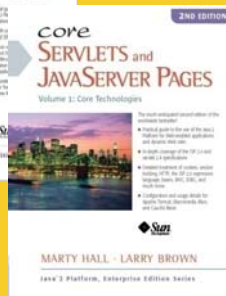
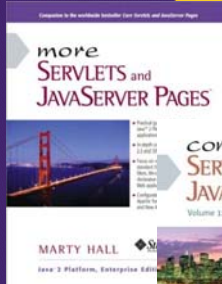




The Google Web Toolkit (GWT): Handling History and Bookmarks (GWT 2.5 Version)

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
- Panel design
- HTML setup
- Setting History tokens
- Responding to History tokens
- 3 most common mistakes in History (handling)
- Advanced issues

4

© 2013 Marty Hall & Yaakov Chaikin



Overview

Those who cannot learn from history handling are doomed to repeat it.
- with apologies to George Santayana

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.

Traditional Web Apps

- **URLs**

- Each “screen” is a separate URL

- **History**

- Pressing the “Back” and “Forward” buttons means cycling through the previous URLs
- History handled automatically by the browser

- **Bookmarks**

- Any “screen” accessed with GET can be bookmarked merely by saving the URL
- The application can often automatically reconstitute a saved state
 - Although not with POST or with session data



6

GWT-Based Web Apps

- **URLs**

- There is only *one* HTML page
- The main URL *never* changes
 - Only possibly the #linkTarget on the end

- **History**

- Users still expect the “Back” and “Forward” buttons to have meaning
- You must explicitly store tokens (markers) in a history object and tell the application how to respond to them

- **Bookmarks**

- User can still save a bookmark (<http://mainURL#token>)
- You must explicitly tell the app what screen to go to in response to the token



7

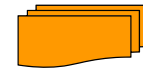
Typical GWT Panel Design

- **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



8

Main Issues

- **Setting History tokens**

- Whenever the user takes an action that changes the “screen” in a way that you want to save, you should store a token
 - `History.newItem("someLinkTarget", false);`

- **Responding to History tokens**

- Whenever the URL ends in `#someToken`, that token is passed to the `onValueChange` method of the History’s `ValueChangeHandler`

```
public void onValueChange (ValueChangeEvent<String> event) {  
    String linkTarget = event.getValue();  
    if (hasCertainProperties(linkTarget)) {  
        switchToSomeScreen();  
    } else ...  
}
```

9

Navigation Design

- **Without history handling**
 - Navigation distributed
 - E.g., pushing a button might directly change the top panel shown in a DeckLayoutPanel. The top-level app does not need to know how to get to all pages and sub-pages.
- **With history handling**
 - Navigation centralized
 - Pushing the button stores a token. The history handler does the real navigation. The history handler must know how to get to all pages and sub-pages.
- **Moral**
 - Decide early if you need history/bookmark support

10

HTML Setup

- **Leave iframe in body**
 - Exactly as built when project created
- **Prepare for panel usage**
 - Body is entirely empty except for iframe
 - All content inserted dynamically

- **Example**

```
<!doctype html>
<html>
<head><title>...</title>...
<script ...></script>
</head><body>
<iframe ...></iframe>
</body></html>
```

Unchanged from auto-generated HTML file.

11



Simple Examples

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.

Setting History Tokens

- **Changing a tab**
 - Add a SelectionHandler to the TabLayoutPanel that stores a token saying which tab was selected
 - Uses History.newItem("token", **false**)
 - Passing false does not trigger history token evaluation
- **Selecting a stack (accordion) label**
 - Add SelectionHandler to the StackLayoutPanel that stores a token saying which stack was selected
 - Uses History.newItem("token", **false**)
 - Passing false does not trigger history token evaluation

Setting History Tokens (Continued)

- **Clicking a hyperlink**
 - Adds a token to the history automatically
 - Triggers history token evaluation
- **Any user event that results in a change to the screen**
 - You decide if the result of this event is something the user would want to be able to bookmark
 - I.e., save its navigational state
 - E.g., pressing a Button might need to cause a new screen to be displayed, so override `onClick` to store a token
 - Use `History.newItem("token")` to trigger history token evaluation

