Sakai and GWT
Toward Improved UX and Easy Web 2.0 Development all in Java

Claude Coulombe
Université de Montréal

Sacha Leprêtre
CRIM & Sakai Québec

Sakai
1-3 JULY 2008
PARIS, FRANCE

OpenSyllabus: HEC MONTRÉAL Université de Montréal
Sakai Québec CRIM
Old fashioned web - Click and wait!
Web 2.0: User Experience (UX)
Web 2.0: User Experience (UX)

- Perceived 2nd generation of web sites and services
- Improved UX is what Web 2.0 is all about
- Students ask for responsive and dynamic web interfaces and web interface similar to desktop interface
- Sakai must evolve toward Web 2 and deliver a better UX
- Improving UX → more complex GUI → more work for developers
- How to keep happy users & developers?
- But, great technology doesn't give great UX...
- The real magicians are the UI designers
OpenSyllabus - Short Demo

- What we have done with OpenSyllabus...
AJAX - A breakthrough!
AJAX - A breakthrough!

- Ajax eliminates painful page loading!
- Ajax stands for Asynchronous JavaScript and XML
- XMLHttpRequest JavaScript Object allows asynchronous requests for data to the server and updates the web page without doing a full page reload
- Invented by Microsoft
- Without Ajax we were still stuck with click and wait interface
- The result is more responsive and dynamic Webapps
- But, Ajax is based on Client-side JavaScript
Looking for a silver bullet...

- Hundreds of JavaScript Libraries and Ajax Frameworks
- Which one will be the good one?
- Survey of Sakai’s Ajax and alternative technologies:
  - UX Richness of the libraries
  - Easy dev, quick learning curve
  - Easy integration with Sakai
  - Open Source License
  - Documentation
  - Endorsement
  - Cross browsing compatibility
  - Java based
  - Dev tools / IDE (eclipse)
  - Debugging/Test

Sakai
1-3 July 2008
Paris, France
Problems with JavaScript...

So, he didn't know JavaScript well enough...
Problems with JavaScript

- Real JavaScript gurus are rare.
- JavaScript implies, working around browser quirks, memory leaks and long load times
- Not a real O O programming language
- Not designed for big programs
- Lack of modularity, scalability and development & debugging tools
- Some JavaScript tools are good and some not very good.
- Good JavaScript libraries like (jQuery or YUI) makes much easier to use JavaScript but it's still JavaScript coding.
What is GWT

http://code.google.com/webtoolkit/
What is GWT

- GWT (Google Web Toolkit) is nothing less than a completely original approach to web development that allows Java developers to create Rich Webapps without having to write any JavaScript.
- GWT Cross-Compiler takes client side Java code and generates cross-browser JS.
- Support Internet Explorer, Firefox, Mozilla, Safari, and Opera browsers
- In short, you write, test and debug your client-side code in Java and let GWT cross-compiler convert it all into cross-browser JS.
- Client-side is then pure JS & HTML.
GWT is all Java

- Java to JS Cross-Compiler
- Friendly Open Source Apache 2 license
- Community support
- Improving quickly
- Hosted Mode using JVM for quick dev. cycle
- IDE integration Eclipse, Netbeans, IntelliJ
- Rich built-in Ajax library
- Rich UI libraries similar to SWING
- Only one language: JAVA
GWT - Architecture Overview

- GUI Library
- Widgets & Panels
- Server Communication
- RPC & Ajax
- XML parser
- History Management
- JUnit Integration

- JSNI JavaScript Native Interface
- Java to JavaScript Compiler
- JRE Emulation Library

GWT API
GWT - Cross-Compiler Java to JavaScript

GWT version 1.5 supports:
* Firefox 1.0, 1.5, 2.x, 3.0
* Internet Explorer 6, 7
* Safari 2.0, 3.0
* Opera 9.0

Write Once...

