Ymirを使ってかんたんWebアプリ開発

Ymirプロジェクト プロジェクトリーダ
（株）アークシステム
横田 健彦（a.k.a. Skirnir）

- 横田 健彦（よこた たけひこ）
  - ネット上ではSkirnir（すきーるにる）と名乗っています
  - （株）アークシステム所属
    - JavaやNETのWeb系技術者募集中！
  - プラグイン機構がウリのCMS「Kvasir/Sora」を開発
    - 2006年度上期IPA未踏ソフトウェア創造事業に採択
  - 北欧神話好き
    - Skirnir、Kvasir、Ymirも北欧神話のキャラクタの名前です
  - 阪神タイガース好き
    - 今シーズンは比較的精神状態良好 ☺ だがここ最近…

今日お話しすること

- Ymirの目指すもの
- Ymirとは
- 新機能
- ロードマップ

Ymirとは

- フィルタ指向のシンプルなWebアプリケーションフレームワーク
  - 読み方は「ゆみる」
    - 北欧神話に出てくる巨人。血肉が世界の元となった。
  - 公式サイト：http://ymir.sandbox.seasar.org/
    - このサイトはKvasir/Soraで作られています
  - 元々はCMS（Kvasir/Sora）への組み込み用に作ったものをWebアプリ開発用に拡張
  - J2SE5以降＋ServletAPI2.4で動作
  - Seasar2.4ベース

特徴

- MVCモデルのCの部分を担当
  - お勧め構成はYmir+FreyjaのZPT実装+DBFlute
    - Freyja…XMLテンプレートエンジン構築F/W。ZPT実装を持っている。
      - http://www.skirnir.net/product/freyja/
- フィルタ指向
- URLとアクションの対応付けを柔軟に定義可
- 強力な自動生成機構（ymir-extension）
  - Seasar2.4のHOT deploy対応

特徴

フィルタ指向

- 画面レンダリングの前に処理を挟み込む
  - サーブレットフィルタとして実装
  - Transitive Path Strategyに基づくWebアプリケーション開発が可能

特徴

フィルタ指向

Transitive Path Strategy

Transitive Path Strategy

- URLとリソースの物理配置を対応付ける戦略
- Webサーバを使って静的リソースを公開する場合の基本形

Transitive Path Strategy(2)

http://www.hoe.com/app/product/list.html

URLの階層構造

ファイルシステムの階層構造

Transitive Path Strategy（3）

- CGI登場
  - コンテンツを動的に生成

  → URLに対応する物理リソースが存在しないケースが増加（ex. path-info）

Transitive Path Strategy(4)