14

Responding to History Tokens

- **Attach handler**

```
History.addValueChangeHandler(new HistoryHandler());
```
- **Respond to tokens**

```
private class HistoryHandler
    implements ValueChangeHandler<String> {
    public void onValueChange
        (ValueChangeEvent<String> event) {
        String linkTarget = event.getValue();
        switchScreensBasedOn(parse(linkTarget));
    }
}
```
- **Notes**
 - Tokens are sent on initial user actions as well as when back/forward button is pressed or bookmarked URL is sent
 - Initial user action (e.g., pasting a URL into the address bar) needs `History.fireCurrentHistoryState()` to trigger `ValueChangeEvent`

15

Example

- **Panel design**
 - Top-level panel is TabLayoutPanel
- **Storing tokens**
 - Every time a tab is selected, do `History.newItem("tabi", false)`, where *i* is the tab number
 - Do this from SelectionHandler attached to the TabPanel
 - Must include 2nd arg as 'false' not to trigger another history event (common mistake!)
- **Resultant URLs**
 - `http://host/app/app.html#tab0`
 - `http://host/app/app.html#tab1` (etc.)
- **Responding to tokens**
 - Parse the token, extract the tab number, and do `mainTabPanel.selectTab(i)`
 - Do this from ValueChangeListener attached to History

16

© 2013 Marty Hall & Yaakov Chaikin



Extended Example

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.

Example: Panel Design Review (extended GwtPanels2 project)

• Top Level Panel

- DockLayoutPanel
 - North: HTML widget
 - West: Navigation panel (NavPanel)
 - Added Hyperlink widgets along the existing Button widgets
 - Center: DeckLayoutPanel (ContentPanel)
 - Contains subpanels

• Subpanels

- TabLayoutPanel: 1st sub-panel inside DeckLayoutPanel in the center
- StackLayoutPanel: 2nd sub-panel inside DeckLayoutPanel in the center
- SplitLayoutPanel: 3rd and 4th sub-panels inside DeckLayoutPanel in the center (Horizontal & Vertical)

18

Example: Panel Design Review (extended GwtPanels2 project)



19

Top-Level Navigation: Setup (HTML)

```
<!doctype html>
<html>
<head><title>GWT History</title>
<link rel="stylesheet" href="./css/styles.css"
      type="text/css"/>
<script type="text/javascript" language="javascript"
        src="gwthistory/gwthistory.nocache.js"></script></head>
<body>
<iframe src="javascript:''" id="__gwt_historyFrame"
        tabIndex='-1'
        style="position:absolute;width:0;height:0;border:0">
</iframe>
<noscript><div style="width: 22em; position: absolute; left:
50%; margin-left: -11em; color: red; background-color:
white; border: 1px solid red; padding: 4px; font-family:
sans-serif">Your web browser must have JavaScript enabled
in order for this application to display correctly.</div>
</noscript></body></html>
```

The script and the iframe are left unchanged from the automatically-generated HTML file. The rest of the HTML is custom, but note that body is totally empty except for the iframe.

20

Top-Level Navigation: Setup (Java)

```
public void onModuleLoad() {
    DockLayoutPanel mainPanel =
        new DockLayoutPanel(Unit.PX);

    ...
    vSplitPanel = new VSplitPanelExample();
    Widget[] contentWidgets =
        {tabPanel, stackPanel, hSplitPanel, vSplitPanel};
    contentPanel = new ContentPanel(contentWidgets);
    mainPanel.add(contentPanel);
    RootLayoutPanel.get().add(mainPanel);
    History.addValueChangeHandler(new HistoryHandler());
    History.fireCurrentHistoryState();
}
```

This wires the history token handler, i.e., handler that parses the history token and manipulates the GUI accordingly.

Triggers evaluation of the history token that is currently on the URL. If this is omitted, pasting a URL of type `http://host/gwtapp/page.html#token` into a new browser window will ignore the 'token' completely.

21

Top-Level Navigation: Navigation Panel

```
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);
    }
}
```

This handler no longer changes the screens, but issues a History event that gets processed by the HistoryHandler.

22

Top-Level Navigation: Navigation Panel (continued)

```
...
navPanel.add(
    new Hyperlink("Show TabLayoutPanel", "tab0"));
navPanel.add(
    new Hyperlink("Show StackLayoutPanel", "stack0"));
navPanel.add(
    new Hyperlink("Show Horizontal SplitLayoutPanel",
        "hsplit"));
navPanel.add(
    new Hyperlink("Show Vertical SplitLayoutPanel",
        "vsplit&range=20"));
return navPanel;
}
```

Hyperlinks automatically add tokens to History and trigger history token evaluation automatically, i.e., no different if you were to type the new URL into the address bar.
Link text. Corresponding History token.

