Apache Wicket 6.x Hands-On

# WicketでWebページを作ってみる

## コンポーネントとモデルの使い方

① org.wicket\_sapporo.handson パッケージに、以下のファイルを作成する

・HomePage.html

**<!DOCTYPE html>**

<html xmlns:wicket="http://wicket.apache.org">

<head>

<meta charset="UTF-8" />

<title>HomePage</title>

</head>

<body>

<p wicket:id="label1">message is here.</p>

</body>

</html>

・HomePage.java

**package** org.wicket\_sapporo.handson**;**

**import** org.apache.wicket.markup.html.WebPage**;**

**import** org.apache.wicket.markup.html.basic.Label**;**

**import** org.apache.wicket.model.IModel**;**

**import** org.apache.wicket.model.Model**;**

**public** **class** **HomePage** **extends** WebPage **{**

**private** **static** **final** **long** serialVersionUID **=** 1L**;**

**public** **HomePage()** **{**

String message **=** "こんにちは。今日はWicketハンズオンです。"**;**

IModel**<**String**>** label1Model **=** **new** Model**<>(**message**);**

Label label1 **=** **new** Label**(**"label1"**,** label1Model**);**

add**(**label1**);**

**}**

**}**

アプリケーションを起動して、ブラウザでhttp://localhost:8080/wicket\_handson/ を表示し、動作を確認する。

「こんにちは。今日は Wicket ハンズオンです。」というメッセージが表示されればOK。

例：



## Formコンポーネントの例

**入力フォームがあるページを作ってみる**

① org.wicket\_sapporo.handson.basicパッケージに以下の2つのファイルを作る。

・FormPage.html（PDF参照）

・FormPage.java（PDF参照）

② HomePage.htmlのBodyタグ内に以下を追加する。

<dl>

<dt>基本編</dt>

<dd><a wicket:id="toFormPage">FormPageへ</a></dd>

</dl>

③ HomePage.javaのコンストラクタに以下を追加する。

Link**<**Void**>** toFormPageLink **=** **new** Link**<**Void**>(**"toFormPage"**)** **{**

**private** **static** **final** **long** serialVersionUID **=** 1L**;**

@Override

**public** **void** **onClick()** **{**

setResponsePage**(new** FormPage**());**

**}**

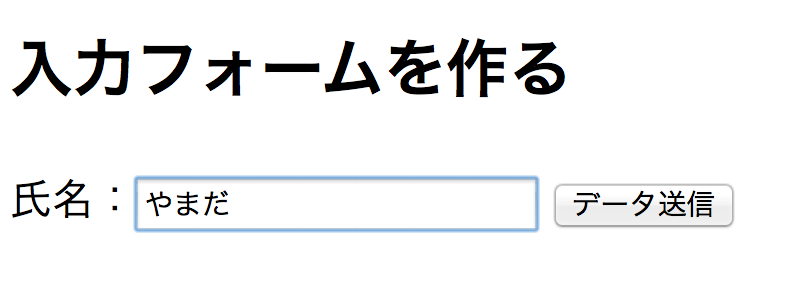
**};**

add**(**toFormPageLink**);**

ブラウザでhttp://localhost:8080/wicket\_handson/ から FormPage に移動して動作を確認する。

フォームに入力して送信した文字列がコンソールに表示されればOK。

例：



↓　データ送信ボタンをクリック

SSD:Users:jyukai:Desktop:スクリーンショット 2013-12-03 18.51.17.png

**入力フォームを増やしてみる**

① FormPage.html の <input type=”text”… の下に、以下のコードを追加する。

<div wicket:id="lunch"></div>

*// lunche の値を格納するModel*

**private** IModel**<**String**>** lunchModel**;**

② FormPage.javaのフィールド変数に以下のコードを追加する。

③ FormPage.javaのコンストラクタに以下のコードを追加する。

lunchModel 変数の初期化はnameModelの初期化の直後に、それ以外はTextField コンポーネントの下へ。

lunchModel **=** **new** Model**<>(**""**);**

*// ラジオボタンの選択肢を準備*

List**<**String**>** lunches **=** Arrays**.**asList**(**"鶏唐揚げ定食"**,** "鳥かつ定食"**,** "鳥ガーリック定食"**);**

*// 選択肢である lunches を格納するModel. List オブジェクト用には ListModel を使う*

IModel**<**List**<**String**>>** lunchesModel **=** **new** ListModel**<>(**lunches**);**

