークスペース所有者変更
    - "WCS Developer"に変更
- VAJ使用可能メモリーの増加
  - ¥[VAJava]¥ide¥program¥ide.ini編集

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# セットアップ手順(3)

## VAJ構成 (cont.)

- Blazeリソースのインポート(readmeより)
  - 以下のプロパティのみをインポート
    - WCS_install_directory/Blaze/AdvSvr31/lib/AdvCommon.jar
    - WCS_install_directory/Blaze/AdvSvr31/lib/Advisor.jar
    - WCS_install_directory/Blaze/AdvIrt31/lib/InnovatorRT.jar
    - WCS_install_directory/Blaze/AdvSvr31/lib/AdvisorSvr.jar
- EJBサーバー構成
  - 右クリック→プロパティーでEJBサーバープロパティを変更
- PNS(Persistent Name Service)用DBを作成
  - DB2CLP起動
    - > db2 create db PNS
- PNS構成
  - WebSphereテスト環境コントロールセンター
    - PNSサーバー・ペインの情報を作成したDB情報で更新
  - ネームサーバーの開始
    - コンソール画面に、『 Server Open for business』を確認
  - データソース構成→追加
    - コンソール画面に、『name = jdbc cname = /』を確認
  - ネームサーバーの再起動
    - コンソール画面に、『 Server Open for business』を確認
- EJBサーバー開始
  - EJBサーバー構成画面
    - EJBサーバー(Server1)を右クリック→サーバーの開始
    - 時間がかかるが、コンソール画面に、『 Server Open for business』を確認するまで待つ

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# セットアップ手順(4)

## VAJ構成 (cont.)

- サーブレットエンジンの開始
  - テスト環境のファイルコピー
    - ¥VAJava¥ide¥project_resources¥IBM WebSphere Test Environment¥hosts¥default_host¥default_app¥web¥[storename]¥*.*を
    - ¥VAJava¥ide¥project_resources¥IBM WebSphere Test Environment¥hosts¥default_host¥default_app¥web¥に
  - SSLモードオフ
    - DB2CLPより
    - > db2 connect to [wcs_db]
    - > db2 update urlreg set https=0
    - > db2 update viewreg set https=0
  - WTEにServlet追加(必要なし)
    - ~~¥VAJava¥ide¥project_resources¥IBM WebSphere Test Environment¥hosts¥default_host¥default_app¥servlets¥default_app.webappを編集~~
  - Servletエンジン→Edit Classpath
    - クラスパスの全追加
  - Servletエンジンの開始
    - コンソール画面に、『 Server Open for business』を確認
- テスト
  - ブラウザーを起動し、以下のURLを実行
    - http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?storeId=10001&catalogId=10001&langId=-1
    - パラメーターはケースセンシティブのため注意

WCS V5.1 GA TECH WS

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# 開発環境セットアップ注意点

## マニュアルに無い点（一部はWCSのReadmeに記載）

- AddFeatureで以下のFeatureを追加しない
  - IBM Common Connector Framework 3.5
  - IBM Java Record Library 3.5.0.1
- wcs51.datのインポート
  - メタデータのエラーが発生するが、無視可能
- Blazeプロパティファイルのインポート
  - 実行しないとテストブラウザーで500エラー
- default_app.webappの編集
  - 必要なし