We'll discuss this construct later in the slides.

23

Top-Level Navigation: Responding to Tab Tokens

```
private class HistoryHandler
    implements ValueChangeListener<String> {
    public void onValueChange(
        ValueChangeEvent<String> event) {
        String historyToken = event.getValue();
        if (historyToken.startsWith("tab")) {
            selectTab(historyToken);
        } else if (...) {
        } else {
            selectTab("tab0");
        }
    }
    ...
}
```

In case the token is missing or malformed, we always want to default to something sensible. So, if the URL is: `http://host/GwtHistory.html#bla-bla-bad-token`, the screen defaults to 'tab0'. See later slides for discussion.

24

Top-Level Navigation: Responding to Tab Tokens (con)

```
private void selectTab(String historyToken) {
    historyToken = historyToken.substring(3);
    int tabIndex = 0;
    try {
        tabIndex = Integer.parseInt(historyToken);
        if (tabIndex < 0 && tabIndex > 3) {
            tabIndex = 0;
        }
    } catch (NumberFormatException e) {
    }

    tabPanel.selectTab(tabIndex);
    contentPanel.showWidget(0);
}
...
}
```

Strip out 'tab' from token 'tab/' so / can be parsed as an integer.

Within TabLayoutPanel, select tab with the corresponding index.

Switch to deck with index 0, i.e., TabLayoutPanel within the DeckLayoutPanel, which resides in the center of the DockLayoutPanel.

25

Tab Panel Navigation: Storing Tokens

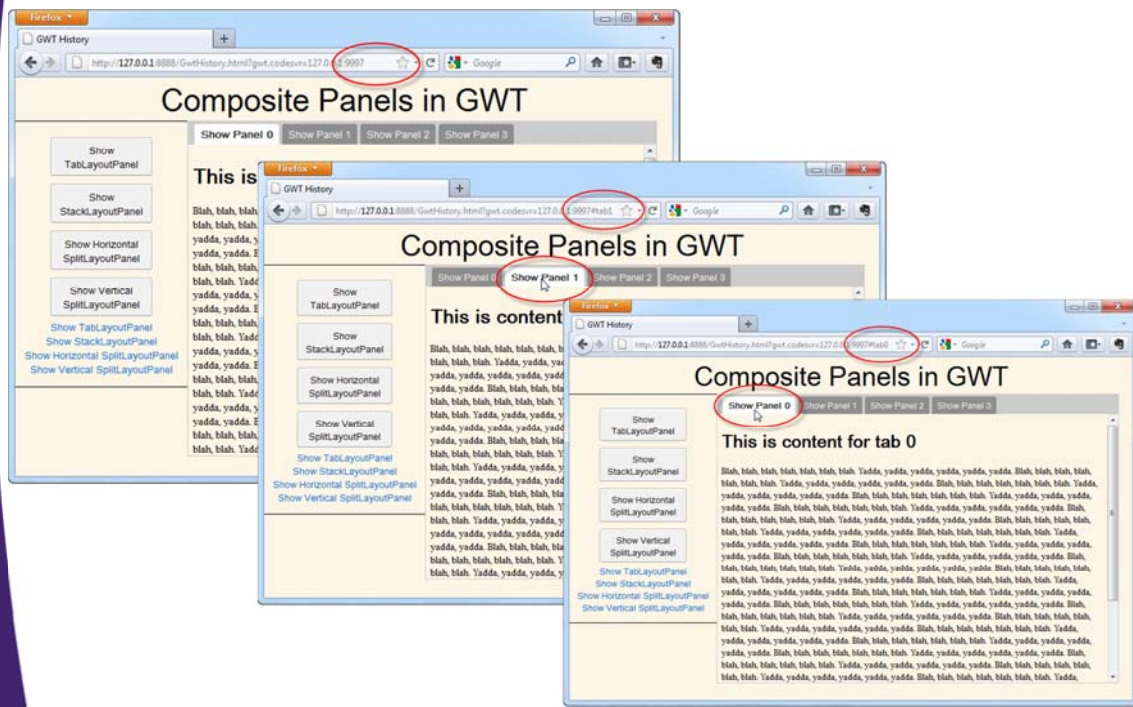
```
public class TabPanelExample extends TabLayoutPanel {
    public TabPanelExample(int numTabs) {
        ...
        addSelectionHandler(new TabSelectionHandler());
    }
    ...

    private class TabSelectionHandler
        implements SelectionHandler<Integer> {
        public void onSelection(SelectionEvent<Integer> event) {
            History.newItem("tab" + event.getSelectedItem(),
                false);
        }
    }
}
```

This flag is very important. It tells GWT that we do NOT want a history event to be fired and therefore processed by the HistoryHandler. All we want to do here is store the token on the URL that represents our current navigational state so we can get to it later, i.e., #tab0, #tab1, etc.

