JTemplate Ver1.00 マニュアル

1. はじめに

JTemplate は、デザイナーとプログラマーの共同作業を実現します。
基本的にプログラムから View を制御するので、JSP のようなコーディングが必要ありません。
読み込んだ XHTML（テンプレートＨＴＭＬ）のタグすべてをプログラム上から取得できますので、より柔軟な動的ＨＴＭＬの生成が可能です。また、テンプレートＨＴＭＬのネストをサポートしています。これにより、ＨＴＭＬの部品化を実現します。
JTemplate は、Struts などのフレームワーク上で利用可能です。

2. JTemplate

Struts などのフレームワーク上で利用可能は、テンプレートエンジンです。

2.1. 使用方法

Servlet --> Action --> JSP の流れを以下に記します。

2.1.1. Servlet

サーブレットの初期化時に、最初にテンプレートホーム、ファイル読み込みエンコードを設定します。JTemplate が、テンプレートホーム配下に work フォルダを自動作成し、html を xml 形式に変換し、work フォルダに xml ファイルを保存します。[1]

```
String contextPath = getServletContext().getRealPath("WEB-INF/template");
ViewFactory.setTemplateHome(contextPath);
ViewFactory.setCharacterEncoding("Windows-31J");
```

2.1.2. Action

最初にファイルを読み込み(２度目からはキャッシュを使用)、データを置換し、最後に display メソッドをよび、リクエストに保存します。

```
View view = ViewFactory.getView("/sample/sample.html");
.
view.display();

request.setAttribute("view", view);
return mapping.findForward("success");
```

2.1.3. JSP

リクエストより View クラスを取得し、toHtml メソッドで HTML 文字列を出力します。

```
<%@page contentType="text/html; charset=Windows-31J"%>
<jsp:useBean id="view" class="org.genesis.view.GenesisView" scope="request"/>
<html>
        <body>
                <%= view.toHtml() %>
        </body>
</html>
```

---

[1] うまく起動しなかった場合は work の xml ファイルを参考にしてもとの html ファイルを修正してください。xhtml 形式であれば問題なくファイルを読み込みます。

2.2. 置換文字

${...}をプログラムで指定された値に置換します。

■テンプレートＨＴＭＬ

```
${name}さんは、${age}才ですね。
```

■ プログラム

二通りの方法でＨＴＭＬの文字を置換できます。

方法１

```
view.put("name", "Genesis");
view.put("age" , "29");
view.display();                    ←View に格納された値で置換します。
```

方法２

```
Map map = new HashMap();
map.put("name", "Genesis");
map.put("age" , "29");
view.display(map);                 ←Map に格納された値で置換します。
```

■実行結果

```
Genesis さんは、29 才ですね。
```

2.3. 独自タグ

以下の４つの独自タグを用いることにより動的な表現が可能になります。

ロジックタグ	概要
<for>	繰り返し
<tree>	XMLの階層をツリー表示します。
<if>	条件判断

2.3.1. FOR

<for>…</for>の間の文字列を java.util.List に格納された java.util.Map の数だけ繰り返します。

■テンプレートＨＴＭＬ

```
<table border="1" width="100%">
  <tr>
    <td>名前</td>
    <td>歳</td>
  </tr>
<for>
  <tr>
    <td>${name}</td>
    <td>${age}</td>
  </tr>
</for>
</table>
```

■ プログラム

```
View forView = view.getFor();

List persons = new ArrayList();

Map mike = new HashMap();
mike.put("name", "Mike");
mike.put("age", "28");
persons.add(mike);

Map tom = new HashMap();
tom.put("name", "Tom");
tom.put("age", "29");
persons.add(tom);

forView.display(persons);
```

■ 実行 HTML ソース

```
<table border="1" width="100%">
      <tr>
            <td>名前</td>
            <td>歳</td>
      </tr>
<for>
   <line>
      <tr>
         <td>Mike</td>
         <td>28</td>
      </tr>
   </line>
   <line>
      <tr>
         <td>Tom</td>
         <td>29</td>
      </tr>>
   </line>
</for>
</table>
```

■実行ＨＴＭＬ画面

名前	歳
Mike	28
Tom	29

2.3.2. TREE

ＸＭＬ階層をＨＴＭＬ形式のツリーで表示します。

■テンプレートＨＴＭＬ

```
<tree>
        <project>
                <font class="xxsmall">${name}</font>
        </project>
        <dir>
                <font class="xxsmall">${name}</font>
        </dir>
        <file>
                <font class="xxsmall" color="black">${name}</font>
        </file>
</tree>
```