http://www.hoe.com/app/product.py/list

path-info

URLの階層構造

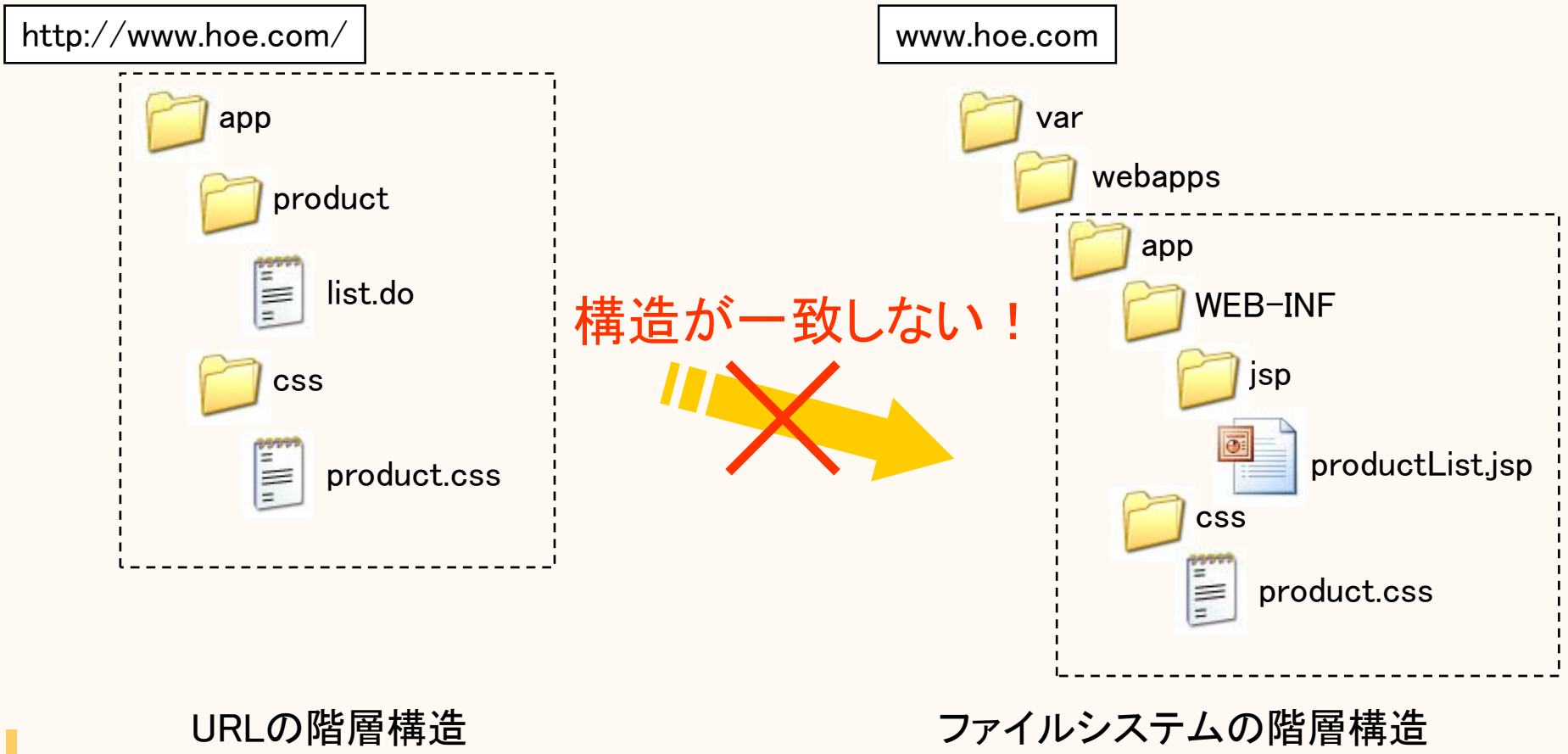
ファイルシステムの階層構造

Transitive Path Strategy（5）

- Javaでよく見る光景

Transitive Path Strategy(6)

http://www.hoe.com/app/product/list.do
http://www.hoe.com/app/css/product.css



URLの階層構造

ファイルシステムの階層構造

Transitive Path Strategy（7）

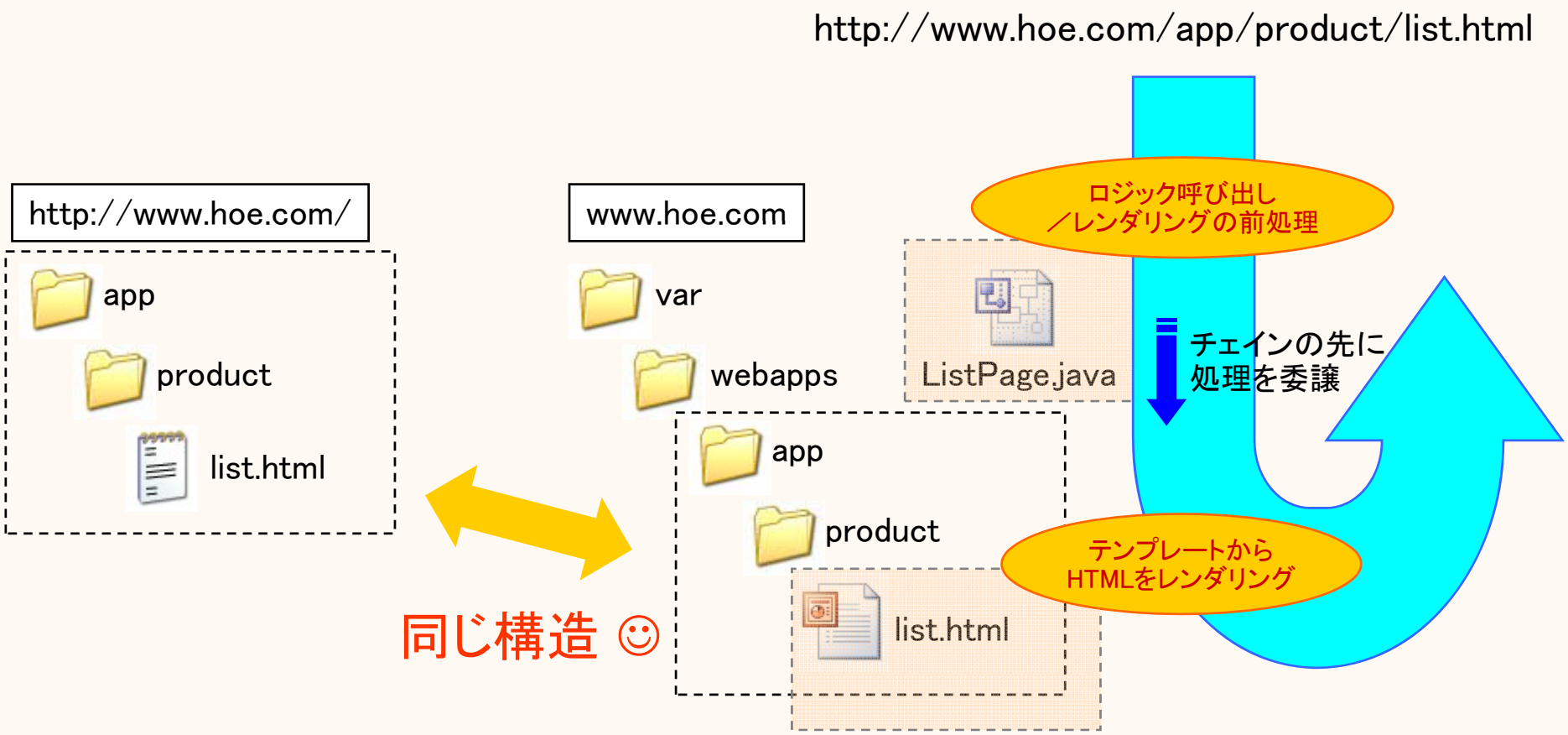
- URLとテンプレート（JSP）のパスが一致しない
  - → ぱっと見で分かりにくい！
  - → クライアントコードもすっきりしない

```
public ActionForward execute(…) {
    …
    return mapping.findForward(“success”);  // ←”success”って何？
}
```

```
public String execute(…) {
    …
    return “/WEB-INF/jsp/productList.jsp”;  // ←テンプレートにスルーさせたいだけなのに…
}
```