26

Tab Panel Navigation: Example Results (Devel. Mode)



27

Top-Level Navigation: Responding to Stack Tokens

```
private class HistoryHandler
    implements ValueChangeListener<String> {
    public void onValueChange(
        ValueChangeEvent<String> event) {
        String historyToken = event.getValue();
        if (historyToken.startsWith("tab")) {
            selectTab(historyToken);
        } else if (historyToken.startsWith("stack")) {
            selectStack(historyToken);
        }
        ...
    } else {
        selectTab("tab0");
    }
} ...
```

28

Top-Level Navigation: Responding to Stack Tokens (con)

```
private void selectStack(String historyToken) {
    historyToken = historyToken.substring(5);
    int stackIndex = 0;
    try {
        stackIndex = Integer.parseInt(historyToken);
        if (stackIndex < 0 && stackIndex > 4) {
            stackIndex = 0;
        }
    } catch (NumberFormatException e) {
    }

    stackPanel.showWidget(stackIndex);
    contentPanel.showWidget(1);
}
```

Strip out 'stack' from token 'stack/' so / can be parsed as an integer.

Within StackLayoutPanel, select stack with the corresponding index.

Switch to deck with index 1, i.e., StackLayoutPanel within the DeckLayoutPanel, which resides in the center of the DockLayoutPanel.

29

Stack Panel Navigation: Storing Tokens

```
public class StackPanelExample extends StackLayoutPanel {  
    public StackPanelExample(int numTabs) {  
        ...  
        addSelectionHandler(new StackSelectionHandler());  
    }  
    ...  
  
    private class StackSelectionHandler  
        implements SelectionHandler<Integer> {  
        public void onSelection(SelectionEvent<Integer> event) {  
            History.newItem("stack" + event.getSelectedItem(),  
                false);  
        }  
    }  
}
```

This flag is very important. It tells GWT that we do NOT want a history event to be fired and therefore processed by the HistoryHandler. All we want to do here is store the token on the URL that represents our current navigational state so we can get to it later, i.e., #stack0, #tstack1, etc.

30

Stack Panel Navigation: Example Results (Devel. Mode)

After loading the page, we press the "Show StackLayoutPanel" button (more on that later). That shows #stack0. Then, we click on heading "Show Panel 1", then heading "Show Panel 4".

31

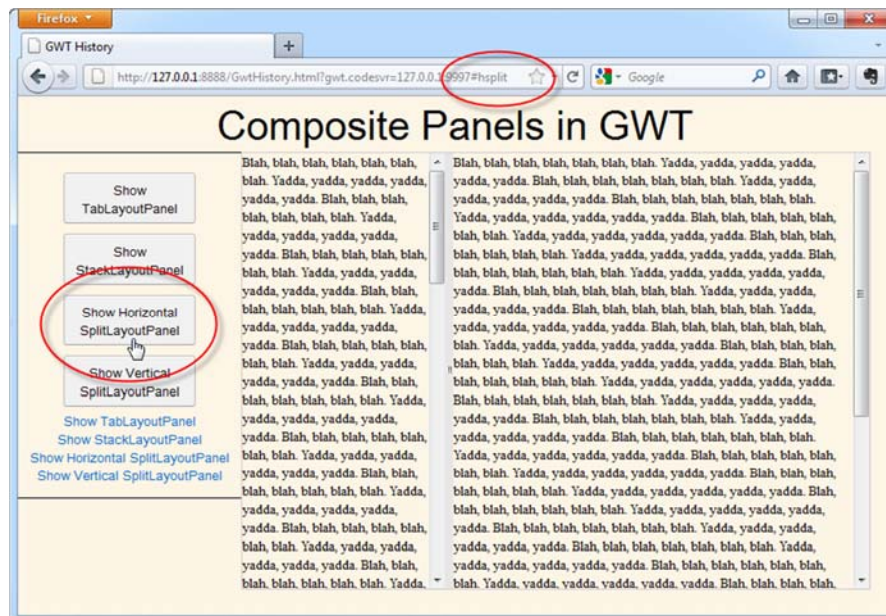
Top-Level Navigation: Responding to 'hsplit' Token

```
private class HistoryHandler
    implements ValueChangeHandler<String> {
    public void onValueChange(
        ValueChangeEvent<String> event) {
        String historyToken = event.getValue();
        ...
    } else if (historyToken.equals("hsplit")) {
        contentPanel.showWidget(2);
    }
    ...
} else {
    selectTab("tab0");
}
} ...
}
```