*// lunches から一つを選択する radio ボタン用のコンポーネント*

RadioChoice**<**String**>** radioChoice **=** **new** RadioChoice**<>(**"lunch"**,** lunchModel**,** lunchesModel**);**

form**.**add**(**radioChoice**);**

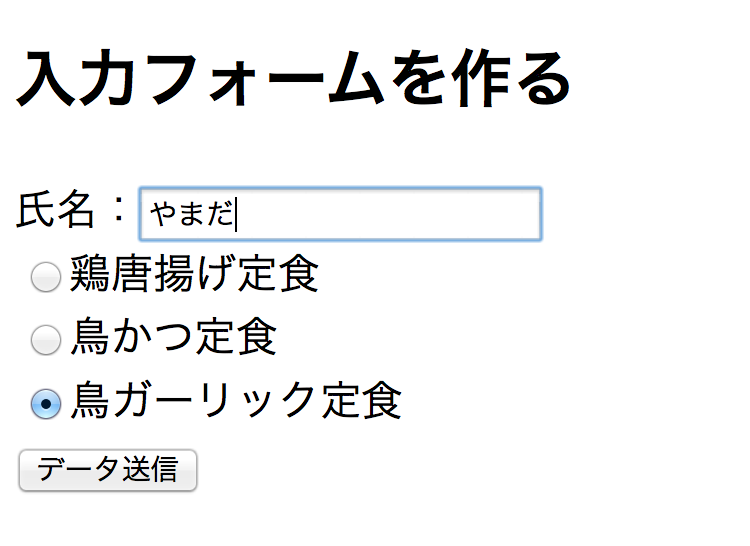
④ FormPage.javaのFormのonSubmit メソッドの内部に、以下のコードを追加する。

System**.**out**.**println**(**"launch : " **+** lunchModel**.**getObject**());**

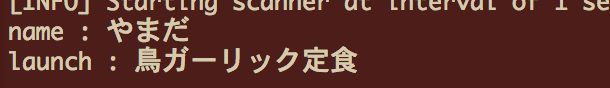
ブラウザで http://localhost:8080/wicket\_handson/ から FormPage に移動し、動作を確認する。

ラジオボタンが増え、フォームから送信した入力フォームとラジオボタンの文字列がそれぞれコンソールに選択表示されればOK。

例：



↓　データ送信ボタンをクリック



**確認ページを作ってみる**

① org.wicket\_sapporo.handson.basicパッケージに以下の2つのファイルを作る。

・ConfirmationPage.html

**<!DOCTYPE html>**

<html xmlns:wicket="http://wicket.apache.org">

<head>

<meta charset="UTF-8" />

<title>ConfirmationPage</title>

</head>

<body>

<h2>Formの投稿結果を表示する</h2>

<p wicket:id="name"></p>

<p wicket:id="lunch"></p>

</body>

</html>

・ConfirmationPage.java

**【練習】 nameModel, lunchModelの内容が表示されるページになるように修正せよ。**

**package** org.wicket\_sapporo.handson**.**basic**;**

**import** org.apache.wicket.markup.html.WebPage**;**

**import** org.apache.wicket.model.IModel**;**

**public** **class** **ConfirmationPage** **extends** WebPage **{**

**private** **static** **final** **long** serialVersionUID **=** 1L**;**

**public** **ConfirmationPage(**IModel**<**String**>** nameModel**,** IModel**<**String**>** lunchModel**)** **{**

**}**

**}**

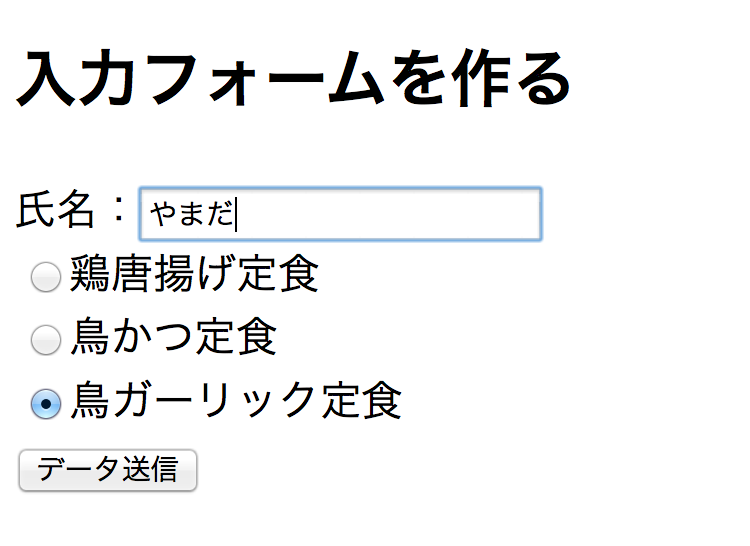
② FormPage.java の Form の onSubmit メソッド内に、以下のコードを追加する。

