



2007/11/6 Cross Community Conference 2007 Fall  
コミュニティパネル「Webアプリケーション開発の今後を占う」


# SPRING FRAMEWORK

[HTTP://WWW.SPRINGFRAMEWORK.ORG/](http://www.springframework.org/)

ARCLAMP.JP YUSUKE



# Agenda

- Springとは
  - SpringによるWebアプリ開発
  - Webアプリ開発の今後
- 

# Springとは

- Springとは
  - ビジネス・オブジェクトをマネージメントするためのフレームワーク
    - ビジネス・オブジェクトを統合するための機能を提供 (IoC、DI、AOP…)
    - 統合するための設定を管理することができる
  - Spring Portfolioのコア
    - Springを前提としたアプリケーション・フレームワーク群

# (参考) Spring Portfolioとは

- Spring Web Flow
- Spring Web Services
- Spring Security (Acegi Security)
- Spring LDAP
- Spring Rich Client
- Spring Extensions (Modules)
- Spring IDE
- Spring BeanDoc
- Spring Dynamic Modules for OSGi(tm) Service Platforms
- Spring JavaConfig
- Spring Batch

# Springとは

- ベスト・プラクティスなプログラミング・スタイルに最適
  - POJOベース
  - インターフェース・ベース
  - IoC “Don't call me, I'll call you.”
  - コードレベルでの依存性削減
  - 単体テストの利用
  - 無用なシングルトンの排除

# Springとは

- Glueとして機能
  - JavaEE APIへの橋渡し
  - 様々なフレームワークと連携可能
  - 既存技術を使いやすく、シンプルに
- 車輪の再発明をしない
  - ログ、コネクションプール、トランザクション・マネージメント、データベース・アクセス…
  - すべて既存のコンポーネントによって提供される
    - 例えばO/Rマップは提供しない。TopLink, Hibernate, JPA, JDOでも、好きなものを使えばいい

# Springとは

- 規約よりも設定重要
  - 明示された設定によりオブジェクトを管理
  - 設定を容易にするための様々なパターンや規約を用意
  
- 小さなお知らせ：次のリリースはSpring2.1ではなくて、Spring2.5と呼ぶことになりました。

# (参考) カスタムXML

- XML名前空間を利用して、設定をパターン化
  - 自分で拡張可能

```
<bean id="dataSource"  
class="org.springframework.jndi.JndiObjectFactoryBean">  
  <property name="jndiName" value="jdbc/jpetsore" />  
</bean>
```



```
<jee:jndi-lookup id="dataSource" jndi-  
name="jdbc/jpetstore"/>
```

# (参考) Spring2.5

- XML属性のprefixとsuffixをルール化
  - “p:”がプロパティ
  - “-ref”が<ref>を代替

```
<bean id="exampleDataAccessObject"  
class="example.ExampleDataAccessObject"  
p:dataSource-ref="myDataSource" />
```

# (参考) Spring2.5

- アノテーション対応

```
public class JdbcOrderRepositoryImpl
    implements OrderRepository {
    @Autowired
    @Qualifier("myDataSource")
    private DataSource orderDataSource; // ...
}
```

# (参考) Spring2.5

- アノテーションと組み合わせたXML設定

```
<beans ...>
```

```
  <context:component-scan
```

```
    base-package="org.example">
```

```
    <context:include-filter type="regex"
```

```
      expression=".*Stub.*Repository"/>
```

```
    <context:exclude-filter type="annotation"
```

```
      expression="org.springframework.stereotype.Re  
pository"/>
```

```
  </context:component-scan>
```

# Springとは

- その他の機能
  - JDBCを簡易的に扱えるフレームワーク
    - データアクセス例外の抽象化
  - 様々なO/Rマッパとの統合
  - トランザクション・マネージメント
  - AOP
  - 様々なMVCウェブ・フレームワークとの統合
  - UnitTestサポート

