

Javaで軽快に使える「軽量フレームワーク」特集

## 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（1）

第20回

Java

WEB用を表示

 ブックマーク

ツイート 0

シェア 1

33

6

掌田 津耶乃[著]

2010/08/19 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」を紹介します。



### はじめに

以前はWebアプリケーションのためのフレームワークといえば、MVCを基本とするアプリケーションフレームワークを示すものでした。しかし、ここ数年、増えてきているのが「プレゼンテーション層のためのフレームワーク」です。データベースアクセスは、HibernateをはじめとするO/Rマッパーなど使えるフレームワークが一通り揃っています。ならば、データベースアクセスはそうしたものに任せ、Javaによるアプリケーション開発でもっとも面倒なプレゼンテーションの部分に絞ったフレームワークでいい、ということなのかもしれません。

今回取り上げる「ZK」も、こうしたプレゼンテーション層のためのフレームワークです。ZKは、Webのビジュアルをいかに簡単な作業で効率よく作成するかに絞って設計されています。独自のタグライブラリにより、シンプルなタグを記述するだけでクールなGUIを構築できます。

しかし、単にGUIを作成するだけなら、jQueryなどのAjaxライブラリでも十分でしょう。ZKは、ただGUIを記述するだけでなく、それを利用した処理をそのままレイアウトページ内にスクリプトとして記述できます。これまでの「クライアント側とサーバ側」を分けて設計したWebアプリケーションの構築とはまったく異なり、一般的なスタンドアロンアプリケーションとほとんど同じ感覚でイベント駆動による処理を作成できます。

### 対象読者

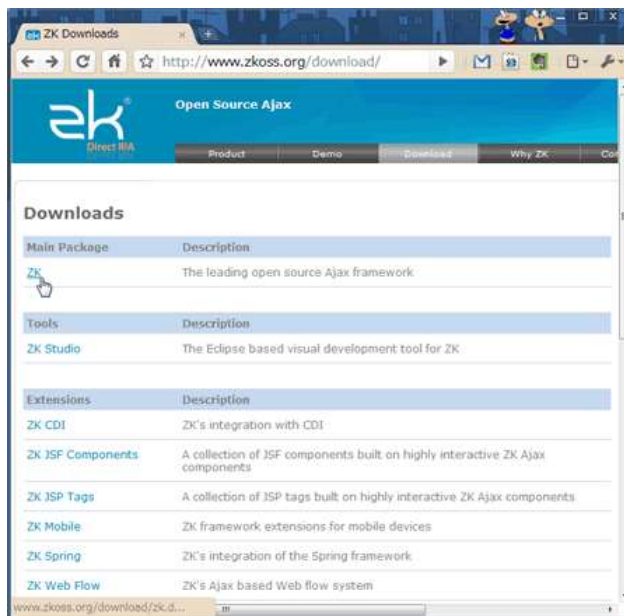
- Javaで手ごろなフレームワークを探している技術者
- 最近のフレームワークをごくざっと理解しておきたい方
- Web開発の手法がどうも気に入らない、と常々考えているJavaプログラマ

### ZKの入手

ZKは、Potix Corp.によって開発されているオープンソースのフレームワークです。現在、**ZK - Direct RIA**にて公開されています。ZKの利用には、まずサイトにアクセスし、右側にある「Download」というリンクからダウンロードページに移動してください。ダウンロードページには、ZKおよび各種のツールやエクステンション類がまとめられています。1番上にある「ZK」という項目だけあれば開発は可能です。

2010年7月現在、「5.0.3」というバージョンが最新版となっています。本書ではこれをベースに解説を行います。ZKのダウンロードページにある「zk-bin-5.0.3.zip」というファイルをダウンロードしてください。これがZKの本体となります。

ZKのWebサイトのダウンロードページ。「ZK」という項目をクリックし、zk-bin-5.0.3.zipをダウンロードする



ダウンロードしたZipファイルを展開すると、フォルダ内に「dist」「doc」という2つのフォルダが保存されているのが確認できます。doc内にはコピーライトやリリースノート、クイックスタートのドキュメントなどがまとめられています。distフォルダが、ZKの本体となります。このフォルダ内には、以下の4つのフォルダが保存されています。

- **lib** : ZKのライブラリです。ここに必要なJarファイル類がまとめられています。
- **src** : ZKのソースコードファイルをまとめたJarです。
- **WEB-INF** : TLDファイルがまとめられています。
- **xsd** : ZKに用意されているZKユーザーインターフェイスマークアップ言語（ZUML）のXSD（XMLスキーマ）です。

