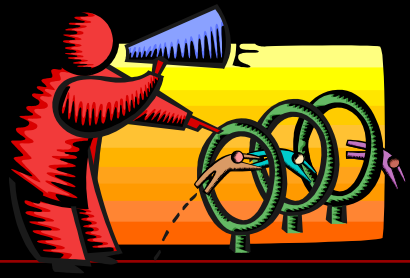
Strutsの最適利用法がわかった！

株式会社アークシステム
黒住　幸光

講師紹介



- 黒住幸光（くろずみゆきみつ）
  - 株式会社アークシステム　ITアーキテクト
  - Open Sourceを利用したJavaEE Applicationの構築とConsulting、企業システムの標準化策定コンサルティングを仕事とする
  - StrutsおよびOpen Source関連の執筆、セミナ講師多数
    - IDG JAPAN　JavaWorld誌 Jakarta活用指南 4年間連載
    - 日経 IT Pro　Jakarta/Apacheウォッチ 連載中
  - 趣味：飲酒

株式会社アークシステムのご紹介

- 1986年設立、社員約150人、CACグループ会社
- アークシステムでは早期よりOSSに取り組み、豊富なノウハウを活かし様々な活動を行っております
- サービスメニュー
  - ＯＳＳ導入Consulting,教育,各種セミナ
  - Web System構築Consultingおよび開発･運用
  - システム標準化コンサルティング
  - System基盤構築（汎用機からオープンシステムまで）
  - System運用（汎用機からオープンシステムまで）
- ご用命は
  - http://www.arksystems.co.jp （お問い合わせ） まで

アジェンダ

- Strutsは難しい？
- Strutsの今
- **最適利用法**
- Struts2とは

Strutsは難しい？

Strutsが難しいという声

- リクエスト処理関連
  - アクションマッピングが複雑になる
  - アクションフォームBeanとDTO間でデータ積み替えが多発する
  - …etc
- レスポンス生成関連
  - 画面仕様(HTML)からJSPを作るのが大変
  - カスタムタグの仕様が複雑で判りにくい
  - ページングの機能がない
  - コンテキスト管理が煩雑となる（保存キーのバッティングなど）
  - …etc
- そもそも機能不足

