Cell widgets (data presentation widgets) are high-performance, lightweight widgets composed of Cells for displaying data.

Examples are [**lists**](http://gwt.google.com/samples/Showcase/Showcase.html#!CwCellList), [**tables**](http://gwt.google.com/samples/Showcase/Showcase.html#!CwCellTable), [**trees**](http://gwt.google.com/samples/Showcase/Showcase.html#!CwCellTree)and [**browsers**](http://gwt.google.com/samples/Showcase/Showcase.html#!CwCellBrowser).

These widgets are designed to handle and display very large sets of data quickly.

A cell widget renders its user interface as an HTML string, using innerHTML instead of traditional DOM manipulation.

This design follows the **flyweight pattern** where data is accessed and cached only as needed and passed to flyweight Cell objects.

A cell widget can accept data from any type of data source.

The data model handles asynchronous updates as well as push updates. When you change the data, the view is automatically updated.

Cells are the basic blocks of a user interface and come in a variety of [available cell types](http://www.gwtproject.org/doc/latest/DevGuideUiCellWidgets.html#available). They render views of data, interpret browser events and can be selected.

A Cell has a type based on the data that the cell represents; for example, DatePickerCell is a Cell<Date> that represents a Date and allows the user to select a new Date.

Cells must implement a render method that renders the typed value as an HTML string. In addition, cells can override onBrowserEvent to act as a flyweight that handles events that are fired on elements that were rendered by the cell.

For example, in the [CellList example](http://gwt.google.com/samples/Showcase/Showcase.html" \l "!CwCellList) of the Showcase, every selectable data record is rendered by a single Cell instance. Notice that the data that a single cell represents can be a composition of different data fields from the data source. In this example, the cell holds data of type ContactInfo, which represents a contact, including name, address and picture.

In the [CellTable example](http://gwt.google.com/samples/Showcase/Showcase.html" \l "!CwCellTable), a different Cell is used to render each Column of a row. The five columns in this example present data from a Boolean and four strings.

1. [Cell Widgets](http://www.gwtproject.org/doc/latest/DevGuideUiCellWidgets.html#Cell_Widgets)
   1. [Demos and Code Examples](http://www.gwtproject.org/doc/latest/DevGuideUiCellWidgets.html#demos)
   2. [Creating a CellList and Setting Data](http://www.gwtproject.org/doc/latest/DevGuideUiCellWidgets.html#celllist)
   3. [Creating a CellTable](http://www.gwtproject.org/doc/latest/DevGuideUiCellWidgets.html#celltable)
   4. [Creating a CellTree](http://www.gwtproject.org/doc/latest/DevGuideUiCellWidgets.html#celltree)
   5. [Creating a CellBrowser](http://www.gwtproject.org/doc/latest/DevGuideUiCellWidgets.html#cellbrowser)
2. [Cells](http://www.gwtproject.org/doc/latest/DevGuideUiCellWidgets.html#Cells)
   1. [Available Cell Types](http://www.gwtproject.org/doc/latest/DevGuideUiCellWidgets.html#available)
   2. [Creating a Custom Cell](http://www.gwtproject.org/doc/latest/DevGuideUiCellWidgets.html#custom-cell)
3. [Selection, Data and Paging](http://www.gwtproject.org/doc/latest/DevGuideUiCellWidgets.html#Selection_Data_Paging)
   1. [Adding Selection Support](http://www.gwtproject.org/doc/latest/DevGuideUiCellWidgets.html#selection)
   2. [Providing Dynamic Data](http://www.gwtproject.org/doc/latest/DevGuideUiCellWidgets.html#data-provider)
   3. [Adding Paging Controls](http://www.gwtproject.org/doc/latest/DevGuideUiCellWidgets.html#paging)
   4. [Updating a Database from Changes in a Cell](http://www.gwtproject.org/doc/latest/DevGuideUiCellWidgets.html#updating-database)

NOTE: CellPanel is not a cell widget. CellPanel is an abstract base class for GWT Panel Widgets that are implemented using a table element.

**Cells**

Available Cell Types

GWT offers a number of concrete Cell implementations that you can use immediately. See the [Cell Sampler](http://gwt.google.com/samples/Showcase/Showcase.html#!CwCellSampler)for examples.

Text

[TextCell](http://www.gwtproject.org/javadoc/latest/com/google/gwt/cell/client/TextCell.html) - A non-editable cell that displays text

[ClickableTextCell](http://www.gwtproject.org/javadoc/latest/com/google/gwt/cell/client/ClickableTextCell.html) - A text field; clicking on the cell causes its ValueUpdater to be called

[EditTextCell](http://www.gwtproject.org/javadoc/latest/com/google/gwt/cell/client/EditTextCell.html) - A cell that initially displays text; when clicked, the text becomes editable

[TextInputCell](http://www.gwtproject.org/javadoc/latest/com/google/gwt/cell/client/TextInputCell.html) - A field for entering text

Buttons, Checkboxes and Menus

[ActionCell<C>](http://www.gwtproject.org/javadoc/latest/com/google/gwt/cell/client/ActionCell.html) - A button that takes a delegate to perform actions on mouseUp