これらの内、実際に開発に必要となるのは「lib」フォルダだけです。それ以外のものは、特に使いません。

## Webアプリケーションを作成する

では、実際にZKを利用したWebアプリケーションを作成してみましょう。ZKには、Eclipseを利用した専用の開発ツールも用意されていますが、簡単な作業でWebアプリケーションを開発することも可能になっています。ここでは、一から作成してみることにしましょう。

Webアプリケーションは、通常いくつかのフォルダとファイルから構成されています。もっともシンプルなWebアプリケーション「zk-sample」を作成してみましょう。サーブレットコンテナの公開ディレクトリ内に、ざっと次のようなフォルダ構成を用意してください。

```
「zk-sample」フォルダ
├──「WEB-INF」フォルダ
│   ├──「classes」フォルダ
│   └──「lib」フォルダ
```

Webアプリケーションのごく基本的なフォルダ構成です。ルートとなる「zk-sample」フォルダ内に「WEB-INF」フォルダがおかれ、その中に「classes」「lib」といったフォルダが置かれる、という形です。ファイルは後から順に作成するので、まずはフォルダの構成だけ用意してください。

続いて、ZKのライブラリを組み込みます。先ほどZKの圧縮ファイルを展開したとき、dist内に「lib」フォルダが作成されました。このフォルダの中身を、すべてWEB-INF内の「lib」フォルダにコピーしてください。

なお、「lib」フォルダの中には、さらに「ext」「zkforge」といったフォルダが用意されていますが、これらはオプションとして用意されているライブラリ類です。これらも利用する場合は、フォルダ内から出して、直接「lib」フォルダ内に配置しておきます。「ext」「zkforge」フォルダの中においたままではライブラリをロードできないので注意してください。

今回、特に忘れてはならないのが「ext」内にある「bsh.jar」です。これはBeanShellというJavaのインタープリタです。今回、Javaのスクリプトを実行するのにこのライブラリが必要となるので、必ず「lib」内に入れておくようにしてください。

## web.xmlの作成

続いて、web.xmlを作成します。WEB-INF内に新たに「web.xml」ファイルを作成し、ZKを利用するために必要な情報を記述します。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  id="WebApp_ID" version="2.5">
  <display-name>ZkSample</display-name>
  <listener>
    <description>Used to cleanup when a session is destroyed</description>
    <display-name>ZK Session Cleaner</display-name>
    <listener-class>org.zkoss.zk.ui.http.HttpSessionListener</listener-class>
  </listener>
  <servlet>
    <description>The servlet loads the DSP pages.</description>
    <servlet-name>dspLoader</servlet-name>
    <servlet-class>
      org.zkoss.web.servlet.dsp.InterpreterServlet</servlet-class>
  </servlet>
  <servlet>
    <description>ZK loader for ZUML pages</description>
    <servlet-name>zkLoader</servlet-name>
    <servlet-class>
      org.zkoss.zk.ui.http.DHtmlLayoutServlet</servlet-class>
    <init-param>
      <param-name>update-uri</param-name>
      <param-value>/zkau</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet>
    <description>The asynchronous update engine for ZK</description>
    <servlet-name>auEngine</servlet-name>
    <servlet-class>
      org.zkoss.zk.au.http.DHtmlUpdateServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>dspLoader</servlet-name>
    <url-pattern>*.dsp</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>zkLoader</servlet-name>
    <url-pattern>*.zul</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>zkLoader</servlet-name>
    <url-pattern>*.zhtml</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>auEngine</servlet-name>
    <url-pattern>/zkau/*</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
    <welcome-file>index.zul</welcome-file>
  </welcome-file-list>
</web-app>
```

かなり長くなりましたが、これがZKを利用するため必要となるタグです。ZKは、サーブレットを利用して働くフレームワークです。全体を制御するためのコントローラーがサーブレットとして用意されており、これをweb.xmlに登録して利用します。

## index.zulの作成

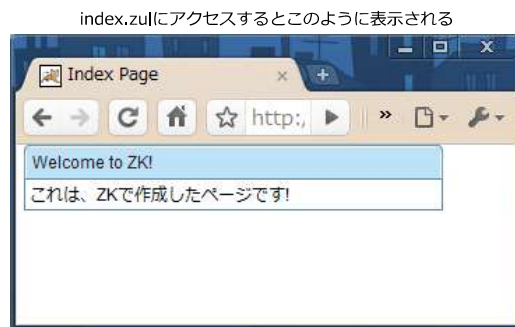
では、さっそくWebページを作成してみましょう。ZKでは、WebページはHTMLは使いません。また、JSPなども利用しません。ZKには「ZUML」というXMLベースのページ記述仕様が用意されています。XMLでは、「XUL（XML based User interface Language）」というマークアップ言語が用意されています。ZUMLも、このXULをベースにしています。

XULでは、XULコンポーネントと呼ばれるUIコンポーネントをXMLのタグとして記述していきます。ZUMLにも「ZULコンポーネント」と呼ばれるコンポーネント群が用意されており、これを記述することでGUIを構築できます。

Webアプリケーションのルート下（「zksample」フォルダ内）に、「index.zul」というファイルを新たに作成してください。そして次のようにスクリプトを記述しましょう。

