Corso di “Sviluppo di applicazioni Web”

Giovanni Grasso

GWT UI

Building the User Interface

GWT user interface classes are similar to those in existing UI frameworks such as Swing and SWT except that the widgets are rendered using dynamically-created HTML rather than pixel-oriented graphics.

Widgets and Panels are client-side Java classes used to build user interfaces.

You construct user interfaces in GWT applications using widgets that are contained within panels.

Widgets and panels work the same way on all browsers; by using them, you eliminate the need to write specialized code for each browser.

- Widgets allow you to interact with the user.
  - Button, TextBox, Tree,... (see showcase)

Understanding Layout

Panels in GWT are much like their layout counterparts in other user interface libraries.

The main difference lies in the fact that GWT panels use HTML elements such as DIV and TABLE to layout their child widgets.

- RootPanel
  - always at the top of the containment hierarchy.
  - The default RootPanel wraps the HTML document's body, and is obtained by calling RootPanel.get().
  - If you need to get a root panel wrapping another element in the HTML document, you can do so using RootPanel.get(String).

TabPanel, Horizontal / Vertical Panel, DockPanel ...

(see showcase)

Creating Custom Widgets

GWT makes it easy to create custom user interface widgets extending the Composite class.

A composite is a specialized widget that can contain another component (typically, a panel)

The Composite widget has to know which widgets it wraps.

You MUST call the initWidget(Widget widget) method, (only once or you will get an error).

The difference between the Composite and simply extending another 'real' widget is that the Composite hides all the wrapped widget's methods and properties.

You can easily combine groups of existing widgets into a composite that is itself a reusable widget.

Some of the UI components provided in GWT are composites: for example, the TabPanel (a composite of a TabBar and a DeckPanel) and the SuggestBox.

Events and Listeners

Events in GWT use the *listener interface* model

A listener interface defines one or more methods that the widget calls to announce an event.

A class wishing to receive events of a particular type implements the associated listener interface and then passes a reference to itself to the widget to *subscribe* to a set of events.

The Button class, for example, publishes click events. The associated listener interface is ClickListener.

```
Button b = new Button("Click Me");
  b.addClickListener(new ClickListener() {
    public void onClick(Widget sender) {
      // handle the click event
    }
  });
```

Image Bundles

An *image bundle* is used to improve application performance by reducing the number of round trip HTTP requests to the server to fetch images.

GWT can package many image files into a single large file to be downloaded from the server and managed as a Java object.

To define an image bundle, the user needs to extend the ImageBundle interface.

```
public interface Images extends ImageBundle {
    AbstractImagePrototype logo();
  }
```

Each method must have the following characteristics:

- The method takes no parameters
- The method has a a return type of AbstractImagePrototype
- The method may have an optional gwt.resource metadata tag which specifies the name of the image file in the module's classpath