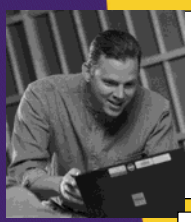
Laying out Windows with Panels: Part II – Composite Panels

(GWT 2.5 Version)

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



Topics in This Section

- **Strategy behind Panel usage**
  - Similar to use of LayoutManagers in desktop Java
- **div-based layout vs. table-based layout**
  - *Blah*LayoutPanel vs *Blah*Panel
- **Composite Panel types in standards mode**
  - TabLayoutPanel
  - StackLayoutPanel
  - SplitLayoutPanel
  - DockLayoutPanel
  - DeckLayoutPanel
- **Advanced issues**
  - Firebug
  - CSS background colors
  - History handling

Overview

Layout Strategies

- **HTML-based layout**
  - Write HTML by hand, designate places for individual controls
  - HTML body contains a lot of content
  - Best when GWT is used for
    - A traditional Web app with some pieces of Ajax-enabled content
    - Complex page where HTML layout does not change
    - You need the HTML content indexed by a search engine
- **Java-based layout (this lecture)**
  - Attaches main layout panel to HTML <body> element
  - Java uses Panels to build up overall layout
    - Similar to way LayoutManagers are used in desktop Java
  - Best when GWT is used to
    - Mimic a desktop application
    - Create an application where HTML layout changes on fly
- **Declarative layout – see UiBinder slides**

Quirks vs. Standards Mode in Browsers Revisited

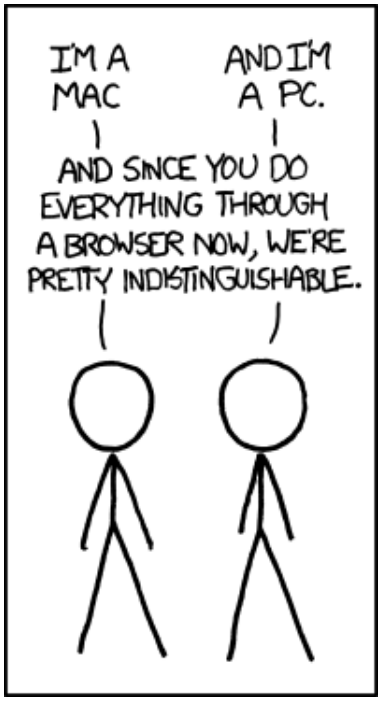
- **Idea**
  - Some browsers render differently when following CSS and HTML standards than they did traditionally
    - Applies especially to Internet Explorer
    - The traditional behavior is called “Quirks Mode”
  - Table-based layouts assume traditional behavior
  - New GWT 2.*x* layout panels are div-based
  - *A few table-based panels are already deprecated*
- **Bottom line for complex panels**
  - Use only standards mode
    - Quirks mode is really only for backward compatibility
  - Use *Blah*LayoutPanel, not *Blah*Panel if available
    - See next slides for advantages

Standards Mode and *Blah*LayoutPanel

Table-Based vs. div-Based Panels

- **In all GWT versions, including 2.*x***
  - DockPanel, TabPanel, StackPanel, etc.
  - Uses HTML tables to control layout
  - Still have moderate control via CSS
- **New in GWT 2.*x***
  - DockLayoutPanel, TabLayoutPanel, StackLayoutPanel,…
  - Uses to control layout
  - Heavily relies on CSS to let you control layout
  - Must use DOCTYPE so browsers run in “standards mode”
  - Goals: see next slide.

Goal: Perfectly Predictable Layouts



GWT 2.x Layout System Goals

- **Perfectly predictable layout behavior**
  - Ironically, in practice, it takes more tinkering to get GWT2 div-based layouts to look the same in all browsers
  - What they mean by "predictable" is *after* you've worked out all the correct CSS rules
    - Forces developers to get *very* familiar with CSS
- **Works correctly in standards mode**
- **Browsers should do almost all layout work using its native layout engine**
  - I.e., no generated JavaScript needed to adjust things
  - Much faster!
- **Smooth, automatic animation**

Main Differences

- **HTML**
  - Must start with DOCTYPE
  - Simplest declaration (HTML 5): <!doctype html>
- **Class names**
  - Contain the word "Layout"
    - DockLayoutPanel, TabLayoutPanel, StackLayoutPanel
- **Starting-point Java**
  - Call RootLayoutPanel.get().add(…)
    - Instead of RootPanel.get().add(…)
  - RootLayoutPanel can only be attached to <body> element
    - Does not allow you to attach widget to a specified id-based region in HTML

