CodeZine
デベロッパーの成長と課題解決に貢献するメディア

Javaで軽快に使える「軽量フレームワーク」特集 ～クールなGUIをシンプルなスクリプトで作成するZK（2）

第21回

Java

掌田 津耶乃[著]

2010/08/26 14:00

注目の軽量フレームワークをフットワーク軽く取り上げていく本連載。今回は、JavaをベースにしたRIA開発のフレームワーク「ZK」の第2回として、コンポーネントをレイアウトするためのコンテナ類と、コンポーネントにおけるデータの扱いを中心に説明します。

はじめに

前回、ZKの基本的なコンポーネントについて説明しました。今回は、もう少し複雑なコンポーネントについて説明を行います。

コンポーネントといっても、単純に「ある部品をそこに配置するだけ」といったものばかりではありません。例えば、JavaのSwingなどでは「コンテナ」がけっこう重要な役割を果たしています。コンテナは、「コンポーネントをまとめるためのコンポーネント」です。このコンテナに相当するものは、ZKにもいろいろと揃っています。

また、コンポーネントの中には、データ表示に関するものもいろいろとあります。ZKにもそうしたコンポーネントがいくつかあります。この種のものは、いかにしてコンポーネントの記述とデータを切り離すか、ダイナミックなデータの更新をどう行うかが重要になります。今回は、「コンテナ」と「データを扱うコンポーネント」について説明します。

対象読者

- Javaで手ごろなフレームワークを探している技術者
- 最近のフレームワークをごくざっと理解しておきたい方
- Web開発の手法がどうも気に入らない、と常々考えているJavaプログラマ

レイアウトコンポーネント

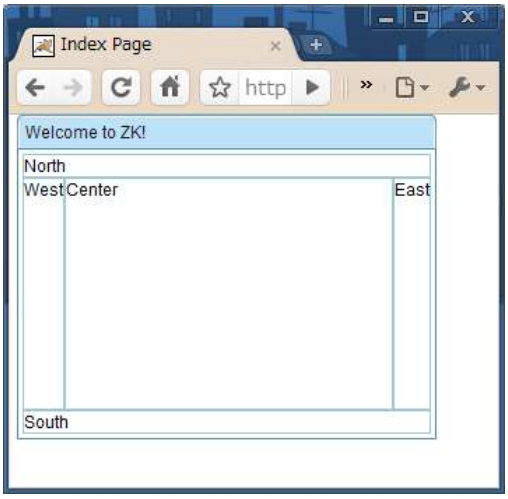
今回は、まずコンポーネントの配置（レイアウト）から説明していきます。**前回**、いくつかのコンポーネントを使ってみましたが、それらはすべて「ずらっと順番に並べる」というだけの配置でした。<separator/>で改行はできましたが、これではあまり複雑なレイアウトはできません。

ZKには、レイアウトのためのコンポーネントというのも用意されています。これを利用することで、コンポーネントをレイアウトし、配置することが可能になります。では、もっとも基本的なレイアウト・コンポーネントである<borderlayout>から使ってみましょう。

```
<?page title="Index Page"?>
<window title="Welcome to ZK!" border="normal" width="300px">
  <borderlayout width="100%" height="200px">
    <north>
      <label>North</label>
    </north>
    <east>
      <label>East</label>
```

```
    </east>
    <center>
      <label>Center</label>
    </center>
    <west>
      <label>West</label>
    </west>
    <south>
      <label>South</label>
    </south>
  </borderlayout>
</window>
```

<borderlayout>によるレイアウト、JavaのBorderLayoutそのままのレイアウトが作成される



アクセスすると、ウインドウ内が5つのエリアに分けられ、それぞれにテキストが表示されます。このレイアウトは、AWTのBorderLayoutとまったく同じ分け方です。このように5つのエリアに整理することで、すっきりとコンポーネントを配置できます。

それぞれのエリアは、<borderlayout>内に、<north>、<south>、<east>、<west>、 といったタグで配置します。これらは、常にすべて用意する必要はありません。例えば<west>、<east>を省略すれば、3列のエリアだけが表示されます。

