

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>
```

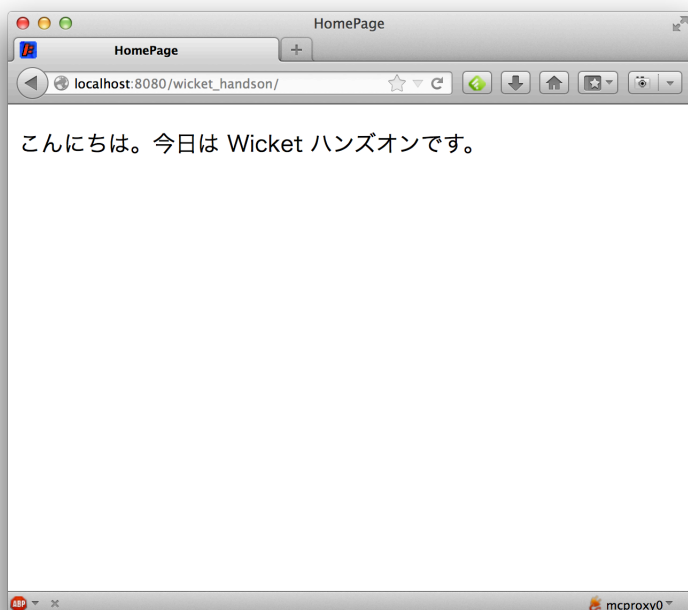
- HomePage.java

```
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：

入力フォームを作る

氏名：

↓ データ送信ボタンをクリック

```
[INFO] Starting scanner at interval of 1
name : やまだ
```

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>
```

② FormPage.java のフィールド変数に以下のコードを追加する。

```
// Lunche の値を格納する Model
private IModel<String> lunchModel;
```

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：

入力フォームを作る

氏名：

☐ 鶏唐揚げ定食

☐ 鳥かつ定食

☒ 鳥ガーリック定食

↓ データ送信ボタンをクリック

```
[INFO] Starting scanner at interval of 1 se
name : やまだ
launch : 鳥ガーリック定食
```

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：

入力フォームを作る

氏名：

- ☐ 鶏唐揚げ定食
- ☐ 鳥かつ定食
- ☒ 鳥ガーリック定食

↓ データ送信ボタンをクリック

Formの投稿結果を表示する

やまだ

鳥ガーリック定食

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：

リストを表示する

- 北海道
- 青森
- 岩手
- 秋田
- 宮城
- 福島

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  棕太", 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：

リストをTableタグで表示する

名前	年齢
宮林 椋太	20
野中 茉莉花	21
川上 優月	19
稲岡 一馬	22

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：

CompoundPropertyModelを使った入力フォームを作る

氏名：

☐ 鶏唐揚げ定食

☐ 鳥かつ定食

☒ 鳥ガーリック定食

↓ データ送信ボタンをクリック

```
[INFO] Starting scanner at interval of 1 se
name : やまだ
launch : 鳥ガーリック定食
```

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

- ・ CPMoelConfirmationPage.html (ConfirmationPage.html の内容をコピーする)
- ・ CPMoelConfirmationPage.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 CPMoelConfirmationPage extends WebPage {
    private static final long serialVersionUID = 1L;

    public CPMoelConfirmationPage(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 CPMoelFormPage.java の Form の onSubmit メソッド内に、以下のコードを追加する。

```
setResponsePage(new CPMoelConfirmationPage(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：

CompoundPropertyModelを使った入力フォームを作る

氏名：

- ☐ 鶏唐揚げ定食
- ☐ 鳥かつ定食
- ☒ 鳥ガーリック定食

↓ データ送信ボタンをクリック

CompoundPropertyModelを使ってFormの投稿結果を表示する

やまだ

鳥ガーリック定食

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 など更新する度に表示される値が変わっていること。

例：

ReadOnlyModelを使って複数回Modelを表示する

AbstractReadOnlyModel

268

86

401

LoadableDetachableModel

936

936

936

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 を試してみる

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：

入力チェックを使った入力フォームを作る

- '氏名' 欄 は必須です。
- 'メニュー' 欄 は必須です。

氏名 :