Typical GWT Panel Design

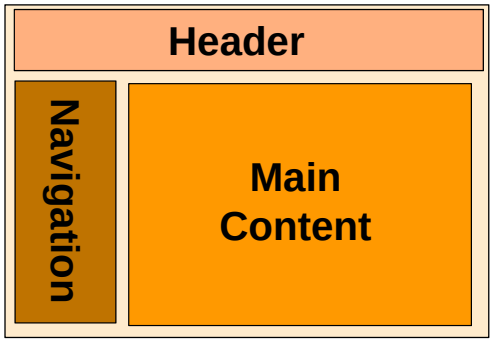
- **Multi-panel types**
  - Either the top-level panel or the main region of the top-level panel usually contains a multi-panel type
    - E.g., DeckLayoutPanel, TabLayoutPanel, StackLayoutPanel
- **Sample designs**
  - Top-level window is a TabLayoutPanel or StackLayoutPanel
    - Every time the user clicks on a tab or stack (accordion) label, you switch "screens"
  - Top-level window is a DeckLayoutPanel
    - Custom controls to navigate from screen to screen
  - Top-level window is a DockLayoutPanel
    - And center is a TabLayoutPanel, StackLayoutPanel, or DeckLayoutPanel

Tutorial Breakdown

- **Previous section: basic panels**
  - Purpose: organize a group of GUI controls
    - All panels can contain other panels, but the first Panel tutorial covers the simplest panel types that are often used to represent a small group of widgets (e.g., a row or column of buttons or an area of text)
    - Examples: HorizontalPanel, VerticalPanel, Grid, FlowPanel, Grid, FlexTable
- **This section: composite panels**
  - Purpose: organize other panels
    - Covers the panels that are used primarily to group other panels, and are typically the top-level panels in an application
      - Examples: TabLayoutPanel, StackLayoutPanel, DockLayoutPanel, DeckLayoutPanel

Examples: Overview

- **Top-level panel: DockLayoutPanel**



- **Examples for content panel**
  - Another DockLayoutPanel
  - TabLayoutPanel
  - StackLayoutPanel
  - SplitLayoutPanel

Examples: HTML

```
<!doctype html>
<html>
<head><title>GWT Panels</title>
<link rel="stylesheet"
      href="./css/styles.css"
      type="text/css"/>
<script type="text/javascript" language="javascript"
        src="gwtpanels2/gwtpanels2.nocache.js"></script>
</head>
<body>
…
<!-- Empty body, i.e., no visible HTML tags. -->
</body></html>
```


DockLayoutPanel

Summary

- **Purpose**
  - Most commonly used for main outer layout of the page
  - Child widgets are "docked" at panel's outer edges
    - You specify which edge you want the widget docked
      - North, south, west, east
      - Center of layout takes up the rest of available space
  - Only works in standards mode
- **Main methods**
  - DockLayoutPanel(Style.Unit)
    - Unit is enum of CSS units of measurement (PX, PCT, etc)
      - Same type must be used for all children
  - addNorth(Widget w, double size), addSouth(Widget w, double size)
  - addWest(Widget w, double size), addEast(Widget w, double size)
    - Each docked child must be sized explicitly
  - add(Widget w)
    - Adds center widget
    - Must be called last: no other widget can be added after this one

Example Code – Main Layout

```
public void onModuleLoad() {
  DockLayoutPanel mainPanel = new DockLayoutPanel(Unit.PX);
  HTML headerPanel = makeHeader();
  mainPanel.addNorth(headerPanel, 55);
  FlowPanel navPanel = makeNavPanel();
  mainPanel.addWest(navPanel, 220);
  TabLayoutPanel tabPanel = new TabPanelExample(4);
  StackLayoutPanel stackPanel = new StackPanelExample(5);
  SplitLayoutPanel hSplitPanel = new HSplitPanelExample();
  SplitLayoutPanel vSplitPanel = new VSplitPanelExample();
  Widget[] contentWidgets = { tabPanel, stackPanel, hSplitPanel,
   vSplitPanel };
  contentPanel = new ContentPanel(contentWidgets);
  mainPanel.add(contentPanel);
  RootLayoutPanel.get().add(mainPanel);
}
```