BOXレイアウト

いくつかのコンポーネントを一列に並べる場合には、「BOXレイアウト」と呼ばれるものを利用するのが良いでしょう。これは<box>というタグとして用意されています。この中に、縦横にコンポーネントを並べていく<hbox>、<vbox>といったタグを使ってコンポーネントを組み込んでいきます。

```
<?page title="Index Page"?>
<window title="Welcome to ZK!" border="normal" width="300px">
  <label>※BOXレイアウト</label>
  <separator />
  <box width="90%">
    <hbox>
      <label style="background-color:#DDF">hbox その１</label>
      <label style="background-color:#DDF">hbox その２</label>
      <label style="background-color:#DDF">hbox その３</label>
    </hbox>
    <hbox>
      <label style="background-color:#DDF">hbox その１</label>
      <label style="background-color:#DDF">hbox その２</label>
      <label style="background-color:#DDF">hbox その３</label>
    </hbox>
  </box>
  <separator /><separator />
</window>
```

<box>によるレイアウト。3×2でラベルが配置される

ここでは、ラベルを横に3つ並べた<hbox>を2行配置してみました。<box>タグの中に<hbox>タグがあり、その中にさらに<label>タグが並べられているのが分かります。ここで使っている<hbox>は、中に組み込んだコンポーネントを横に並べて配置します。同様に、<vbox>タグを使うと、中に組み込んだコンポーネントを縦に配置することもできます。

タブボックス

いくつかの表示をタブで切り替えながら表示する「タブパネル」は、多くのプラットフォームで採用されています。これを実現するのが、<tabbox>というタグです。これは、タブの情報とそれぞれのタブに表示するパネルを設定することで、自動的に切り替えタブを持つパネルを作成できます。

```
<?page title="Index Page"?>
<window title="Welcome to ZK!" border="normal" width="300px">
  <label>※TABBOX</label>
  <separator />
  <tabbox width="250px" height="100px">
    <tabs>
      <tab label="最初"/>
      <tab label="真ん中"/>
      <tab label="最後"/>
    </tabs>
    <tabpanels>
      <tabpanel>
        <label>最初のタブです。</label>
      </tabpanel>
      <tabpanel>
        <label>真ん中のタブです。</label>
      </tabpanel>
      <tabpanel>
        <label>最後のタブです。</label>
      </tabpanel>
    </tabpanels>
  </tabbox>
  <separator /><separator />
</window>
```

<tabbox>によるタブパネル

ここでは3つのタブを用意してみました。タブをクリックすれば、表示が切り替わります。タグを記述するだけなので、ノンプログラミングで作成できることが分かるでしょう。

<tabbox>は、内部に<tabs>と<tabpanels>という2つのタグを持っています。<tabs>は、タブの設定をまとめるもので、この中に<tab>タグとして表示するタブの情報を設定します。タブをクリックしたときに表示される内容を用意するのが<tabpanels>で、この中に<tabpanel>というタグを使って各タブ用の表示を作成します。<tabpanel>はコンテナであり、この中に自由にコンポーネントを配置できます。

moldによるアコーディオンボックス

タブパネルと似たような役割を果たすもので、最近、Webなどで広く使われるようになったインターフェースに「アコーディオンパネル」があります。タイトルをクリックすると、アコーディオンのようにその項目の表示が広がって表示される、というものです。

ZKには、アコーディオンパネルのための専用コンポーネントは用意されていません。しかし、実はタブボックスを使ってアコーディオンパネルを作ることができます。

```
<tabbox width="250px" height="200px" mold="accordion">
```

moldを変更し、アコーディオンにしたタブボックス

先ほどの<tabbox>のタグを上のように修正してみてください。タブパネルの表示が、そのままアコーディオンパネルに変わります。ここでは「mold」という属性に"accordion"を指定しています。これだけでアコーディオンパネルを作ることができます。

アコーディオンパネルとタブパネルは、「いくつかのタブがあり、タブをクリックして選択すると、そのタブの内容が表示される」という機能自体はまったく同じです。そのため、これは別のコンポーネントというより、「同じコンポーネントで、表し方が違っているだけ」と考えた方が自然でしょう。moldは、こうした「あるコンポーネントの、異なる見せ方」を示すためのものです。

