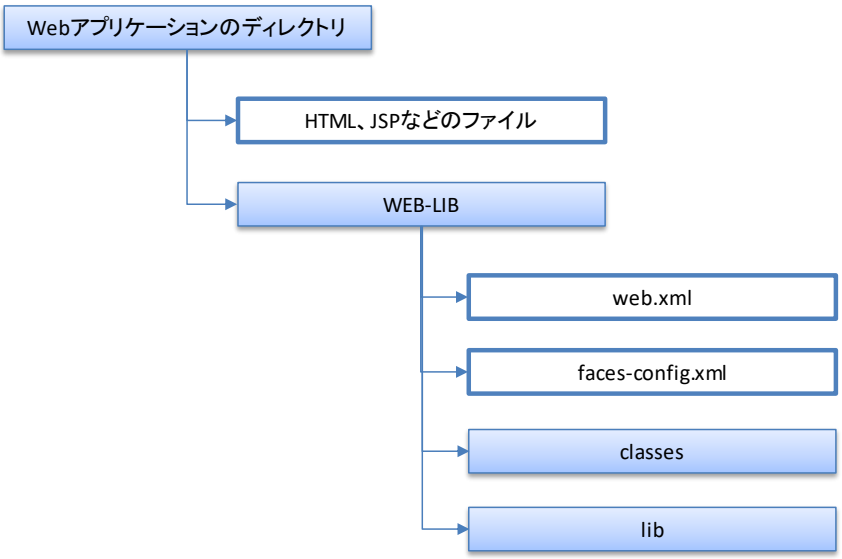
目次

JSF を利用する

JSF における処理の流れ

① Faces サーブレットへの送信
② UI コンポーネントの構築と JSF 内部での処理
③ 値の検証・型変換
④ Bean への受け渡し
⑤ イベント処理
⑥ 送信用 UI コンポーネントツリーの作成
⑦ GUI のレンダリング
⑧ クライアントへの HTML 送信

ファイル構成



以下のライブラリが必要

- jsf-api.jar
- jsf-impl.jar
- jstl-1.2.jar

表示用 JSP の作成

▼Sample01.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>

<html>
<head>
	<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
	<title>Sample01</title>
</head>
<body>
	<f:view>
		<h:form>
			<h:outputText value="What's your name?" binding="#{sample01Bean.text }" />
			<br/>
			<h:inputText size="20" binding="#{sample01Bean.field}" />
			<br/>
			<h:commandButton value="OK" action="#{sample01Bean.action }" binding="#{sample01Bean.button}" />
		</h:form>
	</f:view>
</body>
</html>
```

▼タグ一覧

<f:view>タグ	画面へ表示する要素を view タグ内に記載する
<h:outputText>	表示テキストのタグ
<h:inputText>	入力フィールドのタグ
<h:commondButton>	プッシュボタンのタグ

▼属性一覧

value	表示するテキストを指定
binding	Bean との結びつけの役割
action	ボタンをクリックしたときに発生するイベントで実行する処理との紐づけの役割

管理 Bean クラスの作成

▼SampleBean.java

```
package sample.sample01;


import javax.faces.component.html.HtmlCommandButton;
import javax.faces.component.html.HtmlInputText;
import javax.faces.component.html.HtmlOutputText;


public class SampleBean {
	private HtmlCommandButton button;
	private HtmlOutputText text;
	private HtmlInputText field;


	public String action(){
		String str = (String)field.getValue();
		text.setValue("こんにちは、" + str + "さん");
		return "";
	}


