Taming AJAX with GWT

Scott Blum

Scott Blum — Taming AJAX with GWT
Overview

- Why AJAX?
- Why GWT?
- How GWT Works
- Add GWT to your App
- Advanced Techniques
- Summary
- Q & A
Demo – A Production App

AJ AX Texas Hold’em

http://www.gpokr.com/
AJ AX: Easy on Users

- No installation
  - Every application is just a URL away

- Secure... the mantra we teach our parents
  - installing things = mostly unsafe
  - surfing the web = mostly safe

- Web simplicity
  - AJ AX pages have the familiar web look and feel
  - Web simplicity – back, forward, links, URLs
AJAX: Easy on Servers

- State maintained on client
  - Only send the data, not the presentation
  - Send only deltas instead of entire UI state
  - Clean separation of concerns
  - Allows servers to be effectively stateless

- Leverage user’s CPU and RAM
  - Do everything you can on the client
  - Much faster for the user, too!
Traditional App Cycle

[Diagram showing the Traditional App Cycle process with steps involving User interacts, Event data & Session state, Reconstitute Request Tree, Apply Request Values, Process Validations, UI Logic, Update Model Values, Invoke Application, Render Response, Business Logic, and Network.]
AJ AX App Cycle

Service invoked

User interacts

Browser

UI Logic

“Real-time” UI updates

Service response

Service params

Network

Invoke Service

Business Logic

Response data

Client

Server
Summary and Q&A

- Use AJAX
- The End
Overview

- Why AJAX?
- Why GWT?
- How GWT Works
- Add GWT to your App
- Advanced Techniques
- Summary
- Q & A
AJ AX: Hard on Developers

- JavaScript is the “J” in AJ AX
- No strong type system
  - typos + expandos = bug-o-s
  - Imagine this gem on line 5912 of your script
    
    ```javascript
    x.component = document.getElementById("x");
    // a "spelling bug" that will bite much later
    ```
- Poor IDE support
- Debuggers... for *some* browsers...
AJ AX:   Hard on Developers

- If you support IE6 and FF you’re doing well!
- But then there’s
  - IE7
  - Older Mozillae
  - Safari
  - Opera
  - Konqueror
  - You name it
AJ AX: Hard on Developers

- No installation… sort of
  - In a sense, it’s also *always-reinstall* (better be small)
  - Modularizing JavaScript is really tricky

- Secure… sort of
  - Hard enough just to make AJ AX work at all!
  - Lots of JS = lots of attack surface

- Has web simplicity… sort of
  - History, bookmarks and even hyperlinks misbehave
  - Badly coded AJ AX is worse than traditional HTML
AJ AX: Hard on Developers

- Lots of technologies to worry about (you need regexps to list them all on one page)
  - HTTPS?, [DX]?HTML (3.2|4.0), CSS[1-3]
  - DOM Level[0-3]
  - (Java|ECMA|J |VB)Script
  - (X|VR?|Math)ML
  - SVG, Canvas, Flash
  - JSON, SOAP, XML-RPC
AJ AX: Hard on Developers

- Lots of JS libraries to choose from, but...
  - You often wind up paying for the whole library when you only need 5% of it
  - Inline comments and well-named identifiers add to code size
  - The compiler can’t tell you if your using the library wrong
  - Upgrading to new versions?
Solution: Use GWT

- Write your AJAX code in Java
- Compile it with a real Java compiler
- Debug it in a real Java IDE
- Unit test it with JUnit
- Javadoc it without script size worries
- Reuse it on a class or package basis
- Share it as a JAR
Solution: Use GWT

- Java has a lot of advantages over JavaScript
  - Static type checking
  - Great IDE support
    - Code completion
    - Quick correction
    - Lint, style, and other warnings
    - Refactoring
  - Established code patterns
  - Scales to large projects and large teams
Solution: Use GWT

(+50 more slides about how Java whoops JavaScript)
Overview

- Why AJAX?
- Why GWT?
- How GWT Works
- Add GWT to your App
- Advanced Techniques
- Summary
- Q & A
How GWT Works

- Build your app in Java
- DOM-based Widget framework
- Debug in a real Java IDE
- Compile to JavaScript
- Deploy compiled output as static content
- Built-in RPC based on Servlets
Code Sample – A Simple App

<html>
  <head>
    <meta name='gwt:module' content='com.google.gwt.sample.hello.Hello'>
    <title>Hello</title>
  </head>
  <body>
    <script language="javascript" src="gwt.js"></script>
  </body>
</html>
public class Hello implements EntryPoint {

    public void onModuleLoad() {
        Button b = new Button("Click me",
                               new ClickListener() {
            public void onClick(Widget sender) {
                Window.alert("Hello, AJAX");
            }
        });
        RootPanel.get().add(b);
    }
}
Demo – A Simple App