Run Everywhere!
GWT - Communication library & Ajax

- RPC (Remote Procedure Call)
- REST (REpresentational State Transfer)
- Basic Ajax tools:
  - HTTPRequest
  - RequestBuilder
  - FormPanel
- Support for complex data
  - JAVA SERIALIZATION
- And also:
  - XML
  - JSON (JavaScript Object Notation)
GWT - JSNI: JavaScript Native Interface

- GWT simplifies integration with existing JS code

- JavaScript ↔ Java

- Automatic inclusion of external JavaScript
GWT - User Interface Library

- Widgets & Panels
- Handling Events
- CSS support
- I18N
Widgets & Panels

- Standard HTML tags, such as img, anchor, hyperlink
- Button, radio button, toggle button, checkbox button
- Menus, cascading-menus, drop down menus
- Text Box, Text Area
- Tabs, ListBox, Dialog Box
- Splitters
- Complex widgets such as Tables, File Upload boxe, Tree widgets, Rich Text Editor
- Panels helps for the layout
GW T - Handling Events

- Use the "listener interface"
- Similar to SWING

```java
Button aButton = new Button("Click Me");
aButton.addClickListener(new ClickListener() {
    public void onClick(Widget sender) {
        // handle the click event
    }
});
```
GWT - CSS Support

- Separation of code and presentation
- 3 methods to manage style name
  - setStyleName
  - addStyleName
  - removeStyleName
- Java code:
  ```java
  public ListWidget(String Item) {
    ...
    setStyleName("gwt-ListWidget");
  }
  ```
- CSS file:
  ```css
  .gwt-ListWidget {
    background-color:black;
    color:lime;
  }
  ```
I18N

- Built-in I18N mechanism
- Based on properties files
- GWT generates different versions of your Webapp for each language
- At the runtime, GWT will choose the appropriate version
GW T - Browser’s History Management

- Use the browser's "back" button correctly

- Simple History API based on a stack of tokens

  ```javascript
  History.newItem("newToken")
  ```

- HistoryListener

  ```javascript
  History.addHistoryListener(controller)
  ```
GWT - Software Engineering for Ajax

- Using advanced software engineering
- Using establish O O design patterns
- Using powerful Java IDE
  - Edit / test / debug / refactor cycle
  - Debugging support
  - Compile-time error checking
- Testing & Debugging in “Hosted Mode”
- Logging support
- JUnit support
GWT - “Hosted mode”

- GWT Webapp can run in “Hosted Mode”
- In “Hosted Mode”, a JVM executes the GWT code inside an embedded browser window
- Running GWT Webapp in “Hosted Mode” makes debugging easy
  - Edit your source
  - Refresh
  - See the results
GWT - “Hosted mode”
GWT - “Web Mode” / Deployment

- Once tested in “Hosted Mode”, you can compile your Java source code to JavaScript
- When compiled the Client-side is now pure JavaScript and HTML
- To deploy your Webapp in production, you would move the files in your www/... directory to your web server
GWT integration into SAKAI
GWT integration into SAKAI

- A « Sakai tool » based on GWT
- Sakai platform used as Stateless Server (backend part of Sakai)
- JSP entry role
- RPC servlet
- Spring/Hibernate relations (difficulties ... solutions)
- Client MVC Architecture
- SDATA
GWT integration into SAKAI

Traditional WebApp
GWT integration into SAKAI

Sakai platform used as Stateless Server (backend part of Sakai)

Indirect access to the Service layer of the server

GWT Client 1

Model for A

getModel

update

update

saveModel

GWT Client 2

Model for B

getModel

update

update

saveModel

Sakai Server

Persistent Models

Presentation Layer

Business Logic / Data Access
GW T integration into SAKAI

Sakai platform used as Stateless Server (backend part of Sakai)

A real SAKAI tool!

Javascript running on the Internet Browser

OSYL GWT Client

MVC

client side MVC Approach with in GWT

Presentation Layer

Business Logic / Data Access

Sakai Server

Sakai services

OSYL services

OSYL Servlet(s)

OSYL SAKAI Tool Application

Sakai Framework

DAO

Business logic

RPC

AJAX exchanges of data model
GW T integration into SAKAI

App File structure, what’s new?

- 3 main directories
  - API(interfaces)
  - Impl(implentations)
  - Tool(webapp)
    - src/java
      - RPC servlets
      - RPC Interface
    - src/webapp
      - Index.jsp
      - Folder of the compiled GW T content (js,html... )
GW T integration into SAKAI
Where is GW T source code?