# Springとは

- 僕が思うに
  - あなたのコードはそのままで。
    - +JavaEE、+フレームワークを実現
    - 規定する部分は、なるべく標準仕様で
  - あなたの規約はそのままで。
    - ロジックと組み立ての分離
    - 組み立て方を多様性にして、多様なフレームワークを受け入れる

# Springとは

- 例えばRails + Spring
  - Railsのルールはそのままで、Java、JavaEE、AOPのパワーを提供
  - すべてのJavaEE機能を利用可能。SOAだって、RMIだって、分散トランザクションだってOK
- デモ用にアプリを作ってみた
  - 意味があるかとかは、あとのパネルで

# デモ : JRuby on Rails + Spring

- 作り方

- Warblerを利用

- <http://caldersphere.rubyforge.org/warbler/>

- RailsアプリケーションをWARに

- Goldspikeを利用

- <http://rubyforge.org/projects/jruby-extras>

- 中の人がSpringモジュールを提供しているが使わず

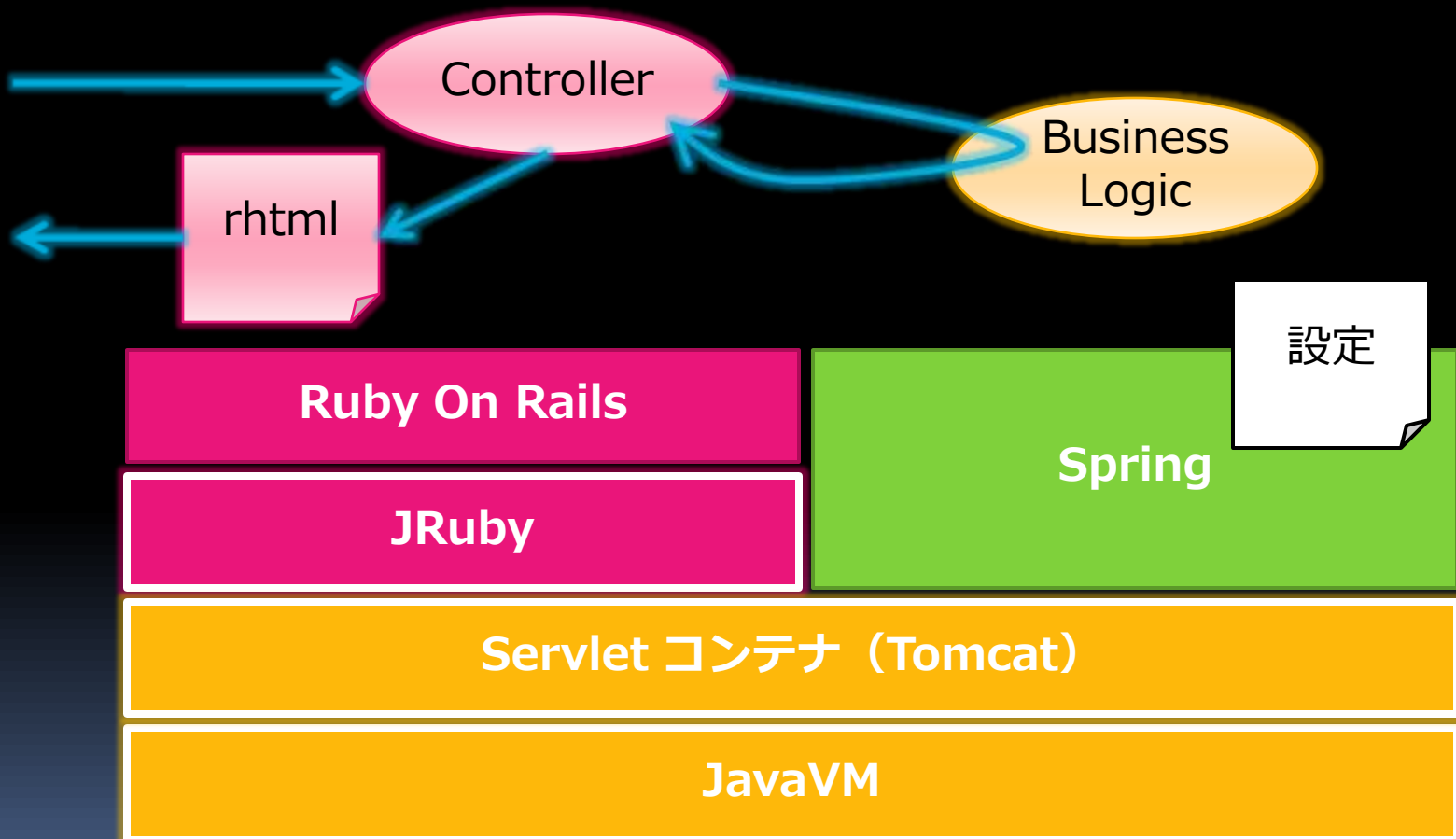
- <http://mysterycoder.blogspot.com/2007/06/spring-jruby.html>