Hello, AJAX
How GWT Works

GWT

Client
- Widget Framework
- JRE Emulation
- Browser Abstraction

Server
- Java Servlet
- Serialization Framework
- Other Servers

Tools
- Java to JavaScript Compiler
- Hosted Mode
- Deferred Binding
How GWT Works

Client

- Widget Framework
- JRE Emulation
- Browser Abstraction
Client Framework

- gwt-user.jar
- Open Source: Apache 2.0
- Browser abstraction layer
- Full set of UI Widgets, styled with CSS
  - Real HTML elements
  - Event driven
  - Like Swing or SWT
- Panels for managing layout
Demo – Widgets and History

Mail and KitchenSink
Client Framework

- Growing subset of the JRE
  - Much of java.lang, java.util
  - No reflection (by design!)
  - What makes sense on a browser
    - No threads, file i/o, etc.
- RPC, XML, JSON, i18n, other acronyms
How GWT Works

Server

- Java Servlet
- Serialization Framework
- Other Servers
GWT’s Built-in RPC

- gwt-servlet.jar
- Open Source: Apache 2.0
- Servlets on the server side
- Explicit server calls
  - This is a good thing 😊
- Declare a remote interface
- Send POJOs, primitives, collections
Code Sample – RPC Interfaces

// Synchronous interface (declared)
public interface SpellingService extends RemoteService {
    String[] suggest(String wordToCheck);
}

// Asynchronous interface (derived)
public interface SpellingServiceAsync {
    void suggest(String wordToCheck, AsyncCallback callback);
}
SpellServiceAsync spell = GWT.create(SpellService.class);

spell.suggest("component", new AsyncCallback() {

    void onSuccess(Object result) {
        String[] alts = (String[]) result;
        if (alts.length > 0)
            showSuggestions(alts);
    }

    void onFailure(Throwable e) {
        reportProblem(e);
    }

});
// Servlet class
public class SpellingServiceImpl
    extends RemoteServiceServlet
    implements SpellingService {

    public String[] suggest(String maybeMisspelledWord) {
        List list = new ArrayList();
        // populate list; run any server-side code you like
        return (String[]) list.toArray(new String[list.size()]);
    }

}
Demo – RPC

DynaTable
Network Usage

- Traditional HTML
- GWT First Run
- GWT Other Runs

Bytes Transferred

Page 1  Page 2  Page 3  Page 4  Page 5  Page 6  Page 7
GWT’s Built-in RPC

- Always asynchronous
- Generated classes perform serialization
- Unpublished super-efficient wire format
- Libraries support custom RPC
How GWT Works

Tools

- Java to JavaScript Compiler
- Hosted Mode
- Deferred Binding
GWT Tools

- `gwt-dev-(windows|linux|mac).jar`
- Free; Currently Closed-Source
- Compiler
- Hosted Mode support
- Common Infrastructure
GWT Tools – Compiler

- Compile Java into JavaScript
- Full Java 1.4 language support
- Deploy compiled scripts on any web server
- Optimize like heck
  - Monolithic compile
  - Type Tightening
  - Aggressive Code Pruning
public class Hello implements EntryPoint {

    public void onModuleLoad() {
        Button b = new Button("Click me",
            new ClickListener() {
                public void onClick(Widget sender) {
                    Window.alert("Hello, AJAX");
                }
            });
        RootPanel.get().add(b);
    }
}
Hello AJAX – Generated Script

// from class Hello
function _$onModuleLoad(_this$static){
    var _b = _$Button(new _Button(), 'Click me',
        _$Hello$1(new _Hello$1(), _this$static));
    _$add(_get(), _b);
}

// from anonymous ClickListener class
function _onClick(_sender){
    _alert('Hello, AJAX');
}
Hello AJAX – Obfuscated

```javascript
function ne(oe){
    var pe=qe(new re(),'Click me',se(new te(),oe));
    ue(ve(),pe);
}

function xe(ye){
    ze('Hello, AJAX');
}
```
GWT Tools – Pay for Use

![Graph showing the script size in bytes for various GWT tools.](image-url)
GWT Tools – Hosted Mode

- Run under a real 1.4 or 5.0 JVM
- Run inside the browser with full DOM access
- Full debugging and IDE support
- Platform support:
  - Windows with IE
  - Linux with Mozilla
  - Mac OS X with Safari may already be available
- Embedded Tomcat serves files from your classpath for easy setup
GWT Tools – Deferred Binding

- Replace classes at compile time
  
  ```java
  DOM dom = (DOM)GWT.create(DOM.class);
  dom.doSomethingPlatformSpecific();
  ```

- At runtime, `dom` actually refers to the most appropriate subclass

Deferred binding compiles in the correct subclass based on selection rules.
GWT Tools – Deferred Binding

