Resource Bundles and Linkers in the Google Web Toolkit

Bob Vawter
Google, Inc
May 28, 2008

What is GWT?

The Google Web Toolkit is a platform for building high-performance AJAX applications.

GWT's mission is to radically improve the web experience for users by enabling developers to use existing Java tools to build no-compromise AJAX for any modern browser.

Presentation Overview

- Compiler Primer
- Resource Bundling
  - Rationale
  - Overview of Generators
  - ImmutableResourceBundles (aka "FooBundle")
  - Additional Bundling Techniques
- Linkers
  - When to build your own Linker
  - Component parts
  - Building a simple Linker
  - Going offline with Google Gears
  - Reducing round-trips
  - Integrating with a Gadgets container

GWT Compiler Primer

How do Generators and Linkers fit into the compilation process?

1. Read source .gwt.xml and .java files
2. Replacing rebinds may cause more Java code to be created:

```
<when-type-assignable class="...">
<generate-with class="com.foo.rebind.SomeGenerator">
GWT.create(Some.class)
```

   - Generators may create more Java source and/or resources
   - Access to the closed type-system via TypeOracle
3. Compile Java to JavaScript
4. Run the Linker chain
   - Linkers operate on a set of Artifacts
   - The Primary Linker is responsible for module bootstrap
   - Pre- and Post-Linkers run before and after the Primary Linker
5. Deploy
   - Output from the compiler may be very different than a standard compile if custom Linkers are used

Resource Bundling

Why do we want to bundle resources together?

- Basically boils down to increasing the efficiency of the app
- Fewer round-trips can mean a snappier app
  - The application's "time-to-ready" can be significantly decreased if required resources are baked-in
- Programmatic access to application resources beneficial to developers
  - Strings and image resources can be externalized
  - Easy hook-point for I18N support
  - Widgets can be written to have resources injected
- Opportunities for compile-time manipulation of the resource
- As the best-practices for resource access change, the framework can be evolved

Overview of Generators

How is bundling implemented?

- Primary means of extending the functionality of the GWT platform
- Compile-time generation of Java source

```
public abstract String generate(
    TreeLogger logger, GeneratorContext context,
    String typeName) throws UnableToCompleteException;
```

- We can use Generators to create glue code at compile time or for wholesale synthesis of new functionality
- Use the Java type system to encode extrinsic information about resources
- Code generation triggered by GWT.create()

```
private final Resources r = GWT.create(Resources.class);
```

- Additional resources available on the web, especially Ray Cromwell's blog entries

ImmutableResourceBundles

(aka "FooBundle")

What are the design goals for a resource bundling system?

- Support "perfect caching" of resources in the client
- Leverage the compiler to pay for only the resources you use
- Allow for optimizations specific to a given content type
- Integrate through existing GWT extension hooks
- Arbitrary extensibility to new types of resources
- Support for resource localization

What are non-goals for a resource bundling system?

- Providing a filesystem abstraction
- Integrating resources not available at compile-time

ImmutableResourceBundles

Integration

How can we make using FooBundles easy?

1. Import the module

```
<inherits name="com.google.gwt.libideas.ImmutableResources" />
```

2. Declare the interface

```
public interface Resources extends ImmutableResourceBundle, Source, Slid
  @Resource("com/google/gwt/bobv/gio/client/r/io_dots.png")
  ImageResource smallDots();
  @Resource("com/google/gwt/bobv/gio/client/r/titleSlide.html")
  TextResource titleSlide();
}
```

3. Use the generated class

```
private final Resources r = GWT.create(Resources.class);
```

ImmutableResourceBundles

I18N / L10N support

How do we support the other NN% of the web?

- ImmutableResourceBundle is sensitive to the locale property
- A best-match algorithm is used to select the source file

```
Locale := en_US_gwt;
@Resource("words.txt");
Search path:
  words_en_US_gwt.txt
  words_en_US.txt
  words_en.txt
  words.txt
```

- Token-based substitution can also be used

```
@Resource("com/foo/resources/%locale%/words.txt");
```

ImmutableResourceBundles

Perfect Caching

How can we leverage existing web-deployment techniques to make resource delivery maximally-efficient?

- Start with strongly-named files. The filenames are an MD5-sum of the contents of the resource

```
548CDF11D6FE9011F3447CA200D7FB7F.cache.txt
```

- Use "cache forever" headers

```
Last-Modified: Thu, 15 Nov 2007 15:37:26 GMT
Content-Type: text/plain
Cache-Control: public, max-age=31536000
Expires: Tue, 19 May 2009 23:33:47 GMT
```

- Use data: URLs and inline the resource into the module. This leverages perfect caching of the module, but trades size for reduced numbers of round-trips and synchronous access.

ImmutableResourceBundles

Leveraging the Compiler

How can we take advantage of the compiler?