[ButtonCell](http://www.gwtproject.org/javadoc/latest/com/google/gwt/cell/client/ButtonCell.html) - A button whose text is the data value

[CheckboxCell](http://www.gwtproject.org/javadoc/latest/com/google/gwt/cell/client/CheckboxCell.html) - A checkbox that can be checked or unchecked

[SelectionCell](http://www.gwtproject.org/javadoc/latest/com/google/gwt/cell/client/SelectionCell.html) - A drop-down menu for selecting one of many choices

Dates

[DateCell](http://www.gwtproject.org/javadoc/latest/com/google/gwt/cell/client/DateCell.html) - A date that conforms to a specified date format

[DatePickerCell](http://www.gwtproject.org/javadoc/latest/com/google/gwt/cell/client/DatePickerCell.html) - A date picker that displays a month calendar in which the user can select a date

Images

[ImageCell](http://www.gwtproject.org/javadoc/latest/com/google/gwt/cell/client/ImageCell.html) - A cell used to render an image URL

[ImageResourceCell](http://www.gwtproject.org/javadoc/latest/com/google/gwt/cell/client/ImageResourceCell.html) - A cell used to render an ImageResource

[ImageLoadingCell](http://www.gwtproject.org/javadoc/latest/com/google/gwt/cell/client/ImageLoadingCell.html) - A cell used to render an image URL. A loading indicator is initially displayed

Numbers

[NumberCell](http://www.gwtproject.org/javadoc/latest/com/google/gwt/cell/client/NumberCell.html) - A number that conforms to a specified number format

Compositions

[CompositeCell<C>](http://www.gwtproject.org/javadoc/latest/com/google/gwt/cell/client/CompositeCell.html) - A composition of multiple Cells.

Decorators

[IconCellDecorator<C>](http://www.gwtproject.org/javadoc/latest/com/google/gwt/cell/client/IconCellDecorator.html) - A decorator that adds an icon to another Cell

***Creating a Custom Cell***

If you want more control, you can subclass AbstractCell, or you can implement the Cell interface directly to define how your Cell is rendered and how it responds to events. See the instructions in the [Creating Custom Cells Dev Guide](http://www.gwtproject.org/doc/latest/DevGuideUiCustomCells.html) for detailed information.

**Demo** - [CwCellList example](http://gwt.google.com/samples/Showcase/Showcase.html" \l "!CwCellList) shows a CellList<ContactInfo> (on the left). Each list item is a custom type ContactCell<ContactInfo>. The right-hand widget is a normal Composite widget that renders the data for a selected contact.

**Selection, Data and Paging**

Adding Selection Support

The [SelectionModel](http://www.gwtproject.org/javadoc/latest/com/google/gwt/view/client/SelectionModel.html) is a simple interface that views use to determine if an item is selected. Cell widgets provide several selection models for selecting the children of a node: DefaultSelectionModel, NoSelectionModel, SingleSelectionModel and MultiSelectionModel. One of these is likely to fit your need.

For demonstrations of selection, the [CwCellList](http://gwt.google.com/samples/Showcase/Showcase.html" \l "!CwCellList) widget creates a SingleSelectionModel, whereas CwCellTable implements a MultiSelectionModel using checkboxes.

Views or application code can call setSelected() to select an item. Views call isSelected() to determine if an item is selected. Views also subscribe to the SelectionModel so they can be informed of selection changes that arrive from outside the view. In fact, you can extend DefaultSelectionModel and override isDefaultSelected().

This simple approach offers a lot of flexibility. A complex implementation can handle "select all" across multiple pages using a boolean to indicate that everything is selected, and then keep track of negative selections.

By using a subscription model, we can link selection across multiple views. If multiple views subscribe to a single SelectionModel, then selecting a row in one view will select the row in other views. This behavior is optional and can be avoided by using a single SelectionModel instance per view.

**Demo** - [CwCellList example](http://gwt.google.com/samples/Showcase/Showcase.html" \l "!CwCellList) shows a cell widget that has a SelectionModel added to it. Clicking on an item selects it.

**To Add a Selection to a Cell Widget:**

1. Create a cell widget.
2. Choose a standard [SelectionModel](http://www.gwtproject.org/javadoc/latest/com/google/gwt/view/client/SelectionModel.html) (or roll your own).
3. Add this SelectionModel to the cell widget using[setSelectionModel](http://www.gwtproject.org/javadoc/latest/index.html?com/google/gwt/user/cellview/client/AbstractHasData.html)(SelectionModel).
4. Create a [SelectionChangeEvent.Handler](http://www.gwtproject.org/javadoc/latest/com/google/gwt/view/client/SelectionChangeEvent.Handler.html) implementing onSelectionChange.
5. Add this handler to the SelectionModel using [addSelectionChangeHandler](http://www.gwtproject.org/javadoc/latest/com/google/gwt/view/client/SelectionModel.html" \l "addSelectionChangeHandler(com.google.gwt.view.client.SelectionChangeEvent.Handler)).

**Code Example** - The example of SelectionModel below is available at[CellListExample.java](http://google-web-toolkit.googlecode.com/svn/trunk/user/javadoc/com/google/gwt/examples/cellview/CellListExample.java).