## その他注意点

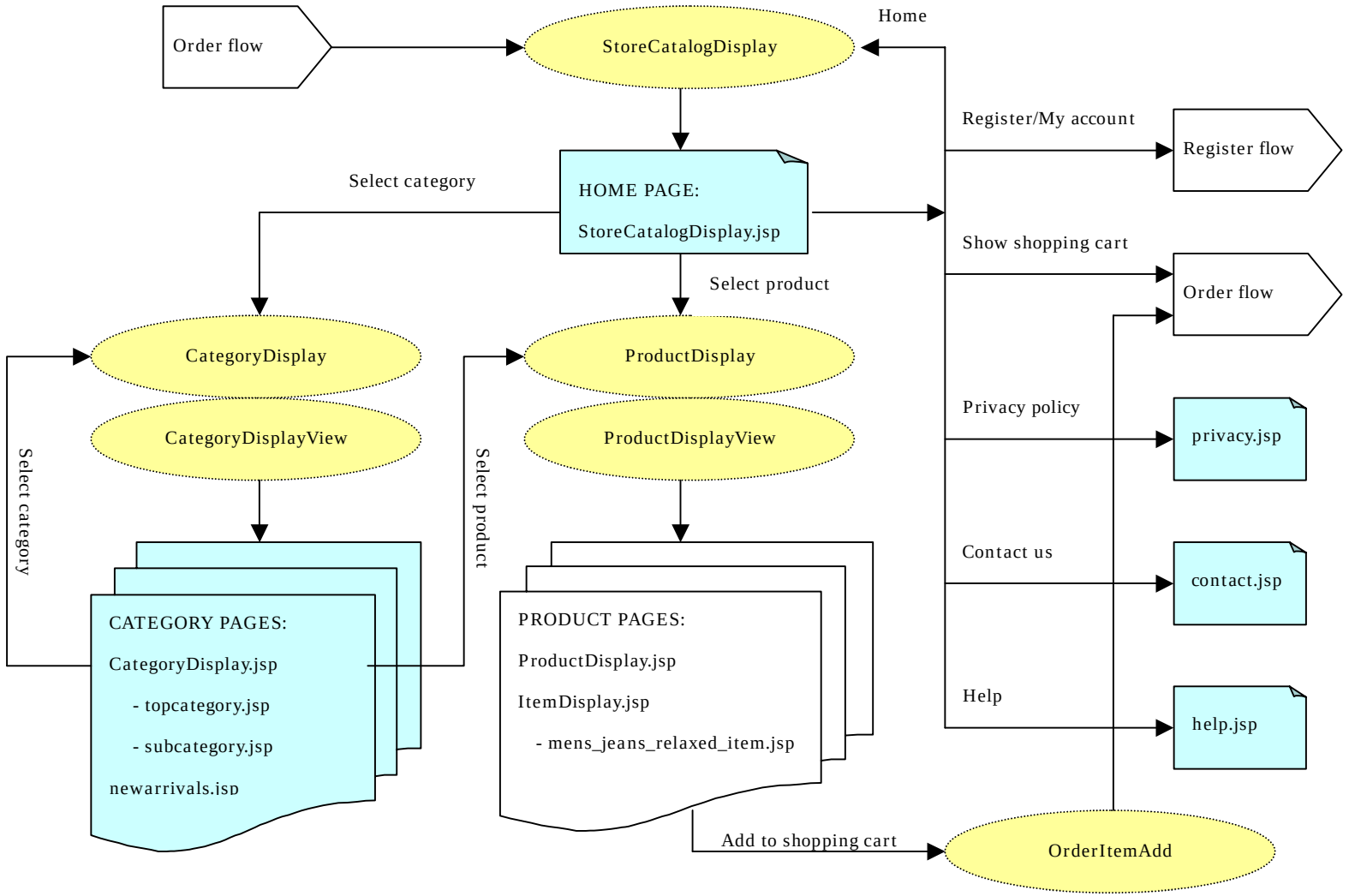
- VAJインストール途中で停止したように見えるが、ThinkPad600X(650MHzPIII,576MB RAM)で40分以上かかるので手を出さずに待つ
- EJBサーバーのプロパティ→データソース名とWTEのデータソース設定のデータソース名が完全に一致するようにコピー&ペーストを推奨
  - demo.xmlの読み込みを行うので、実働WCS環境と同じデータソース名にすること
- テスト環境が500エラーを返すときは以下を確認
  - Blaze Propertyファイルのインポートをしたか？
  - WTE/EJBサーバーの設定はあっているかもう一度確認
  - EJBサーバーが本当に起動しているか？（時間がかかるのでコンソールを確認）
  - EJB起動後にServletエンジンを起動したか？
- テスト環境構築失敗でフォールバック時、最悪VAJを入れなおすが、入れなおし手順はStudioインストールマニュアル『VisualAge for Javaの手動インストールおよび構成』を参照