32

HSplitPanel Panel Navigation: Example Results (Devel. Mode)

After loading the page, we press the "Show Horizontal SplitLayoutPanel" button (more on that later). That shows #hsplit.



33

Using Buttons & Hyperlinks to Trigger Navigation

- **Button widgets**

- Designate Button for screen navigation
- Assign ClickHandler to handle onClick event
- Option 1: No history handling
 - Affect UI directly inside the ClickHandler
- Option 2: History handling
 - Add new history token to URL, force URL token evaluation
 - I.e., call History.newItem("token"), without 2nd arg as false

- **Hyperlink widgets**

- Designate the Hyperlink with the label and value as the token to navigate to
 - E.g., new Hyperlink("Go Home", "home"); // no # needed
- Clicking on the link automatically changes the URL and triggers URL history token evaluation

34

Top-Level Navigation: Navigation Panel

```
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);
    }
}
```

This handler no longer changes the screens, but issues a History event that gets processed by the HistoryHandler.

35

Top-Level Navigation: Navigation Panel (continued)

```
...
navPanel.add(
    new Hyperlink("Show TabLayoutPanel", "tab0"));
navPanel.add(
    new Hyperlink("Show StackLayoutPanel", "stack0"));
navPanel.add(
    new Hyperlink("Show Horizontal SplitLayoutPanel",
        "hsplit"));
navPanel.add(
    new Hyperlink("Show Vertical SplitLayoutPanel",
        "vsplit&range=20"));
return navPanel;
}
```

Hyperlinks automatically add tokens to History and trigger history token evaluation automatically, i.e., no different if you were to type the new URL into the address bar.
Link text. Corresponding History token.

We'll discuss this construct in later in the slides.

36

History Aware Button ClickHandler

```
private class ButtonHandler implements ClickHandler {
    private int subPanelIndex;

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

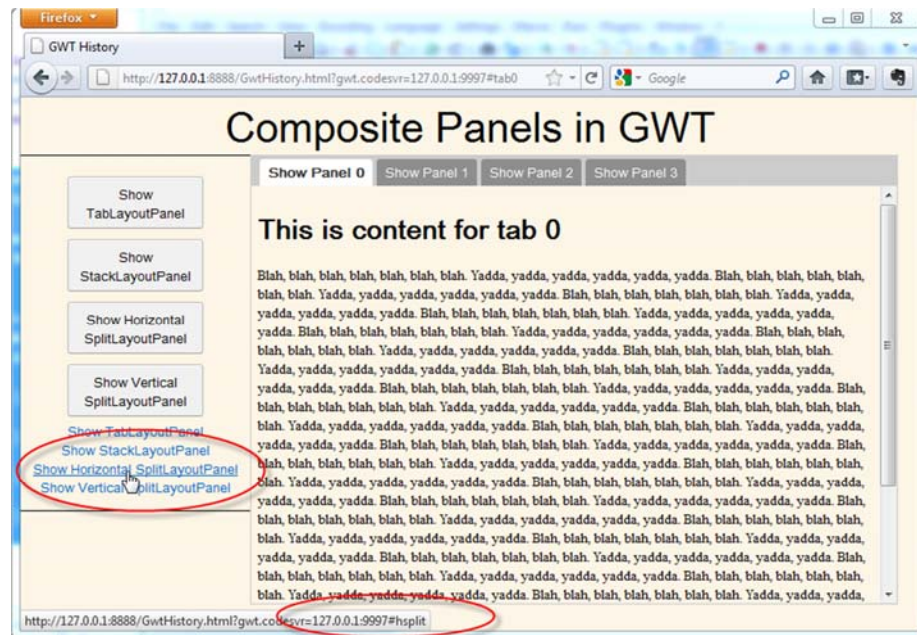
    public void onClick(ClickEvent event) {
        if (subPanelIndex == 0) {
            History.newItem("tab0");
        } else if (subPanelIndex == 1) {
            History.newItem("stack0");
        } else if (subPanelIndex == 2) {
            History.newItem("hsplit");
        } else {
            History.newItem("vsplit&range=10");
        }
    }
}
```

Causes 'tab0' to be the new history token on the URL, e.g.,
http://host/GwtHistory.html#tab0 and fires an event to force
HistoryHandler to evaluate the URL and act accordingly.