- 2 projects:
  - One for the backend part (sakai webapp shown previously).
  - One project for the presentation layer, the client part (i.e. the GW T Source code).
GW T integration into SAKAI
Compilation/deployment scripts steps

- 3 possible use cases:
  A. Client update only
  B. Backend update only
  C. RPC interface update or Full compilation

  ➔ A only or B only make faster the development cycle

  Note: A only can either run in hosted or web mode.

To save time, use hosted mode as much as possible for client code developments
GWT integration into SAKAI

JP main roles

- Gives access to the GWT compiled $\mathcal{S}$ (slots)
  - Can also use some logic to choose between different javascript compiled application

```html
<!-- This script loads our GWT compiled module. -->
<script language='javascript' src='org.sakaiquebec.opensyllabus.OsylEditorEntryPoint/org.sakaiquebec.opensyllabus.OsylEditorEntryPoint.nocache.js'></script>
```
GW T integration into SAKAI

**JSP main roles**

- **Provides the CSS Link**
  ```html
  <link rel="stylesheet" type="text/css"
  href="osylcoconfigs/default/skin/osylcore.css" />
  ```

- **Initializes some META to set the language (I18N)**
  ```html
  <%@ page import="org.sakaiproject.util.ResourceLoader" %>
  ...
  <%
  ResourceLoader rb = new ResourceLoader();
  Locale sessionLocale = rb.getLocale();
  String locale = sessionLocale.toString();
  ...
  %>
  <html>
  <head>
  <meta name="gwt:property" content="locale=<%=locale%>"/>
  ...
GWT integration into SAKAI

JSF main roles

- Controls the tool display size

... 
<html>
<head>
...

<!-- Headers from Sakai -->
<%= request.getAttribute("sakai.html.head") %>
<script> // Size of the JS application
function myLoad() {
setTimeout("<%= request.getAttribute("sakai.html.body.onload") %>", 500);
}
</script>
</head>
<body onload="myLoad()"
GWT integration into SAKAI
Servlets main roles

Servlets are used for:
- RPC from GWT client
- Indirect Spring services access
- Security management
- Extra request information access
GW T integration into SAKAI
Servlet and tool configuration

- **Sakai tool**
  - Tool Registration is done by providing the tool xml file.
    Ex: `webapp/tools/sakai.opensyllabus.tool.xml` file
  - And configuring the `web.xml`

  ```xml
  ...  
  <listener>
    <listener-class>org.sakaiproject.util.ToolListener</listener-class>
  </listener>
  ...
  ```

- **GW TRPC Servlet:**

  ```java
  public class OsylEditorGwtServiceImpl extends RemoteServiceServlet implements OsylEditorGwtService {
  ```

- **Other web.xml configuration**: Sakai filters, servlet definitions …:
GW T integration into SAKAI

SPRING: How to join spring services from the RPC servlet?

3 explored possibilities:

- Centralized approach: Using the applicationContext with one main backing bean able to join other services
- Not centralized: Call the Spring services as your need from your RPC servlet:

```java
import org.springframework.web.context.WebApplicationContext;
...
void myRPCmethod(){
    WebApplicationContext context = WebApplicationContextUtils.getWebApplicationContext(getServletContext());
    osylSiteExplorerService = (OsylSiteExplorerService)
    context.getBean("org.sakaiquebec.osylsiteexplorer.api.OsylSiteExplorerService");
    ...
}
```

- Other approach Spring service mapping:
  http://g.georgovassilis.googlepages.com/usingthespringgwtcontroller
GW T integration into SAKAI

SPRING: Centralized approach for Tool

- Session
- Authz
- Site
- Content
- ContentService
- SecurityService
- DAOs

OsylBean (service main Bean)

- Index.jsp
- * .js GWT...
- OsylServlet
- OtherServlet

Sakai Services

- OSYL Services
- Top Down Spring injection

Included in the SPRING CONTEXT

Service Usage

Spring injection
- No injection

1-3 JULY 2008
PARIS, FRANCE
GW T integration into SAKAI

Spring backing bean

- **Main backing bean, included in Spring context**
  - Define a webapps/WEB-INF/applicationContext.xml
  
  - Include a contextLoaderListener into your webapps/WEB-INF/web.xml:

