UI Frameworks for GWT

Dr. Lofi Dewanto
https://lofidewanto.blogspot.de

Agenda

- Widget-based
- Elemental2-based
- Canvas-based

Widget-based

- Layout / “Grid”
  - Each framework has its own layout mechanism
- Widget class: foundation of all UI classes
  http://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/Widget.html
- Composite Widgets: combination of other widgets into a new component
  https://gist.github.com/lofidewanto/77fa30f220f9ec54cdecf86474e6bc4b
- Using DOM
  - Provide a feature in your user interface that GWT does not support
  - Write a new Widget class
  - Access an HTML element defined directly in the host page
  - Handle browser Events at a low level
  - Perform some filtering or other processing on an HTML document loaded into the browser

Widget-based: UIBinder

- Helps productivity and maintainability — it’s easy to create UI from scratch or copy/paste across templates;
- Makes it easier to collaborate with UI designers who are more comfortable with XML, HTML and CSS than Java source code;
- Provides a gradual transition during development from HTML mocks to real, interactive UI;
- Encourages a clean separation of the aesthetics of your UI (a declarative XML template) from its programmatic behavior (a Java class);

Widget-based: UIBinder

- Performs thorough compile-time checking of cross-references from Java source to XML and vice-versa;
- Offers direct support for internationalization that works well with GWT’s i18n facility;
- Encourages more efficient use of browser resources by making it convenient to use lightweight HTML elements rather than heavier-weight widgets and panels;
- **“Android-like” UI elements in XML**

Widget-based: UIBinder

```
 | under the License.
-->
<ui:UiBinder xmlns:ui="urn:ui:com.google.gwt.uibinder"
             xmlns:g="urn:import:com.google.gwt.user.clie
             xmlns:m="urn:import:gwt.material.design.clie
    <g:HTMLPanel>
        <m:MaterialHeader top="0"...>

        <!-- Panel -->
        <m:MaterialPanel>
            <m:MaterialRow>
                <m:MaterialColumn grid="s12 m4">
                    <m:MaterialTitle title="Material is t
                                     description="The met
                </m:MaterialColumn>
                <m:MaterialColumn grid="s12 m4">
                    <m:MaterialTitle title="Bold, graphic
                                     description="Element
                </m:MaterialColumn>
                <m:MaterialColumn grid="s12 m4">
                    <m:MaterialTitle title="Motion provid
                                     description="Motion
                </m:MaterialColumn>
            </m:MaterialRow>
            <m:MaterialRow...>
            <m:MaterialRow...>
        </m:MaterialPanel>
```

GWTMaterialDesign

```
-->
<ui:UiBinder xmlns:ui="urn:ui:com.google.gwt.uibinder"
             xmlns:g="urn:import:com.google.gwt.user.client.ui"
             xmlns:b="urn:import:org.gwtbootstrap3.client.ui"
             xmlns:select="urn:import:org.gwtbootstrap3.extras.select.client.ui">
    <ui:style>
        .margin {
            margin-top: 20px;
            margin-left: 20px;
            margin-right: 20px;
        }
    </ui:style>

    <g:ScrollPanel>
        <b:Container fluid="true" addStyleNames="{style.margin}">
            <b:Row>
                <b:Column size="MD_12">
                    <b:Row>
                        <!-- Filter Panel -->
                        <b:Panel>
                            <b:PanelHeader>
                                <b:Heading size="H4" text="Filter"/>
                            </b:PanelHeader>
                            <b:PanelBody>
                                <b:Container>
                                    <b:Form type="HORIZONTAL"...>
                                </b:Container>
                            </b:PanelBody>
                        </b:Panel>

                        <!-- Content Panel -->
                        <b:Panel>
```

GWTBootstrap3

Widget-based: Examples

- Standard GWT, included in GWT libs
- GWTBootstrap3
- GWTMaterialDesign
- GXT
- SmartGWT
- ...