```
<?page title="Index Page"?>
<window title="Welcome to ZK!" border="normal" width="300px">
  <label value="これは、ZKで作成したページです。"/>
</window>
```

これで完成です。実際にサーブレットコンテナを起動し、Webアプリケーションのルートにアクセスしてみましょう。例えばTomcatの場合、公開ディレクトリの「webapps」内に「zksample」を配置してあるならば、<http://localhost:8080/zksample/>にアクセスします。



これで、index.zulにアクセスし、ごくシンプルなメッセージが表示されます。ZUMLでは、このように「zul」という拡張子のファイルを用意し、そこに記述したZULコンポーネントを記述してページを作成します。

## ZULコンポーネントを調べる

では、ここで使われているタグについて簡単に説明をしていきましょう。まず最初に、<?page ?>というタグが記述されています。XMLの場合、冒頭に<?xml ?>といったタグが書かれますが、XUMLでは、ページの属性を定義するため、このように冒頭に<?page ?>というタグを記述できます。

この<?page ?>では、titleという属性が用意されています。これは、このWebページのタイトル（通常、HTMLでは<head>内に<title>として用意されるもの）を示すのに用いられます。

このスクリプトでは、その後に、<window>というタグが記述されています。これは、文字どおり「ウインドウ」のコンポーネントです。ブラウザからこのページにアクセスすると分かりますが、ブラウザ内に擬似的なウインドウが表示され、その中にメッセージが表示されていました。ZKでは、このようにまずウインドウを用意し、その中にコンポーネントを配置していきます。この<window>タグでは、以下の属性が用意されています。

- **title** : ウインドウのタイトル表示を示すものです。
- **border** : ボーダー（ウインドウの輪郭）を指定するものです。ここでは一般的なボーダーとして"normal"が指定されています。
- **width** : ウインドウの横幅を指定します。省略すると、ブラウザの横幅にあわせて最大限の大きさで表示されます。

この<window>タグ内に、GUIのコンポーネントが配置されます。ここでは<label>というタグが用意されていますが、これはテキストを表示するためのコンポーネントです。ここでは、valueという属性に、表示するテキストを指定しています。この<label>タグには以下のように2通りの書き方が用意されています。

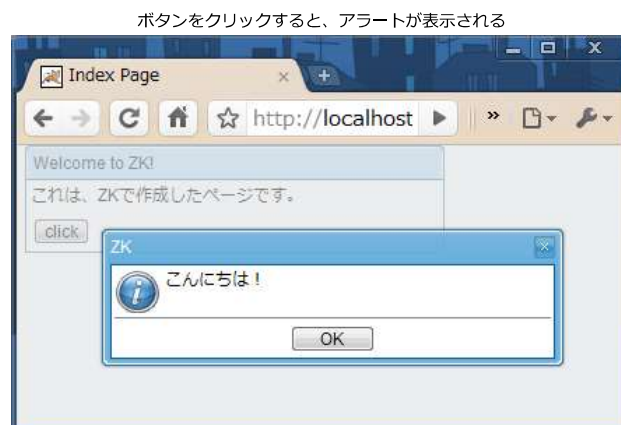
```
<label value="これは、ZKで作成したページです。"/>
<label>これは、ZKで作成したページです!</label>
```

どちらでも表示は同じです。後者の場合、テキストをダブルクォートでくくる必要がないため、例えば、「say "Hello"」なども直接記述できますが、前者では"say &quot;Hello&quot;"というようにクォートを展開しなくてははいけません。こうした細かな点で記述に違いがあるので注意してください。難しい記述法ではないので、どちらでも必要に応じて使い分けられるようにしておきましょう。

## イベント処理とアラート表示

では、ZULコンポーネントの基本的な使い方が分かったところで、次に簡単なイベント処理を行ってみましょう。プッシュボタンを用意し、クリックしたら簡単なメッセージを表示させてみます。