- JavaでTransitive Path Strategyを使うために
  - テンプレートエンジンサーブレットをURLとマッピング
  - ロジック呼び出し／レンダリングの前処理はサーブレットフィルタで
  - 処理後はフィルタチェインの次フィルタに処理を委譲

Transitive Path Strategy(9)



Transitive Path Strategy（10）

- クライアントコードもすっきり

```
public void _get() {
    …
}
```

特徴

URLとアクションの対応付け

- 呼び出すロジックはどう決定するか？
  - URLとHTTPメソッドから決定
  - URLパターンとPage名の構築式とアクションメソッド名の構築式の組（パスマッピング）を登録しておく
  - リクエストURLにマッチするパスマッピングについて、Page名の構築式とアクションメソッド名の構築式から処理対象のPageクラスと呼び出すべきメソッドを決定

URLとアクションの対応付け(2)

URLパターン:

^/([a-zA-Z_][a-zA-Z_0-9]*)/([a-zA-Z_][a-zA-Z_0-9]*)¥.html$

Page名の構築式:

${1}_${2}Page

アクションメソッド名の構築式:

_${method}

リクエストHTTPメソッド/URL:

コンテキストパス以降とマッチング

GET http://www.hoe.com/app/product/list.html

マッチ: ${1}="product", ${2}="list"

Page名とアクションメソッド名:

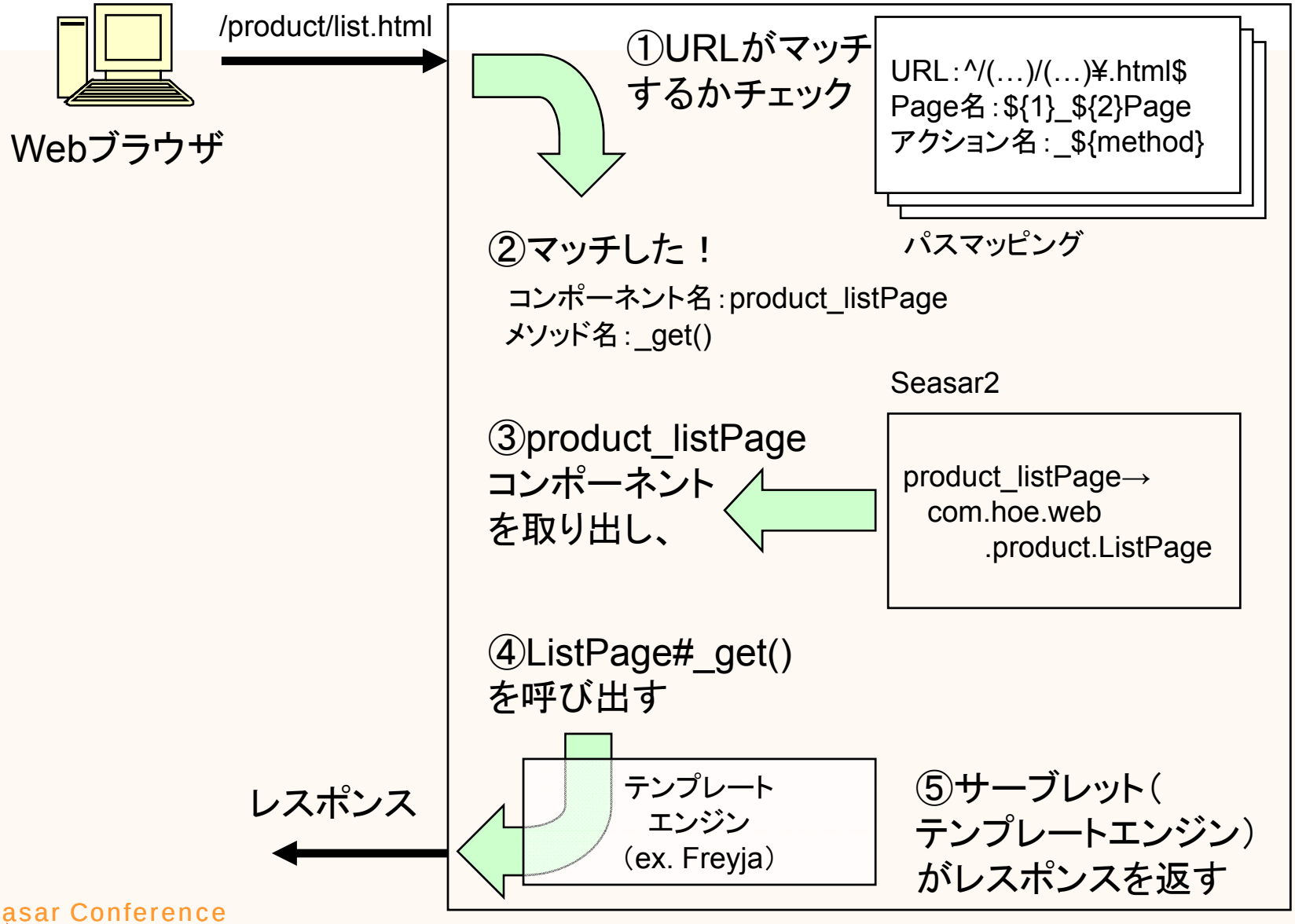
"product_listPage", "_get" → com.hoe.web.product.ListPage#_get()