All child sizes are set to be in pixels

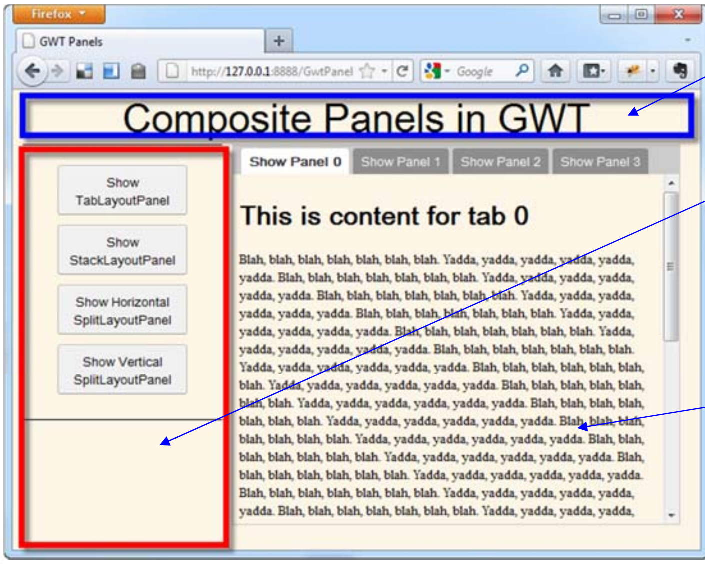
Each child's size is set explicitly. These numbers are in pixels since panel was created with Unit.PX

Example Result

mainPanel.addNorth(headerPanel, 55);

mainPanel.addWest(navPanel, 220);

mainPanel.add(contentPanel);



Example: Navigation Panel

```
…
public class NavPanel extends FlowPanel {
  @Override
  public void add(Widget w) {
    // Force to be block-level element
    SimplePanel wrapper = new SimplePanel();
    wrapper.addStyleName("center-block");
    wrapper.add(w);
    super.add(wrapper);
  }
}
```

Adds CSS class to the widget

```
.center-block {
       text-align: center;
}
```

Center-aligns the single element added to the SimplePanel within the SimplePanel itself.

Example: Creating NavPanel – Java Code

```
private FlowPanel makeNavPanel() {
   NavPanel navPanel = new NavPanel();
   navPanel.addStyleName("nav-panel");
   Button[] navButtons = {
       makeNavButton("Show TabLayoutPanel"),
       makeNavButton("Show StackLayoutPanel"),
       makeNavButton("Show Horizontal SplitLayoutPanel"),
       makeNavButton("Show Vertical SplitLayoutPanel") };
   for (int i = 0; i < navButtons.length; i++) {
     Button button = navButtons[i];
     button.addClickHandler(new ButtonHandler(i));
     navPanel.add(button);
   }
   return navPanel;
 }
```

Uses DeckLayoutPanel to switch between panels in the main content region. More on that later.

Example: Creating NavPanel – CSS Code

```
.nav-panel {
      height: 260px;
      padding-top: 15px;
      border-top: 1px solid black;
      border-bottom: 1px solid black;
}
```

Would it be better to add this CSS style class inside NavPanel itself?

```
this.addStyleName("nav-panel");
```

Example: Creating Buttons for NavPanel (Java & CSS)

```
public Button makeNavButton(String label) {
   Button navButton = new Button(label);
   navButton.addStyleName("nav-button");
   return navButton;
}
```

Creating spacing between buttons

```
.nav-button {
      width: 130px;
      height: 50px;
      margin: 5px !important;
}
```


TabLayoutPanel

Summary

- **Purpose**
  - Place panels on top of each other, select by clicking on tabs
  - Only works in standards mode
- **Main methods**
  - TabLayoutPanel(double barHeight, Style.Unit barUnit)
    - Constructor
  - add(Widget w, String tabText)
    - Adds a widget (usually HTML or a Panel), in left-to-right order. Note *two* args for "add".
    - You can also supply HTML or a Widget to be shown in tab at top
  - selectTab(int tabNumber)
    - Programmatically selects a panel
  - setAnimationDuration(long duration)
    - Set the duration of the animated transition between tabs (in ms)
  - setAnimationVertical(boolean isVertical)
    - Set transitions to slide vertically or horizontally (default: false)