```
<?page title="Index Page"?>
<window title="Welcome to ZK!" border="normal" width="300px">
  <label>これは、ZKで作成したページです。</label>
  <separator/>
  <button label="click" onClick="alert(&quot;こんにちは！&quot;);" />
</window>
```



先ほどのindex.zulをこのように修正してください。アクセスすると、「click」というボタンが表示されます。これをクリックすると、ブラウザ内に「こんにちは！」と表示されたアラートウィンドウが現れます。

ここでは、<button>というタグでプッシュボタンのコンポーネントを配置しています。このタグでは、labelでボタンに表示するテキストを指定し、「onClick」でクリック時に実行する処理を設定しています。

今回は、"alert(&quot;こんにちは！&quot;);"というテキストを実行しています。これは、「alert("こんにちは！")」という処理になります。このalertを実行するだけで、メッセージをアラートウィンドウで表示できます。

このalertは、JavaScriptで多用されているalert関数ではありません。「グローバル関数」と呼ばれるもので、ZKに独自に用意されているものです（グローバル関数については後述します）。

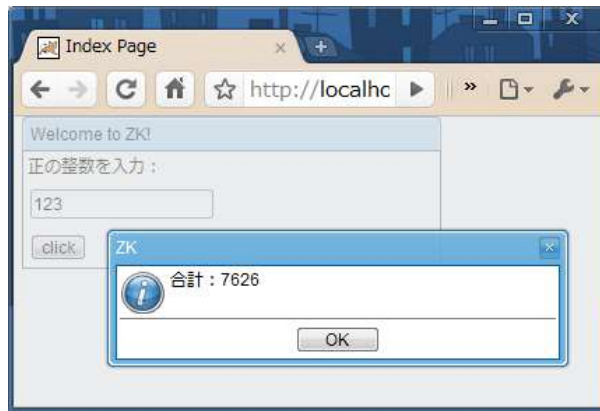
最後に、<separator/>についても触れておきましょう。これは、改行するためのタグです。ZKでは、コンポーネント類を<window>内に組み込むと、そのまま横に並べられていきます。改行する場合は、明示的に<separator/>を記述します。

## ユーザーからの入力を処理する

今度は、ユーザーからの入力を利用して処理を行ってみましょう。テキストの入力フィールドを用意し、そこに入力した値を使って計算を行ってみます。