■プログラム

```
View treeView = view.getTree();

Freedom appli = new Freedom("project");
appli.set("name", "アプリケーション");

Freedom office = appli.newChild("dir");
office.set("name", "Office");
Freedom excel = office.newChild("file");
excel.set("name", "エクセル");
Freedom word = office.newChild("file");
word.set("name", "ワード");

Freedom other = appli.newChild("dir");
other.set("name", "その他");
Freedom hide = other.newChild("file");
hide.set("name", "秀丸");

treeView.display(appli);
```

動的 XML の作成

■動的ＸＭＬ

```
<project name="アプリケーション">
    <dir name="Office">
        <file name="エクセル"/>
        <file name="ワード"/>
    </dir>
    <dir name="その他">
        <file name="秀丸"/>
    </dir>
</project>
```

■実行結果

```
アプリケーション
  Office
      エクセル
      ワード
  その他
      秀丸
```

2.3.3. IF

ネスト可能な条件分岐を実現します。

■ テンプレートＨＴＭＬ

```
<if name="Mike">
  １組
</if>
<if name="Tom">
    ２組
</if>
<else>
    -
</else>
```

■ プログラム

```
List persons = new ArrayList();

Map mike = new HashMap();
mike.put("name", "Mike");
mike.put("age", "28");
mike.put("score", "30");

view.display(mike);
```

■ 実行結果

```
<if name=”Mike”>
  １組
</if>
```

2.4. ＨＴＭＬタグ

2.4.1. SELECT

動的にセレクトタグを作成することができます。

OptionTag をインターフェースに持たせてください。

■テンプレートＨＴＭＬ

```
<select name=”id”>
   <option value=””></option>
</select>
```

■プログラム

```
String selectedValue = “1”;
List list = new ArrayLIst();

FruitsBean apple = new FruitsBean();
apple. setFruitsCode (“0”);
apple. setFruitsName (“りんご”);
list.add(apple);

FruitsBean peach = new FruitsBean();
peach. setFruitsCode (“1”);
peach. setFruitsName (“もも”);
list.add(peach);

view.selectTag(“id”, list, selectedValue);
```

■実行後

```
<select name=”id”>
   <option value=””></option>
   <option value=”0”>りんご</option>
   <option value=”1” selected=””>もも</option>
</select>
```

2.4.2. RADIO

動的にラジオタグを作成します。

OptionTag をインターフェースに持たせてください。

■ テンプレートＨＴＭＬ

```
<input type=”radio” name=”price_code”/>
```

■プログラム

```
List prices = new ArrayList();

PriceBean high = new PriceBean();
high.setPriceCode("high");
high.setPriceName("松");
prices.add(high);

PriceBean middle = new PriceBean();
middle.setPriceCode("middle ");
middle.setPriceName("竹");
prices.add(middle);

PriceBean low = new PriceBean();
low.setPriceCode("low");
low.setPriceName("梅");
prices.add(low);

view.radioTag("price_code", prices, " low ");
```

■ 実行後

○松　○竹　●梅

2.4.3. CHECKBOX

動的にチェックボックスを作成します。

OptionTag をインターフェースに持たせてください。

■テンプレートＨＴＭＬ

```
<input type=”checkbox” name=”place_code”/>
```

■プログラム

```
List places = new ArrayList();

PlaceBean tokyo = new PlaceBean();
tokyo.setPlaceCode("0");
tokyo.setPlaceName("東京");
places.add(tokyo);

PlaceBean paris = new PlaceBean();
paris.setPlaceCode("1");
paris.setPlaceName("パリ");
places.add(paris);

PlaceBean london = new PlaceBean();
london.setPlaceCode("2");
london.setPlaceName("ロンドン");
places.add(london);

view.checkboxTag("place_code", places, new String[]{"0", “1”});
```

■実行後

■東京　■パリ　□ロンドン

2.5.　HTML 操作

プログラム上からすべてのタグが、取得、登録、更新、削除が可能です。

2.5.1.　SELECT

すべてのタグを select(タグ名、属性名、属性値)で取得可能です。

■　テンプレートＨＴＭＬ

```
<table>
    <tr>
        <td>好きな果物</td>
        <td name=”fruits”>
            <select name=”fruits_code”>
                <option value=””></option>
            </select>
        </td>
    </tr>
</table>
```

■　プログラム

```
Freedom fruits = view.select("td", "name", "fruits");
```

■取得ＨＴＭＬ

```
        <td name=”fruits”>
            <select name=”fruits_code”>
                <option value=””></option>
            </select>
        </td>
```

2.5.2. INSERT

すべてのタグに newChild(タグ, 属性名, 属性値)で新しくタグを挿入することが可能です。

■テンプレートＨＴＭＬ

```
<table>
    <tr>
        <td>好きな果物</td>
        <td name=”fruits”>
            <select name=”fruits_code”>
                <option value=””></option>
            </select>
        </td>
    </tr>
</table>
```