- Springの起動はweb.xmlの範囲で

- 呼び出し方がダサいので、あとで高井さんに相談

- thanks to よういちろう

# デモ : JRuby on Rails + Spring



# デモ : JRuby on Rails + Spring

- Javaのビジネスロジック

```
public class BusinessLogicImpl
    implements BusinessLogic {
    private String message;
    public void setMessage(String message) {
        this.message = message;
    }
    public String showMessage() {
        return message;
    }
}
```

# デモ : JRuby on Rails + Spring

- Springの設定ファイル

```
<beans xmlns="...  
  <bean id="mybl" class="...BusinessLogicImpl" >  
    <property name="message"  
      value="Hello Spring" />  
  </bean>  
</beans>
```

# デモ : JRuby on Rails + Spring

- Web.xml

```
<web-app>
```

```
...
```

```
<listener>
```

```
  <listener-class>
```

```
    org.jruby.webapp.RailsContextListener
```

```
  </listener-class>
```

```
</listener>
```

```
<listener>
```

```
  <listener-class>
```

```
    org.springframework.web.context.ContextLoaderListener
```

```
  </listener-class>
```

```
</listener>
```

# デモ : JRuby on Rails + Spring

- Railsのコントローラー

```
class CccController < ApplicationController
  include Java
  import ...WebApplicationContextUtils
  def index
    app_context =
      WebApplicationContextUtils.getWebApplicationCon
      text($servlet_context)
    @mybl = app_context.getBean 'mybl'
  end
end
```