```
<?page title="Index Page"?>
<window title="Welcome to ZK!" border="normal" width="300px">
  <label>正の整数を入力 : </label>
  <separator/>
  <textbox id="text1" />
  <separator/>
  <button label="click" onClick="doAction()" />
  <zscript>
    void doAction() {
      String s = text1.value;
      int n = Integer.parseInt(s);
      int re = 0;
      for(int i = 0; i <= n; i++) {
        re += i;
      }
      alert("合計 : " + re);
    }
  </zscript>
</window>
```

ボタンをクリックすると、ゼロから入力した数字までの合計を計算し表示する



ここでは、ゼロから入力した数字までの合計を計算して表示させています。スクリプトを見ると、<textbox>というタグが用意されています。これが、入力フィールドのコンポーネントです。また<button>では、onClick="doAction()"というように修正されています。これにより、クリックするとdoActionという、Javaのメソッドに相当するものが呼び出されるようになります。（このdoActionが何かは、この後で）

このdoActionは、その後にある<zscript>というタグの中で定義されています。この<zscript>は、スクリプト言語を使った処理を記述するためのものです。これにより、GUIを記述しているzulファイル内に処理を埋め込むことができるわけです。

## 使用言語は「Java」！？

では、ここで使われているのはどういう言語なのでしょう。よく見ると、どこかで見覚えのある感じがしませんか？ Integer.parseIntなんて、なかなか他の言語ではお目にかからないものじゃないでしょうか。そう、これは「Java」なのです！

といっても、完全なJavaというわけではありません。よく見ると、繰り返し部分がfor(int i = 0; i <= n; i++)と書かれています。これは、Javaならばfor(int i = 0; i <= n; i++)となるべきものです。

この1文を見てピンときた人は、かなりなJava通でしょう。正解は「BeanShell」です。BeanShellは、Javaのインタプリタです。Javaのソースコードをその場で実行することが可能です。それだけでなく、クラスやメソッドとして定義しなくとも、ただ行わせたい処理をそのまま記述するだけで実行できます。ここでのdoActionなどは「グローバル関数」といって、クラスなどを用意せず、いきなり関数を定義して利用することができるBeanShellの機能を使っています。

このように、Javaという言語をそのままスクリプト言語として利用できるようにしたものがBeanShellである、といってもよいでしょう。ただ単に文を書いて実行するだけでなく、面倒な定義などを極力省略して使えるように再構築されたJava、それがBeanShellです。

ZKでは、Java（BeanShell）だけでなくさまざまな言語を使って処理を記述することが可能です。<zscript>タグに「language」という属性を用意することで、Java以外の言語を使うことができます。標準で利用可能なものとしては、JavaScript、Jython、JRuby、Groovyといったものがあ

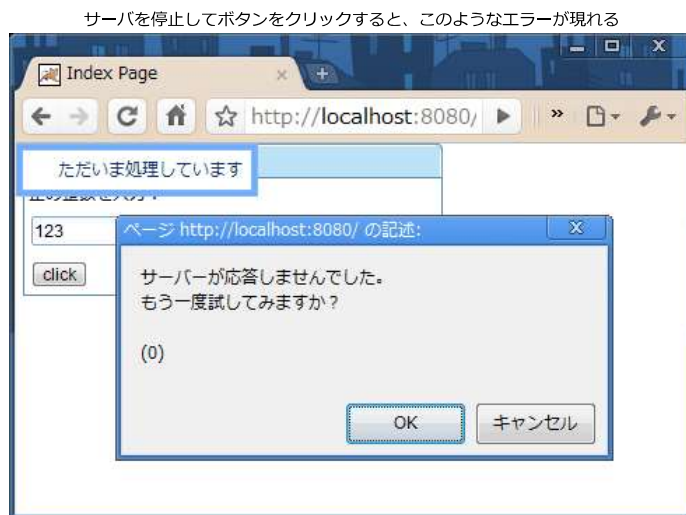
ります。これらはZKライブラリの「ext」フォルダ内に用意されているJarファイルを使うことで利用可能となります。先に、「bsh.jar」をWEB-INF/lib内に移動しましたが、これがBeanShellのライブラリファイルだったのです。

先程、alertというものを利用しましたが、これもJavaにはない機能で、BeanShellで用意されたグローバル関数です。BeanShellを利用していればこそ、こうした安直な機能の拡張が可能なのです。

## 処理はどこで実行している？

イベントを使った処理は、非常にシンプルで分かりやすいものです。特に引かかる点もないでしょう。では、ここで質問です。このdoActionの処理は、一体どこで実行されているのでしょうか。クライアント側でしょうか、それともサーバ側でしょうか。

まるでJavaScriptのような利用の仕方からして「クライアント側だろう」と思った人も多いことでしょう。しかし実際は、処理はサーバ側で実行されています。実際に、ページを表示した後、サーバを停止してからボタンをクリックしてみてください。クライアント側で処理が行われているならそのまま動くはずですが、実際にはエラーを示すアラートが表示されます。



ZKを利用する最大の利点は、ここにあります。すなわち、開発者は、その処理がサーバ側で動いているのか、それともクライアント側かといったことを一切考える必要がないのです。開発者が考えるべきは、ただ「どのイベントでどのような処理を実行するか」といったことだけです。すなわち、AWTやSwingなどのGUIアプリケーションとまったく同じ感覚で処理を実装できます。

Ajaxを利用したWebアプリケーションの開発では、常に「クライアントとサーバの処理の切り分け」を意識しなければなりません。jQueryなどのライブラリの登場により、クライアントサイドの処理を実装するための技術はここ数年で格段に進歩しました。しかし、どんなに進歩しても、いわゆるAjaxライブラリでは「ここまではクライアント、ここからサーバ側」といった切り分けを行い、それぞれで実装するしかありませんでした。それらは当然、クライアント側がJavaScript、サーバ側が何か別のサーバサイドの言語というように、異なる言語で実装されることになるでしょう（サーバサイドJavaScriptというものないわけではありませんが）。

しかし、ZKでは、サーバとクライアントの処理の切り分けなどを考えることなく、一貫してただ1つのスクリプト言語で処理を記述できます。

## チェックボックスとラジオボタン

では、その他のZULコンポーネントを使ってみましょう。まずはチェックボックスとラジオボタンからです。チェックボックスは<checkbox>、ラジオボタンは<radio>というタグとして用意されています。ただしラジオボタンの場合、グループ化をする<radiogroup>というタグと併用する必要があります。