(Page名→FQCN変換はSeasar2による)

特徴

動作のまとめ

動作のまとめ



特徴

自動生成機能

- リクエスト駆動
  - アプリケーションを実際に動作させる中でJavaコードを生成
  - 生成に関してWebブラウザ上で指示
- 自動生成できない部分はEclipse上で開発
  - HOT deploy機能によって自動生成フェーズと手動開発フェーズをシームレスに結合

開発の流れ

開発の流れ

- アプリケーションスケルトンをダウンロード
  - http://maven.seasar.org/maven2/org/seasar/ymir/ymir-skeleton/0.9.6/ymir-skeleton-0.9.6.zip
- Eclipseへインポート
- Tomcatに配備
- Tomcatを起動
- ブラウザでアクセス
  - 初期設定画面が表示されるので初期設定を行なう
- Tomcatを再起動（最初だけ）

初期設定画面

SYSTEM NOTIFICATION

Ymirへようこそ！

Webアプリケーションを構築する前に、プロジェクトの設定を行ないます。以下の項目を埋めて「OK」ボタンを押して下さい。

基本情報

rootPackageName

このWebアプリケーションのJavaソースコードのルートパッケージ名を入力して下さい。(例:com.example)

com.example

自動生成

extension.sourceCreator.superclass

Pageクラスの共通の親クラス名を指定して下さい。何も指定しないと親クラスは設定されません。

extension.sourceCreatoruseFreyjaRenderClasses

DTOを生成する際、Freyjaが持つレンダリングクラスが使える場合はレンダリングクラスを使うようにするかどうかを指定して下さい。例えば種別を選択するためのラジオボタンの組をレンダリングするためのオブジェクトを返す「typeRadioInputTags」というプロパティが ある場合、この設定が「利用する」になっていればプロパティの型は生成されたTypeRadioInputTagsDtoクラスではなく Freyjaが持つRadioInputTagsクラスになります。

○ 利用する ◉ 利用しない

beantable.enable

Bean定義からテーブルを自動生成するBeantable機能を利用するかどうかを指定して下さい。

○ 利用する ◉ 利用しない

extension.sourceCreator.feature.createFormDto.enable

HTML中のformタグにnameが指定されている場合に同じ名前でformに対応するDTOを生成するかどうかを指定して下さい。

- http://localhost:8080/example/ にアクセス
  - URLに対応するPageクラスがないのでPageクラス生成画面が表示される
  - 「/index.html」にリダイレクトするように設定して「設定」ボタンを押す

Pageクラス生成画面

SYSTEM NOTIFICATION

リクエストされたパスに対応するPageクラスcom.example.web._RootPageが見つかりませんでしたので、クラスを作成します。

既に存在しているテンプレートに処理をフォワードさせたい場合や別のパスに処理をリダイレクトさせたい場合は、下のフォームに遷移先のパス(コンテキスト相対)を入力した上で「設定」ボタンを押して下さい。遷移先がない場合は何も指定する必要はありません。

処理後の遷移先: /index.html

☑ リダイレクトさせる

設定 スキップ

Pageクラス生成画面(2)

- http://localhost:8080/example/index.html にリダイレクトされる
  - URLに対応するHTMLテンプレートがないのでHTMLテンプレート生成画面が表示される

HTMLテンプレート生成画面(1)