Widget-based: Task

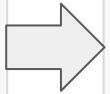
- Create a simple GWT app with GWTMaterialDesign or GWTBootstrap3!
- Example:
  https://github.com/gwtboot/gwt-boot-samples/tree/master/gwt-boot-sample-ui-gwtmaterial
  https://github.com/gwtboot/gwt-boot-samples/tree/master/gwt-boot-sample-ui-gwtbootstrap3

Elemental2-based

- Elemental2 provides type checked access to all browser APIs for Java code. This is done by using closure extern files and generating JsTypes, which are part of the new JsInterop specification that is both implemented in GWT and J2CL (Java to Closure).
- Closure Tools are collection of optimized JS tools
  - https://developers.google.com/closure
- Elemento simplifies working with Elemental2
  - https://github.com/hal/elemento

Elemento: Instead of Document.createElement()...

```html
<section class="main">
    <input class="toggle-all" type="checkbox">
    <label for="toggle-all">Mark all as complete</label>
    <ul class="todo-list">
        <li>
            <div class="view">
                <input class="toggle" type="checkbox" checked>
                <label>Taste Elemento</label>
                <button class="destroy"></button>
            </div>
            <input class="edit">
        </li>
    </ul>
</section>
```



```
import static org.jboss.gwt.elemento.core.Elements.*;
import static org.jboss.gwt.elemento.core.InputType.checkbox;
import static org.jboss.gwt.elemento.core.InputType.text;

HTMLElement section = section().css("main")
        .add(input(checkbox).id("toggle-all").css("toggle-all"))
        .add(label()
                .apply(l -> l.htmlFor = "toggle-all")
                .textContent("Mark all as complete"))
        .add(ul().css("todo-list")
                .add(li()
                        .add(div().css("view")
                                .add(input(checkbox)
                                        .css("toggle")
                                        .apply(cb -> cb.checked = true))
                                .add(label().textContent("Taste Elemento"))
                                .add(button().css("destroy")))
                        .add(input(text).css("edit"))))
        .asElement();
```

Elemento: HTML Template

```html
<!doctype html>
<html lang="en">
<head>
    <link rel="stylesheet" href="<path-to>/node_modules/todomvc-common/base.css">
    <link rel="stylesheet" href="<path-to>/node_modules/todomvc-app-css/index.css">
</head>
<body>
<section data-element="todos" class="todoapp">
    <header class="header">
        <h1>todos</h1>
        <input data-element="newTodo" class="new-todo" placeholder="What needs to be done?" autofocus>
    </header>
    <section data-element="main" class="main">
        <input data-element="toggleAll" class="toggle-all" id="toggle-all" type="checkbox">
        <label for="toggle-all">Mark all as complete</label>
        <ul data-element="list" class="todo-list">
            <!-- Todo items are mapped to an extra template class -->
        </ul>
    </section>
    <footer data-element="footer" class="footer">
        <span data-element="count" class="todo-count"><strong>0</strong> item left</span>
        <ul class="filters">
            <li><a data-element="all" href="#/">All</a></li>
            <li><a data-element="active" href="#/active">Active</a></li>
            <li><a data-element="completed" href="#/completed">Completed</a></li>
        </ul>
        <button data-element="clearCompleted" class="clear-completed">Clear completed</button>
    </footer>
</section>
[...]
</body>
</html>
```

Todo.html

```java
@Templated("Todo.html#todos")
abstract class ApplicationElement implements IsElement<HTMLElement> {

    // @formatter:off
    static ApplicationElement create(TodoItemRepository repository) {
        return new Templated_ApplicationElement(repository);
    }

    abstract TodoItemRepository repository();
    // @formatter:on


    @DataElement HTMLInputElement newTodo;
    @DataElement HTMLElement main;
    @DataElement HTMLInputElement toggleAll;
    @DataElement HTMLElement list;
    @DataElement HTMLElement footer;
    @DataElement HTMLElement count;
    @DataElement("all") HTMLElement filterAll;
    @DataElement("active") HTMLElement filterActive;
    @DataElement("completed") HTMLElement filterCompleted;
    @DataElement HTMLButtonElement clearCompleted;
```

Template class

Elemental2-based

- DominoUI
- VueGWT
- GWTReact bzw. React4Java
- Errai
- ...

Elemental2-based: Task

- Create a simple GWT app with VueGWT or Elemento Template!
- Example:
  https://github.com/gwtboot/gwt-boot-samples/tree/master/gwt-boot-sample-ui-vue-gwt
  https://github.com/gwtboot/gwt-boot-samples/tree/master/gwt-boot-sample-elemento-template

Canvas-based

- HTML Canvas drawing
- Example from Animatron: https://www.slideshare.net/gwtcon/ui-framework-development-using-gwt-and-html-canvas-by-iarosla-kobyliukh

References

- http://www.gwtproject.org/doc/latest/DevGuideUiCustomWidgets.html
- http://www.gwtproject.org/doc/latest/DevGuideUiBinder.html