We'll discuss this construct in later in the slides.

37

Example: Hyperlink Results



38

3 Most Common Mistakes with History Handling

- **At startup, assigning ValueChangeHandler, but forgetting to fire event that forces URL token evaluation**
 - Must call `History.fireCurrentHistoryState()`
 - If not done, pasted URL with tokens will not be evaluated, and token will be ignored
- **On click of some widget that already changes UI state, calling `History.newItem("newToken")` without second argument of false**
 - Without false, fires a history handling event and forces reevaluation of the newToken history token
 - No new event is needed, just saving navigational state on the URL
 - At best evaluates token again and wastes processing
 - At worst, leaves an extra history token in the history
- **Correcting missing or malformed token**
 - See next slide

39

Responding to Tokens: Missing or Malformed Tokens

- If token is missing or malformed, we always want to pick something sensible to navigate to
 - E.g., ...} else { selectTab("tab0"); }
- Don't correct the URL by setting correct token
 - If you do this, and browser has important history before the malformed token, it will be very hard for user to “jump over” the bad token to get back
 - Every time the “back” button is clicked and user ends up on malformed token, your app will force browser back to correct token
 - So, don't do `History.newItem("goodToken")` and even `History.newItem("goodToken", false)`
 - Leave the “bad/missing token” URL alone. All you care about is consistent navigational state, e.g.: bad token = default screen

40

© 2013 Marty Hall & Yaakov Chaikin



Advanced Issues

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.

Complex Navigation

- **Session data: option 1**
 - Suppose you have an online exam. Merely storing which page the user is on is not sufficient, because if you navigate back to that page, you must also recreate that user's answer. So, store a session identifier as well as a page identifier in session.
- **Session data: option 2 – don't have any!**
 - True beauty of Web 2.0 Ajax applications!
 - Make page number, student ID tokens in your URL
 - <https://exams.com/#studentId=23&page=3>
 - Obviously, student would have to be authenticated first
 - Store everything in some persistent storage, e.g., database
 - When URL is requested recreate full state (view & data)
 - **Your app is now server and client agnostic**
 - No need for clustering. Simple load-balancing scales!

42

Complex Navigation (Continued)

- **Bypassing entries that shouldn't be saved**
 - Not every click and view needs to have its own token
 - Some pages are inherently transient (e.g., error messages) and should not be recorded in the history.
- **Responding to complex actions**
 - You might want to save some state after a drag-and-drop, or after user answers x correct questions. You must store enough information to identify the screen to show, and you must have code that will display that screen.

43



Recreating Navigational State from Server Data

Customized Java EE Training: <http://courses.coreservlets.com/>

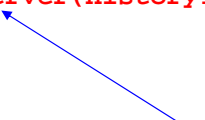
GWT, Java 7 and 8, JSF 2, PrimeFaces, HTML5, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, REST, Hadoop, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Main Idea

- **History tokens contain not only name of screen, but also some data (e.g., range)**
- **In order to retrieve the same UI state, server needs to be contacted**
 - Need custom parser to separate tokens
 - Very similar to parsing request parameters, where one of the parameters is the name of the view
 - Need to pass token data to some mechanism to retrieve with GWT RPC
 - Need to show some intermediate state of the view with some message, e.g., “Loading...”, so user knows what’s going on
- **Visiting the same URL produces random number within the *same range***

Parse History Token That Contains Data

```
private class HistoryHandler implements ValueChangeListener<String> {
    public void onValueChange(ValueChangeEvent<String> event) {
        String historyToken = event.getValue();
        ...
    } else if (historyToken.startsWith("vsplit&range=")) {
        selectVSplit(historyToken);
    } else {
        selectTab("tab0");
    }
}
...
private void selectVSplit(String historyToken) {
    historyToken = historyToken.substring(13);
    vSplitPanel.displayRandomFromServer(historyToken);
    contentPanel.showWidget(3);
}
}
```



