The Google Web Toolkit (GWT): Introduction to Cell Widgets  
(GWT 2.5 Version)

Originals of Slides and Source Code for Examples:  
http://courses.coreservlets.com/Course-Materials/gwt.html

Customized Java EE Training: http://courses.coreservlets.com/ 
GWT, Java 7 and 8, JSF 2, PrimeFaces, HTML5, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, REST, Hadoop, Android. 
Developed and taught by well-known author and developer. At public venues or onsite at your location.

For live Ajax & GWT training, see training courses at http://courses.coreservlets.com/. 

Taught by the author of Core Servlets and JSP, More Servlets and JSP, and this tutorial. Available at public venues, or customized versions can be held on-site at your organization.

- Courses developed and taught by Marty Hall  
  - JSF 2, PrimeFaces, servlets/JSP, Ajax, jQuery, Android development, Java 7 or 8 programming, custom mix of topics  
  - Courses available in any state or country. Maryland/DC area companies can also choose afternoon/evening courses.
- Courses developed and taught by coreservlets.com experts (edited by Marty)  
  - Spring, Hibernate/JPA, GWT, Hadoop, HTML5, RESTful Web Services  
  - Contact hall@coreservlets.com for details
Topics in This Section

- Motivation
- Idea: Flyweight Design Pattern
- Simple CellList example
  - Just a simple CellList with days of the week
- Pagable CellList example
  - Thousands of rows with a pager
  - Incredibly fast paging!
- Other available Cell widgets and Cell types

Motivation

- Regular widgets are not designed to be fast
  - They build complex UI objects and are not meant for large sets of data
  - They use DOM APIs to build up and render each piece of data
  - They don’t scale
    - When the above process is repeated on thousands of data objects, the result is slow performance
Idea: Flyweight Design Pattern

• Big idea: build one object that stores a group of strings
  – Use just one Cell object to build HTML strings for each piece of data in the data set
  – Create multiple DOM elements from those strings
  – Any event that occurs on those elements goes back to the flyweight object to be handled
    • Flyweight object gets the event and the parent DOM element that event occurred on

Analogy: Word Processor

• Think of a word processor document and each letter in a large document
  – Each char is not a Letter object with its own font, size, ...
    • If it were, it would be prohibitively expensive to store even a 100K word document
  – Instead, every char is a reference to a flyweight Letter object shared by every instance of the same char
    • Only the position of each character needs to be stored

• GWT analogue
  – In the case of Cell widgets, this boils down to being able to render a huge data table as a single HTML string
Example: CellList

- **Explicit data via a List**
  - `CellList<String> cellList = new CellList<String>(...);`
  - `cellList.setRowCount(...);`
  - `cellList.setRowData(0, someListOfStrings);`

- **Data via a DataProvider**
  - `CellList<String> cellList = new CellList<String>(...);`
  - `ListDataProvider<String> dataProvider = ...;`
  - `List<String> data = dataProvider.getList();`
  - `data.add(...);`
  - `dataProvider.addDataDisplay(cellList);`

- **Other widgets have similar approach**
  - `CellTable, DataGrid, CellTree, CellBrowser, etc.`
Steps

- **Make a CellList to store text**
  - CellList<String> cellList =
    new CellList<String>(new TextCell());

- **Add event handlers, if any**
  - cellList.setKeyboardSelectionPolicy(...);
  - final SingleSelectionModel<String> selectionModel = ...
  - cellList.setSelectionModel(selectionModel);
  - selectionModel.addSelectionChangeHandler(...);

- **Put data in list**
  - List<String> data = ...;
  - cellList.setRowCount(data.size(), true);
  - cellList.setRowData(0, data);

---

Example: CellList to Show Days (Adapted from GWT Docs)

```java
public class GwtCellWidgets1 implements EntryPoint {
    private static final List<String> DAYS =
        Arrays.asList("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday");

    public void onModuleLoad() {
        CellList<String> sampleCellList = makeSimpleCellList();
        RootPanel.get("simple-cell-list").add(sampleCellList);

        FlowPanel pagableList = makePagableList();
        RootPanel.get("pagable-cell-list").add(pagableList);
    }
}
```
private CellList<String> makeSimpleCellList() {
    // Create a CellList that renders text cells.
    CellList<String> cellList =
        new CellList<String>(new TextCell());
    cellList.setKeyboardSelectionPolicy
        (KeyboardSelectionPolicy.ENABLED);
}