setResponsePage**(new** ConfirmationPage**(**nameModel**,** lunchModel**));**

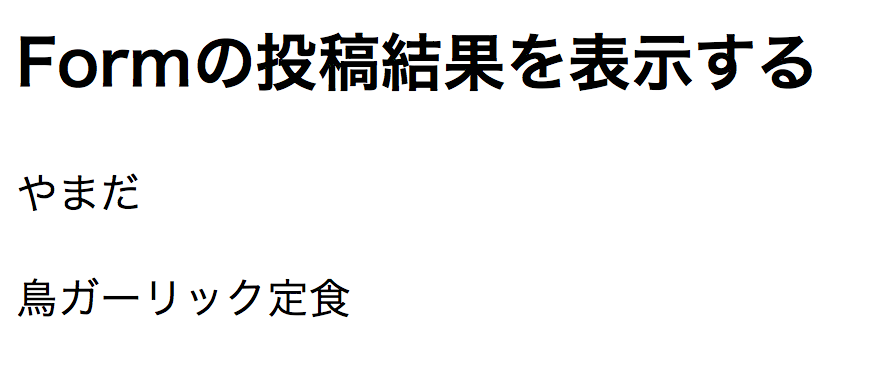
ブラウザでhttp://localhost:8080/wicket\_handson/ からFormPageに移動し、動作を確認する。

入力フォームから送信すると画面遷移し、送信した値が表示されればOK。

例：



↓　データ送信ボタンをクリック



## ListViewコンポーネントの例

**データのリストアップ的なページを作ってみる**

① org.wicket\_sapporo.handson.listViewパッケージに以下の2つのファイルを作る。

・ListViewPage.html（PDF参照）

・ListViewPage.java（PDF参照）

② HomePage.htmlのdlタグ内に以下を追加する。

<dd><a wicket:id="toListViewPage">ListViewPageへ</a></dd>

③ HomePage.javaのコンストラクタに以下を追加する。

Link**<**Void**>** toListViewPageLink **=** **new** Link**<**Void**>(**"toListViewPage"**)** **{**

**private** **static** **final** **long** serialVersionUID **=** 1L**;**

@Override

**public** **void** **onClick()** **{**

setResponsePage**(new** ListViewPage**());**

**}**

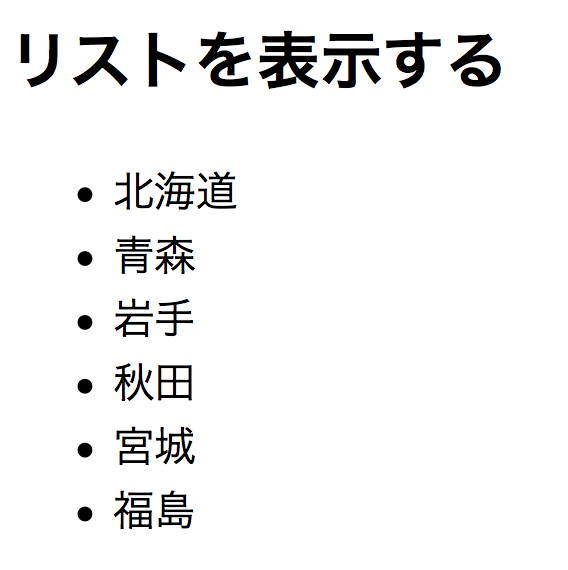
**};**

add**(**toListViewPageLink**);**

ブラウザでhttp://localhost:8080/wicket\_handson/ からListViewPageに移動し、動作を確認する。

北海道〜福島の県名が縦にリストで表示されれば成功。

例：



**ListViewでテーブル（表）を作ってみる**

① org.wicket\_sapporo.handson.listViewパッケージに以下の2つのファイルを作る。

* ListViewTablePage.html（PDF参照）
* ListViewTablePage.java

**【練習】 getUsersメソッドの結果がHTMLに併せて表示されるように修正せよ。**

**package** org.wicket\_sapporo.handson**.**listview**;**

**import** java.util.ArrayList**;**

**import** java.util.List**;**

**import** org.wicket\_sapporo.handson.beans.User**;**

**import** org.apache.wicket.markup.html.WebPage**;**

**public** **class** **ListViewTablePage** **extends** WebPage **{**

**private** **static** **final** **long** serialVersionUID **=** 1L**;**

**public** **ListViewTablePage()** **{**

**}**

*// データベースなどから取得してきた体で*

**public** List**<**User**>** **getUsers()** **{**

List**<**User**>** list **=** **new** ArrayList**<>(**4**);**

list**.**add**(new** User**(**"宮林　椋太"**,** 20**));**

list**.**add**(new** User**(**"野中　茉莉花"**,** 21**));**

list**.**add**(new** User**(**"川上　優月"**,** 19**));**

list**.**add**(new** User**(**"稲岡　一馬"**,** 22**));**

**return** list**;**

**}**

**}**

② HomePage.htmlのdlタグ内に以下を追加する。

<dd><a wicket:id="toListViewTablePage">ListViewTablePageへ</a></dd>

③ HomePage.javaのコンストラクタに以下を追加する。

Link**<**Void**>** toListViewTablePageLink **=** **new** Link**<**Void**>(**"toListViewTablePage"**)** **{**