Example: Java Code

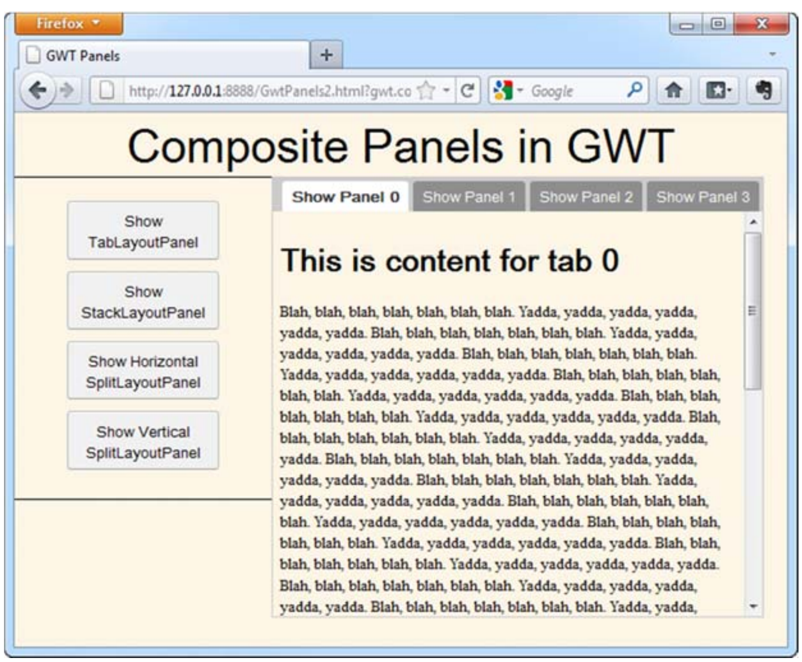
```
public class TabPanelExample extends TabLayoutPanel {
  public TabPanelExample(int numTabs) {
    super(30, Unit.PX);
    for (int i = 0; i < numTabs; i++) {
      add(tabContent(i), tabLabel(i));
    }
    addStyleName("tab-panel");
    setAnimationDuration(500);
  }

  private Widget tabContent(int index) {
    String content = "<h1>This is content for tab " +
              index + "</h1>" + PanelUtils.randomText();
    return (new ScrollPanel(new HTML(content)));
  }

  private String tabLabel(int index) {
    return ("Show Panel " + index);
  }
}
```

Example: CSS Code

These style rules will apply to these CSS classes

```
.tab-panel, .stack-panel, .split-panel {
  margin-bottom: 25px;
  margin-right: 20px;
}
```


Example Result



StackLayoutPanel

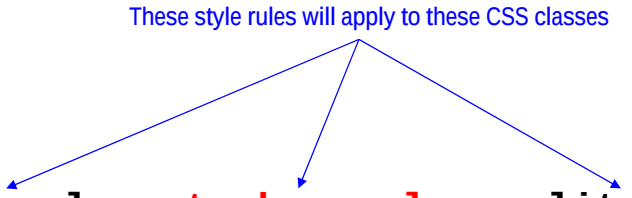
Summary

- **Purpose**
  - To show one panel at a time, and select by clicking on header. Headers go up and down instead of across as with tabbed panels. Called "accordion panels" in some libraries.
  - Only works in standards mode
  - Unlike many JavaScript libraries, one panel is *always* open
- **Methods**
  - StackLayoutPanel(Style.Unit headerSizeUnit)
    - Constructor
  - add(Widget widget, String header, double headerSize)
    - Adds a widget, in top-to-bottom order. Note *three* args for "add"
  - showWidget(int index)
    - Programmatically selects a panel based on order added
  - setAnimationDuration(long duration)
    - Set the duration of the animated transition between stacks
    - Transitions are always vertical

Example: Java Code

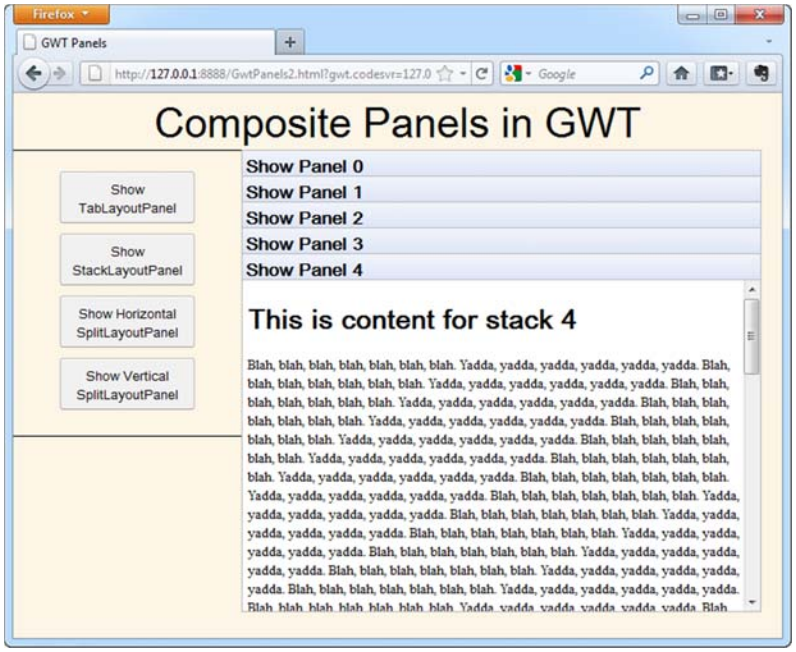
```
public class StackPanelExample extends StackLayoutPanel {
  public StackPanelExample(int numTabs) {
    super(Unit.PX);
    for (int i = 0; i < numTabs; i++) {
      add(stackContent(i), stackLabel(i), 25);
    }
    showWidget(numTabs - 1);
    addStyleName("stack-panel");
    setAnimationDuration(500);
  }

  private Widget stackContent(int index) {
    String content = "<h1>This is content for stack " + index +
    "</h1>“ + PanelUtils.randomText() + PanelUtils.randomText();
    ScrollPanel p = new ScrollPanel(new HTML(content));
    return (p);
  }

  private String stackLabel(int index) {
    return ("Show Panel " + index);
  }
}
```


