NoVDL: Write your JSF views in pure Java

By Roger Keays, 17 March 2016

NoVDL is a View Description Language that makes your JSF views more robust by implementing them in Java. By using Java to write your views you automatically gain many features.

- type safety
- compile-time code verification
- stack traces that actually make sense
- ide autocomplete support
- ide refactoring support
- java static methods for composite components
- java function composition for decorating views
- java inheritance for decorating views
- java object references to access components
- java method references to invoke action listeners
- java namespacing for component libraries
- high performance
- less reliance on flaky EL expressions
- less reliance on managed beans
- no XML configuration, taglibs, or magic annotations
- one language to learn instead of three (Java + XML + EL)

Example View

```
public class OrderForm extends BasePage implements Activity {
```

```
    @Override
    public void buildView(FacesContext faces, UIViewRoot root) {
        List<Product> products = ProductDAO.getAllProducts();
        UIPanelGroup main =
          h_panelGroup().children(
            f_verbatim("<h1>Order Form</h1>"),
            h_form().id("order").children(
              h_dataTable().id("products").value(products).var("product").widtl
                h_column().header(f_verbatim("Description")).children(
                  h_outputLink().id("link").valuex("${product.page.linkURL}").c
                    h_outputText().valuex("${product.description}")
                  )
                ),
                h_column()
                    .header(f_verbatim("Quantity"))
                    .children(h_inputText().id("quantity").converter(new Intege
                    .footer(h_commandButton().id("submit").value("Create Order'
                                    .actionListener(createOrder()))
                )
              )
            )
          );


        /*
         * If your master template is implemented in java you'll create a
         * separate function to build it, e.g.
         *
         * build_master_view(faces, root, main);
         *
         * NoVDL also includes a function to execute a facelet which can
         * then include the partial view you created here using a
         * binding=".." attribute.
         */
        build_from_facelet(faces, root, getMasterTemplate());
    }
}
```

Mapping Routes

The current implementation maps the view id to its Java implementation by converting the url path to a class name. E.g.

```
http://localhost/com/example/Demo.xhtml
```

is routed to the class

```
com.example.Demo
```

If you need different functionality, you could rewrite get_view_class_name() and recompile NoVDL. If there is enough demand, I'll make this function configurable.

Value Expressions

NoVDL allows value expressions to be set by using the setter method which ends in *x*. e.g.

```
h_outputText().value("Hello World"); // or
h_outputText().valuex("${bean.greeting}");
```

This was chosen partly because the NoVDL components are automatically generated from the JSF API, and partly because EL is discouraged in NoVDL. Ideally, we'd like to be able to implement ValueExpressions using lambda functions.

Composite Components with a Static Factory

It is very easy to make composite components using a static factory method.

```
public static UIPanel create_data_grid(Data current) {
    HPanelGrid grid = h_panelGrid().id("ox_related").columns(4);
    for (Match match : data.getMatches()) {
        HPanelGroup panel = h_panelGroup().id("ox_related_" + match.getId());
        if (match.getThumbnail() != null) {
            panel.children(
                h_outputLink().value(match.getLinkURL()).children(
                    h_graphicImage().value(match.getThumbnail())
                )
            );
        }
        panel.children(
            f_verbatim("<p>"),
            h_outputLink().value(match.getLinkURL()).children(
                h_outputText().value(match.getTitle())
            ),
            f_verbatim("</p>")
        );
```

```
        grid.getChildren().add(panel);
    }
    return grid;
}
```

This is a good example of why XML is a questionable choice for user interfaces. Is it declarative code or is it procedural? Actually, it's a mixture, and there is not much advantage to declarative languages when you start to mix in procedural (or functional) code.

Decorating Views using Functions

Same idea as above. e.g.

```
public static void decorate_view(FacesContext faces, UIViewRoot root,
        UIHead head, UIComponent content) {
    root.getChildren().add(head);
    root.getChildren().add(
        h_body().children(
            content
        );
}
```

Decorating Views using Inheritance

Function composition is preferred over inheritance. If you like inheritance, all you need to do is make an abstract superclass that defines the sections of your view. e.g.

```
public abstract class MasterView implements Activity {

    @Override
    public void buildView(FacesContext faces, UIViewRoot root) {
        root.getChildren().add(build_head(faces));
        root.getChildren().add(
            h_body().children(
                build_content(faces)
            );
    }

    public abstract UIHead build_head(FacesContext faces);
    public abstract UIComponent build_content(FacesContext faces);
}
```

Mixing Facelets and Java Views

Use the function

```
build_from_facelet(faces, root, template);
```

to execute a Facelet from your buildView function. Inside that template, you'll have to reference the components you built using NoVDL. e.g.

```
<h2>Current Data</h2>
<div class="web_noprint">

  <!-- datagrid is prebuilt with NoVDL -->
  <h:panelGroup binding="${someBean.datagrid}"/>
</div>
```

Editing Views on the Fly

Facelets lets you edit views on the fly without recompiling or restarting theapplication. You should be able to use HotSwapAgentor JRebel to achieve the same thing with NoVDL.

Installation

NoVDL is automaticallydetected when you add the following dependency. View IDs that cannot be mapped to Java implementations will fall back on the default view decoration language (normally Facelets).

```
<dependency>
  <groupId>au.com.ninthavenue.jamaica.faces</groupId>
  <artifactId>novdl</artifactId>
  <packaging>jar</packaging>
  <version>1.0</version>
</dependency>
```

Note: NoVDL requires Java 8.

Get the Source

https://github.com/rogerkeays/novdl

Spend some time experimenting with NoVDL and let me know how it works for you.

About Roger Keays

Roger Keays is an artist, an engineer, and a student of life. He has no fixed addressand has leftfootprints on 40-something different countries around the world. Roger is addicted to surfing. His other interests are music, psychology, languages, the proper use of semicolons, and finding good food.