Download exactly what you need in a single can't-go-wrong chunk

Single Java Code Base

Cache it on the client until the sun explodes
GWT Tools – Deferred Binding

- Why do at runtime what you can do at compile time?
- Because of type tightening, polymorphic calls can be inlined
- Generators provide an optimized alternative to runtime reflection
  ➢ Our own RPC is built on it
Overview

- Why AJAX?
- Why GWT?
- How GWT Works
- Add GWT to your App
- Advanced Techniques
- Summary
- Q & A
Add GWT to Your App

- Attach a `<meta>` tag to your HTML
- Include the `gwt.js` script
- Add an `id` to elements you’re interested in
- Deploy compiler output as static content
- Works with any HTML-generation approach
Demo – Adding GWT

A Simple JSP Example
Overview

- Why AJAX?
- Why GWT?
- How GWT Works
- Add GWT to your App
- Advanced Techniques
- Summary
- Q & A
Advanced Techniques

- Effective RPC
- Hanging RPC
- JavaScript native interface (JSNI)
Effective RPC

- Calls are Asynchronous only
  - Think event-driven

- Limit the number of invocations
  - Think “chunky”, not “chatty”
  - Ideally, no more than 1 call per user action

- Limit to 2 outstanding calls at once
  - Browser-imposed limit
Hanging RPC

- Imagine a chat room app
- Naïve solution:
  - Clients poll on timer
  - Most calls return empty
  - Slow perceived response
  - Wasted processing/bandwidth
Hanging RPC

- *a.k.a. “server push” or “comet”*
  - Not really possible with HTTP; this fakes it

- Basic algorithm
  - Client calls to check for events
  - Server blocks until an event occurs
  - Client calls again to wait for next event

- Pro: Less processing/bandwidth than polling

- Con: Each client constantly uses a TCP connection and server thread; not as easy to throttle clients
More Info on RPC

- Ron Bodkin’s *AJAX Performance and Monitoring*
  - More depth on these topics
  - Jetty continuations and hanging RPC
JavaScript Native Interface

- Hand-write JavaScript into your Java class
  - Like inline assembly in C
- Foundation of GWT’s framework
- Interact with browser directly
- Call an external script
  - Wrap existing JavaScript with type-safe Java signatures
- Constructs JS supports better than Java
  - Function pointer table
  - String-key map
Code sample – JSNI

native void alert(String msg) /*-{alert(msg);}*/;
JSNI Features

- Implement a Java method in JavaScript
- Call Java methods from JavaScript
- Read and write Java fields from JavaScript
- Marshal values between Java/JavaScript
  - Super efficient in production
- Throw exceptions across Java/JavaScript boundaries
- Our entire client framework is based on it
  - If you don’t like ours, you can write your own 😊
native JavaScriptObject lazyInitMap() /*-{ 
  var map = this.@com.example.foo.client.Foo::stringMap;
  if (map == null) {
    map = this.@com.example.foo.client.Foo::stringMap = {};
  }
  return map;
}*/;

native String getValue(String value) /*-{ 
  var map = this.@com.example.foo.client.Foo::lazyInitMap()();
  var result = map[key];
  return (result == null) ? null : result;
}*/;

native void putValue(String key, String value) /*-{ 
  var map = this.@com.example.foo.client.Foo::lazyInitMap()();
  map[key] = value;
}*/;
Overview

- Why AJAX?
- Why GWT?
- How GWT Works
- Add GWT to your App
- Advanced Techniques
- Summary
- Q & A
Growing GWT Community

- Community and Support
  - 5000+ members on the developer forum
  - Books, articles, blogs
  - Meta-sites (e.g. http://gwtpowered.org)
  - 645,000 results for “google web toolkit”

- Libraries and Applications
  - 25 projects on Google Code Project Hosting
  - GWT Widgets on SourceForge
  - JetBrains' JET markup framework for GWT

- Tools
  - JetBrains IntelliJ IDEA support for GWT built into Version 6.0
  - Googlipse, an open source Eclipse plug-in for GWT
  - VistaFei for GWT – Visual Designer
  - Instantiations GWT Designer – Visual Designer; round-trip to Source!
Instantiations GWT Designer

[Diagram of a GWT Designer interface with various components and controls, including button, checkBox, and other GUI elements.]

Scott Blum — Taming AJAX with GWT
Production Apps

- [http://www.dimdim.com](http://www.dimdim.com)
  Web-based conferencing

- [http://www.indigopuzzles.com](http://www.indigopuzzles.com)
  High-quality online puzzles

- [http://www.notetwonote.com](http://www.notetwonote.com)
  Student self-testing
Conclusion

- AJAX is hard: you tame it... or *vice-versa*
  - You need leverage to take full advantage of AJAX
- PhD in browser quirks is no longer a prereq
- We will share our best work and ideas with you, and we hope you will return the favor
- Much more to come... see you online!

http://code.google.com/webtoolkit/
Q & A

Ask me anything! 😊

scottb@google.com