**private** **static** **final** **long** serialVersionUID **=** 1L**;**

@Override

**public** **void** **onClick()** **{**

setResponsePage**(new** ListViewTablePage**());**

**}**

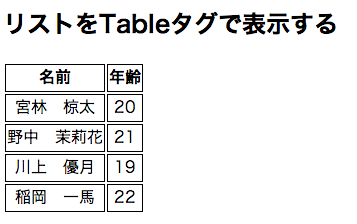
**};**

add**(**toListViewTablePageLink**);**

ブラウザでhttp://localhost:8080/wicket\_handson/ からListViewTablePageに移動し、動作を確認する。

ユーザの情報がテーブル形式で表示されればOK。

例：



# いろいろなModelを使ってみる

## CompoundPropertyModelの例

① org.wicket\_sapporo.handson.model\_usage パッケージに以下の二つのファイルを作成する

・CPModelFormPage.html（FromPage.html の内容をコピーする）

・CPModelFormPage.java（PDF参照）

② HomePage.htmlのdlタグ内に以下を追加する。

<dt>いろいろなModel</dt>

<dd><a wicket:id="toCPModelFormPage">CompoundPropertyModelFormPageへ</a></dd>

③ HomePage.javaのコンストラクタに以下を追加する。

Link<Void> toCPModelFormPageLink = new Link<Void>("toCPModelFormPage") {

private static final long serialVersionUID = 1L;

@Override

public void onClick() {

setResponsePage(new CPModelFormPage());

}

};

add(toCPModelFormPageLink);

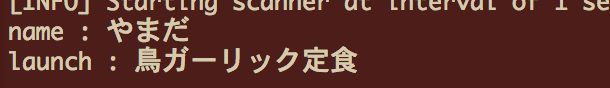
ブラウザでhttp://localhost:8080/wicket\_handson/ から CPModelFormPage に移動して動作を確認する。