- ☐ 鶏唐揚げ定食
☐ 鳥かつ定食
☐ 鳥ガーリック定食

ClearURL を試してみる

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>
```

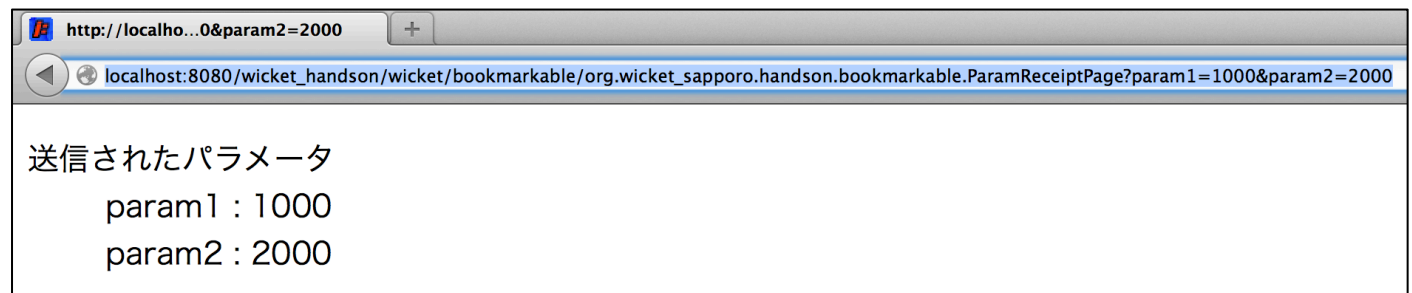
③ HomePage.java のコンストラクタに以下を追加する。

```
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：



送信されたパラメータ

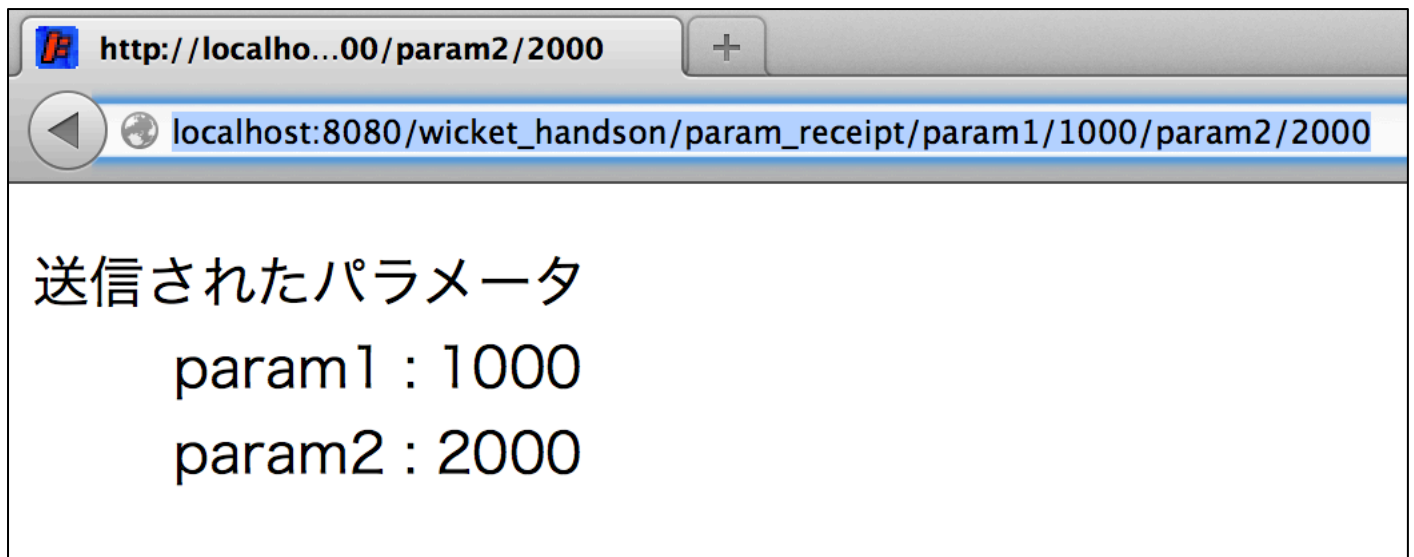
- param1 : 1000
- param2 : 2000

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 UriPathPageParametersEncoder()));
```

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 を試してみる

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：

HTTP ERROR 403

Problem accessing /wicket_handson/. Reason:

Forbidden! You must be login!

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：

ユーザーID :

パスワード :

↓ ログインして移動ボタンをクリック

yamadaさん、ようこそ！

[ログアウト](#)

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 を試してみる

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](#)

上のリンクを押すと表示が切り替わります

※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。