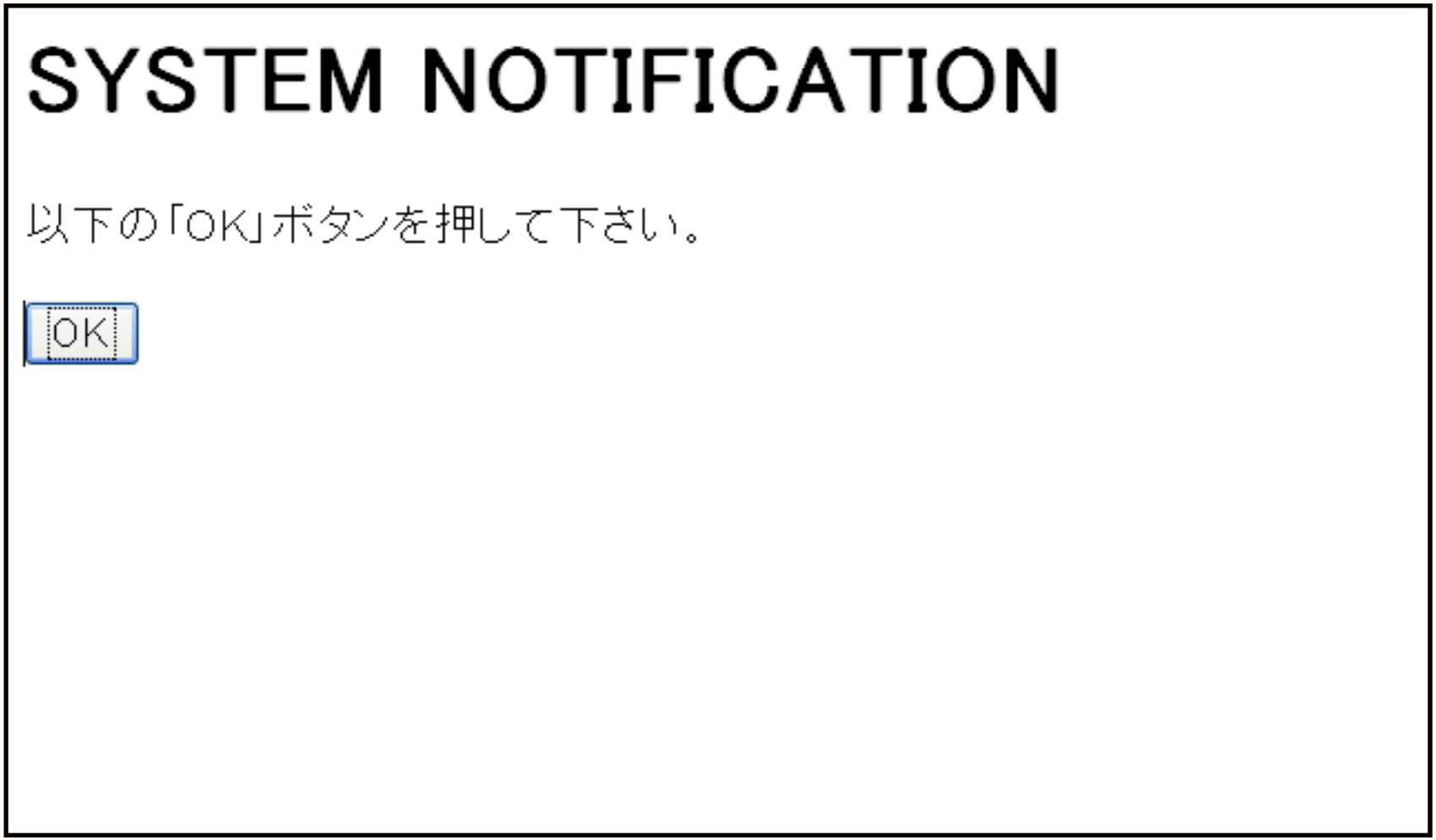
SYSTEM NOTIFICATION

リクエストされたパス/index.htmlに対応するテンプレート/index.htmlが見つかりませんでした。

テンプレートを作成する場合は「作成」ボタンを押して下さい。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja" lang="ja">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <title>サンプル</title>
</head>
<body>
  <h1>サンプル</h1>
  <p>Hello, <span tal:content="self/message">MESSAGE</span>!</p>
</body>
</html>
```

HTMLテンプレート生成画面(2)



- HTMLテンプレートが生成されたことが検出される
  - HTMLテンプレートからPageクラスを生成するための画面が表示される

Pageクラス生成画面

SYSTEM NOTIFICATION

テンプレート/index.htmlの変更を検出しました。テンプレートに関連する以下のクラスを追加・更新します。追加・更新したくないものについてはチェックを外して下さい。

クラスのプロパティ型、Pageの親クラス、Dtoに対応するEntityクラスには以下のものを指定することができます。

- プリミティブ型(例:boolean)
- パッケージ名つきクラス名(例:com.example.dto.AaaDto)
- 空文字列(java.lang.Objectを指定したのと同じになります)
- Genericタイプを指定した型名(例:java.util.List<java.lang.String>)
- パッケージ名を省略したクラス名。パッケージ名が指定されなかった場合は以下の規則にしたがってクラスが見つかるまで探索します。
  1. パッケージ名として「java.lang」「java.util」を補完