```
<?page title="Index Page"?>
<window title="Welcome to ZK!" border="normal" width="300px">
  <label>チェックボックスとラジオボタン</label>
  <separator/>
  <checkbox onCheck="alert(self.isChecked())" label="チェック！" />
  <separator/>
  <radiogroup id="group1" onCheck="doAction()">
    <radio label="iphone"/>
    <radio label="android"/>
```



```

    <radio label="webOS"/>
</radiogroup>
<zscript>
void doAction(){
    var item = group1.selectedItem;
    alert("選択した項目 : " + item.label);
}
</zscript>
</window>

```

チェックボックスとラジオボタンの例。それぞれクリックして選択するとアラートが表示される



ラジオボタンとチェックボックスの簡単なサンプルです。それぞれクリックして選択すると、画面にアラートが表示されます。チェックボックスの場合にはチェックの状態を示す真偽値が、ラジオボタンの場合は選択した項目名がそれぞれ表示されます。

まず<checkbox>のタグを見てみましょう。ここでは、「onCheck」という属性が用意されています。これは、チェック状態が変更された時のイベント処理を行うものです。ここではalertを使って「self.isChecked()」を表示しています。「self」は、このコンポーネント自身を示すオブジェクトで、「isChecked」はチェックの状態を返すメソッドです。

続いてラジオボタンです。こちらは、<radiogroup>というグループ設定のためのタグが用意されており、この中に<radio>タグを使ってラジオボタンのコンポーネントを配置しています。ここでもチェックが変更された時の処理を行っていますが、注意すべきは<radio>ではなく<radiogroup>側にonCheckが用意されている、という点です。これで、グループ内のラジオボタンのどれをクリックしても、doActionが呼び出されるようになります。

このdoActionでは、選択されたラジオボタンのラベルをalertで表示させています。まず、グループに用意されているプロパティ「group1.selectedItem」を使い、選択されている項目を取得します。そして、そのlabelプロパティをalertで表示しています。

また、よく見るとselectedItemの値を変数に収める際、「var item」という形で変数を宣言しています。ここにも、JavaでありながらJavaではない、BeanShell特有の表現が見られます。BeanShellでは、変数はJavaのようにタイプを明確に指定するだけでなく、「var」を使って特定の型を指定しない（いわゆるバリエーション型）で利用することも可能なのです。

## スライダーとプログレスメーター

ボタンなどは、HTMLにも標準で用意されていますが、ZKではもちろんHTMLにないコンポーネントも用意されています。例えば、スライダーなどは、HTMLにはありませんが、パソコンの世界ではほぼ標準のGUIとっていいほどに多用されています。また進行状況をバーで示すプログレスメーター（プログレスバー）もほぼ標準的なGUIとっていいほどに普及しています。これらを使ってみましょう。

```

<?page title="Index Page"?>
<window title="Welcome to ZK!" border="normal" width="300px">
    <label>スライダーとプログレスメーター</label>
    <separator/>
    <label id="msg" value="スライダーの値 : 0" />
    <separator/>
    <slider id="slider" onScroll="doAction()" />
    <separator/><separator/>
    <progressmeter id="progress" value="0" width="200px"/>
    <separator/>
    <timer id="timer" delay="1000" repeats="true"
    onTimer="doTimer()" />
</zscript>

```

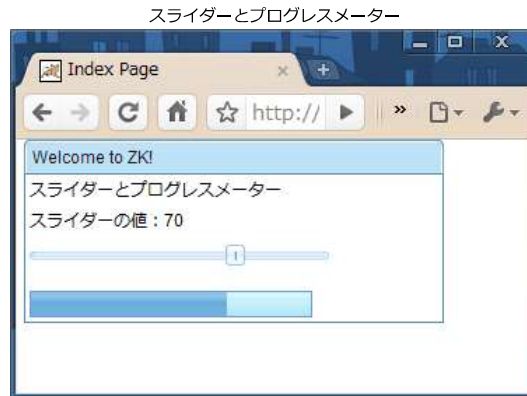