# デモ : JRuby on Rails + Spring

- Railsのrhtml

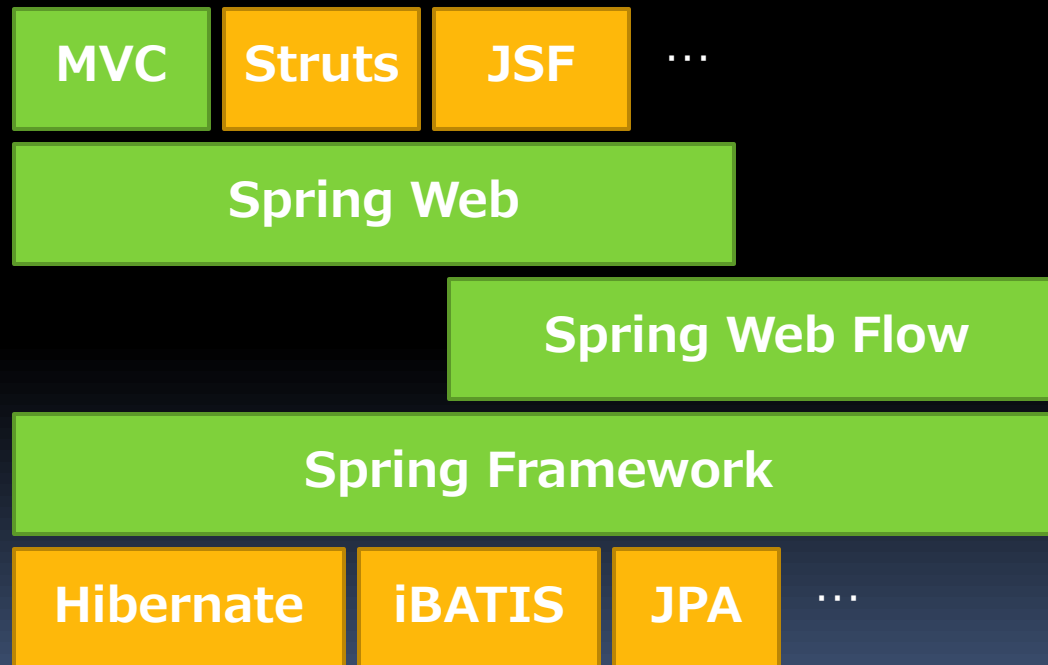
```
<%= @mybl.showMessage %>
```

# SpringによるWebアプリ開発

- そんなSpringが考えたWebアプリ開発
  - 設定にできるところは徹底的に設定で
  - Webアプリの基本的なパターンを取り込み
    - MVC
    - フローとカンバセーション・スコープ
  - データアクセスはお好きに