フォームに入力して送信した文字列がコンソールに表示されればOK。

例：



↓　データ送信ボタンをクリック



④ org.wicket\_sapporo.handson.model\_usage パッケージに以下の二つのファイルを作成する

・CPModelConfirmationPage.html（ConfirmationPage.html の内容をコピーする）

・CPModelConfirmationPage.java

**【練習】 UserLunchの内容が表示されるページになるように修正せよ**

**package** org.wicket\_sapporo.handson**.**model\_usage**;**

**import** org.wicket\_sapporo.handson.beans.UserLunch**;**

**import** org.apache.wicket.markup.html.WebPage**;**

**import** org.apache.wicket.model.CompoundPropertyModel**;**

**import** org.apache.wicket.model.IModel**;**

**public** **class** **CPModelConfirmationPage** **extends** WebPage **{**

**private** **static** **final** **long** serialVersionUID **=** 1L**;**

**public** **CPModelConfirmationPage(**IModel**<**UserLunch**>** model**)** **{**

*// setDefaultModel(IModel) メソッドで、Modelをページにセットする*

setDefaultModel**(new** CompoundPropertyModel**<>(**model**));**

**}**

**}**

⑤ CPModelFormPage.java の Form の onSubmit メソッド内に、以下のコードを追加する。

setResponsePage**(new** CPModelConfirmationPage**(**getModel**()));**

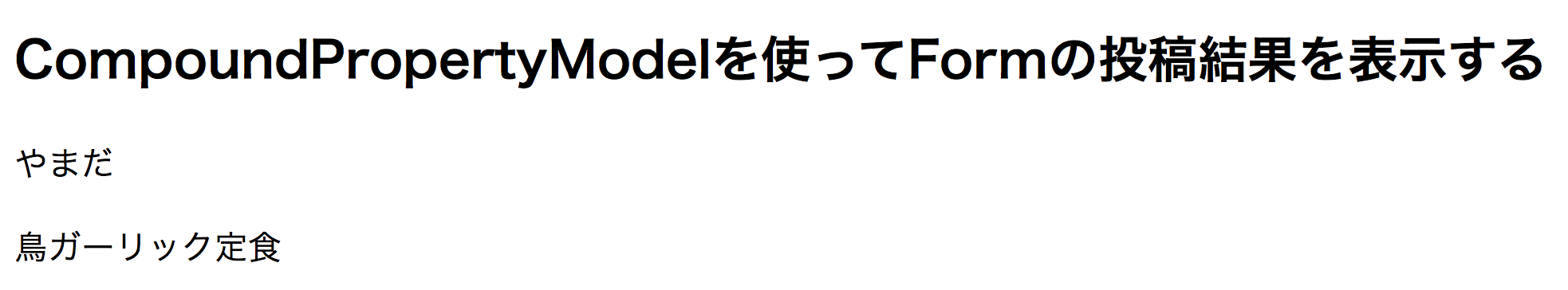
改めてブラウザでhttp://localhost:8080/wicket\_handson/ からFormPageに移動し、動作を確認する。

入力フォームから送信すると画面遷移し、送信した値が表示されればOK。

例：



↓　データ送信ボタンをクリック



## Read-OnlyなModelの例

① org.wicket\_sapporo.handson.model\_usage パッケージに以下の二つのファイルを作成する

・ReadOnlyModelPage.html（PDF参照）

・ReadOnlyModelPage.java（PDF参照）

② HomePage.htmlのdlタグ内に以下を追加する。

<dd><a wicket:id="toROModelPage">ReadOnlyModelPageへ</a></dd>

③ HomePage.javaのコンストラクタに以下を追加する。

Link<Void> toROModelPageLink = new Link<Void>("toROModelPage") {

private static final long serialVersionUID = 1L;

@Override

public void onClick() {

setResponsePage(new ReadOnlyModelPage());

}

};

add(toROModelPageLink);

ブラウザでhttp://localhost:8080/wicket\_handson/ から ReadOnlyModelPage に移動して動作を確認する。

確認事項は2つ。1. AbstractReadOnlyモデルを用いたコンポーネントはコンポーネントごとに異なる乱数が表示されており、LoadableDetachableモデルを用いたコンポーネントはコンポーネントごとに値が統一されていること。2. 1の条件はそのままに、画面をF5などで更新する度に表示される値が変わっていること。

例：



## Modelの実践的な使い方

① org.wicket\_sapporo.handson.model\_usage パッケージに以下のファイルを作成する

・ModelfulListViewPage.html（ListViewTablePage.html の内容をコピーする）

・ModelfulListViewPage.java（空のファイル）

② **【練習】listViewパッケージの ListViewTablePage.java を参考に、以下の条件を満たすような ModelfulListViewPage.java を作成しなさい。**

1. getUsers()メソッドの呼び出しに LoadableDetachableModel を使う  
2. ListView#populateItemメソッドで各ユーザのデータを表示する部分にはCompoundPropertyModelを使う

③ HomePage.htmlのdlタグ内に以下を追加する。

<dd><a wicket:id="toModelfulListViewPage">ModelfulListViewPageへ</a></dd>

④ HomePage.javaのコンストラクタに以下を追加する。

Link<Void> toModelfulListViewPageLink = new Link<Void>("toModelfulListViewPage") {

private static final long serialVersionUID = 1L;

@Override

public void onClick() {

setResponsePage(new ModelfulListViewPage());

}

};

add(toModelfulListViewPageLink);

ブラウザでhttp://localhost:8080/wicket\_handson/ から ModelfulListViewPage に移動して、ListViewTablePage と同じ画面が表示されるか確認する。

# Validationを使ってみる

## ValidatorとFeedbackメッセージの例

① org.wicket\_sapporo.handson.validation パッケージに以下の３つのファイルを作成する

・ValidationFormPage.html（PDF参照）

・ValidationFormPage.java（PDF参照）

・ValidationFormPage.properties（PDF参照）

※Eclipse では、propertiesファイルはプロパティエディタプラグインを使うと編集しやすい。（Pleiades All in Oneの場合は同様のプラグインが入っているかも？）

② HomePage.htmlのdlタグ内に以下を追加する。

<dt>入力チェック</dt>

<dd><a wicket:id="toValidationFormPage">ValidationFormPageへ</a></dd>

③ HomePage.javaのコンストラクタに以下を追加する。

Link**<**Void**>** toValidationFormPageLink **=** **new** Link**<**Void**>(**"toValidationFormPage"**)** **{**