```
<listener>
  <listener-class>org.sakaiproject.util.ContextLoaderListener</listener-class>
</listener>
```
GW T integration

Client Design principles

• Big paradigm shift for traditional web development (server based to client oriented)
• Main client design pattern MVC + Observer + Event driven
• Reusable views components (gwt.Composite)
• Client model as JAVA classes (DATA->POJO)
• Asynchronous programming (Callbacks)
• RPC, REST / Java Serialization, XML, JSON
• Security: server based
GW T integration
GW T Composite class

- Reusable views components (gwt.Composite)
- Aggregation of GW T widgets but also Composites

```java
... import com.google.gwt.user.client.ui.Composite;
...
public class OsylTreeView extends Composite {
    private Tree osylTree;
    ...
}
```
GW T integration
Client MVC

OSYL
GWT Client

Model: POJO(GWT)
Notify events to subscribers (views) on model update

View:
Display the data and manage the user event on GUI.
In GWT: composite

Controller:
Makes the relationship between models and views
Listen to view events and manage user’s actions

M
update
notify
access
data exchange
event

V

C
GWT integration
Client MVC specific approach

- More a M&VC approach
- Hierarchical nature of our data (tree) ➔ Hierarchical views (HMVC)
  - Composite view aggregates other composite views and so on...
  - And each view are based on a sub-model part of the model
  - Tight relation between model update and related view update (thanks to events)
GW T integration into Sakai
Leveraging Sakai technology

- **SDATA**
  - A REST Servlet in Sakai, Data exchanges in JSON format (site data, resource data etc..)
  - Used to create or read data from the Sakai’s resources repository
  - JSON Parser of GW T
GW T integration into Sakai

Difficulties & solutions

- Spring - GW T wiring $\Rightarrow$ ApplicationContext
- Dealing with Hosted Mode vs Web mode $\Rightarrow$ Mockup data for the hosted mode
- I18N not dynamic $\Rightarrow$ develop our own mechanism
- CSS: still browser’s dependent $\Rightarrow$ check with different browser
- GW T is a toolkit not a framework, this means that GW T does not prescribe a way to build an application $\Rightarrow$ established our own design guidelines
- Third party libraries: GW T-ext license change, different quality of components $\Rightarrow$ be careful
- Third party tools: GW T-Designer: very promising but $\Rightarrow$ we recommend to use it for mockup and UI design.
Advantages

- Development time efficiency
- Quicker response to user’s actions
- Powerful & efficient in resources usage both network & server
- Good to add Ajax to Webapps
- Good to build complex “desktop-like” applications
- Rapid development and debugging with common IDEs as Eclipse
- Open source, free and well documented
- Only one language : JAVA
- Rich libraries of components
- Familiar to Java developers
- Supported by GOOGLE...
- Not magic but has the potential to be the "next big thing"
Disadvantages

- Needs good knowledge of Java programming
- Components (Widgets) are from different sources and qualities
- Depends on cross-compiler performances
- Few cross-browser compatibility problems (really?)
- Need to learn CSS & restrictions of browser-based apps
- We have to keep an eye on security issues!
- GWT is a toolkit not a framework
- GWT won't solve every problem you may encounter creating Ajax or RIA
What’s next?

- Release our code to the SAKAI community
- Rewrite using Java 1.5 and generic types
- Experiment Sakai portal tool (OSYLSiteExplorer)
- Create SAKAI GWT library
- Explore Accessibility using GWT 1.5
- Improving performance
- SOLO mode (Google Gears)
- Improve benefit from the OpenSource Google API ecosystem
  - Simplified GWT APIs for: Ajax Search, Gears, Gadgets and Maps
  - http://code.google.com/p/gwt-google-apis/
SAKAI + GWT - The next big thing!

[Image of Sakai and GWT logos]
Resources and books

- **Google Web Toolkit Applications**
  by Ryan Dewsbury
  Prentice Hall
  (December 15, 2007)
  www.gwtapps.com

- **Google Web Toolkit Solutions**
  by David Geary, Rob Gordon
  Prentice Hall
  (November 17, 2007)
  www.coolandusefulgwt.com