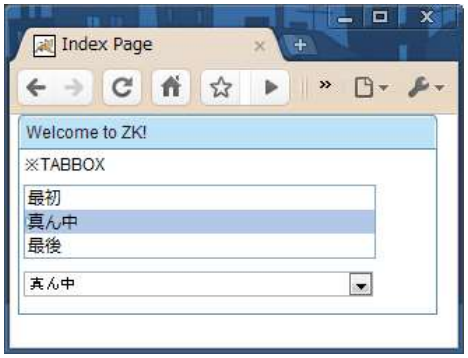
2種類のリストボックス

アコーディオンパネル以外にも、moldを利用するコンポーネントはあります。例えば、一覧リストを表示するための<listbox>というコンポーネントがありますが、これはmoldによって、HTMLのリストとして表現させることもできるようになります。

```
<?page title="Index Page"?>
<window title="Welcome to ZK!" border="normal" width="300px">
  <label>※TABBOX</label>
  <separator />
  <listbox width="250px" mold="default">
    <listitem label="最初"/>
    <listitem label="真ん中"/>
    <listitem label="最後"/>
  </listbox>
  <separator />
  <listbox width="250px" mold="select">
    <listitem label="最初"/>
    <listitem label="真ん中"/>
    <listitem label="最後"/>
  </listbox>
```

```
  <separator />
</window>
```

moldを使って表示した2種類の<listbox>タグ



これは、同じ<listbox>を、片方はmold="default"、もう一方はmold="select"に設定したものです。前者は後述する<grid>によるテーブル表示と同じような形でリスト表示され、後者はHTMLの<select>タグによるコンボボックスとして表示されます。moldによって、表現の仕方が変わっていることが分かるでしょう。

<listbox>は、表示する項目を<listitem>というタグとして内部に用意します。これで、いくつかの選択項目を用意して選ぶインターフェイスが作成できます。多数の項目を選択するためのものとして覚えておきたいコンポーネントです。

リストボックスによる複数列表示

このリストボックスは、HTMLの<select>タグによる一覧リストの表示とは明らかに違います。動作は同じようなもので、一覧からクリックして1行だけ（あるいは複数行）選択でき、クリックしたイベント処理をonSelectで行わせることもできます。

しかし、少なくとも表示においては、<listbox>の表示は<table>タグによるテーブル表示と同じものといえます（mold="default"の場合）。先の例では1列のリストだけを表示しましたが、実は複数列の表示も簡単に作成できます。

```
<?page title="Index Page"?>
<window title="Welcome to ZK!" border="normal" width="300px">
  <label>※TABBOX</label>
  <separator />
  <listbox width="250px" mold="default">
    <listitem label="支社名">
      <listcell label="東京" />
      <listcell label="大阪" />
      <listcell label="名古屋" />
    </listitem>
    <listitem label="前期売上">
      <listcell label="12345" />
      <listcell label="6789" />
      <listcell label="1010" />
    </listitem>
    <listitem label="後期売上">
      <listcell label="9876" />
      <listcell label="5432" />
      <listcell label="1010" />
    </listitem>
  </listbox>
  <separator /><separator />
</window>
```

<listcell>を用い、複数列のテーブルを表示したリストボックス

ここでは、複数の列のデータを表のようにして表示しています。見れば分かるように、<listbox>と<listitem>の構造は同じですが、ここでは<listitem>の中にさらに<listcell>というタグが組み込まれています。

これは、各列のデータを記述するためのもので、これを利用することで、1行を複数の列に分割し表示できるようになります。実際にやってみると、ヘッダーの部分などもまったく同じ表示になっており、表としてはのっぺりした感じになりますが、「クリックして選択できる表」を作りたい場合にはとても重宝できるでしょう。

グリッドによるテーブル作成

多数のコンポーネントを配置するケースとして、もっとも多いのは「フォームのレイアウト」でしょう。いくつもの入力用コンポーネントをきれいに整列させるのに、<table>タグを使うことはよくあります。

こうした「縦横にコンポーネントを整列させる」という場合に役立つのが<grid>です。これは、HTMLの<table>タグなどによるテーブル作成とほぼ同じようなレイアウトを提供しますが、<table>などよりずっときれいに表示されます。

```
<?page title="Index Page"?>
<window title="Welcome to ZK!" border="normal" width="300px">
  <label>※入力フォームのレイアウト</label>
  <separator />
  <grid width="90%">
    <columns>
      <column label="タイトル" width="35%" />
      <column label="入力フィールド" />
    </columns>
    <rows>
      <row>
        <label>名前</label>
        <textbox width="98%" />
      </row>
      <row>
        <label>Eメール</label>
        <textbox width="98%" />
      </row>
      <row>
        <label>電話番号</label>
        <textbox width="98%" />
      </row>
    </rows>
  </grid>
  <separator /><separator />
</window>
```