**private** **static** **final** **long** serialVersionUID **=** 1L**;**

@Override

**public** **void** **onClick()** **{**

setResponsePage**(new** ValidationFormPage**());**

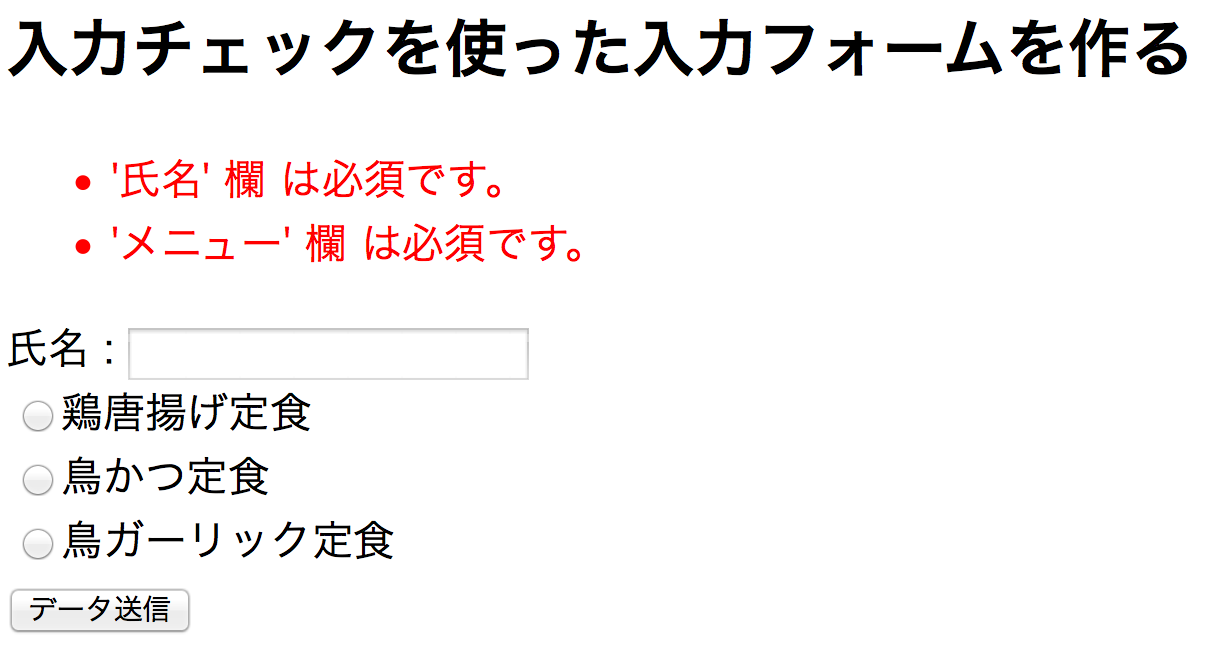
**}**

**};**

add**(**toValidationFormPageLink**);**

ブラウザでhttp://localhost:8080/wicket\_handson/ から ValidationFormPageに移動して、いろいろな値を入れてみて、エラー文が表示される事を確認する。

例：



# ClearURLを使ってみる

## BookmarkablePageLinkとPageMountの例

① org.wicket\_sapporo.handson.bookmarkable パッケージに以下の4つのファイルを作成する

・ParamSendPage.html（PDF参照）

・ParamSendPage.java（PDF参照）

・ParamReceiptPage.html（PDF参照）

・ParamReceiptPage.java（PDF参照）

② HomePage.htmlのdlタグ内に以下を追加する。

<dt>Bookmarkable（CleanURL）</dt>

<dd><a wicket:id="toParamSendPage">ParamSendPageへ</a></dd>

③ HomePage.javaのコンストラクタに以下を追加する。

Link**<**Void**>** toParamSendPageLink **=** **new** Link**<**Void**>(**"toParamSendPage"**)** **{**

**private** **static** **final** **long** serialVersionUID **=** 1L**;**

@Override

**public** **void** **onClick()** **{**

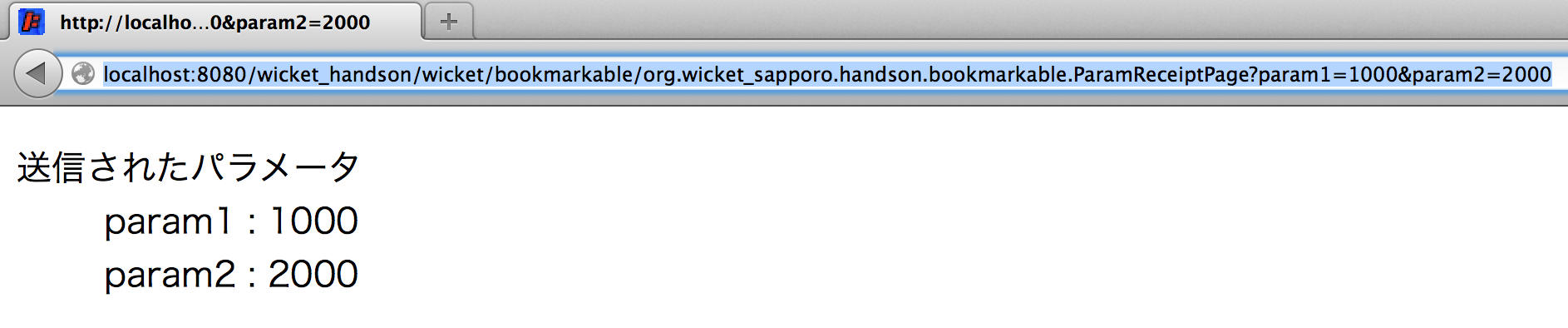
setResponsePage**(new** ParamSendPage**());**