	/* setter, getter */
	public HtmlCommandButton getButton() {
		return button;
	}
	public void setButton(HtmlCommandButton button) {
		this.button = button;
	}
	public HtmlOutputText getText() {
		return text;
	}
	public void setText(HtmlOutputText text) {
		this.text = text;
	}
	public HtmlInputText getField() {
		return field;
	}
	public void setField(HtmlInputText field) {
		this.field = field;
	}
}
```

▼使用するクラス一覧

HtmlCommandButton	<h:commandButton>タグによって構築される、プッシュボタンの UI コンポーネント
HtmlOutputText	<h:outputText>タグによって構築される、テキスト表示のための UI コンポーネント
HtmlInputText	<h:inputText>タグによって構築される、テキスト入力のための UI コンポーネント

▼HtmlOutputText, HtmlInputText のメソッド

Object getValue()	値の取得
void setValue()	値の設定

faces-config.xml の作成

JSF の設定ファイル

→ここでは用意した管理 Bean クラスの登録

▼faces-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>


<faces-config
        xmlns="http://java.sun.com/xml/ns/java33"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
        http://java.sun.com/xml/ns/javaee/web-facesconfig_1_2.xsd"
        version="1.2">


        <managed-bean>
                <managed-bean-name>sample01Bean</managed-bean-name>
                <managed-bean-class>sample.sample01.SampleBean</managed-bean-class>
                <managed-bean-scope>session</managed-bean-scope>
        </managed-bean>
</faces-config>
```

▼タグ一覧

<managed-bean-name>JSP での名前</managed-bean-name>

<managed-bean-class>クラスの指定</managed-bean-class>

<managed-bean-scope>変数のスコープ</managed-bean-scope>

web.xml の作成

▼web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
          xmlns="http://java.sun.com/xml/ns/javaee"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
          http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" >
          <display-name>jsf</display-name>
          <servlet>
                    <servlet-name>Faces Servlet</servlet-name>
                    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
                    <load-on-startup>1</load-on-startup>
          </servlet>
          <servlet-mapping>
                    <servlet-name>Faces Servlet</servlet-name>
                    <url-pattern>*.jsf</url-pattern>
          </servlet-mapping>
          <welcome-file-list>
                    <welcome-file>index.html</welcome-file>
          </welcome-file-list>
</web-app>
```

▼サーブレットの指定

```
<servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
```

JSF の UI コンポーネント・タグ

テキスト入力関連タグ

▼<h:inputText>タグ

<textarea>に変換される。cols, rows で縦横の文字列を指定できる。

▼<h:inputSecret>タグ

<input type=”password”>に変換される。

入力された文字列はすべてアスタリスクで表示される。

▼<h:inputHidden>タグ

<input type=”hidden”>に変換される。

チェックボックスとラジオボタン

▼<h:selectBooleanCheckbox>タグ

<input type=”checkbox”>に変換される。

テキストは次のような要素を利用する。　<h:outputLabel>ラベル</h:outputLabel>

値は Boolean で受け取る。

▼<h:selectManyCheckbox>タグ

複数のチェックボックスを表示するためのタグ。

値は String[]で受け取る。

<f:selectItem>と合わせて使用する。

```
<h:selectManyCheckbox binding="#{sample03Bean.manyCheckbox}">
        <f:selectItem itemLabel="Item1" itemValue="item1" />
        <f:selectItem itemLabel="Item2" itemValue="item2" />
</h:selectManyCheckbox>
```

リストの利用

▼<h:selectOneListbox>タグ、<h:selectManyListbox>タグ

一つまたは複数項目が選択可能なリストボックスを表示。

size 属性で表示する項目数を設定できる。

<f:selectItem>タグで選択項目を記述する。

▼<h:selectOneMenu>タグ、<h:selectManyMenu>タグ

選択可能なコンボボックスを表示。

<f:selectItem>タグで選択項目を記述する。

▼sample04.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>

<html>
<head>
	<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
	<title>Sample03</title>
</head>
<body>
	<f:view>
		<h:form>
			<span>▼listbox</span><br>
			<h:selectOneListbox size="1" binding="#{sample04Bean.list}">
				<f:selectItem itemLabel="label1" itemValue="item1" />
				<f:selectItem itemLabel="label2" itemValue="item2" />
				<f:selectItem itemLabel="label3" itemValue="item3" />
			</h:selectOneListbox><br>
			<br>
			<span>▼selectManyListBox</span><br>
			<h:selectManyListbox size="3" binding="#{sample04Bean.manyList}">
				<f:selectItem itemLabel="label1" itemValue="item1" />
				<f:selectItem itemLabel="label2" itemValue="item2" />
				<f:selectItem itemLabel="label3" itemValue="item3" />
			</h:selectManyListbox><br>
			<br>
			<span>▼selectOneMenu</span><br>
			<h:selectOneMenu binding="#{sample04Bean.menu}">
				<f:selectItem itemLabel="label1" itemValue="item1" />
				<f:selectItem itemLabel="label2" itemValue="item2" />
				<f:selectItem itemLabel="label3" itemValue="item3" />
```

```
                        </h:selectOneMenu><br>
                        <br>
                        <span>▼selectManyMenu</span><br>
                        <h:selectManyMenu binding="#{sample04Bean.manyMenu}">
                                <f:selectItem itemLabel="label1" itemValue="item1" />
                                <f:selectItem itemLabel="label2" itemValue="item2" />
                                <f:selectItem itemLabel="label3" itemValue="item3" />
                        </h:selectManyMenu><br>
                        <br>
                        <h:commandButton binding="#{sample04Bean.button }" action="#{sample04Bean.action }"
value="送信" /><br>
                        <br>
                        <span>▼outputText</span><br>
                        <h:outputText binding="#{sample04Bean.out }"/>
                </h:form>
        </f:view>
</body>
</html>
```