  2. 所属クラス(Pageの親クラスについてはPage、Dtoに対応するEntityクラスについてはDto、プロパティ型についてはプロパティを持つクラス)のパッケージ、その親パッケージ、…を補完
  3. ルートパッケージ以下のクラスのうち名前が同じもの(複数ある場合は最初に見つかったもの)

自動生成を行なう場合は「OK」ボタンを、今回は自動生成を行なわない場合は「スキップ」ボタンを、今回は自動生成を行なわずに今後もこのテンプレートを自動生成の対象外にする場合は「無視」ボタンを押して下さい。

OK スキップ 無視

追加されるクラス:

- ☑ com.example.web.IndexPage　親クラス:

  messageプロパティ String

OK スキップ 無視

Pageクラス生成画面(2)

SYSTEM NOTIFICATION

以下のクラスが追加されました:

- com.example.web.IndexPage

以下の「OK」ボタンを押して下さい。

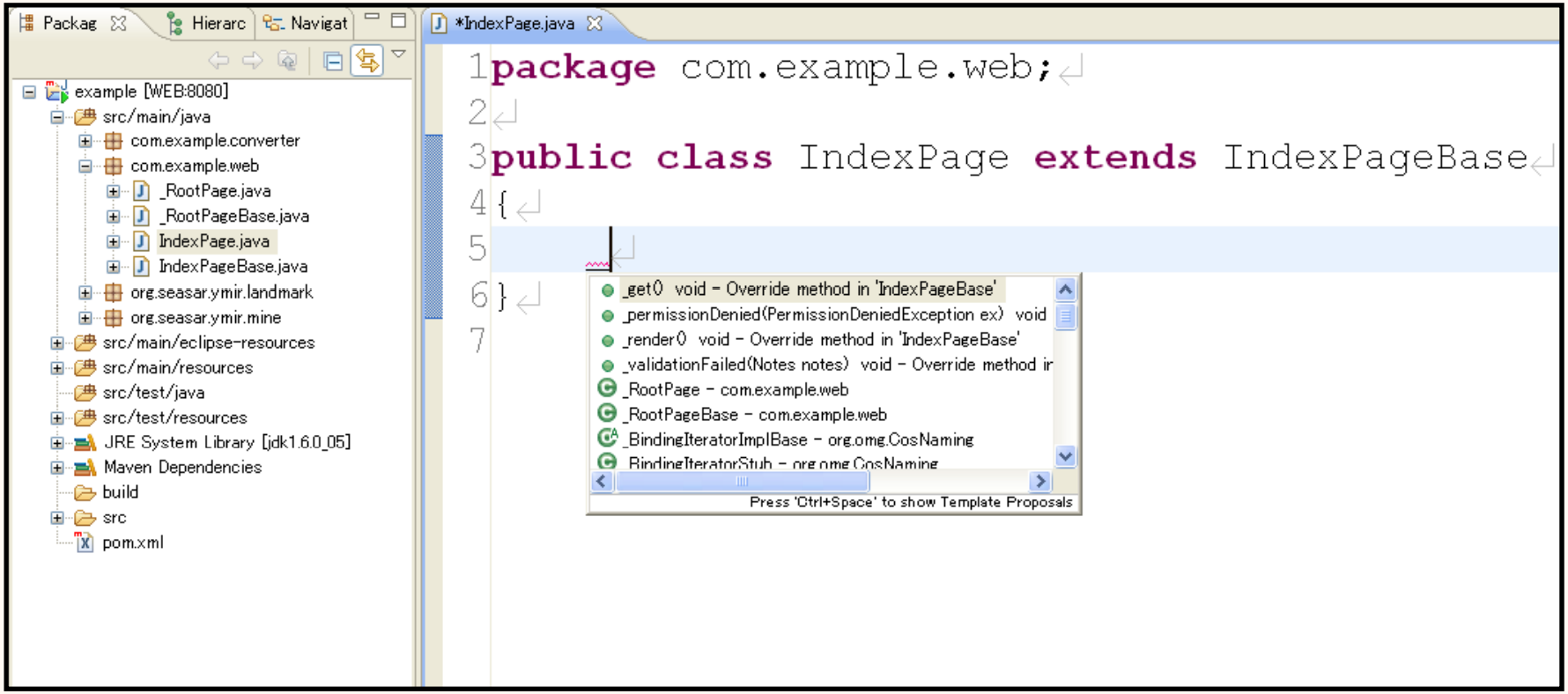
OK

次にあなたがすべき事の例:

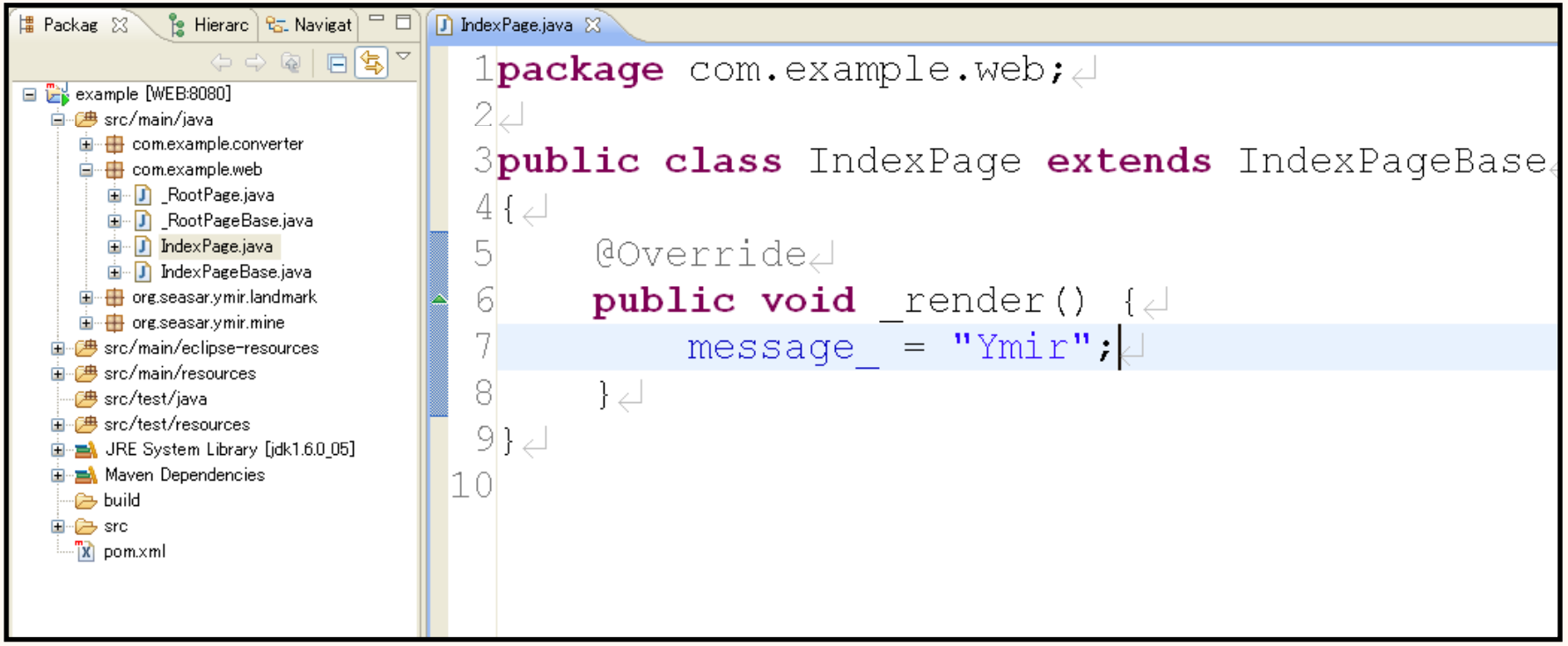
- IndexPageクラスのBaseクラスにリクエストを処理するための_getのようなメソッドが生成されています。これをGapクラスでオーバライドして処理を実装して下さい。(この時必要に応じてBaseクラスのメソッドの返り値の方を変更しても構いません。メソッドの返り値の型はBaseクラスが更新されても保持されます。)
- IndexPageクラスのBaseクラスに_renderメソッドが生成されている場合、これをGapクラスでオーバライドして処理を実装して下さい。_renderメソッドはテンプレートがレンダリングされる前に呼び出されるメソッドです。このメソッドの中でテンプレートを表示するために必要な各種プロパティの準備を行なうようにして下さい。

- Eclipse上でPageクラスを変更
- ブラウザで再度アクセス

Pageクラスの変更



Pageクラスの変更(2)