- Intrinsic resource metadata can be hoisted into the compilation unit
  - Example: Image height and width baked into emitted JS

```
@Resource("com/google/gwt/bobv/gio/client/r/io_dots.png")
ImageResource smallDots();
```

```
$ImageResourcePrototype(new ImageResourcePrototype(),
    this$static.imageResourceBundleUrl,
    0, 0, 1248, 646);
```

- The GWT compiler aggressively prunes unused code; this can be extended to resources as well

ImmutableResourceBundles

CssResource Example

Compile-time CSS generation allows no-runtime-cost extensions to CSS

```
@def MAX-HEIGHT 748px;
@def MAX-WIDTH 1004px;
.slide {
  height: MAX-HEIGHT;
  width: MAX-WIDTH;
```

```
@if user.agent safari {
@media screen {
  pre {
    \-webkit-border-radius: 10px;
    \-webkit-box-shadow: 10px 10px 10px darkGray;
  }
```

```
/* Refers to function ImageResource smallLogo(); */
@sprite background smallLogo;
/* It's css, so we can define more properties */
.background {
```

ImmutableResourceBundles

Extensibility

How can we support new resource types or optimizations in a forward-compatible way?

- Instead of having type-per-interface (e.g. ImageBundle), specify the resource type via the method declaration
- This has the interesting property of allowing the different types of resources to cooperate

```
interface MyResources extends ImmutableResourceBundle {
  @Resource("myCss.css")
  CssResource myCss();
  @Resource("myCursor.cur")
  DataResource dataResourceFunction();
}
```

```
@url myUrl dataResourceFunction;
.someClass { cursor: myUrl, pointer; }
```

- As capabilities around a particular type of resource increase, new methods can be added to existing interfaces

Additional Bundling Techniques

JSON-style

How can we make a data-driven app start faster?

- Inline the initial data payload into the host page
- Add a json payload

```
<script>window.myPayload = {'key' : 'value', ...}</script>
```

- Combine with GWT 1.5 JavaScriptObject support

```
class MyPayloadJSO extends JavaScriptObject {
  public native String getKey() /*-{
    return this.key;
  }-*/;
}
```

```
private native MyPayloadJSO getPayload() /*-{
  return $wnd.myPayload;
}-*/;
```

Additional Bundling Techniques

RPC-style

How can we make a GWT RPC app start faster?

- Similar to inlining a JSON payload, but using GWT RPC payloads
- Use a servlet to construct the host HTML page
- Assume a synchronous RPC interface

```
interface MyRpcInterface extends RemoteService {
  MyPayload someMethod(int a, int b);
}
```

- Construct the embedded payload using

```
com.google.gwt.user.server.rpc.RPC.encodeResponseForSuccess()
```

- Place the payload into the host HTML page

```
<script>window.myPayload = "RPC string"</script>
```

- Cast an the asynchronous proxy to SerializationStreamFactory

```
SerializationStreamFactory f = GWT.create(MyRpcInterface.class);
MyPayload p = f.createStreamReader(getPayloadString()).readObject();
```

GWT Compiler Primer

How do Generators and Linkers fit into the compilation process?

1. Read source .gwt.xml and .java files
2. Replacing rebinds may cause more Java code to be created:

```
<when-type-assignable class="...">
<generate-with class="com.foo.rebind.SomeGenerator">
GWT.create(Some.class)
```

   - Generators may create more Java source and/or resources
   - Access to the closed type-system via TypeOracle
3. Compile Java to JavaScript
4. Run the Linker chain
   - Linkers operate on a set of Artifacts
   - The Primary Linker is responsible for module bootstrap
   - Pre- and Post-Linkers run before and after the Primary Linker
5. Deploy
   - Output from the compiler may be very different than a standard compile if custom Linkers are used

Linkers

Overview

- Linkers are new in GWT 1.5 and are used to separate the Java-to-JavaScript compilation process from packaging and bootstrap.
- All files in the output directory are placed there by the Linker stack.
- Built-in Linkers:
  - Standard iframe "std": Used in all prior versions of GWT
  - Cross-site "xs": Originally introduced in GWT 1.4 as -xs.nocache.js
  - Single-script output "sso": New in GWT 1.5
- Linkers are only run during a web-mode compile
- The Linker mechanism is intended to be extensible by GWT developers:

```
import com.google.gwt.core.ext.linker.*;
```

Linkers

When to build your own Linker

You would typically write a Linker to:

- Synthesize additional output-dependent files
  - Google Gears ManagedResourceStore manifests
- Integrate with environments beyond simple web pages
  - iGoogle or OpenSocial containers
- Provide an alternate bootstrap or packaging system
  - Gadgets and single-stage bootsrap
- Fulfilling a specific requirement for the structure of the output
- Generally solve "GWT and ..." integration

Linkers

Component parts

- Artifact
  - CompilationResult: Represents a unique chunk of JavaScript.
  - EmittedArtifact: Anything written into the output directory
    - GeneratedResource: Any file created by tryCreateResource()
    - PublicResource: Any file in the module's public path
  - ScriptReference and StylesheetReference: Represent tags in the .gwt.xml files
  - Custom subclasses can be used by Generators to communicate with the Linker stack via commitArtifact()
- Linker: Transforms one ArtifactSet into another, executed in a stack
- LinkerContext: Global data, such as module name and all SelectionProperties.

