AJAX Applications with Google Web Toolkit (GWT)

Svetlin Nakov
Director Training and Consulting Activities
National Academy for Software Development
academy.devbg.org
What is Google Web Toolkit (GWT)?
What is GWT?

• **Google Web Toolkit (GWT)** is:
  • Open source framework for easy creating rich AJAX applications with Java
  • Powerful JavaScript code generator
  • You write code in Java and it is compiled to JavaScript
    • Pure JavaScript / DHTML at the client side
    • Supports Firefox, IE and Safari
    • Pure Java at the server side
Why GWT?

- No need to learn/use JavaScript language
  - Leverage existing Java skills
- No need to handle browser incompatibilities
  - GWT handles them for you
- No need to learn/use DOM APIs
  - Use pure Java APIs
- No need to handle forward/backward buttons
  - GWT handles this for you
- No need to build commonly used widgets
  - GWT provides most of them
GWT Features

- GWT provides to developers:
  - API for creating GUI applications, similar to Swing and SWT
  - API for manipulating the Web browser's DOM (Document Object Model)
  - Java-to-JavaScript compiler
    - JavaScript skills not required
  - Environment for running and debugging GWT applications called **GWT Shell**
Hosted and Web Mode

- GWT applications can run in hosted and Web mode
  - Hosted mode
    - Java bytecode runs in the JVM, in the GWT Shell environment
    - Allows debugging
  - Web mode
    - JavaScript/HTML runs in the Web browser
GWT Architectural Components

Class Libraries
- JRE emulation library: java.lang and java.util
- GWT Web UI class library

Development Tools
- Java-to-JavaScript compiler
- GWT Shell hosted mode environment

Source: http://code.google.com/webtoolkit/
Getting Started with GWT

Downloading and Installing GWT
Creating GWT Projects
• GWT is free and open source

• Download the latest version from: http://code.google.com/webtoolkit/
  • It is ZIP file, e.g. gwt-windows-1.4.60.zip

• Extract the archive. It contains:
  • applicationCreator, projectCreator – create Eclipse/Ant based GWT projects
  • samples directory – example apps
  • doc – reference docs and javadocs
Creating GWT Projects

- Creating GWT project for building with **ant**
  ```cmd
  projectCreator.cmd -ant MyProject -out c:\myproject
  ```

- Creating Eclipse project for GWT app
  ```cmd
  projectCreator.cmd -eclipse MyProject -out c:\myproject
  ```

- Creating GWT module and scripts for compilation and execution
  ```cmd
  applicationCreator.cmd -eclipse MyProject -out c:\myproject example.client.MyProject
  ```
Creating GWT Projects

Live Demo
Creating GWT Modules

Creating Modules, Adding Widgets, Handling Events
GWT Modules

- **GWT module** is a set of files that define the client side and server side of a GWT application or library
  - **Client side**
    - HTML, CSS and images
    - Java code that is compiled to JavaScript
  - **Server side**
    - Contains server side Java code, e.g. RPC services invoked from the client code
GWT Module Structure

- Each module has:
  - Name: `nasd.example.Sumator`
  - Base directory: `nasd.example`
    - Contains the module structure
  - Public directory: `nasd.example.public`
    - Contains HTML, CSS and images
  - Client code package: `nasd.example.client`
    - Compiled to JavaScript
  - Server code package: `nasd.example.server`
  - GWT module descriptor: `Sumator.gwt.xml`
• GWT Sumator is simple application for calculating sum of two integers
  • Adding widgets to the root panel

```java
public class Sumator implements EntryPoint {
  private TextBox textBoxNumber1 = new TextBox();
  private TextBox textBoxNumber2 = new TextBox();
  private TextBox textBoxSum = new TextBox();

  public void onModuleLoad() {
    RootPanel rootPanel = RootPanel.get();
    rootPanel.add(textBoxNumber1, 12, 36);
    rootPanel.add(textBoxNumber2, 88, 35);
    rootPanel.add(textBoxSum, 168, 35);
  }
}
```
• Handling events

```java
Button buttonCalcSum = new Button();
buttonCalcSum.setText("Calculate Sum");
rootPanel.add(buttonCalcSum, 12, 64);

buttonCalcSum.addClickListener(new ClickListener() {
    public void onClick(Widget sender) {
        int number1 = Integer.parseInt(textBoxNumber1.getText());
        int number2 = Integer.parseInt(textBoxNumber2.getText());
        int sum = number1 + number2;
        String sumStr = Integer.toString(sum);
        textBoxSum.setText(sumStr);
    }
});
```
Creating GWT Sumator