// Add a selection model to handle user selection.
final SingleSelectionModel<String> selectionModel =
    new SingleSelectionModel<String>();
cellList.setSelectionModel(selectionModel);
selectionModel.addSelectionChangeHandler(
    new SelectionChangeEvent.Handler() {
        public void onSelectionChange(
            SelectionChangeEvent event) {
            String selected =
                selectionModel.getSelectedObject();
            if (selected != null) {
                Window.alert("You selected: " + selected);
            }
        }
    });
// Set the total row count. This isn't strictly necessary,
// but it affects paging calculations, so its good habit
// to keep the row count up to date.
cellList.setRowCount(DAYS.size(), true);

// Push the data into the widget.
cellList.setRowData(0, DAYS);
cellList.addStyleName("simple-cellList");

return(cellList);

// You can use keyboard's up and down arrow and the
// spacebar to select elements.
Pagable CellList

Steps

- **Make a CellList to store text**
  - CellList<String> cellList = new CellList<String>(new TextCell());

- **Make a data provider to supply data for list**
  - ListDataProvider<String> dataProvider = new ListDataProvider<String>();
  - List<String> data = dataProvider.getList();
  - data.add(…);
  - dataProvider.addDataDisplay(cellList);

- **Make a pager**
  - SimplePager pager = new SimplePager();
  - pager.setDisplay(cellList);
  - pager.setPageSize(20);

- **Add both pager and cell list to display**
public class GwtCellWidgets1 implements EntryPoint {
    ...
    public void onModuleLoad() {
        CellList<String> sampleCellList = makeSimpleCellList();
        RootPanel.get("simple-cell-list").add(sampleCellList);
        FlowPanel pagableList = makePagableList();
        RootPanel.get("pagable-cell-list").add(pagableList);
    }
}

private FlowPanel makePagableList() {
    // Create a CellList.
    CellList<String> cellList =
        new CellList<String>(new TextCell());
    cellList.addStyleName("pagable-cellList");
    // Add a cellList to a data provider.
    ListDataProvider<String> dataProvider =
        new ListDataProvider<String>();
    List<String> data = dataProvider.getList();
    for (int i = 0; i < 2000; i++) {
        data.add("Item " + i);
    }
    dataProvider.addDataDisplay(cellList);
    Whatever is added to the ListDataProvider's list is automatically displayed in the widget (if that portion is made visible by the SimplePager – see next)
// Create a SimplePager.
SimplePager pager = new SimplePager();

// Set the cellList as the display.
pager.setDisplay(cellList);
pager.setPageSize(20);
pager.setWidth("300px");

// Add the pager and list to the page.
FlowPanel flowPanel = new FlowPanel();
flowPanel.add(new SimplePanel(pager));
flowPanel.add(new SimplePanel(cellList));

return flowPanel;

Pagable CellList: Result

Select one of the items (gets a yellow background). Now, press "Page Down" key and hold it down. See how blazing fast the new set of items loads.
Wrap-Up

Other Cell Widgets

• CellTable
• DataGrid
  – CellTable with fixed header and footer
• CellTree
• CellBrowser
• Many types of GWT premade cell types
  – Text
  – EditText
  – Date, etc.
  – Or create your own custom cell
  – See list: http://gwt.google.com/samples/Showcase/Showcase.html#!CwCellSampler
Summary

• Explicit data via a List
  – CellList<String> cellList = new CellList<String>(…);
  – cellList.setRowCount(…);
  – cellList.setRowData(0, someListofStrings);

• Data via a DataProvider
  – CellList<String> cellList = new CellList<String>(…);
  – ListDataProvider<String> dataProvider = …;
  – List<String> data = dataProvider.getList();
  – data.add(…);
  – dataProvider.addDataDisplay(cellList);

• Can add pager
  – Usually to CellList that uses data provider

Questions?

Customized Java EE Training: http://courses.coreservlets.com/
GWT, Java 7 and 8, JSF 2, PrimeFaces, HTML5, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Android, Hadoop, and other Java EE training.
Developed and taught by well-known author and developer. At public venues or onsite at your location.