```

timer.start();

void doAction() {
    var val = slider.curpos;
    msg.value = "スライダーの値 : " + val;
}
void doTimer() {
    var val = progress.value;
    progress.value = val + 10 > 100 ? 0 : val + 10;
}
</zscript>
</window>

```



スライダーは、<slider>というタグとして用意されています。タグを置くだけで、マウスでドラッグして数値を入力するスライダーのGUIが生成されます。この<slider>には、以下のような属性が用意されています。

- **curpos** : 現在の値を示す
- **maxpos** : 最大値を示す

最小値はゼロ固定です。また省略すると、maxposは100に設定されます。マウスでドラッグすると、現在の値がリアルタイムに左下に表示され、離して値が確定すると「onScroll」イベントが発生します。ここでは「slider.curpos」で現在の値を取得し、それをラベルに表示させています。

プログレスメーターは、<progressmeter>タグとして用意されています。これは0～100の範囲内で値を設定することができ、それによってバーの長さが変更される仕組みになっています。現在の値はvalueプロパティとして用意されています。

今回は、プログレスメーターの値を一定間隔で変化させています。これは「タイマー」を利用して行っています。このタイマーも、実はZULコンポーネントなのです。<timer>というタグが、タイマーのタグです。ここで以下の属性を指定しておきます。

- **delay** : タイマーが発動するまでの待ち時間（ミリ秒単位）
- **repeats** : 繰り返しタイマーを発動させるかどうかを示す真偽値
- **onTimer** : タイマーが発動したイベントにより実行される処理

タイマーのスタートは、<zscript>タグ内にある「timer.start();」で行っています。これで、delayで指定した時間が経過するとonTimerの処理を実行するようになります。途中でタイマーを終了する場合は、「stop」メソッドを呼び出します。

## カレンダーの表示

入力関係で意外と面倒なのが日付です。入力された値が正しい日付の値となっているかどうかをチェックするのは面倒ですし、といって年月日の値を入力するフィールドを並べるのもあまりクールとは言えません。

ZUMLには、日付を入力するためのZULコンポーネントが2種類用意されています。1つはカレンダーを画面に表示するもの、もう1つはポップアップでカレンダーが現れるものです。これらを利用すれば、簡単に日付の入行を行うことができます。

```

<?page title="Index Page"?>
<window title="Welcome to ZK!" border="normal" width="300px">
    <label>カレンダー 2種</label>
    <separator/>

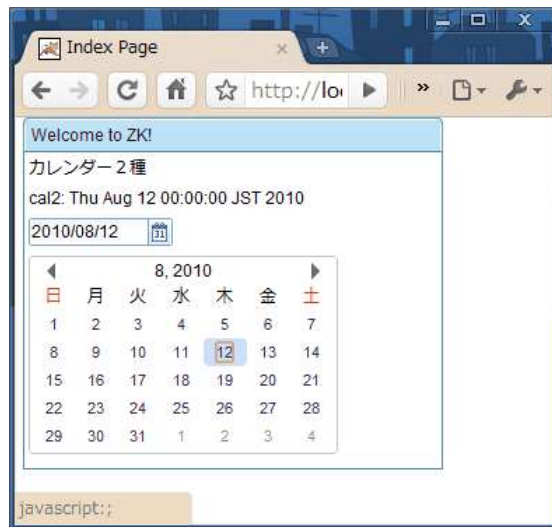
```

```

<label id="msg" value="選択した日付: " />
<separator/>
<datebox id="cal1" onChange="doAction1()" />
<separator/>
<calendar id="cal2" onChange="doAction2()" />
<separator/>
<zscript>
void doAction1() {
    var val = cal1.value;
    msg.value = "cal1: " + val;
}
void doAction2() {
    var val = cal2.value;
    msg.value = "cal2: " + val;
}
</zscript>
</window>

```

2種類のカレンダー



上にある入力フィールドは、直接日付を入力することもでき、右側にあるアイコンをクリックすることでカレンダーをポップアップで呼び出し選択することもできます。その下のカレンダーは説明は必要ないと思います。日付を選択すると、その日付が上のラベルに表示されます。

ポップアップでカレンダーを表示したのは<datebox>という特殊な入力フィールドのコンポーネントです。カレンダーの表示は、<calendar>というタグで行っています。いずれもデフォルトで今日の日付が選択された状態となっています。

日付を選択したときのイベント処理は「onChange」という属性に用意します。また選択されている日付は、valueで取り出すことができます。このvalueで得られる値は、テキストなどではなく、java.util.Dateインスタンスです。従って、valueのオブジェクトを利用してもっと柔軟な日付の処理をさせることも可能です。

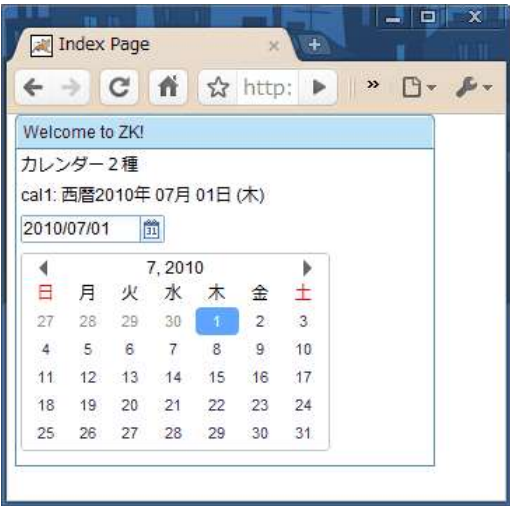
```