■プログラム

```
Freedom fruits = view.select("td", "name", "fruits");
Freedom a      = fruits.newChild(a, “href”, “http://www.sample.co.jp”);
a.set(“target”, “_blank”);
a.setNodeValue(“ホームページ”);
```

■ＨＴＭＬ

```
        <td name=”fruits”>
            <select name=”fruits_code”>
                <option value=””></option>
            </select>
            <a href=”http://www.sample.co.jp” target=”_blank”>
              ホームページ
            </a>
        </td>
```

2.5.3. UPDATE

すべての文字、タグ名を set(属性名，属性値)、setNodeValue(タグ値)、
setNodeName(タグ名)で更新可能です。

■テンプレートＨＴＭＬ

```
<table>
    <tr>
        <td name=”title”>好きな果物</td>
        <td name=”fruits”>
            <select name=”fruits_code”>
                <option value=””></option>
            </select>
        </td>
    </tr>
</table>
```

■プログラム

```
Freedom title = view.select("td", "name", "title");
title.set(“name”, “dessert”);
title.setNodeValue(“好きなデザート”);
title.setNodeName(“span”);
```

■HTML

```
<table>
    <tr>
        <span name=”dessert”>好きなデザート</span>
        <td name=”fruits”>
            <select name=”fruits_code”>
                <option value=””></option>
            </select>
        </td>
    </tr>
</table>
```

2.5.4. DELETE

すべてのタグを delete(タグ名，属性名，属性値)で削除可能です。

■テンプレートＨＴＭＬ

```
<table>
    <tr>
        <td name=”title”>好きな果物</td>
        <td name=”fruits”>
            <select name="fruits_code">
                <option value=""></option>
            </select>
        </td>
    </tr>
</table>
```

■　プログラム

```
view.delete("select", "name", "fruits_code");
```

■実行後ＨＴＭＬ

```
<table>
    <tr>
        <td name=”title”>好きな果物</td>
        <td name=”fruits”>
        </td>
    </tr>
</table>
```

2.6.　HTML 部品化

テンプレートＨＴＭＬファイルに複数のファイルをネストすることが可能です。
タグ名に name 属性をつけ、一意なタグならどんなタグでも可能です。
ネストするファイルには、最初のタグが、テンプレートＨＴＭＬと同じタグ名と name 属性を持つファイルを用意してください。

■テンプレートＨＴＭＬ(nest.html)

```
<table>
    <tr>
        <td>
            <table name="type"/>      ← ネスト1対象タグ
        </td>
        <td>
          <table name="shop"/>        ← ネスト2対象タグ
        </td>
        <td>
            <table name="layout"/>    ← ネスト3対象タグ
        </td>
    </tr>
</table>
```

■ネスト１ＨＴＭＬ(type.html)

```
<table name="type">
    <tr>
        <td>築年数</td>
        <td>
            <select name="type">
                <option value="0">築５年以内</option>
                <option value="1">築１０年以内</option>
                <option value="2">その以外</option>
            </select>
        </td>
    </tr>
</table>
```

■　ネスト２ＨＴＭＬ(shop.html)

```
<table name=”shop”>
    <td>
        <span>場所</span>
        <select name="shop_code">
            <option value="tokyo">東京</option>
            <option value="nagoya">名古屋</option>
            <option value="osaka">大阪</option>
        </select>
      </td>
</table>
```

■ネスト３ＨＴＭＬ(layout.html)

```
<table name="layout">
        <tr>
                <td>間取り</td>
                <td>
                        <select name="layout">
                                <option value="0">1K</option>
                                <option value="1">1LDK</option>
                                <option value="2">2DK</option>
                        </select>
                </td>
        </tr>
</table>
```

■プログラム

```
View view     = ViewFactory.getView("/sample/nest.html");
View type     = view.getView("/sample/type.html");
View shop    = view.getView(“/sample/shop.html”);
View layout = view.getView("/sample/layout.html");
```

view.getView が実行されると同時にネストします。

■　実行結果

```
<html>
    <body>
        <table>
            <tr>
                <td>
                    <table name="type">
                      .
                      .
                      .
                    </table>
                </td>
                <td>
                    <table name="shop">
                      .
                      .
                      .
                    </table>
                </td>
                <td>
                    <table name="layout">
                      .
                      .
                      .
                    </table>
                </td>
            </tr>
        </table>
    </body>
```

ネストＨＴＭＬの内容

ネストＨＴＭＬの内容

ネストＨＴＭＬの内容