Live Demo
GWT Widgets and Panels

Creating and Using GWT Widgets
GWT Widgets

• GWT widgets are UI components
  • Like Swing components but rendered to HTML, CSS and JavaScript
  • Written purely in Java
  • Can manipulate the DOM and modify it dynamically
  • CSS / HTML formatting rules still apply
• Widgets dramatically facilitate creating rich user interfaces in Web
Standard GWT Widgets (1)

- **Button**
  - Normal Button
  - Disabled Button

- **TextBox**
  - text box...

- **ListBox**
  - List 0
  - List 1
  - List 2
  - List 3
  - List 4

- **Tree**
  - foo@example.com
  - Inbox
  - Drafts
  - Templates
• CheckBox

• TextArea

• Hyperlink

• MenuBar
Standard GWT Widgets (3)

- RichTextArea

- Table

- DialogBox
• GWT panels are container controls that layout widgets in various ways

• FlowPanel

• TabPanel
• DockPanel

• HorizontalSplitPanel

• AbsolutePanel
GWT KitchenSink Sample

Live Demo
Creating Custom Widgets

- To create a custom widget you can extend the *Widget* or *Composite* class

```java
public class GroupBoxPanel extends FlowPanel {
    private Element legend;
    public GroupBoxPanel() {
        Element fieldset = DOM.createFieldSet();
        this.legend = DOM.createLegend();
        DOM.appendChild(fieldset, legend);
        setElement(fieldset);
    }
    public String getCaption() {
        return DOM.getInnerText(this.legend);
    }
    public void setCaption(String caption) {
        DOM.setInnerText(this.legend, caption);
    }
}
```
GroupBoxPanel

- Example of GroupBoxPanel

Step 1: User Information

First Name: 
Last Name: 
E-Mail: 

Step 2: Preferences

☐ Receive emails about our new services
☐ List me in the public directory

OK Cancel
Invoking RPC Services

Creating and Invoking Server Side Functionality
What is RPC Service?

- **RPC = Remote Procedure Call**
  - Invoke remote methods (through HTTP)
  - Unidirectional: clients calls the server

- RPC calls in GWT run asynchronously
  - The client is notified when the invocation complete (by callback)
Why RPC Calls?

- RPC allows dynamic interaction with the server side, e.g. retrieve data from database
- RPC allows separating the presentation and business logic
  - UI logic runs at the client side
  - Business logic runs on the server side
- RPC calls are fast, need low bandwidth
  - Only data is transferred, no UI elements
  - Efficient data format
Data Marshaling

- Data marshaling / unmarshaling
  - Java/JavaScript objects are serialized and transferred to the other side
  - On the other side the objects are deserialized and used
- You can marshal built-in types by default
  - int, float, boolean, String, etc.
- User defined types must implement IsSerailizable to be marshaled
How RPC Works?