リンク関係タグ

▼<h:outputLink>タグ

ハイパーテキストの機能を提供するためのタグ。

value 属性に遷移先の URL を記載する。

<a>タグに変換される。

リンク部分のテキストは<h:outputText>タグなどを用意。

▼<h:commandLink>タグ

アクションを実行するためのタグ。

action 属性に管理 Bean のメソッドをバインド。

リンク部分のテキストは<h:outputText>タグなどを用意。

その他の主なタグ

※使用例は後述

▼<h:graphicImage>タグ

イメージを表示するためのタグ。

「utl」または「value」属性でイメージファイルを指定する。

▼<h:message><h:messages>タグ

メッセージを表示するためのタグ。

for 属性で関連するコンポーネントの id 属性値を指定する。

▼<h:panelGroup>タグ

複数のコンポーネントをまとめて扱うためのタグ。

▼<h:panelGrid>タグ

複数のコンポーネントを整列して配置し扱うためのタグ。

Columns 属性で列数を指定する。

コンポーネントの利用

コンポーネントの表示スタイル

コンポーネントにスタイスシートを適用したい場合について

▼sample06.jsp

```
<span>▼埋め込み</span><br>
<h:outputText value="text" style="color: #f05"/><br>
<span>▼クラスを指定</span><br>
<h:outputText value="text" styleClass="outtext"/><br>
```

style	CSS を直接記述
styleClass	class 属性を設定し、CSS は別途記述

コンポーネントの状態変更イベント

▼sample07.jsp

```
<f:view>
    <h:form>
        <span>▼出力テキスト</span><br>
        <h:outputText binding="#{sample07Bean.out }"/><br>
        <span>▼入力欄（イベント監視）</span><br>
        <h:inputText binding="#{sample07Bean.text }" value="defalt"
            valueChangeListener="#{sample07Bean.textValueChange }"/><br>
    </h:form>
</f:view>
```

▼イベント監視

valueChangeListener	binding 属性と同様の記述方法で、 管理 Bean のイベント発生時に呼び出すメソッドを指定

▼sample.sample07.SampleBean

```
public void textValueChange(ValueChangeEvent event){
    String oldValue = (String)event.getOldValue();
    String newValue = (String)event.getNewValue();
    out.setValue("oldValue: " + oldValue + ", newValue: " + newValue);
}
```

▼ValueChangeEvent クラス

主なメソッド

Object getSource()	発生した UI コンポーネントを取得
Object getOldValue()	変更される前の value を取得
Object getNewValue()	変更後の新しい value を取得

イベントの強制送信

▼sample08.jsp

```
<h:selectManyListbox valueChangeListener="#{sample08Bean.textValueChange }"
                                                    onclick="this.form.submit();">
```