Example: CSS Code




```
.tab-panel, .stack-panel, .split-panel {
  margin-bottom: 25px;
  margin-right: 20px;
}
```

Example Result



SplitLayoutPanel

Summary

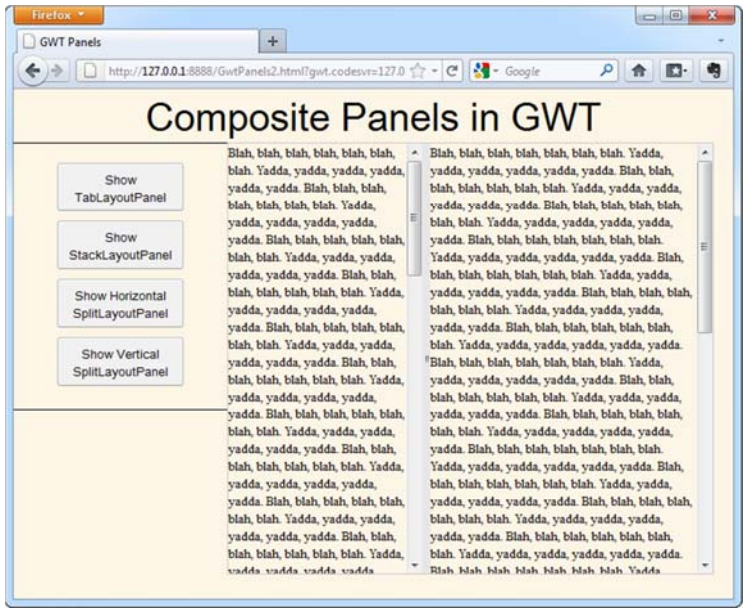
- **Purpose**
  - Identical to DockLayoutPanel but with draggable splitter between each section
  - Only works in standards mode
  - Only supports pixel units
- **Main methods**
  - SplitLayoutPanel(), SplitLayoutPanel(int splitterSize)
    - No arg defaults to 8px for splitter
  - addNorth(Widget w, double size), addSouth(Widget w, double size)
  - addWest(Widget w, double size), addEast(Widget w, double size)
    - Each docked child must be sized explicitly
  - add(Widget w)
    - Adds center widget
    - Must be called last, i.e., no other widget can be added to the panel after this one

Example: SplitLayoutPanel (Horizontal)

```
public class HSplitPanelExample extends SplitLayoutPanel {

  public HSplitPanelExample() {
    addWest(makeScrollableText(), 200);
    add(makeScrollableText());
    addStyleName("split-panel");
  }

  private Widget makeScrollableText() {
    return new ScrollPanel(new HTML(PanelUtils.randomText()));
  }
}
```

```
.split-panel {
   border: 1px solid #CCC;
}
.tab-panel, .stack-panel, .split-panel {
   margin-bottom: 25px;
   margin-right: 20px;
}
```