ServiceDefTarget (interface)

RemoteService (interface)

RemoteServiceServlet (class)

YourServiceAsync (interface)

YourService (interface)

YourServiceImpl (class)

YourServiceProxy (class)

Translatable Java code (runs as JavaScript on client)

Standard Java code (runs as bytecode on server)

Source: http://code.google.com/webtoolkit/
Creating RPC Service (1)

- Define the service interface

```java
public interface TasksService extends RemoteService {
    public Task[] getAllTasks();
    public Task[] getTasksByCategory(String category);
    public void addTask(Task task);
}
```

- Define the corresponding asynchronous interface

```java
public interface CategoriesServiceAsync {
    public void getAllTasks(AsyncCallback callback);
    public void getTasksByCategory(String category, AsyncCallback callback);
    public void addTask(Task task, AsyncCallback callback);
}
```
Creating RPC Service (2)

- Defining the data transfer objects (DTO)

```java
public class Task implements IsSerializable {
    private String title;
    private String description;
    private String category;

    public Task(String title, String description, String category)
        public Task(String title, String description, String category)
    {
        this.title = title;
        this.description = description;
        this.category = category;
    }

    public String getTitle() { ... }
    public String getTitle() { ... }
    public void setTitle(String title) { ... }
    public void setTitle(String title) { ... }
    public String getDescription() { ... }
    public String getDescription() { ... }
    public void setDescription(String description){...}
    public void setDescription(String description){...}
    public String getCategory() { ... }
    public String getCategory() { ... }
    public void setCategory(String category) { ... }
    public void setCategory(String category) { ... }
}
```
• Defining the service implementation

```java
public class TasksServiceImpl
    extends RemoteServiceServlet
    implements TasksService {

    @Override
    public Task[] getAllTasks() { ... }

    @Override
    public Task[] getTasksByCategory(String category) {
        ... }

    @Override
    public void addTask(Task task) { ... }
}
```
• Register the service implementation class as RPC service servlet
  • In the `.gwt.xml` deployment descriptor:

```
<servlet path="/TasksService"
  class="nasd.example.server.SomeServiceImpl"/>
```

• Obtain the asynchronous service proxy:

```
TasksServiceAsync tasksServiceAsync =
  (TasksServiceAsync) GWT.create(TasksService.class);
```

• This is slow operation!
• Use caching if possible
Calling RPC Services (2)

• Assign service entry point servlet:

```java
ServiceDefTarget target = (ServiceDefTarget) tasksServiceAsync;
target.setServiceEntryPoint(GWT.getModuleBaseURL() + "/TasksService");
```

• Call the service:

```java
tasksServiceAsync.getAllTasks(new AsyncCallback() {
    public void onSuccess(Object result) {
        Task[] tasks = (Task[]) result;
        // Process results here ... 
    }
    public void onFailure(Throwable caught) {
        Window.alert("Error: "+ caught.toString());
    }
});
```
Invoking RPC Services

Live Demo
GWT: The Dark Side
Frequent Problems with GWT
Drawbacks of GWT (1)

- Supports Java 1.4 only
  - No support for Java 5 generics, etc.
- Client can not import source files from directory which is not subdirectory of the "client" directory
  - Code duplication is sometimes required
- Runs very slowly in hosted mode
  - Startup time for large projects can reach 1-2 minutes
Drawbacks of GWT (2)

- Standard widgets are not good enough
  - No good table, no good dialogs, no drag-and-drop, no , etc.
- RPC calls can not be run synchronous
  - GWT is AJAX-based technology!
- No good framework for modal dialogs
  - Issues not addressed: dialog creation, data validation, returning data, handling [OK] / [Cancel] buttons, glass panel, etc.
Drawbacks of GWT (3)

- Not well established technology
  - The community is not very large
- Have bugs
  - Sometimes the GWT Shell hangs-up
- No backward compatibility
- Good free UI designers not available
  - Instantiations have commercial product GWT Designer
- Takes lots of resources on the client PC
GWT Advanced Table

GWT Table Widget with Paging, Sorting and Data Filtering
My Contribution

• GWT Advanced Table widget
  • Allows paging, sorting, data filtering
GWT Advanced Table

Live Demo
Links

• Google Web Toolkit – Official Web Site
  • [http://code.google.com/webtoolkit/](http://code.google.com/webtoolkit/)

• GWT Advanced Table

• GWT Widget Library

• Svetlin Nakov's Blog
  • [http://www.nakov.com/blog/](http://www.nakov.com/blog/)
Questions?

http://www.nakov.com/blog/