<zscript>
var format = new java.text.SimpleDateFormat("Gyyyy年 MM月 dd日 (E)");

void doAction1() {
    var val = cal1.value;
    msg.value = "cal1: " + format.format(val);
}
void doAction2() {
    var val = cal2.value;
    msg.value = "cal2: " + format.format(val);
}
</zscript>

```

カレンダーを選択すると、「西暦xxxx年 xx月 xx日 (曜日)」といったフォーマットで表示されるようになる



例えば<zscript>タグ部分をこのように修正してみましょう。日付を選択すると、「西暦xxxx年 xx月 xx日 (曜日)」という形で表示されます。ここではSimpleDateFormatインスタンスを用意しておき、これを使ってカレンダーのvalueをフォーマットします。

まとめ

ZKは、Ajaxを駆使したクールなGUIを、ごく簡単なタグによるzulファイルによって生成できます。しかし、「GUIがきれい」ということばかりに目を奪われてはいけません。ZKは、Webアプリケーションの開発から「クライアントとサーバ」という概念を葬り去ろうとしています。

ZKのコードを書いているとき、それがクライアント側かサーバ側かどちらで動いているのか？ということ意識することはまずありません。それは、ZKのシステムによって適当に処理されるべき問題であり、開発者が頭を悩ませるべきものではない、と考えているかのようです。

また、ZKではすべてのコードはインタープリタによって実装されます。Javaのプログラミングに慣れていると、つい「ビルドして再デプロイ」といったことを考えてしまいがちですが、ZKではスクリプトを書き換えればその場ですぐに反映されます。もちろん、JavaにもJSPがありますが、ZKではJava以外にもさまざまなスクリプト言語を利用できます。

「クールなGUI」+「シンプルなスクリプトによる処理」を考えた結果、誕生したのがZKなのでしょう。では、今回はもう少しZKの機能について掘り下げていくことにしましょう。

[バックナンバー](#)

[WEB用を表示](#)

[ブックマーク](#)

[ツイート](#) { 0 }

[シェア](#) 1

33

{ 6 }

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  
Article copyright © 2010 Syoda Tuyano, Shoeisha Co., Ltd.

PR  
PR  
PR

[ページトップへ](#)

CodeZineについて


[各種RSSを配信中](#)

プログラミングに役立つソースコードと解説記事が満載な開発者のための実装系Webマガジンです。  
掲載記事、写真、イラストの無断転載を禁じます。  
記載されているロゴ、システム名、製品名は各社及び商標権者の登録商標あるいは商標です。




- [ヘルプ](#)
- [スタッフ募集！](#)
- [人事](#)
- [書籍・ソフトを買う](#)
- [広告掲載のご案内](#)
- [メンバー情報管理](#)
- [教育ICT](#)
- [電験3種対策講座](#)

<a href="#">著作権・リンク</a>	<a href="#">メールバックナンバー</a>	<a href="#">マネー・投資</a>	<a href="#">電験3種ネット</a>
<a href="#">免責事項</a>	<a href="#">マーケティング</a>	<a href="#">ネット通販</a>	<a href="#">第二種電気工事士</a>
<a href="#">会社概要</a>	<a href="#">エンタープライズ</a>	<a href="#">イノベーション</a>	
<a href="#">サービス利用規約</a>		<a href="#">セールス</a>	
<a href="#">プライバシーポリシー</a>		<a href="#">クリエイティブ</a>	
		<a href="#">プロダクトマネジメント</a>	
		<a href="#">ホワイトペーパー</a>	

 [メンバーメニュー](#)

|

 [ログアウト](#)