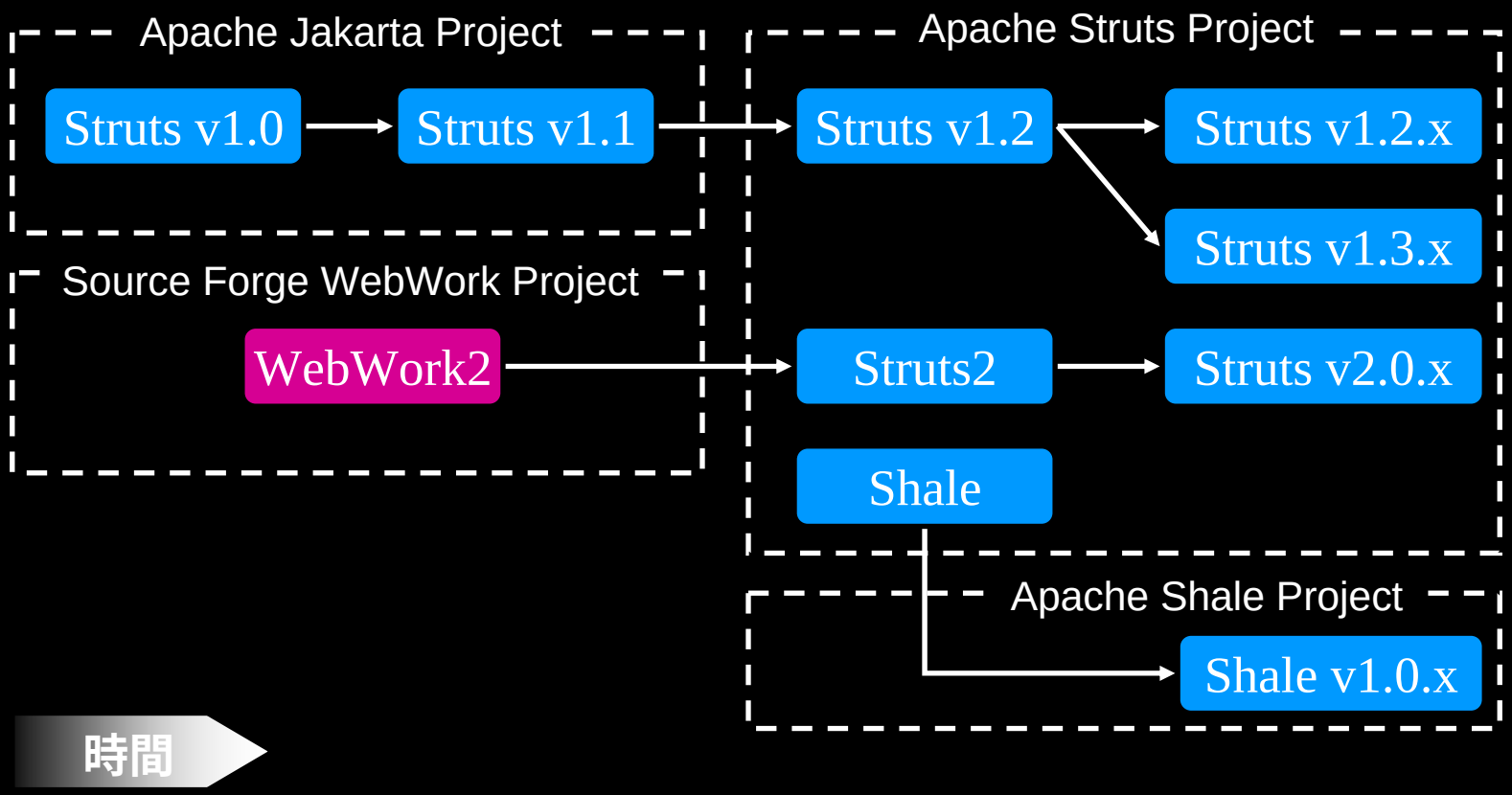
はい、難しいです

- 実装方式は1つではありません
  - 知れば知るほど、ベストな利用方法が知りたくなる
- Strutsはフレームワークのフレームワーク
  - 不足機能は補う必要があります
- システム全体のほんの一部
  - システムの全体アーキテクチャの考慮が必要
- Strutsカスタムタグは非推奨？
  - Struts1.3ではStrutsカスタムタグは非推奨です
- Strutsが沢山、どれを選べばよい？
  - このあと詳しく

以降で、解決策をご紹介！

Strutsの今

Strutsの系統図



Strutsの種類

- Shale
  - JSFの機能を強化するWebフレームワーク
  - 現在、TLPのShale Project にて開発が継続されている
  - 今日は無視
- Struts2 v2.0.x
  - WebWork2ベースのv1系とはまったく異なるフレームワーク
  - JavaSE 5.0、Servlet2.4/JSP 2.0以上に対応
  - まだ利用環境が未成熟
- Struts1 v 1.2.x
  - 元祖Strutsの流れを受け継ぐもの
  - Servlet2.2/JSP1.1以上に対応
  - メンテナンスのみが継続されている
- Struts1 v1.3.x
  - 内部構造をリファクタリングした正統派Struts
  - Servlet2.3/JSP1.2以上に対応
  - 現在の主力プロダクトの1つ

どっちのStrutsを使えばよい？

- 選択のポイント
  - アプリケーションサーバの対応バージョンのみ！

- Struts v1.3系を利用しましょう！
  - 便利な機能が盛りだくさん（マッピングの継承とか）
  - リクエストプロセッサの拡張が簡単にできる
  - APIに大きな変更なし（旧非推奨クラスが消滅したくらい）