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# 参考(2) InFashionフロー

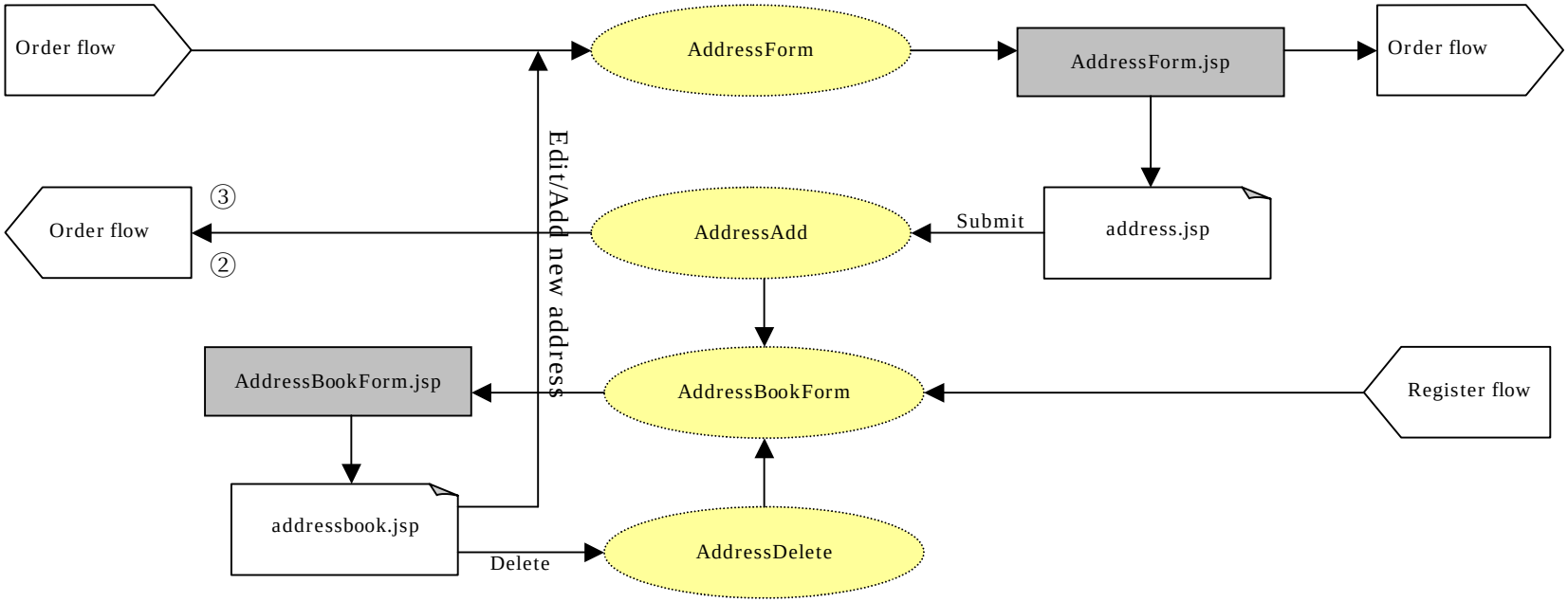
(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# Catalog Flow