# SpringによるWebアプリ開発

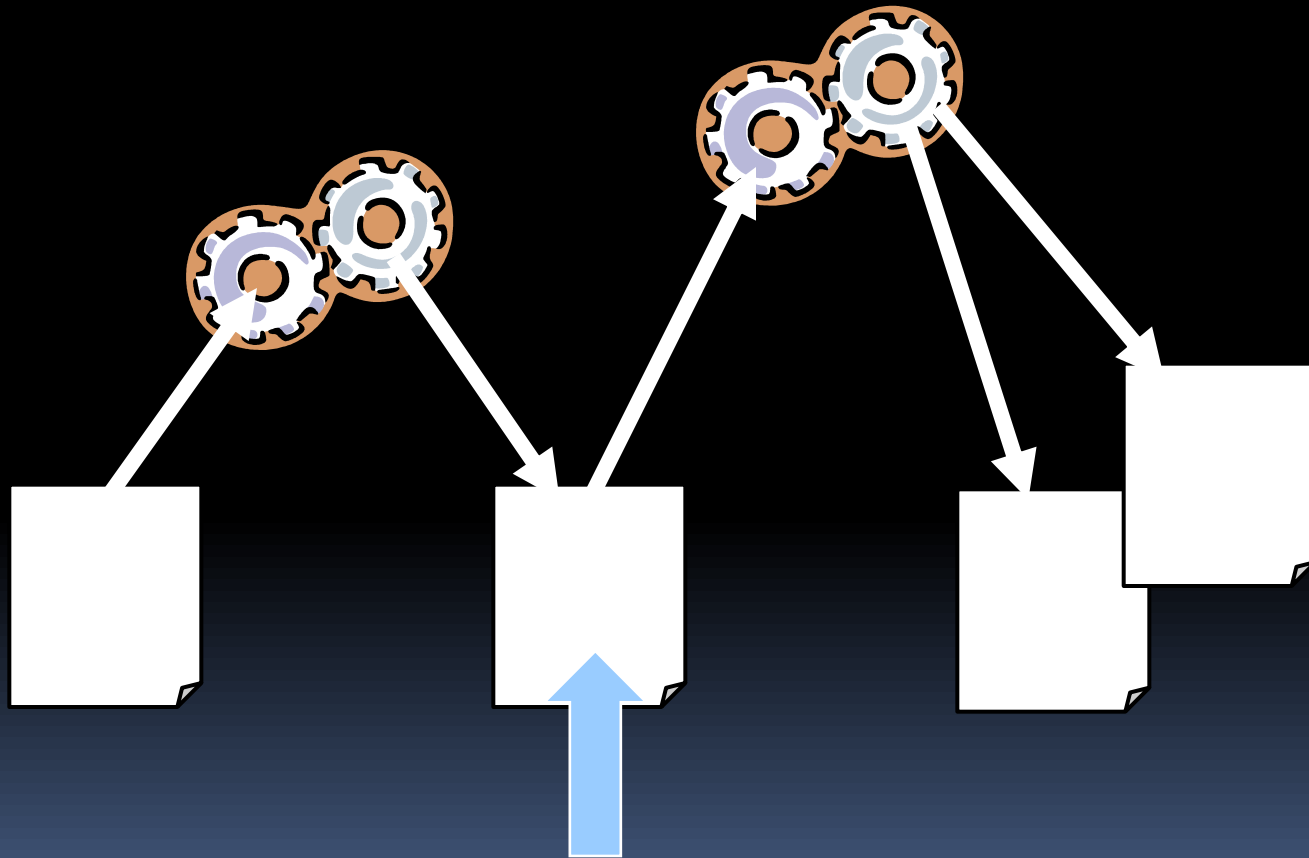
- 標準的なPortfolio



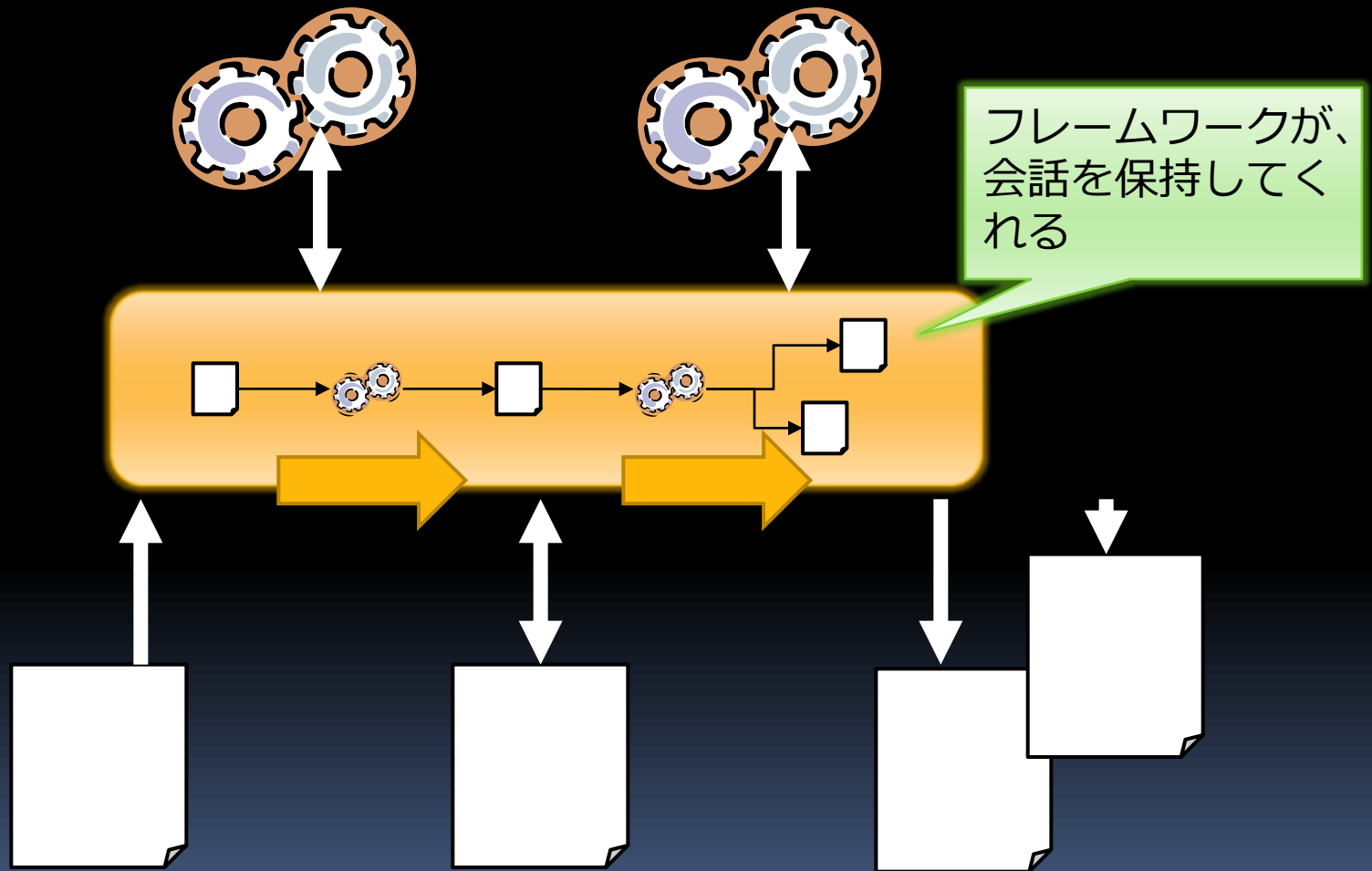
# SpringによるWebアプリ開発

- Spring Web Flow
  - Webアプリケーションの画面遷移のルールを各ページやサービスから分離することができるフレームワーク
  - <http://www.springframework.org/webflow>
  - Spring Web MVC、JSF、Strutsと統合して利用できる
  - 資料 : thanks to Kaz Kawamura

# SWF : 従来のWebアプリ



# SWF : SWFによるWebアプリ



# SWF : Flowが利用するBean

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
  xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">
  <bean id="loginFormAction"

    class="org.springframework.webflow.action.FormAction">
    <property name="formObjectName" value="loginBean"/>
    <property name="formObjectClass"
value="sample.bean.LoginBean"/>
    <property name="formObjectScope" value="FLOW"/>
    <property name="validator">
      <bean class="sample.validator.LoginValidator"/>
    </property>
  </bean>
</beans>
```

# SWF : フローの定義 (1/3)

```
<?xml version="1.0" encoding="UTF-8"?>
<flow
  xmlns="http://www.springframework.org/schema/webflow"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    http://www.springframework.org/schema/webflow
    http://www.springframework.org/schema/webflow/spring-webflow-1.0.xsd">
<start-state idref="login"/>

<view-state id="login" view="login">
  <entry-actions>
    <action bean="loginFormAction" method="setupForm"/>
  </entry-actions>
