**Custom Widgets**

GWT makes it easy to create custom user interface elements. There are three general strategies to follow:

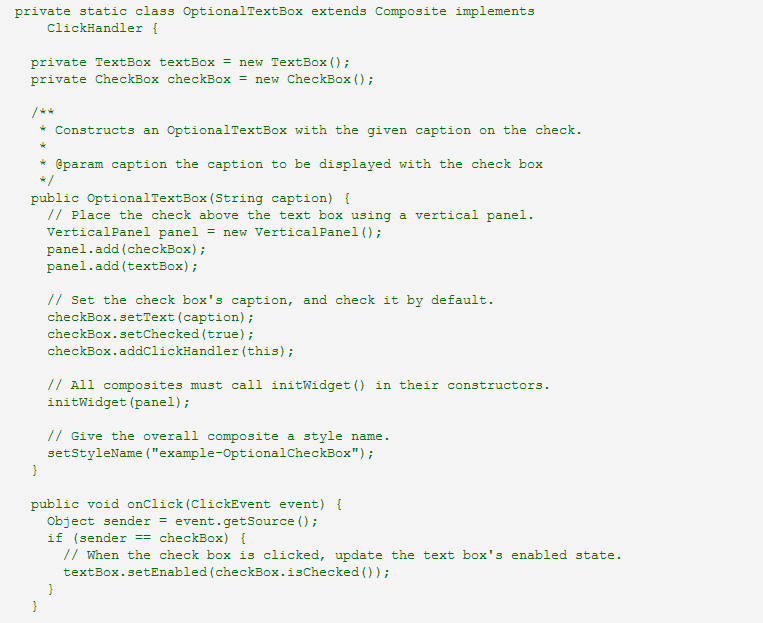
* [Create a widget that is a composite](http://www.gwtproject.org/doc/latest/DevGuideUiCustomWidgets.html#composite) of existing widgets.
* [Create an entirely new widget](http://www.gwtproject.org/doc/latest/DevGuideUiCustomWidgets.html#new) written in the Java language.
* [Create a widget that wraps JavaScript](http://www.gwtproject.org/doc/latest/DevGuideUiCustomWidgets.html#javascript) using [JSNI](http://www.gwtproject.org/doc/latest/DevGuideCodingBasics.html#DevGuideJavaScriptNativeInterface) methods.

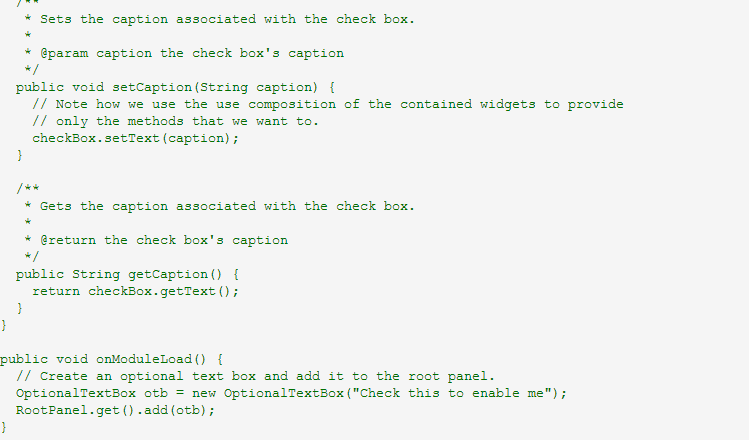
**Building Composites**

The most effective way to create new widgets is to extend the [Composite](http://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/Composite.html) class. A composite is a specialized widget that can contain another component (typically, a[Panel](http://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/Panel.html)) but behaves as if it were its contained widget. 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](http://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/TabPanel.html)(a composite of a TabBar and a DeckPanel) and the [SuggestBox](http://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/SuggestBox.html).

Rather than create complex widgets by subclassing [Panel](http://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/Panel.html) or another [Widget](http://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/Widget.html) type, it's better to create a composite because a composite usually wants to control which methods are publicly accessible without exposing those methods that it would inherit from its Panel superclass.





**From Scratch in Java Code :**

It is also possible to create a widget from scratch, although it is trickier since you have to write code at a lower level. Many of the basic widgets are written this way, such as [Button](http://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/Button.html) and [TextBox](http://google-web-toolkit.googlecode.com/svn/javadoc/latest/com/google/gwt/user/client/ui/TextBox.html). Please refer to the implementations of these widgets to understand how to create your own.

To understand how to create your own, refer to the implementations of these widgets in the com.google.gwt.user.client.ui package. The source code is in gwt-user.jar.

**Using JavaScript :**

When implementing a custom widget that derives directly from the [Widget](http://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/Widget.html) base class, you may also write some of the widget's methods using JavaScript. This should generally only be done as a last resort, as it becomes necessary to consider the cross-browser implications of the native methods that you write, and also becomes more difficult to debug. For an example of this pattern in practice, see the [TextBox](http://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/TextBox.html) widget and the underlying JavaScript implementation of some of its methods in the TextBoxImpl class. You should use [deferred binding](http://www.gwtproject.org/doc/latest/DevGuideCodingBasics.html#DevGuideDeferredBinding) to isolate browser specific code.