46

Create DataService and DataServiceAsync Interfaces

```
package coreservlets.client;
...
@RemoteServiceRelativePath("data-service")
public interface DataService extends RemoteService {
    public RandomNumber getRandomFromServer(String range);
}

```

```
package coreservlets.client;
...
public interface DataServiceAsync {
    void getRandomFromServer(String range,
        AsyncCallback<RandomNumber> callback);
}

```

47

Create DataServiceImpl Servlet

```
package coreservlets.server;
..
public class DataServiceImpl extends RemoteServiceServlet implements
    DataService {
    ...
    public RandomNumber getRandomFromServer(String rangeString) {
        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
        }
        return (new RandomNumber(rangeString));
    }
}

<!-- web.xml snippet -->
<servlet>
    <servlet-name>DataService</servlet-name>
    <servlet-class>coreservlets.server.DataServiceImpl</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>DataService</servlet-name>
    <url-pattern>/gwhistory/data-service</url-pattern>
</servlet-mapping>
```

On the server, pause for 5 seconds to simulate prolonged processing.

48

Create DataService Proxy

```
public class GwtHistory implements EntryPoint {
    ...
    public static DataServiceAsync SERVICE_PROXY;
    public void onModuleLoad() {
        SERVICE_PROXY = GWT.create(DataService.class);
        ...
    }
    ...
}
```

One way to share central service proxy with the rest of components in the application.

49

VSplitPanelExample Modified with Server Data Retrieval

```
public class VSplitPanelExample extends SplitLayoutPanel {
    private SimplePanel centerContainer;

    public VSplitPanelExample() {
        addNorth(makeScrollableText(), 200);
        centerContainer = new SimplePanel();
        add(centerContainer);
        addStyleName("split-panel");
    }
    ...
}
```

50

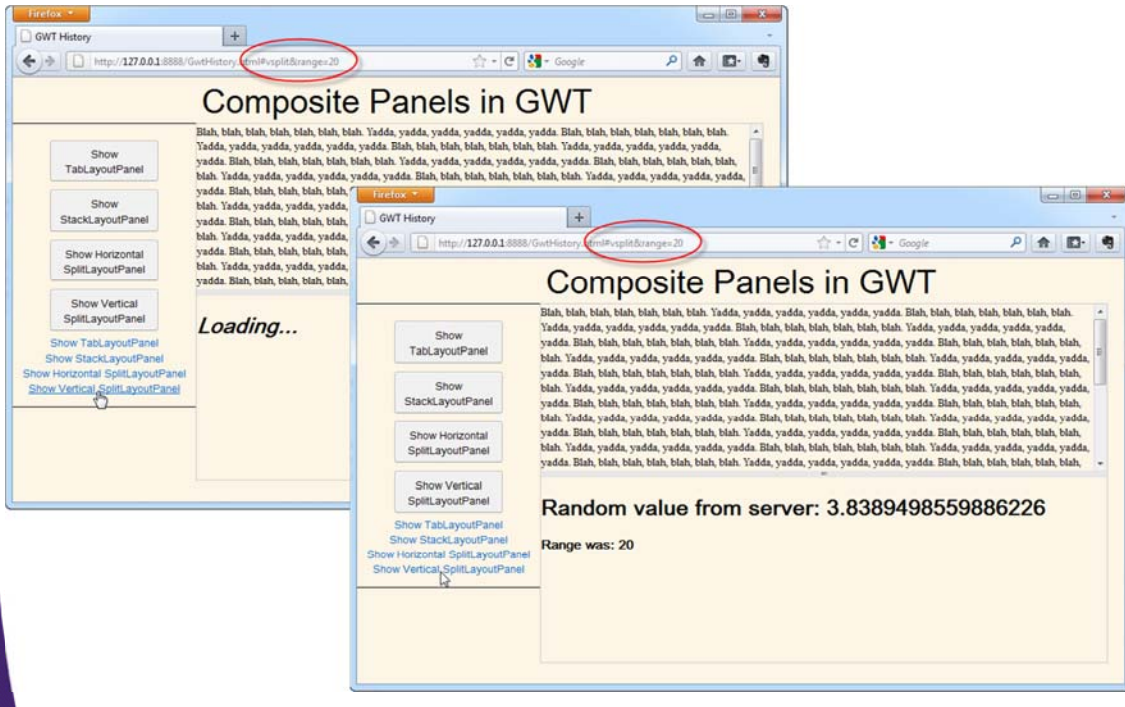
VSplitPanelExample Modified with Server Data Retrieval

```
public void displayRandomFromServer(String rangeString) {
    centerContainer.clear();
    centerContainer.add(new HTML("<h1><i>Loading...</i></h1>"));
    GwtHistory.SERVICE_PROXY.getRandomFromServer(rangeString,
        new AsyncCallback<RandomNumber>() {
            public void onSuccess(RandomNumber result) {
                centerContainer.clear();
                centerContainer.add(
                    new HTML("<h1>Random value from server: "
                        + result.getValue() + "</h1><h3>Range was: "
                        + result.getRange() + "</h3>"));
            }

            public void onFailure(Throwable caught) {
                Window.alert("Unable to communicate with server.");
            }
        }); ...
}
```