  <transition on="login" to="processLogin">
    <action bean="loginFormAction" method="bindAndValidate"/>
  </transition>
  <transition on="goodbye" to="goodbye"/>
</view-state>
```

# SWF : フローの定義 (2/3)

```
<action-state id="processLogin">
  <bean-action bean="loginService" method="login">
    <method-arguments>
      <argument expression="${flowScope.loginBean.loginname}"/>
      <argument expression="${flowScope.loginBean.password}"/>
    </method-arguments>
  </bean-action>
  <transition on="yes" to="success"/>
  <transition on="no" to="fail"/>
</action-state>
```

# SWF : フローの定義 (3/3)

```
<view-state id="success" view="success">  
  <transition on="logout" to="goodbye"/>  
</view-state>
```

```
<view-state id="fail" view="fail">  
  <transition on="retry" to="login"/>  
</view-state>
```

```
<end-state id="goodbye" view="goodbye"/>
```

```
<import resource="login-beans.xml"/>  
</flow>
```

# SWF : ページの変更 (フォーム)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head> ...
</head>
<body>
<form:form commandName="loginBean">
  ...
  <input type="hidden" name="_flowExecutionKey" value="{flowExecutionKey}"/>
  <table>
    ...
    <tr>
      <td colspan="2">
        <input type="submit" name="_eventId_login" value="Login">
        <input type="submit" name="_eventId_goodbye" value="Good Bye">
      </td>
    </tr>
  </table>
</form:form>
</body>
</html>
```