<grid>によるレイアウト。表の形式でレイアウトするのに用いる

簡単な入力フォームを<grid>でレイアウトしたサンプルです。<grid>では、ヘッダーと内容部分を設定できます。また内容の表示では、1行ごとに背景色が自動的に変更され非常に見やすくなります。<listbox>でも表は作成できましたが、こちらはヘッダーなども立体的になり、非常に表として見やすくなっています。

この<grid>は、その中に<columns>と<rows>の2つのタグを持っています。<columns>は列情報のタグで、<column>タグで各列の設定を記述します。ここで用意したものがそのままヘッダーとして表示されるようになっています。

<rows>は、<row>タグを使って各行の表示内容を記述します。この<row>内に配置したコンポーネントは、<columns>内に用意した<column>の列に1つずつはめこまれ、表示されます。

ここでは、フォームのレイアウトサンプルとして作成したのでただ各項目を表示しているだけですが、<grid>は本格的な作表にも利用できます。<grid>には、サイズの自動調整やソート順を変更するための機能が用意されており、動的に表示順などを変更できる表を作ることができます。

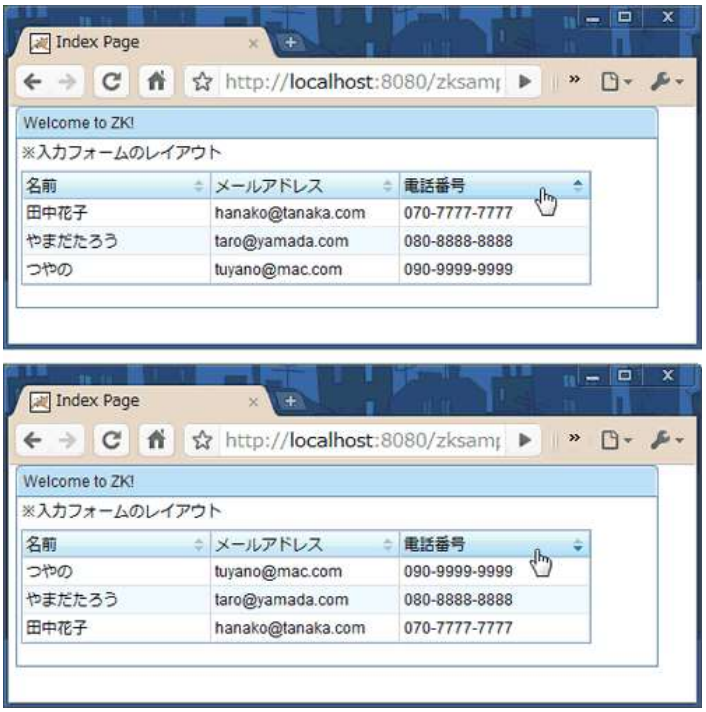
データを並べ替える

では、<grid>で表示した項目に、列幅の調整と行の並べ替え機能を追加してみましょう。

```
<?page title="Index Page"?>
<zk>
  <zscript>
    class GridComparator implements Comparator {
      boolean asc;
      public GridComparator(boolean ascending) {
        asc = ascending;
      }
      public int compare(Object o1, Object o2) {
        Row r1 = (Row)o1;
        Row r2 = (Row)o2;
        return 1;
      }
    }
    Comparator asc = new GridComparator(true);
    Comparator dsc = new GridComparator(false);
  </zscript>
  <window title="Welcome to ZK!" border="normal" width="500px">
    <label>※入力フォームのレイアウト</label>
    <separator />
    <grid width="90%">
      <columns sizable="true">
        <column label="名前"
          sortAscending="${asc}" sortDescending="${dsc}" />
        <column label="メールアドレス"
          sortAscending="${asc}" sortDescending="${dsc}" />
        <column label="電話番号"
          sortAscending="${asc}" sortDescending="${dsc}" />
      </columns>
      <rows>
        <row>
```

```
            <label>つやの</label>
            <label>tuyano@mac.com</label>
            <label>090-9999-9999</label>
    </row>
    <row>
            <label>やまだたろう</label>
            <label>taro@yamada.com</label>
            <label>080-8888-8888</label>
    </row>
    <row>
            <label>田中花子</label>
            <label>hanako@tanaka.com</label>
            <label>070-7777-7777</label>
    </row>
        </rows>
      </grid>
      <separator /><separator />
    </window>
</zk>
```