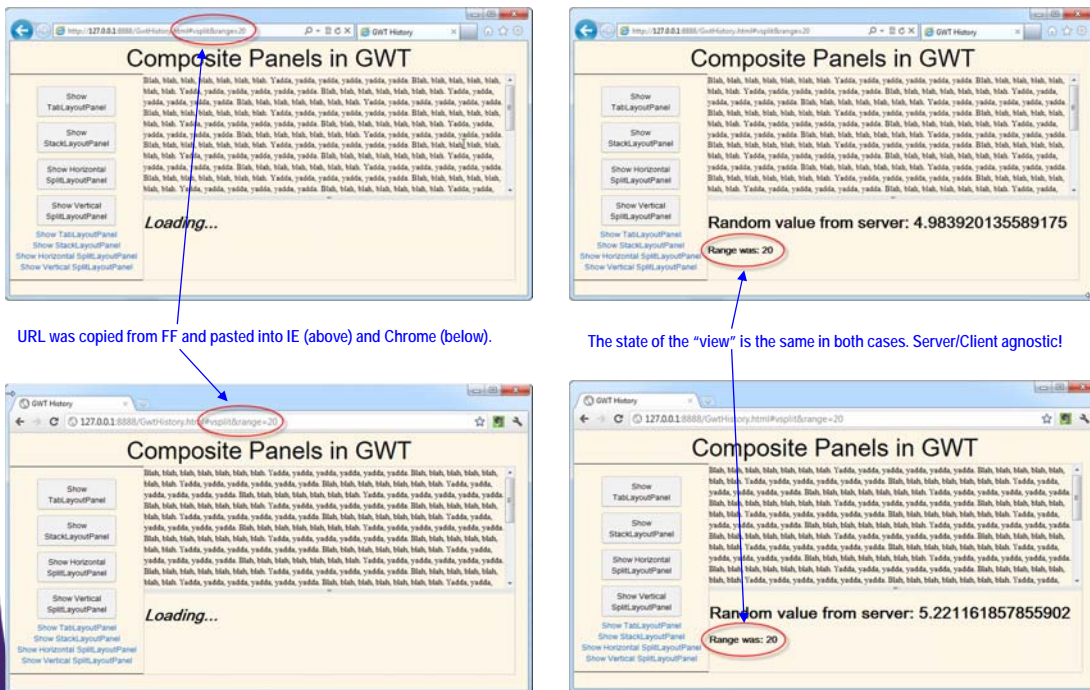
51

VSplitPanelExample: Result (Hosted Mode - Firefox)



52

VSplitPanelExample: Result (Hosted Mode – IE, Chrome)



53



Wrap-up

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.

Summary

- **HTML setup**
 - Keep “iframe” in the body
- **Java setup**
 - `History.addValueChangeHandler(new HistoryHandler());`
 - `History.fireCurrentHistoryState();`
- **Storing history tokens**
 - `History.newItem("someString", false)`
- **Responding to history tokens**
 - In `onValueChange` of `HistoryHandler`
 - `String linkTarget = event.getValue();`
 - `navigateBasedOn(linkTarget);`
 - Tokens sent on initial user action as well as from back button or entering a bookmarked URL (since token is on end of URL)
 - Don't forget initial `History.fireCurrentHistoryState`
- **Planning ahead**
 - Top-level app needs to know how to get to all sub-pages that can be bookmarked or access with “Back” button



Questions?

[JSF 2, PrimeFaces, Java 7 or 8, Ajax, jQuery, Hadoop, RESTful Web Services, Android, HTML5, Spring, Hibernate, Servlets, JSP, GWT, and other Java EE training.](#)

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.