最適利用法

Strutsが難しい、いくつかの理由

- チーム開発のための方法論が流通していない
- １つの機能の実装方式が複数存在する
- JSPの開発が難しい

設計や実装の方針(最適利用法)を決める事で、ある程度解決

誰が決めるか？

アーキテクトと呼ばれる人

Struts最適利用

- モジュール設定ファイル
- 入力検証方式
- アクションフォームBean
- アクションマッピングの定義
- 機能拡張
- トランザクション制御方式

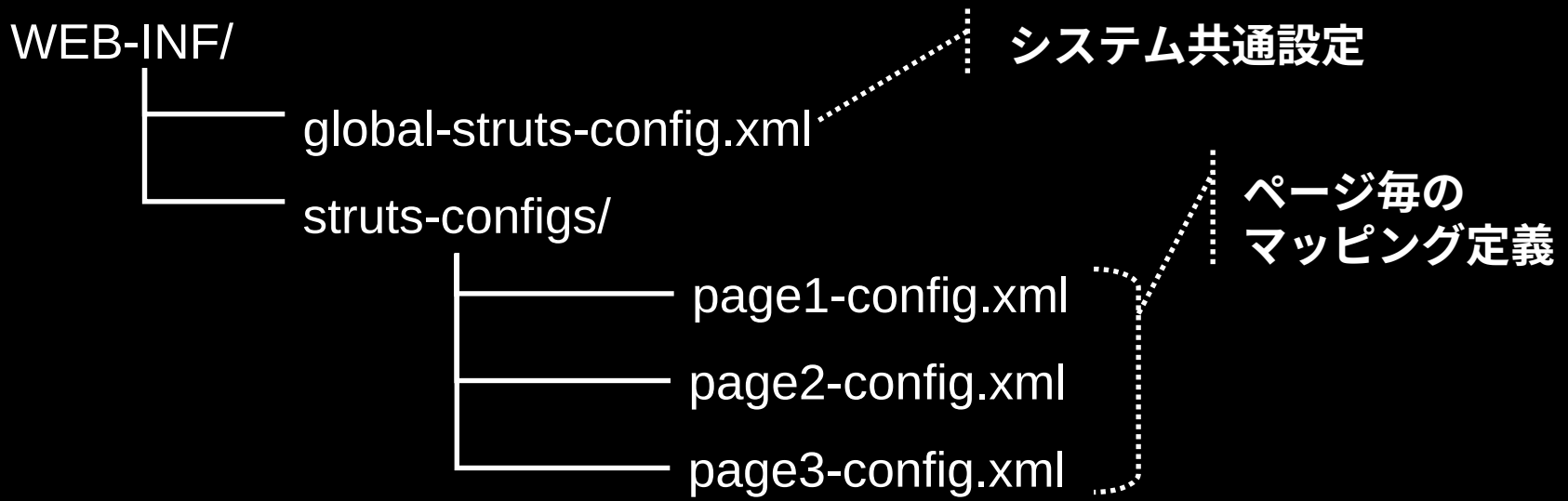
モジュール設定ファイルの定義方針

モジュール設定ファイルの定義方針

- チーム開発では、1Webページ1ファイル
- ＋モジュール共通定義 (プラグインなど)
- 各ファイルに定義する要素
  - アクションフォームBean
  - アクションマッピング
- このポリシによって、編集のコンフリクトやファイルの巨大化、メンテナンス性の低下が防げる

モジュール設定ファイルの定義方針

- 以下のようなファイル配置になる
- 設定ファイルのコンフリクトなし！(web.xmlに注意)



入力検証方式

複数の方法が存在する

- アクションフォームBeanに実装する
- Struts-Validatorを利用する
- アクションクラス内で実装する

Struts-Validatorとアクションクラス内実装の併用がおすすめ

なぜ併用なのか

- 基本はStruts-Validatorで全ての入力検証を実装する
- 足りない検証ルールは、追加ルールを開発
- しかし、表のサブミットなどで全てのエラーを検出して表示する必要がある場合に対応しきれない
  - → アクションで入力検証の一部を実装