Example Result

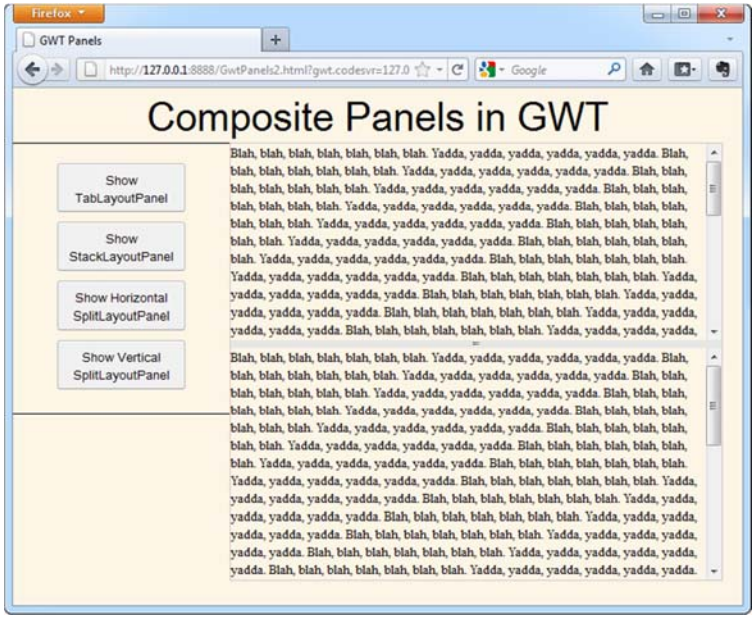


Example: SplitLayoutPanel (Vertical)

```
public class VSplitPanelExample extends SplitLayoutPanel {

  public VSplitPanelExample() {
    addNorth(makeScrollableText(), 200);
    add(makeScrollableText());
    addStyleName("split-panel");
  }

  private Widget makeScrollableText() {
    return new ScrollPanel(new HTML(PanelUtils.randomText()));
  }
}
```

```
.split-panel {
   border: 1px solid #CCC;
}
.tab-panel, .stack-panel, .split-panel {
   margin-bottom: 25px;
   margin-right: 20px;
}
```

Example Result



DeckLayoutPanel

Summary

- **Purpose**
  - To put widgets on top of each other and show one at a time
  - You must add your own logic for selecting screens
    - Like a TabPanel with no tabs
      - In fact, DeckLayoutPanel is used by the TabLayoutPanel
    - It is very common in GWT to use a DeckLayoutPanel as the main "content" panel or possibly even the top-level panel. This is the standard way to programmatically switch "screens".
  - Only works in standards mode
- **Main methods**
  - add(Widget w)
    - Adds a widget, usually another Panel.
  - showWidget(int index)
    - Displays the widget (sub-panel)
    - You must call showWidget(index) or nothing is displayed initially
  - getVisibleWidget()
    - Returns the index of the widget currently displayed
  - setAnimationDuration(long duration)
    - Set the duration of the animated transition between tabs (in ms)
  - setAnimationVertical(boolean isVertical)
    - Set transitions to slide vertically or horizontally (default: false)

Example: Main Content Panel

```
public class ContentPanel extends DeckLayoutPanel {
  public ContentPanel(Widget[] panels) {
    for (Widget panel : panels) {
      System.out.println("Adding to content panel: "
                         + panel);
      add(panel);
    }
    showWidget(0);
    setAnimationDuration(800);
    setAnimationVertical(true);
  }
}
```

Output generated HTML of the widget to Eclipse console.

Animate switching between panel in a "deck" a bit slower and make the transitions vertical (horizontal is default).

Example: Adding Widgets to Main Content Panel

```
…
public class GwtPanels2 implements EntryPoint {
  private DeckLayoutPanel contentPanel;
  …
  public void onModuleLoad() {
    DockLayoutPanel mainPanel =
                new DockLayoutPanel(Unit.PX);
    HTML headerPanel = makeHeader();
    mainPanel.addNorth(headerPanel, 55);
    FlowPanel navPanel = makeNavPanel();
    mainPanel.addWest(navPanel, 220);
    TabLayoutPanel tabPanel = new TabPanelExample(4);
    StackLayoutPanel stackPanel =
                new StackPanelExample(5);
    SplitLayoutPanel hSplitPanel =
                new HSplitPanelExample();
    SplitLayoutPanel vSplitPanel =
                new VSplitPanelExample();
    Widget[] contentWidgets =
        { tabPanel, stackPanel, hSplitPanel, vSplitPanel };
    contentPanel = new ContentPanel(contentWidgets);
    mainPanel.add(contentPanel);
    RootLayoutPanel.get().add(mainPanel);
  }
```


Example: Switching Between Panels

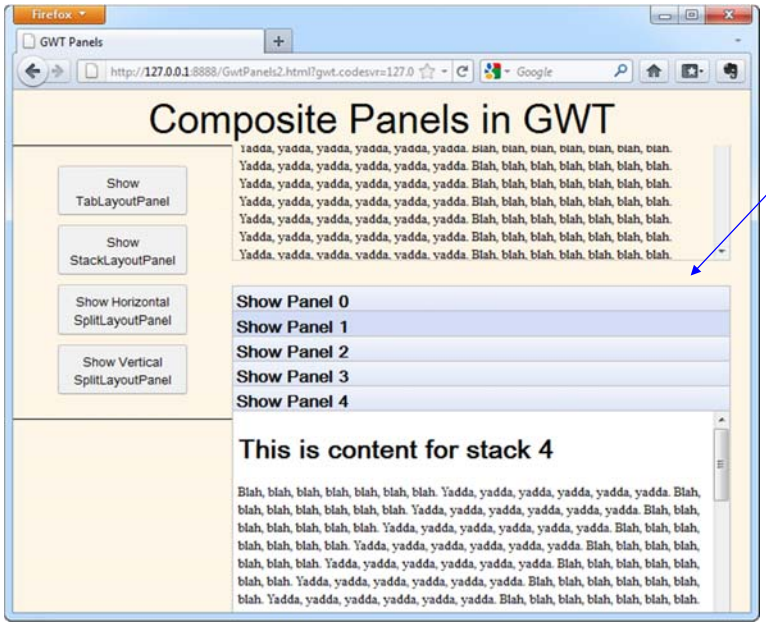
```
private FlowPanel makeNavPanel() {
   NavPanel navPanel = new NavPanel();
   navPanel.addStyleName("nav-panel");
   Button[] navButtons = {
       makeNavButton("Show TabLayoutPanel"),
       makeNavButton("Show StackLayoutPanel"),
       makeNavButton("Show Horizontal SplitLayoutPanel"),
       makeNavButton("Show Vertical SplitLayoutPanel") };
   for (int i = 0; i < navButtons.length; i++) {
     Button button = navButtons[i];
     button.addClickHandler(new ButtonHandler(i));
     navPanel.add(button);
   }

   return navPanel;
 }
```