**}**

**};**

add**(**toParamSendPageLink**);**

ブラウザでhttp://localhost:8080/wicket\_handson/ から ParamSendPageに移動して、「パラメータなしリンク」「パラメータありリンク」をクリックすることで、ParamReceiptPage に移動し、URLがパッケージ＋クラス名で表示されて、パラメータが従来の形（?param1=…）で渡されていることを確認する。

パラメータありリンク押下後の例：  


④ WicketApplication#init() メソッドに、MountMapperを追加する。

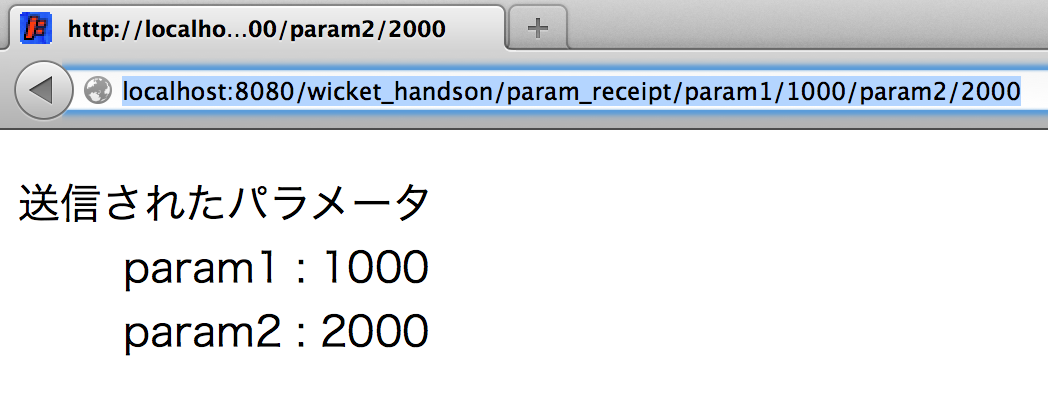
*// ParamReceiptPageをCleanURLに設定する*

mount**(new** MountedMapper**(**"/param\_receipt"**,** ParamReceiptPage**.**class**,**

**new** UrlPathPageParametersEncoder**()));**

改めてブラウザでhttp://localhost:8080/wicket\_handson/ から ParamSendPageに移動して、「パラメータなしリンク」「パラメータありリンク」をクリックすることで、 ParamReceiptPage に移動し、かつURLが短縮されてパラメータがCleanURLの形で渡されていることを確認する。

パラメータありリンクの例：



# Sessionを使ってみる

## ログインフォームの例

① org.wicket\_sapporo.handson.session パッケージに以下の5つのファイルを作成する

・MySession.java（PDF参照）

・SignInPage.html（PDF参照）

・SignInPage.java（PDF参照）

・SecurePage.html（PDF参照）

・SecurePage.java（PDF参照）

② HomePage.htmlのdlタグ内に以下を追加する。

<dt>Session</dt>

<dd><a wicket:id="toSigninPage">SigninPageへ</a></dd>

<dd><a wicket:id="toSecurePage">認証せずに直接SecurePageへ</a></dd>

③ HomePage.javaのコンストラクタに以下を追加する。

*// SignInPageはステートレスページにしているので、BookmarkablePageLinkで遷移させる*

BookmarkablePageLink**<**Void**>** toSigninPageLink **=**

**new** BookmarkablePageLink**<>(**"toSigninPage"**,** SignInPage**.**class**);**

add**(**toSigninPageLink**);**

Link**<**Void**>** toSecurePageLink **=** **new** Link**<**Void**>(**"toSecurePage"**)** **{**

**private** **static** **final** **long** serialVersionUID **=** 1L**;**

@Override

**public** **void** **onClick()** **{**

setResponsePage**(new** SecurePage**());**

**}**

**};**

add**(**toSecurePageLink**);**

④ WicketApplication#init() メソッドに、MountMapperを追加する。

mount**(new** MountedMapper**(**"/signin"**,** SignInPage**.**class**,** **new** UrlPathPageParametersEncoder**()));**