項目の並べ替えを追加する。ヘッダーをクリックすることで並び順が変わる



ヘッダーとヘッダーの境界部分をマウスでドラッグすると、それぞれの列幅を調整できます。また、各列のヘッダーをクリックすると、その列の値を基準にデータを並べ替えることができます。1度クリックすると昇順に、さらにもう1度クリックすると降順になります。

まず列幅の調整からです。これは非常に簡単で、<columns>タグにsizable="true"という属性を追加するだけです。問題は、並び順の調整です。これは、各<column>に「sortAscending」「sortDescending」といった属性を用意します。これらには、sortAscending="${asc}" sortDescending="${dsc}"というように値が設定されています。この${○○}という書き方は、JSPで見覚えがあると思いますが、式言語（Expression Language）です。ZULファイルでは、このJSPの式言語と同じ書き方で変数を埋めこむことが可能なのです。

ここでsortAscending／sortDescendingに式言語を使って指定しているのは、「Comparator」のインスタンスです。Comparatorというのは、Javaプログラマならば、オブジェクトの並び替えを行うの用いられるインターフェイスであることは知っていると思います。ソートのためのComparatorクラスを用意し、これをsortAscending／sortDescending属性に設定することでソートが可能となります。

ここでは、GridComparatorというクラスを定義し、そのインスタンスを使っています。一応、昇順と降順での違いを処理できるようにascという真偽値のフィールドを用意してあります（ここでは特に使っていません）。compareメソッドでは2つのObjectが渡されますが、これは<grid>の列の並べ替えでは「Row」というクラスのインスタンスとして渡されます。ここでは、単純に1を返しているだけですが、昇順降順や列に応じて並び順のチェックを変更することもできます。

なお、今回は、<window>タグの手前に<zscript>タグを置いています。そのため、まず<zk>というタグをルートに用意し、その中に<zscript>と<window>を用意してあります。ページ内に<window>が1つあるだけというシンプルな構造なら<window>がルートで良いのですが、そうでない場合には、このように<zk>タグから始めるのが良いでしょう。

繰り返しタグによる表示の作成

この<grid>による表の作成は、多数のデータを利用する代表的なコンポーネントといって良いでしょう。もう少し「データを表示する」ということについて、この<grid>を使って掘り下げてみましょう。

先の例では、表示するすべてをタグとして記述していました。しかし、このやり方では、データと表示をうまく切り分けることができません。こうしたコンポーネントでは、データはデータでまとめて管理したいものです。

データを配列などの形でまとめておき、そこから繰り返し値を取り出して表示していく、というようなやり方ができれば、データの管理は楽になります。これは、ZUMLに用意されている繰り返しのための属性を利用することで可能となります。

```
<?page title="Index Page"?>
<zk>
  <zscript>
    String[][] datas = new String[][]{
      new String[]{"一郎","one@1.com","111-1111"},
      new String[]{"次郎","two@2.com","222-2222"},
      new String[]{"三朗","three@3.com","333-3333"}};
  </zscript>
  <window title="Welcome to ZK!" border="normal" width="500px">
    <label>※入力フォームのレイアウト</label>
    <separator />
    <grid width="90%">
      <columns sizable="true">
        <column label="名前" />
        <column label="メールアドレス" />
        <column label="電話番号" />
      </columns>
      <rows>
        <row forEach="${datas}">
          <label>${each[0]}</label>
          <label>${each[1]}</label>
          <label>${each[2]}</label>
        </row>
      </rows>
    </grid>
    <separator /><separator />
  </window>
</zk>
```

