

Taking a Look at the Google Web Toolkit

Sarasota Java User Group
March 19, 2008

Your Speaker

R.J. Salicco {

Axiomatic IT Incorporated
rj.salicco@axiomaticit.com
<http://thejavajar.com>

}

Google Web Toolkit (GWT)

- Introduction
- Installation
- Creating a Project
- The UI, Events and Listeners
- Running Your GWT Application
- Services
- Architectural Considerations
- Questions

Google Web Toolkit (GWT)

What is GWT?

GWT is Java Web development, literally.

```
...  
    button1.addClickListener(new ClickListener() {  
        public void onClick(Widget sender) {  
            label1.setText("Hello World!");  
        }  
    });  
...
```

Your Java code is compiled into AJAX (HTML and JavaScript) Web applications like Google Maps and Google Mail.

What? Yes, you write your front-end code in Java*.

*Well, you need to write some HTML.

Getting Started with GWT

Where do we start?

1) Install the Java SDK

2) Download GWT

<http://code.google.com/webtoolkit/download.html>

3) Unzip the Package

4) Create Your Application

Lets Write Some Code

Coding By Convention

Your Package Structure

com/mysite/myapp/

Project root containing module XML files.

com/mysite/myapp/client/

Client-side source files and sub-packages. Compiled into JavaScript.

com/mysite/myapp/server/

Server-side code and sub-packages. Compiled to Java. (Servlets, Data Access Objects)

com/mysite/myapp/public/

Static resources that can be served publicly. (HTML, CSS, Images)

Lets Write Some Code

1) projectCreator

Used to make an Eclipse project for one of the samples that comes with GWT

projectCreator

```
[-ant projectName] - Ant build file to compile source.  
[-eclipse projectName] - Generate an Eclipse project.  
[-out dir] - Directory to write output files.  
[-overwrite] - Overwrite any existing files.  
[-ignore] - Ignore existing files, do not overwrite.
```

2) applicationCreator

Generates a starter application and scripts for launching hosted mode and compiling to JavaScript

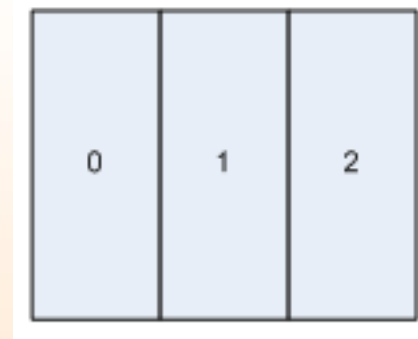
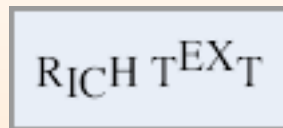
applicationCreator

```
[-eclipse projectName] - Debug launch configurations for Eclipse project.  
[-out dir] - Directory to write output files.  
[-overwrite] - Overwrite any existing files.  
[-ignore] - Ignore existing files, do not overwrite.  
className - Fully-qualified name of the application class to write.
```

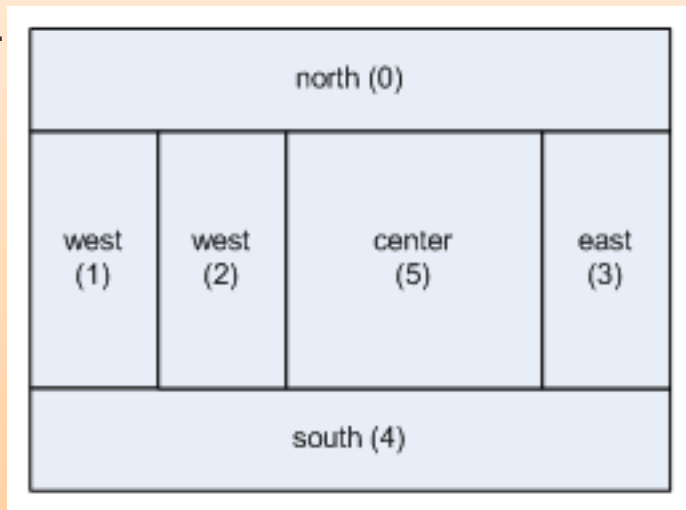
The User Interface

Widgets and Panels

Think HTML in terms of Swing and SWT...