アクションフォームBeanの選択

複数のアクションフォームBeanが存在する

- 静的アクションフォームBean(ActionForm)
- 宣言型(動的)アクションフォームBean(DynaActionForm)
- POJOのアクションフォームBean
- マップバックドアクションフォームBean

静的アクションフォームBean(ValidatorForm)がおすすめ

なぜ「一番シンプルな方法」なのか

- シンプルだからメンテナンス性が高い
- シンプルだから開発/デバッグが容易
- Struts-Validatorの不足検証（先ほどのアクションに実装する部分）をvalidateメソッド内に実装できる

アクションマッピングの定義方針

アクションマッピングの定義方針

- ページ内のアクションマッピングを定型化する
- 利用するアクションクラス、アクションフォームクラスも定型化する
- 複雑なページも簡単なページも同じパターン
- アクションマッピングに関わる全ての論理名に命名規則を定義

- 開発効率UP!　メンテナンス性向上！

アクションマッピングの定型化(1/2)

- ページに関するイベント(初期化/サブミット/リンククリック)毎に個別のアクションマッピング(Commandマッピング)を用意する
  - 例) /ページ名Eventイベント名.do
  - マッピングで呼び出すメソッドを定義できるMappingDispatchActionの継承クラスで実装
- 自ページを表示するJSPを呼び出す専用のアクションマッピング(Renderマッピング)を用意する
  - forwardマッピングで定義(アクションクラス不要)
- ページ内のイベント(サブミット/リンククリック)は1つのアクションマッピングが受け取る(EventDispatchマッピング)
  - 例) /ページ名.do
  - 呼び出すメソッドをJSPで指定できるEventDispatchActionの継承クラスで実装

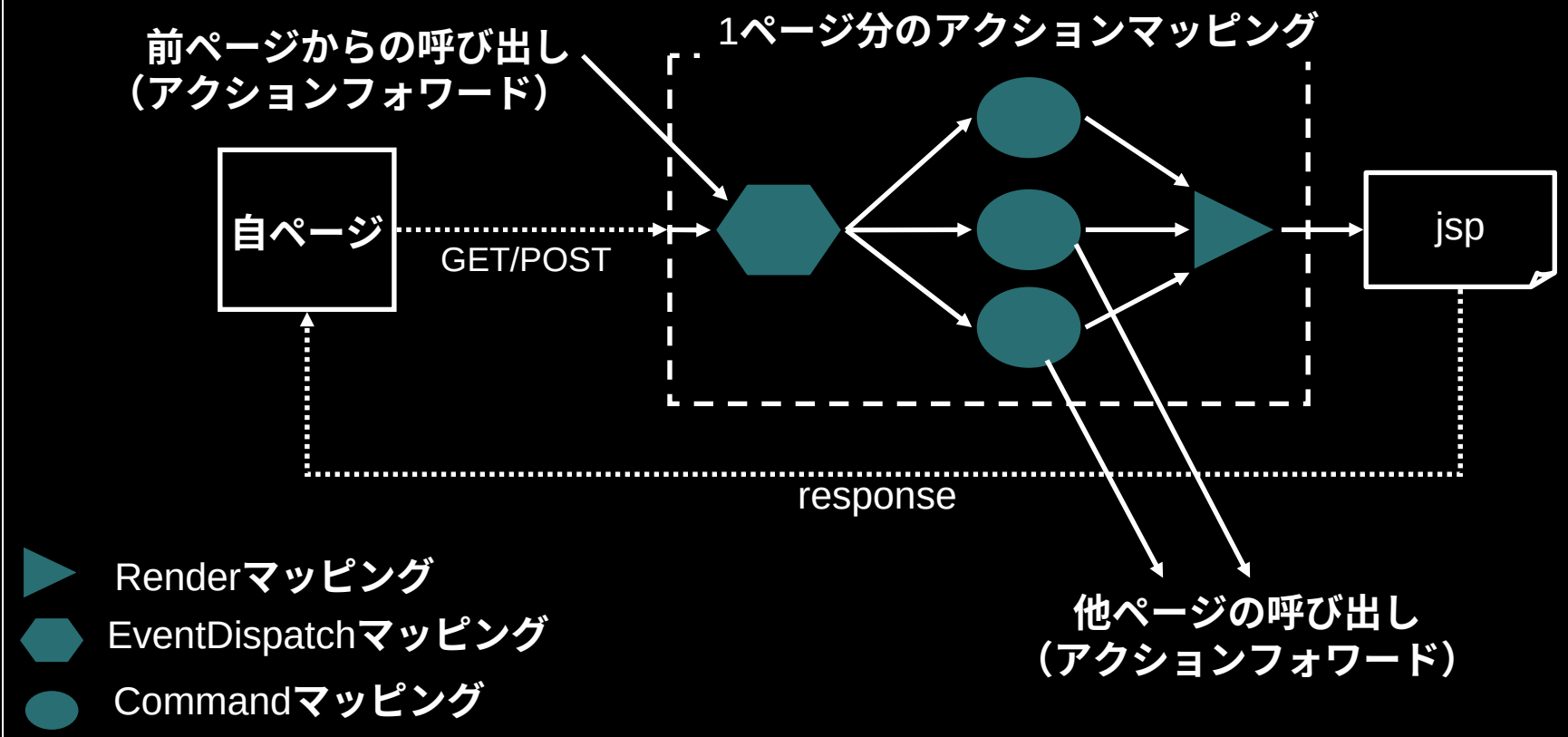
アクションマッピングの定型化(2/2)