Example: Switching Between Panels (continued)

```
…
public class GwtPanels2 implements EntryPoint {
  private DeckLayoutPanel contentPanel;
  …

 private class ButtonHandler implements ClickHandler {
    private int subPanelIndex;

    public ButtonHandler(int subPanelIndex) {
      this.subPanelIndex = subPanelIndex;
    }

    @Override
    public void onClick(ClickEvent event) {
      contentPanel.showWidget(subPanelIndex);
    }
  }
}
```

Animation gets triggered when switching between panels

Example Result



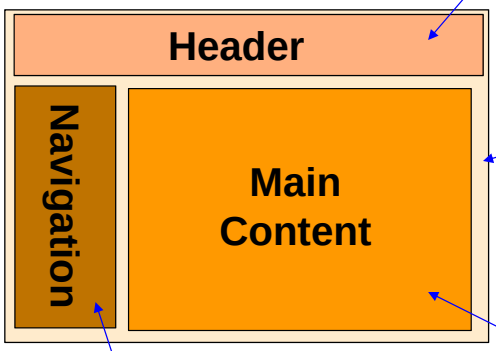
In the middle of a transition sliding up.

Nesting Panels

Idea

- **One panel is rarely enough**
  - It is very rare to have just one or two panel types in your application.
  - Instead, you progressively divide up the screen into smaller and smaller pieces
- **Top-level panel**
  - Most often DockLayoutPanel or TabLayoutPanel
    - But could be DeckLayoutPanel or StackLayoutPanel
  - Remember: tabs don't have to look like tabs!
    - Use CSS to make them look like buttons, links, etc.
- **Area where screen changes**
  - Most often a DeckLayoutPanel, which in turn contains other panels

Summary of Example Design



Header: an HTML widget placed into the NORTH Section of the main DockLayoutPanel.

Top-level panel: DockLayoutPanel.

Content panel: a DeckLayoutPanel placed into the CENTER section of the panel. This DeckLayoutPanel then contains four entries (the four examples shown earlier). The buttons in the navigation panel control which of the four entries is displayed.

Navigation panel: a FlowPanel made to behave like a VerticalPanel, placed into the WEST section of the main DockLayoutPanel. Using CSS, widgets are center-aligned.