...this is where GWT uses Widgets and Panels to dynamically create HTML...



...Widgets and Panels can be found in the GWT user-interface library...more information can be found on the GWT site...these images are there too!

GWT Events and Listeners

Event Driven HTML Applications

Your Widgets make calls to one or more methods that are defined within a listener interface.

If your Class wants to receive events of a particular type, the Class should implement the appropriate listener interface and then pass a reference of itself to the Widget to “subscribe” to a set of events.

...right off of the GWT site...

Anonymous Inner Class

```
public void anonClickListenerExample() {  
  
    Button b = new Button("Click Me");  
    b.addClickListener(new ClickListener() {  
        public void onClick(Widget sender) {  
            // handle the click event  
        }  
    });  
}
```

GWT Events and Listeners

Creating a Listener Class

```
public class ListenerExample extends Composite implements ClickListener {

    private FlowPanel fp = new FlowPanel();
    private Button b1 = new Button("Button 1");
    private Button b2 = new Button("Button 2");

    public ListenerExample() {
        initWidget(fp);
        fp.add(b1);
        fp.add(b2);
        b1.addClickListener(this);
        b2.addClickListener(this);
    }

    public void onClick(Widget sender) {
        if (sender == b1) {
            // handle b1 being clicked
        } else if (sender == b2) {
            // handle b2 being clicked
        }
    }
}
```

Running Your GWT Application

Development vs Integration Testing

1) Hosted Mode

Hosted mode is where you will be developing and testing your application. You are interacting with GWT without your application being translated into JavaScript. Your JVM is running your application through GWT plumbing keeping your development environment productive.

2) Web Mode

Web Mode refers to the testing you will do when you are running in a normal browser where your application runs as pure JavaScript as it is intended to be deployed. Your application runs completely as JavaScript and does not require any browser plug-ins or JVM.

Google Web Toolkit (GWT)

Remote Procedure Calls (RPC)

Working with Data

GWT runs as an application within the browser, making calls to service, or server-side code, which is based on the Servlet Architecture.

Actually Making a Call

The process of making an RPC from the client always involves the exact same steps.

1. Instantiate the service interface using [GWT.create\(\)](#).
2. Specify a service entry point [URL](#) for the service proxy using [ServiceDefTarget](#).
3. Create an [asynchronous callback](#) object to be notified when the RPC has completed.
4. Make the call.

GWT RPC

Creating Your First Service

What do we need? 2 interfaces and a Servlet implementation.

```
package com.mysite.myapp.client.service;  
  
public interface MyService extends RemoteService {  
    public Return Type methodName(ParamType1 param1, ParamType2 param2);  
}
```

```
package com.mysite.myapp.client.service;  
  
public interface MyServiceAsync {  
    public void methodName(ParamType1 param1, ParamType2 param2,  
        AsyncCallback callback);  
}
```

GWT RPC

Creating Your First Service

```
package com.mysite.myapp.server.service;  
  
public class MyServiceImpl extends RemoteServiceServlet implements MyService {  
    public ReturnType methodName(ParamType1 param1, ParamType2 param2) {  
        Do some work.  
        return ReturnType;  
    }  
}
```

Don't Forget, MyServiceImpl is a Servlet! It needs to be configured in your **web.xml** file.

If you want to be able to run your Services/Servlets during Hosted mode development, you need to add the Servlet to your project module XML file. Something like this:

```
...  
<servlet path='/MyService' class='com.thejavajar.server.service.MyServiceImpl'/>  
...
```

GWT RPC

Calling Your Service

```
MyServiceAsync myService = (MyServiceAsync) GWT.create(MyService.class);
```

```
ServiceDefTarget endpoint = (ServiceDefTarget) myService;  
String moduleRelativeURL = GWT.getModuleBaseURL() + "MyService";  
endpoint.setServiceEntryPoint(moduleRelativeURL);
```

```
AsyncCallback callback = new AsyncCallback() {  
    public void onSuccess(Object result) {  
        // do some UI stuff to show success  
    }  
  
    public void onFailure(Throwable caught) {  
        // do some UI stuff to show failure  
    }  
};
```

```
myService.methodName(param1, param2, callback);
```

GWT RPC

Putting It All Together

- We have our Client-side Code - JavaScript and HTML
- We have our Server-side Code - Java

How about deployment? Maintenance?