- EventDispatchマッピングでは個別のイベントを処理するアクションマッピングにforwardするのみ
  - 例)

```
<forward name="success"  path="/page1EventButton1.do" />
```

- Commandマッピング実行後、自ページを表示する場合はRenderマッピングへforwardする
- Commandマッピング実行後、別ページを表示する場合は別ページのEventDispatchマッピングへforwardする

定型パターンでのマッピングイメージ



このパターンの利点

- ボタン毎に入力検証ルールが変更できる
- JSP名は1箇所にのみ定義
- JSPに呼び出す処理名を指定できる
- 別ページ遷移時の相手先マッピング名が自明
- ページの初期表示用処理が用意されている
- 初期表示用処理とリクエスト処理が分離
- 深く考えなくても、マッピングが設計できる！

モジュール設定ファイルの定義例

```
<action-mappings>
   <action path="//page1BaseMapping" type="sample.Page1MappingDispatchAction"
         name="page1Form" scope="request" validate="true" />
   <!-- EventDispatchマッピング -->
   <action path="/page1"   extends="sample.Page1EventDispatchAction"
         name="page1Form" scope="request" validate="false" />
      <forward name="unspecified" path="/page1EventInit.do"/>
      <forward name="button1" path="/page1EventButton1.do"/>
      <forward name="button2" path="/page1EventButton2.do"/>
   </action>
   <!-- Render マッピング -->
   <action path="/page1PageRender"  extends="//page1BaseMapping«
            prefix="$$" forward="/pages/page1.jsp"/>
   つづく・・・
```

モジュール設定ファイルの定義例

```
    <!-- Command マッピング -->
    <action path="/page1EventInit" extends="//page1BaseMapping"
          parameter="init" >
       <forward name="success" path="/page1PageRender.do"/>
    </action>
    <action path="/page1EventButton1" extends="//page1BaseMapping"
          parameter="button1" >
       <forward name="success" path="/menu.do"/>
    </action>
    <action path="/page1EventButton2" extends="//page1BaseMapping"
          parameter="button2" >
       <forward name="error"   path="/page1PageRender.do"/>
       <forward name="success" path="/page2.do"/>
    </action>
  </action-mappings>
```