- WARファイルの生成
  - コマンドラインからmvn packageするだけ

新機能

- 最新バージョン(0.9.6)で実現された主な機能
  - クラスタリング対応
  - 複数ウィンドウ対応
  - アノテーションの別名定義
  - プラグイン機能
  - 自動生成機能の強化
    - Converterの自動生成機能
    - 自動生成機能のGenerics対応
    - その他自動生成機能周りの細かな改善

新機能

クラスタリング対応

- セッションレプリケーションが行なわれるクラスタリング環境でも動作
- セッションにバインドしたオブジェクトを外部で変更しても自動的に他ノードに通知
  - MapをHttpSession#getAttribute()して書き換えた後に通知のためにHttpSession#setAttribute(Map)する処理をフレームワークが自動実行

新機能

複数ウィンドウ対応

複数ウィンドウ対応

- フレーム使うタイプのアプリケーションや別ウィンドウを開くタイプのアプリケーションに対応
  - リクエストパラメータとしてウィンドウのIDを渡すことでウィンドウを区別
    - それぞれのウィンドウを開く際にウィンドウのIDをつけるようにする必要あり

新機能

アノテーションの別名定義

アノテーションの別名定義

- アノテーションに別名をつけたり束ねたりすることが可能
  - エイリアスアノテーション
  - コレクションアノテーション

- 単一のアノテーションに別名をつける

@In(scopeClass=RequestParameterScope.class)

@Param

エイリアスアノテーション(2)

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
@Alias
public @interface Param {
    In z_alias() default @In(scopeClass = RequestParameterScope.class);

    String name() default "";
}
```