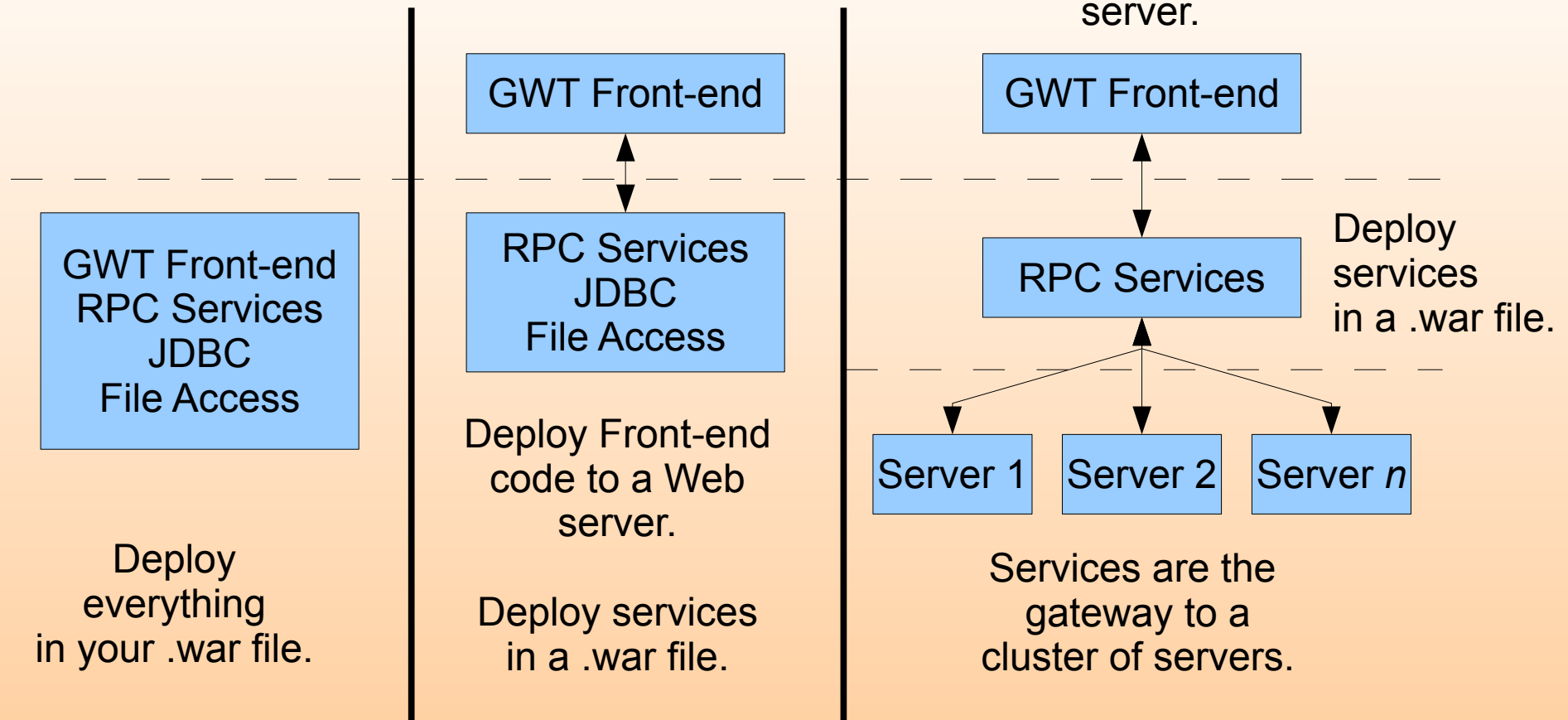
What else does GWT support?

- Unit Testing
- Internationalization
- JavaScript Native Interface (JSNI)
 - JavaScript in Java classes to access low-level browser functionality

GWT RPC

Architectural Considerations

Paths to Consider When Using GWT RPC Services:



Google Web Toolkit

Questions?

Comments?