このパターンのコツ

- EventDispachマッピングのクラスはページ毎に作らないように、汎用のクラスを作る
- アクションフォームBean名やスコープ、検証の有無の属性が冗長にならぬよう、継承マッピングを用いる

機能拡張

機能拡張の方法

- 全てのアクションに同じ機能（セキュリティチェックやログなど）を追加する手段はない
- アクション制御下以外でも処理を追加するにはリクエストプロセッサを変更するのが最適
- Struts1.3では「Chain of Responsibility」パターンを採用しており、簡単に処理が差し込める
- BaseActionはやめましょう（Struts1.3では同名が既にある）

トランザクション制御方式

トランザクション制御方式

- アクションマッピングを分離すると、EJBやAOPを用いたトランザクション制御では境界が複数できる？

NO!　JTAで解決

JTAを積極的に利用

- JTAにてトランザクションの開始と終了を明示的に制御する
- JTAの制御はリクエスト処理の始めの方で実施（アクション呼び出しよりも、もっと前）
- おすすめはサーブレットフィルタ内

Struts最適利用まとめ

最適利用のコツ

- 定型パターン化
- 有識者による方針決め
- Strutsの提供機能を詳しく理解する

おすすめ ^^) →

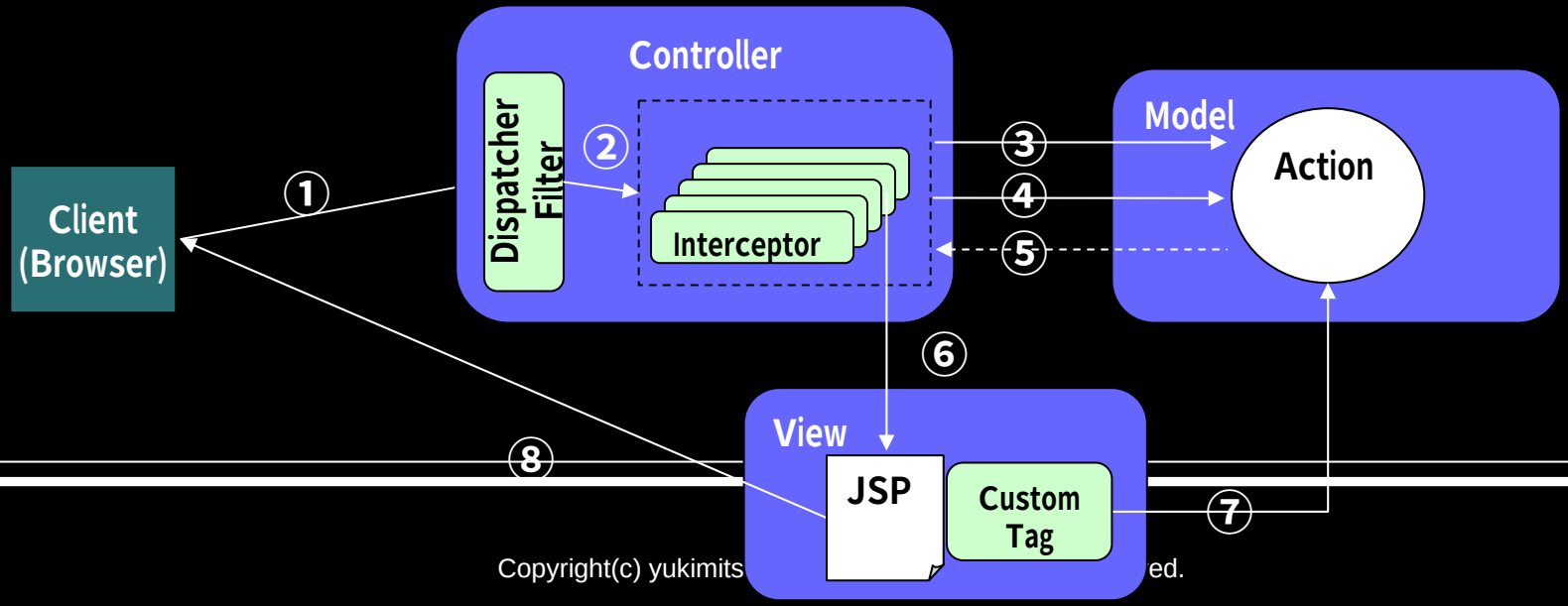
Struts2

Struts2(WebWork)