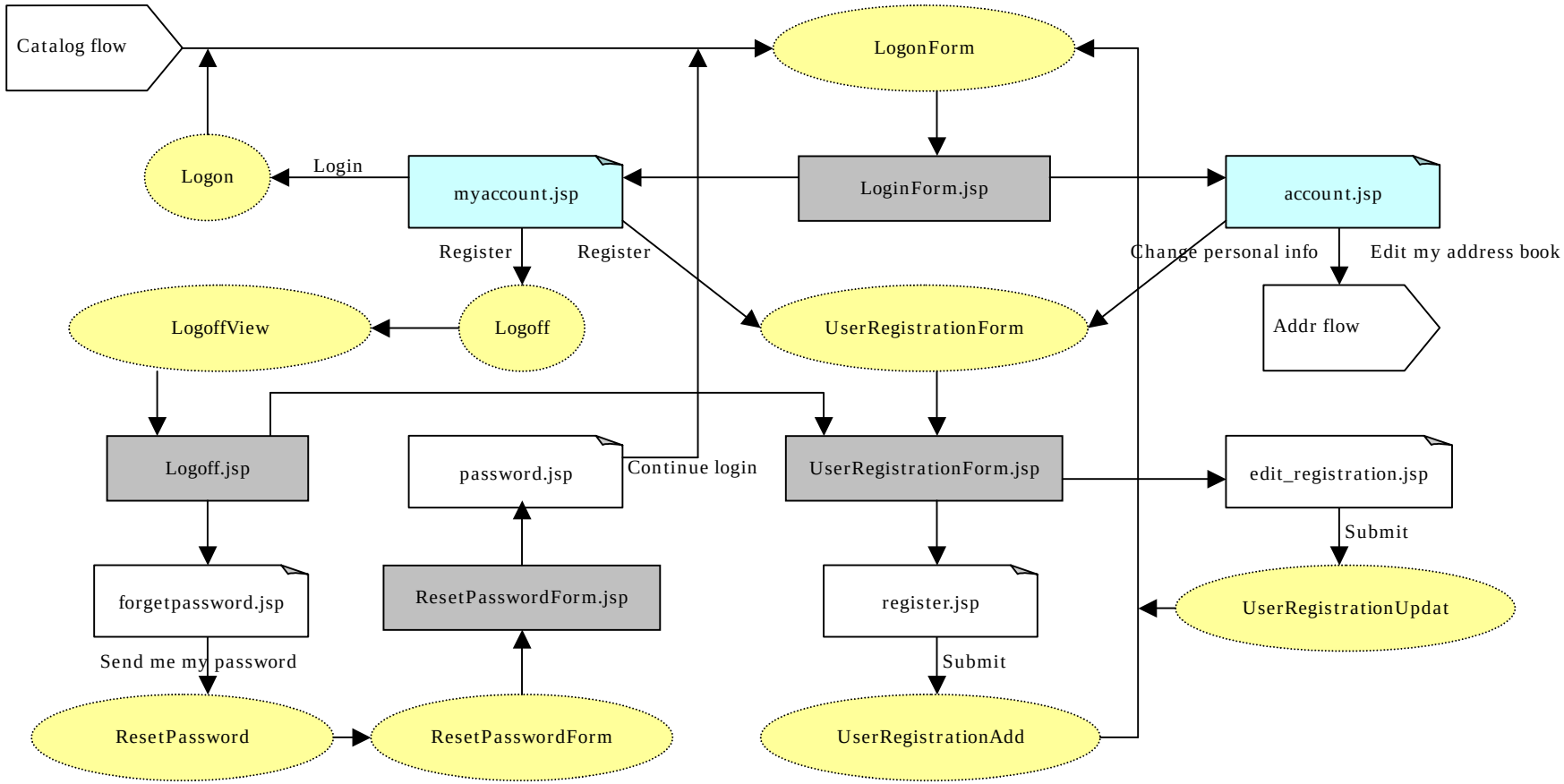

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# Address Registration Flow



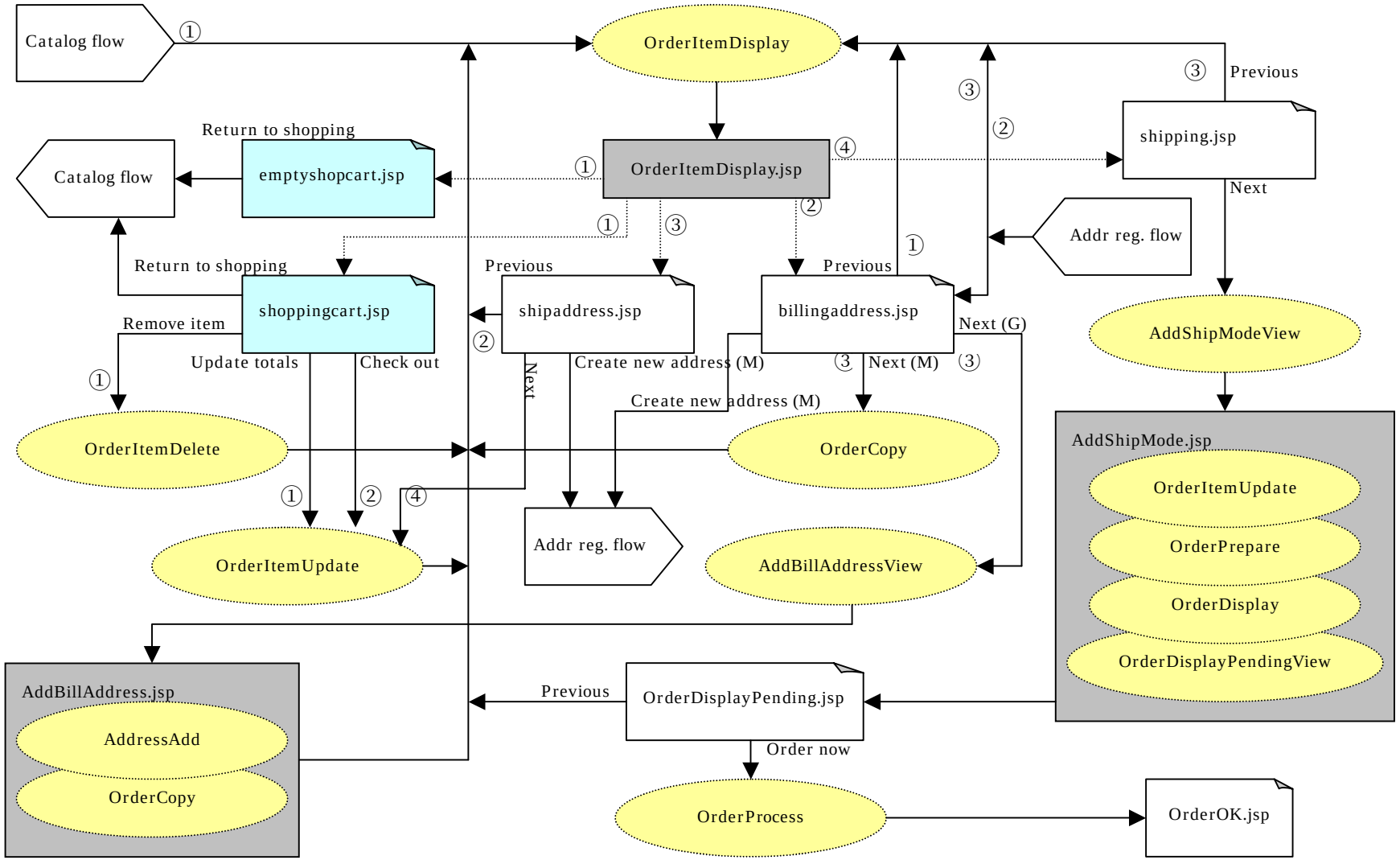

(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# User Registration Flow




(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社

# OrderFlow




(c) Copyright 日本アイ・ビー・エム・システムズエンジニアリング株式会社