2次元配列を元にデータを取り出して繰り返し表示していく

ここでは、表示するデータはdatasという2次元配列にまとめられています。ここから順に要素を取り出し、それを各行に表示していくわけです。しかし、<rows>内には1つの<row>タグしかありません。ここでは、<row>タグ内に「forEach="${datas}"」という属性が指定されています。このforEachが、今回のポイントです。

forEachは、設定されている配列から順に要素を取り出し、「each」という変数に設定してから<row>タグを実行する、という働きをします。<row>内では、${each[0]}というようにしてeachの配列から各要素をラベルとして出力しています。こうして、行のデータ表示を行っていたのです。

このforEachのように、制御を行うための属性というのがZKのコンポーネントにはいろいろと用意されています。「if ～ else ～」に相当する「if」「unless」属性、switch文に相当する「switch」「case」属性といったものです。これらを利用することで、コードを使わずに表示の制御を行うことができるのです。

ダイナミックなデータのバインド

<grid>のように、多くのデータを扱うコンポーネントの場合、最初から静的にデータを持たせることの方が少ないでしょう。データの追加や削除を自由に行うことができないと困るので、必要に応じてダイナミックに表示データを扱えるようにしなければ実用的とは言えません。

これにはいくつかの方法があるのですが、今回の例のように複数の列を持つデータの場合、データの構造とレンダリング方法をそれぞれ<grid>が理解できなければ正しくデータを扱うことができません。

<grid>には、「model」と「rowRenderer」という属性が用意されています。前者は表示するデータのモデルで、これはListModelというインターフェイスを実装したクラスを用意して使います。またrowRendererは行のレンダリングを行うためのもので、これはRowRendererというインターフェイスを実装したクラスを用意します。では、実際にやってみましょう。

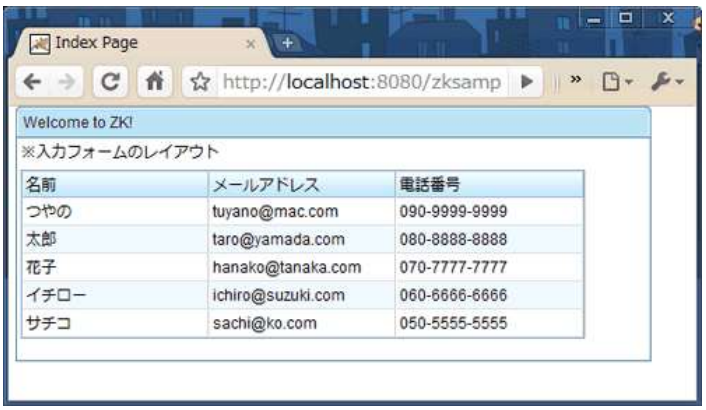
```
<?page title="Index Page"?>
<zk>
  <zscript>
    class SampleRowData {
      private String name;
      private String mail;
      private String tel;

      public SampleRowData(String s1,String s2,String s3){
        name = s1;
        mail = s2;
        tel = s3;
      }

      public String getName(){ return name; }
      public void setName(String s){ name = s; }
      public String getMail(){ return mail; }
      public void setMail(String s){ mail = s; }
      public String getTel(){ return tel; }
      public void setTel(String s){ tel = s; }
    }
    class SampleRowRenderer implements RowRenderer {
      public void render(Row row, java.lang.Object obj){
        SampleRowData data = (SampleRowData)obj;
        new Label(data.getName()).setParent(row);
        new Label(data.getMail()).setParent(row);
        new Label(data.getTel()).setParent(row);
      }
    }
    // モデルとレンダラーの作成
    ListModelList modellist = new ListModelList(new ArrayList());
    modellist.add(new SampleRowData("つやの","tuyano@mac.com","090-9999-9999"));
    modellist.add(new SampleRowData("太郎","taro@yamada.com","080-8888-8888"));
    modellist.add(new SampleRowData("花子","hanako@tanaka.com","070-7777-7777"));
    modellist.add(new SampleRowData("イチロー","ichiro@suzuki.com","060-6666-6666"));
    modellist.add(new SampleRowData("サチコ","sachi@ko.com","050-5555-5555"));
    SampleRowRenderer renderer = new SampleRowRenderer();
  </zscript>
  <window title="Welcome to ZK!" border="normal" width="500px">
    <label>※入力フォームのレイアウト</label>
    <separator />
    <grid width="90%" model="${modellist}" rowRenderer="${renderer}">
```

```
      <columns sizable="true">
        <column label="名前" />
        <column label="メールアドレス" />
        <column label="電話番号" />
      </columns>
    </grid>
    <separator /><separator />
  </window>
</zk>
```

modelとrowRendererを利用してデータを表示させる



ここでは、表示データを管理する「SampleRowData」というクラスを用意し、これを「ListModelList」というモデル用クラスに組み込みモデルとして利用しています。またSampleRowRendererというレンダラークラスを用意し、これでSampleRowDataの表示を行わせています。