@Alias エイリアスアノテーションであることを宣言する

@In(scopeClass = RequestParameterScope.class); エイリアスの内容を書く

String name() default ""; 可変にしたいプロパティを書く

- 可変プロパティも定義可能

@Param(name="paramName")

@In(scopeClass=RequestParameterScope.class, name="paramName")

- 複数のアノテーションを束ねる

```
@ In(scopeClass=SessionScope.class)
@ In(scopeClass=RequestScope.class)
```

```
@SessionOrRequest
```

コレクションアノテーション(2)

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
@Collection
public @interface SessionOrRequest {
    In[] value() default {@In(SessionScope.class), @In(RequestScope.class);
}
```

@Collection コレクションアノテーションであることを宣言する

内容を書く

新機能

自動生成機能の強化

- Converterの自動生成機能
  - S2Dxoの代わりに利用可能
    - リフレクションベースのS2Dxoだとデグレが心配、後でソースコードを見ても分かりにくい…という場合に利用
- 自動生成機能のGenerics対応
  - プロパティ型などに型パラメータつき型を指定可能
  - List → List<String>
- その他自動生成機能周りの細かな改善

ロードマップ

ロードマップ

- http://ymir.sandbox.seasar.org/roadmap
- 0.9.x系
  - 0.9.6リリース（2008年9月）
  - 0.9.6で最後。以降はメンテナンスリリース
- 1.0.x系
  - 廃止予定の機能を廃止など全般的な整理を行なう
  - 最初のバージョンのリリースは2008年10月頃を予定
- 1.1.x系（構想レベル）
  - 純粋なWebアプリフレームワーク化
    - 組み込み用途のための機能を取り除く
  - DIコンテナを差し替え可能に
  - HOT deploy機能の廃止
    - ハマりポイントなので…
    - リロードでも十分なのでは？
  - 最初のバージョンのリリースは2009年夏頃の予定

ご清聴
ありがとうございました

ZPT

- Zope Page Template
- Python（Zope）の世界で生まれたテンプレート言語
- 独自属性（TAL、METAL）でHTMLの書き換え指示を記述

ZPTの例

```
<html>
<body>
  <h1 tal:content="self/title">題名</h1>
  <ul tal:condition="self/messages">
    <li tal:repeat="message self/messages"
        tal:content="message">繰り返しメッセージ</li>
  </ul>
</body>
</html>
```

ZPTの属性

- **tal:attributes**…属性を置き換え
  -
    →
- **tal:content**…タグの中身を置き換え
  - 題名
    → *aaa*
- **tal:replace**…タグ全体を置き換え
  - 題名
    → *aaa*
- **tal:omit-tag**…値が真ならタグだけ除去
  - 内容
    → *内容*

ZPTの属性(2)

- **tal:repeat**…繰り返し
  - 中身
    → *名前1* *名前2* *名前3*

- **tal:condition**…値が真なら描画
  - エラーメッセージ
    → *実際のエラーメッセージ*
- **tal:define**…変数を定義
  -
    → *"a/b/c"の評価結果*

適用事例

- Kvasir/Soraのアプリケーションプラグイン
  - 管理ツール、イベント申し込み受付機能、…
- 業務アプリケーション
  - 某リソース予約サイト（B2C系、国際化対応あり）
    - 約50画面規模×4アプリ（携帯アプリ含）

あると便利なツール達

- 最低限用意したいもの
  - サーブレットコンテナ
  - Webブラウザ
  - Eclipse
  - Maven2
- あると便利なもの
  - ResourceSynchronizer
  - WebLauncher
  - m2eclipseプラグイン
  - Maven2 Additionalプラグイン

あると便利なツール達(2)

- サーブレットコンテナ
  - 動作確認のために必要
  - WebLauncher(後述)があれば不要
- Webブラウザ
  - 動作確認・自動生成のために必要
    - Eclipseの組み込みブラウザでもできなくはないが…
- Eclipse
  - 自動生成＋HOT deployを活用するには必須
    - コード修正→即自動ビルド
- Maven2
  - 開発中は不要だが手軽にWARを生成するために必要

- ResourceSynchronizer
  - Eclipseの管轄外で行なわれたリソースの変更に関する通知を受けて内部情報を同期してくれるEclipseプラグイン
    - http://eclipse.seasar.org/updates/3.3/
  - 自動生成後にEclipseのプロジェクトを手動で同期する必要がなくなる
- WebLauncher
  - Webアプリケーションをサーブレットコンテナに配備して動作させてくれるEclipseプラグイン
    - http://werkzeugkasten.googlecode.com/svn/trunk/werkzeugkasten.update/
  - Sysdeo Tomcatプラグインと違い1アプリ:1コンテナで起動してくれる
  - WTPより手軽

- m2eclipseプラグイン
  - EclipseにMaven2を統合するためのEclipseプラグイン
    - http://m2eclipse.codehaus.org/update/（0.0.12系）
    - 0.9.5系（http://m2eclipse.sonatype.org/update/）は若干不安定＆Maven2 Additionalプラグインと相性が悪いかも
  - JARの依存関係をpom.xmlで一元管理可能
- Maven2 Additionalプラグイン
  - 依存するJARをWEB-INF/libにコピーしてくれるEclipseプラグイン
    - http://www.skirnir.net/eclipse/updates/3.2/
  - WebLauncherとの組み合わせで威力を発揮