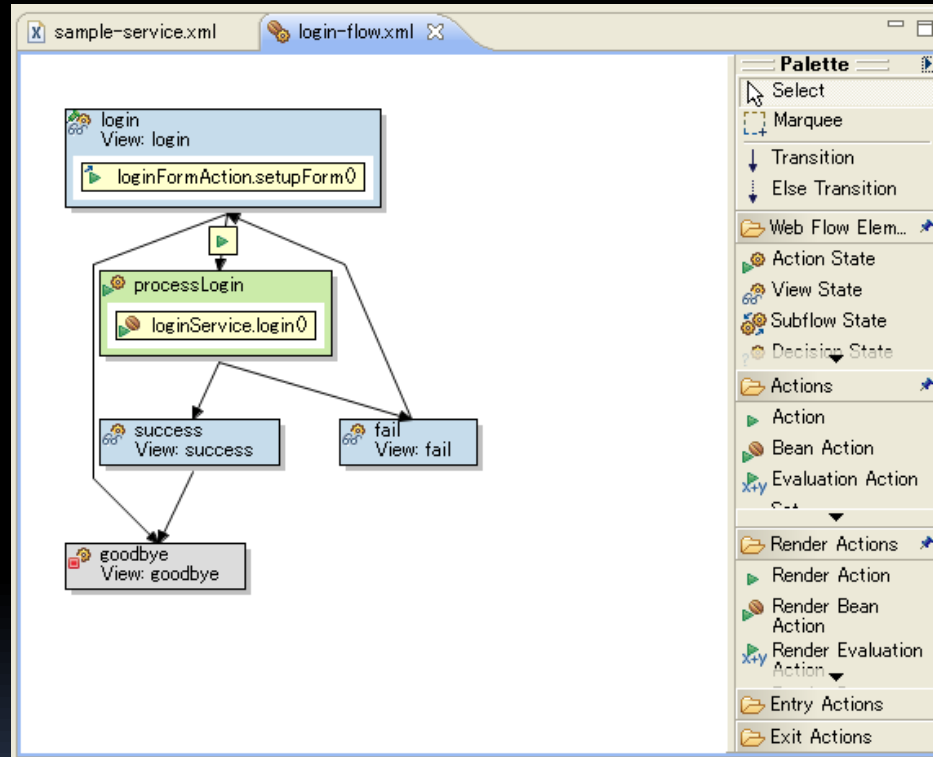
# SWF : ページの変更 (リンク)

```
<%@ page language="java" contentType="text/html; charset=Shift_JIS"
    pageEncoding="Shift_JIS"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=Shift_JIS">
    <title>Login Failed</title>
</head>
<body>
    <h1>Login failed</h1>
    <p>
        Please <a href=
            "login.flow?_flowExecutionKey=${flowExecutionKey}&_eventId=retry"
            >try again!</a>
    </p>
</body>
</html>
```

# SWF : フローのはじめ方

```
<html>
<head>
  <title>Redirect</title>
  <meta http-equiv="Pragma" content="no-cache">
  <meta http-equiv="refresh"
    CONTENT="0;URL=login.flow?_flowId=login-flow">
  <meta http-equiv="Cache-Control" content="no-cache">
</head>
<body>
</body>
</html>
```

# SWF : フローエディタ



# Webアプリ開発の今後

- パラメタ的な設定は重要
  - 裏返せばパターン化
  - 暗黙的な設定としての規約も使えるが、ほどほどに
  - 自分の規約は取り込んでいく
- 中間層のビジネス・オブジェクトはマネージメントする
  - ロジックは型付きで記述するJavaの特性
  - 構造的に管理されるのがよい (=XML)
- IDE重要
  - Spring Tool Suite整備中

# Webアプリ開発の今後

- システム間、システム内を統合する手段を使い分ける
  - システムは全体で意味を持つ
  - 統合手段は使い分けられるほうが便利。抽象化層としてのSpring
  - + JavaEEを活用する
- 多様性を受け入れる
  - 1つの方法が正しいわけではない
  - でも、統合は必要なのでコンテナが必要

# おわり

- JJUG (日本Springユーザー会)
  - <http://www.springframework.jp/>