Advanced Issues

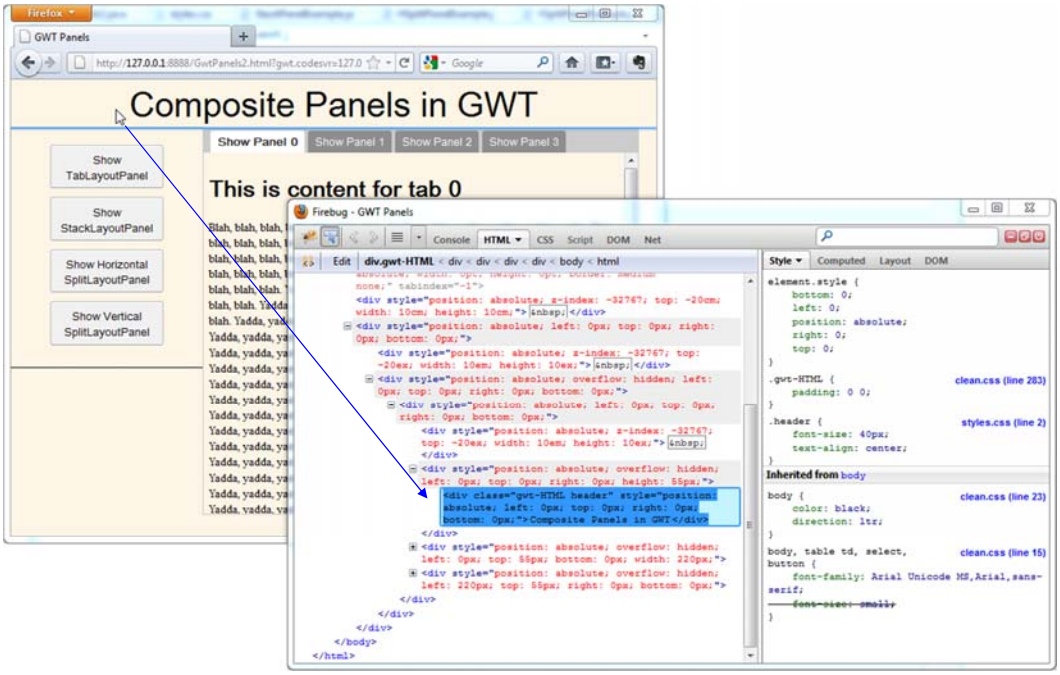
History Handling

- **There is only one HTML page for app**
  - The main URL never changes
- **Back button**
  - Do you want the browser's "back" and "forward" buttons to have any meaning?
- **Bookmarking**
  - Do you want to let users "bookmark" a "screen" and navigate directly there later?
- **History handling**
  - Instead of controls doing direct navigation, they store a token on end of URL (http://host/app/app.html#token)
  - History handler does navigation based on the token
  - See separate tutorial section on History

Using Firebug

- **Firefox browser plugin**
  - Lets you call JavaScript interactively
  - Has JavaScript debugger and error console
  - Traces all XmlHttpRequests or general network requests
  - DOM explorer
  - CSS explorer (adjust CSS for immediate visual feedback!)
  - Lots more
  - See http://getfirebug.com
  - Debugger in Chrome now almost (?) as good as Firebug
- **Firebug in traditional Ajax apps**
  - *Totally* indispensable
- **Firebug in GWT apps**
  - Still very valuable for seeing what DOM looks like after panels have been assembled

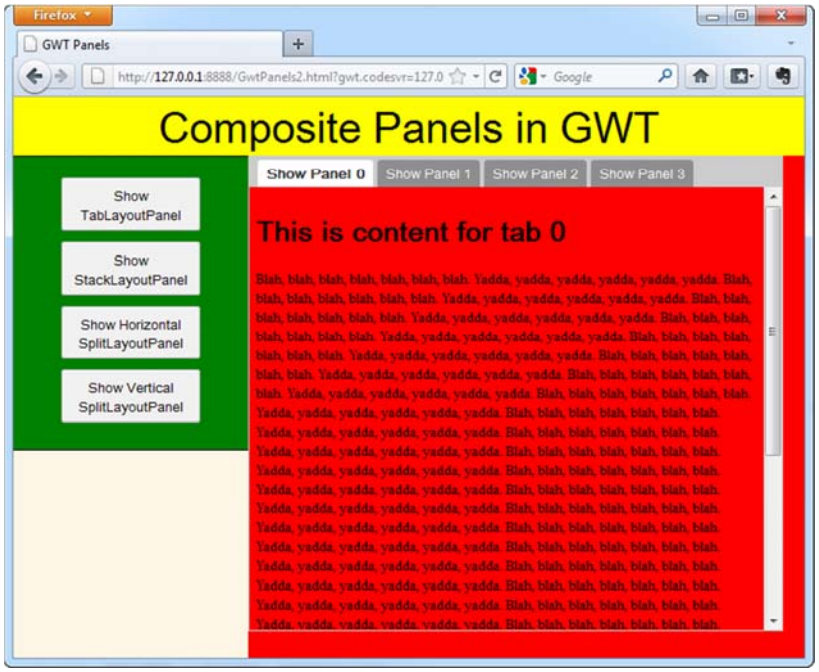
Firebug: Example



Using CSS for Debugging

- **Problem: things are wrong sizes**
  - It is hard to know what you did wrong when you don't actually know where each element is
    - Using Firebug helps most of the time
- **Solution**
  - Use CSS to put a border around each subpanel or give it a different background color
  - CSS
    - .red { background-color: red; }
  - Java
    - myPanel.addStyleName("red");

Using CSS for Debugging



Wrap-Up

Summary

- **Pick panel for top-level layout**
  - Often DockLayoutPanel or TabLayoutPanel
- **Pick panel type to hold screens that will change**
  - Often Tab, Deck, or Stack Panels
- **Nest panels deeply**
  - Build up overall design a piece at a time
  - Very similar strategy to use of layout managers in Swing
- **Debug by looking at real HTML**
  - Use Firebug and CSS background colors

Questions?