まず、モデルから見ていきましょう。今回は、名前・メールアドレス・電話番号といったものを扱いますので、これらの値をプロパティにもつSampleRowDataクラスを定義しています。そして、リストのモデル（ListModel）を管理するListModelListというクラスを作成し、SampleRowDataを組み込んでいきます。

```
ListModelList modellist = new ListModelList(new ArrayList());
modellist.add(new SampleRowData("つやの","tuyano@mac.com","090-9999-9999"));
……以下略……
```

ListModelListは、モデルであるListModelと、多数のデータを管理するためのListインターフェイスを実装しています。つまり、ListModelとして扱えると同時に、Listとして保管データの管理を行うことだできるようになっているのです。インスタンス作成時に、データ保管用のArrayListを引数に渡しておき、以後、addメソッドでデータを追加しています。

続いて、レンダラーです。レンダラーは、RowRendererインターフェイスを継承します。これは「render」というメソッドをもっており、ここでデータ表示の際のレンダリング処理を行ないます。メソッドでは、引数として表示する行のRowインスタンスと、表示データ（ここではSampleRowDataインスタンス）が渡されます。表示データのオブジェクトから必要な値を取得し、それをもとに表示する内容を生成していくわけです。ここでは取り出したデータを、以下のようにしてRowに組み込んでいます。

```
new Label(表示データ).setParent(row);
```

Labelインスタンスを作成し、その「setParent」を呼び出しています。これは、このコンポーネントが組み込まれる親を設定するもので、引数にrowを指定することで、このrow内にLabelが組み込まれることになります。こうして必要なデータを元にLabelを作成し、すべてrowに組み込むことで、データの表示を作成しているのです。

ListModel（ここではListModelListですが）を利用することで、データをダイナミックに操作することができるようになります。また、必要に応じてデータを取得し表示すれば良いため、元データがデータベースにあろうがテキストファイルであろうが、関係なく表示に利用できるようになります。

まとめ

今回は、コンポーネント類をまとめるコンテナ関係と、データを扱うためのコンポーネントを中心に説明しました。前回と今回で、ZKに用意されている主なコンポーネント類については一通り理解できたのではないでしょうか。このほかに、ツリー表示やメニュー関係などがまだ残っていますが、基本的なGUIはこれで大抵作れるようになったはずです。

コンテナやデータ表示のためのコンポーネントというのは、タグだけでなく、処理の部分が多くなりがちです。しかし、ZKでは、なるべくタグだけで済むようにし、極力コードを書かずにGUIが構築できるよう設計されていることがよく分かります。例えば、表関係では横幅の調整などまで属性の設定だけで行うことができるようにしていたり、配列を元にデータを表示する場合もforEach属性で必要最小限のコードで済むようになっています。

またコードを記述する際も、Java本来のコーディングであれば常にクラスを定義するところから始まりますが、グローバル関数の利用などによりなるべくシンプルなコードで済むようになっています。

ZKはJavaベースとはいえ、各種言語でコードを記述できますから、これを「Javaのフレームワークである」と言い切ってしまう方が無理があります。「Javaも使える、スクリプト言語ベースのフレームワーク」と割り切り、気軽に利用するのが賢い使い方なのでしょう。

バックナンバー　WEB用を表示

著者プロフィール

掌田 津耶乃（ショウダ ツヤノ）

三文ライター＆三流プログラマ。主にビギナーに向けたプログラミング関連の執筆を中心に活動している。 ※現在、入門ドキュメントサイト「libro」、カード型学習サイト「CARD.tuyano.com」を公開中。またGoogle+プロフィールはこちら。

※プロフィールは、執筆時点、または直近の記事の寄稿時点での内容です