▼イベント一覧

onblur	マウスが離れたとき
onchange	フォームの内容が変更された時
onclick	クリックされた時
ondbclick	ダブルクリックされた時
onfocus	フォーカスされた時
onkeydown	キーが押し下げられたとき
onkeypress	キーが押し下げたままの状態のとき
onkeyup	キーが離された時
onmousedown	マウスボタンが押し下げられたとき
onmousemove	マウスポインタが移動しているとき
onmouseout	マウスポインタがアンカーから離れたとき
onmouseover	マウスポインタがアンカーに重なったとき
onmouseup	マウスボタンが離された時
onselect	テキストが選択された時
onreset	フォームがリセットされた時
onsubmit	フォームが送信された時

HTML タグを利用した表示の作成

▼sample09.jsp

```
<h:outputText binding="#{sample09Bean.out }" escape="false"/>
```

→escape 属性を false に設定することでタグがエスケープされなくなる

ナビゲーション・コンバータ・バリデータ

ナビゲーションの利用

▼faces-config.xml

```
    <navigation-rule>
        <from-view-id>/jsp/sample/sample10_1.jsp</from-view-id>
        <navigation-case>
            <from-outcome>next</from-outcome>
            <to-view-id>/jsp/sample/sample10_2.jsp</to-view-id>
        </navigation-case>
    </navigation-rule>
```

▼タグ一覧

<navigation-rule>	ひとまとまりのページ遷移ルール
<from-view-id>	遷移元のページ URL を示す情報 ※省略可能で、省略するとすべての遷移元に対してルールが適用される
<navigation-case>	遷移先に関する情報を記述
<from-outcome>	ナビゲーションの名前を記述
<to-view-id>	遷移先の URL を記述

ナビゲーションを JSF タグに設定する

▼sample10_1.jsp

```
<h:commandButton value="submit" action="next" />
```

→設定ファイルでしていしたナビゲーション名を指定するだけ

▼sample10.SampleBean

```
    public String action(){
        return "next";
    }
```

→Java クラスのメソッドでも使用可能

コンバータの利用

→コンバータを利用することで値の型変換（String→？）を行ってくれる。

▼faces-config.xml

```
        <converter>
                <converter-id>intConverter</converter-id>
                <converter-class>javax.faces.convert.IntegerConverter</converter-class>
        </converter>
```

→IntegerConverter の内部で NumberFormatException 例外が投げられるか監視している

▼使用タグ一覧

<converter-id>	コンバータの識別名
<converter-class>	コンバータのクラス

▼sample11.jsp

```
<h:inputText binding="#{sample11Bean.text }" converter="intConverter"/>
```

→converter 属性に、設定ファイルで指定した識別名を記述する

▼コンバータクラス一覧

BooleanConverter	真偽値の入力
ByteConverter	バイトデータの入力
CharacterConverter	文字データの入力
DateTimeConverter	日付の値の入力
DoubleConverter	double 値の入力
FloatConverter	float 値の入力
IntegerConverter	int 値の入力
LongConverter	long 値の入力
NumberConverter	数値全般の入力
ShortConverter	short 値の入力

バリデータの利用

▼faces-config.xml

```
        <validator>
                <validator-id>longRange</validator-id>
                <validator-class>javax.faces.validator.LongRangeValidator</validator-class>
        </validator>
```

▼使用タグ一覧

<validator-id>	バリデータの識別名を指定
<validator-class>	バリデータのクラスを指定

▼sample12.jsp

```
<h:inputText binding="#{sample12Bean.text }">
        <f:validator validatorId="longRange" />
        <f:validateLongRange maximum="10" minimum="0" />
</h:inputText>
```

→validator 属性でバリデータを指定できるが、通常はその他に値の指定が必要

▼バリデータクラス一覧

LongRangeValidator	整数の値の範囲を指定するバリデータ。 値の上限と下限を指定できる。
DoubleRangeValidator	実数の値の範囲を指定するバリデータ。 値の上限と下限を指定できる。
LengthValidator	入力テキストの長さを指定するバリデータ。 最大の文字数を指定できる。

データベースと DataTable

DataTable の基本

用語集

UI

ユーザインタフェース。

コンピュータとその利用者の間で情報のやりとりをするためのインタフェース。