- MVCモデル2アーキテクチャを採用したフレームワーク
- Struts1とは異なるコードベース（元WebWork）
- コントローラにはServletFilterを利用
- 各処理はInterceptorという複数のクラスで処理されてゆく
- Struts1の弱点を考慮している
- 長所
  - Servlet API 非依存（全てをWebコンテナ非依存で開発可能）
  - アクションとアクションフォームを1つのクラスで実現可能
- 短所
  - 専用のカスタムタグを利用
  - 利用ノウハウ、情報が不足

Struts2アプリケーションの動作概要

①全てのHTPPリクエストはDispatcher Filterがうけとる
②Dispatcher FilterがInterceptor群を呼び出す
③Interceptorがリクエストパラメータをアクションにセットする
④Interceptorがアクションを実行する
⑤アクションの処理が終了した場合、次に実施する処理を指定
⑥レスポンスを生成するためのJSPの実行
⑦⑤でアクションが保持している結果をカスタムタグで参照
⑧HTTPレスポンスの返却



Struts2の JSP

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ page pageEncoding="Windows-31J" %>
<%@ taglib uri="/struts-tags" prefix="s" %>
<html>
<head>
    <title>こんにちは</title>
</head>
<body>
<h2>はじめましてSturts2です</h2>
<s:actionerror />
<s:form theme="simple">
   ユーザ名：<s:textfield key="username" /><br>
   パスワード：<s:password  key="password" />
 <p>
   <s:submit value="Login"        action="Top_login" />
   <s:submit value="GuestLogin"   action="Top_guestLogin"/>
 </p>
</s:form>
</body>
</html>
```

アクションマッピングの定義

- アクション名と、アクションを実装したクラス（アクションクラス）の対応定義
- アクションの処理結果(OUTCOME)と次の処理を定義

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
   "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
   "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
   <package name="example" namespace="/example" extends="struts-default">
      <action name="Top" class="example.Top">
         <result name="success" >/example/Top.jsp</result>
      </action>

      <action name="Top_login"  method="login" class="example.Top" >
         <result name="input"   >/example/Top.jsp</result>
      </action>
   </package>
</struts>
```

アクションクラスのコード

- POJOで作成する
- イベント処理用のメソッドと、フォームの値を保持する属性を記述する

```
public class Top {

    private String username;
    private String password;

    public String getPassword() { return password; }
    public void setPassword(String password) {  this.password = password;  }
    public String getUsername() { return username; }
    public void setUsername(String username) {  this.username = username;  }

    public String login() throws Exception {
       // ログイン処理
        if( 成功 ) {  return “success”; }
       return "input";
    }
}
```

Struts1 vs Struts2

アクションサーブレット	Dispatcher Filterが同等の機能を提供
リクエストプロセッサ	インターセプタ群が同等の機能を提供
struts-config.xml	struts.xml に同等の定義を記述、またはアノテーション
アクションマッピング	ほぼ同じ(XMLの記述が異なる)
アクションクラス	POJOで記述できる
アクションフォームビーン	アクションクラスに集約された
アクションフォワード	ほぼ同じ(outcomeとして文字列で扱う)
Sturtsカスタムタグ	まったく別のカスタムタグのセットが用意され、OGNLが利用できる
メッセージリソース	ほぼ同じ(ファイルを分割定義できる)
メッセージ操作	ほぼ同じ (アクションクラスでのAPIが異なる)
入力検証	ほぼ同じ (XML記述やAPIが異なる)
Session/Requestコンテキスト	直接アクセスする事は、ほぼない

Thank You