④ WicketApplication#newSession () メソッドをオーバーライドする。

@Override

**public** Session **newSession(**Request request**,** Response response**)** **{**

*// return super.newSession(request, response);*

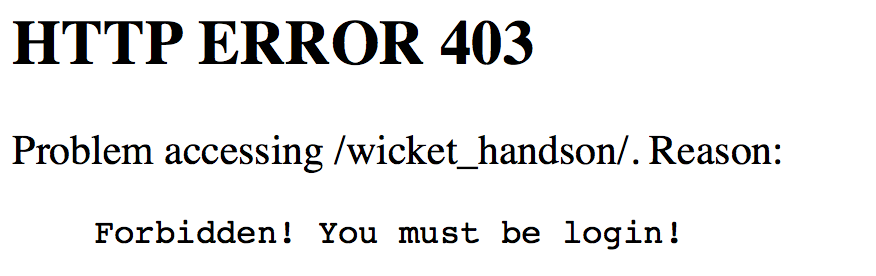
*// 独自に拡張したSessionの利用*

**return** **new** **MySession(**request**);**

**}**

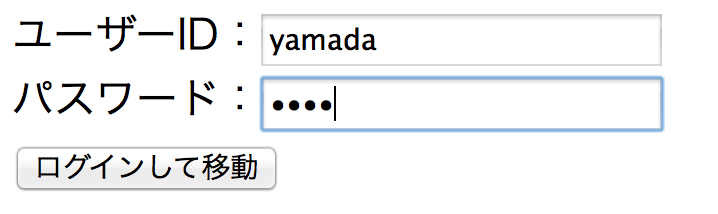
まず、ブラウザでhttp://localhost:8080/wicket\_handson/ から「認証せずに直接SecurePageへ」のリンクでSecurePageに移動して、403エラーが表示されることを確認する。

403エラーの例：

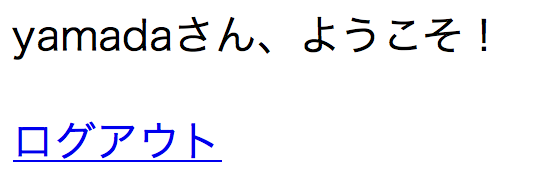


次に、「SigninPageへ」のリンクでSigninPageに移動した上で、ユーザ名は自由、パスワードは1234を入力することで SecurePage に移動できることを確認する。

例：



↓　ログインして移動ボタンをクリック



さらにログアウトリンクを押した後、http://localhost:8080/wicket\_handson/ に戻り、「認証せずに直接SecurePageへ」のリンクで再び403エラーが表示されることを確認する。

# Ajaxを使ってみる

## コンポーネントの表示・非表示の変更例

① org.wicket\_sapporo.handson.ajax パッケージに以下の２つのファイルを作成する

・VisibleChangePage.html（PDF参照）

・VisibleChangePage.java（PDF参照）

② HomePage.htmlのdlタグ内に以下を追加する。

<dt>Ajax</dt>

<dd><a wicket:id="toVisibleChangePage">VisibleChangePageへ</a></dd>

③ HomePage.javaのコンストラクタに以下を追加する。

Link**<**Void**>** toVisibleChangePageLink **=** **new** Link**<**Void**>(**"toVisibleChangePage"**)** **{**

**private** **static** **final** **long** serialVersionUID **=** 1L**;**

@Override

**public** **void** **onClick()** **{**

setResponsePage**(new** VisibleChangePage**());**

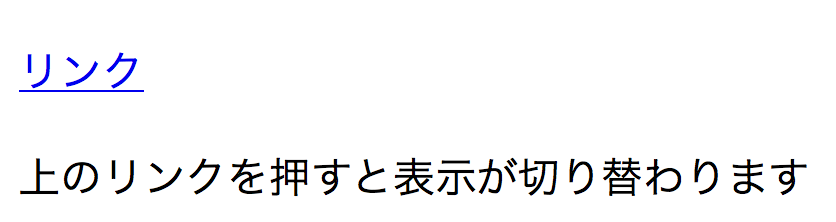
**}**

**};**

add**(**toVisibleChangePageLink**);**

ブラウザでhttp://localhost:8080/wicket\_handson/ から VisibleChangePageに移動して、リンクの押下でページ遷移せずにメッセージの表示・非表示が切り替わることを確認する。

例：



※リンクを押すと、ページ遷移をせずに「上のリンクを押すと表示が切り替わります」という文書が隠れたり表示されたりする。