Linkers

Building a simple Linker

What's the simplest Linker that can be built?

```
<define-linker name="no_op"
 class="com.google.gwt.bobv.gio.linker.NoOpLinker" />
<add-linker name="no_op" />
```

```
@LinkerOrder(Order.PRE)
public class NoOpLinker extends AbstractLinker {
  @Override
  public String getDescription() {
    return "No-op Linker";
  }
  @Override
  public ArtifactSet link(TreeLogger logger, LinkerContext context,
      ArtifactSet artifacts) throws UnableToCompleteException {
    return artifacts;
  }
}
```

Linkers

Going offline with Google Gears

The GALGWT project has Offline.gwt.xml; how does it work?

```
<define-linker name="gearsManifest"
    class="com.google.gwt.gears.offline.linker.GearsManifestLinker" />
<add-linker name="gearsManifest" />
```

```
@LinkerOrder(Order.POST) public final class
GearsManifestLinker extends AbstractLinker {
```

```
public ArtifactSet link(TreeLogger logger, LinkerContext context,
    ArtifactSet artifacts) throws UnableToCompleteException {
  ArtifactSet toReturn = new ArtifactSet(artifacts);
  SortedSet<EmittedArtifact> emitted = toReturn.find(EmittedArtifact.class)
  toReturn.add(emitManifest(logger, context, userManifest, emitted));
  return toReturn;
}
```

```
return emitBytes(logger, Util.getBytes(out.toString()),
    context.getModuleName() + ".nocache.manifest");
```

Linkers

Execution order

- Linkers are ordered into Pre-, Primary-, and Post-Linkers
- Exactly one Primary Linker, which is generally responsible for module bootstrap
- Pre-Linkers are executed in lexicographical order before the Primary
- Post-Linkers are executed in reverse lexicographical order after the Primary
- The most-inherited Pre-Linker runs first and the most-inherited Post-Linker runs last

```
<add-linker name="pre_1" />
<add-linker name="post_4" />
<inherits name="between" />
<add-linker name="pre_2" />
<add-linker name="post_3" />
```

- The exact ordering is controlled by the relative positions of <add-linker> tags verses <inherits> tags

Linkers

Reducing round-trips

How can we improve serving efficiency by inlining the selection script into the host html page?

```
public ArtifactSet link(TreeLogger logger, LinkerContext context,
    ArtifactSet artifacts) throws UnableToCompleteException {
  artifacts = new ArtifactSet(artifacts);
  Map<String, EmittedArtifact> scripts = getScripts(artifacts);
  artifacts.removeAll(scripts.values());
  for (EmittedArtifact e : artifacts.find(EmittedArtifact.class)) {
    if (e.getPartialPath().toLowerCase().endsWith(".html")) {
      EmittedArtifact newArtifact = replaceScripts(logger, e, scripts);
      artifacts.replace(newArtifact);
    }
  }
  return artifacts;
}
```

Linkers

Reducing round-trips

How can we improve serving efficiency by inlining the selection script into the host html page?

```
private EmittedArtifact replaceScripts(TreeLogger logger, EmittedArtifact e
    Map<String, EmittedArtifact> artifacts) throws UnableToCompleteExceptio
  String pageContents = Util.readStreamAsString(e.getContents(logger));
  // ... replacements happen ...
  EmittedArtifact toReturn = emitString(logger, pageContents,
      e.getPartialPath());
  return toReturn;
}
```

Linkers

Server-side selection

How can permutation selection be moved to the server?

- A Linker exports the permutation properties for each CompilationResult

```
for (CompilationResult r : artifacts.find(CompilationResult.class)) {
  EmittedArtifact out = emitWithStrongName(logger,
      Util.getBytes(r.getJavaScript()), "rawJs/", ".cache.js");
  artifacts.add(out);
  for (Map<SelectionProperty, String> map : r.getPropertyMap()) {
    mapContents.append(out.getPartialPath() + " = {");
    for (Map.Entry<SelectionProperty, String> e : map.entrySet()) {
      mapContents.append("\n " + e.getKey().getName() + ":" + e.getValue
    }
    mapContents.append("\n}\n");
```

```
EmittedArtifact exportArtifact = emitString(logger, contents,
    "selection.map");
artifacts.add(exportArtifact);
```

- Requires server-side infrastructure to make permutation decision

Generators + Linkers

Integrating with a Gadgets container

How does the GWT4Gadgets stack work?

- Combination Generator + Linker
- The Generator extracts information from the type-system
- The Linker is responsible for final assembly of the Gadget manifest
- Gadgets have a much richer runtime environment to take advantage of, but a very different deployment and bootstrap model
  - Different Gadgets use different features
  - The Gadget must be described by a manifest file
- Use a "Generator sandwich" with dependency-injection:

Generated subclass with feature injectors

Developer's Gadget subclass

Gadget base class

Generators + Linkers

Integrating with a Gadgets container

How can we derive the contents of the manifest from the type-system?

```
@ModulePrefs(title = "__UP_title__", author = "BobV")
public class HelloGadgets extends Gadget<HelloPreferences>
    implements NeedsIntrinsics, NeedsSetPrefs, NeedsSetTitle {
    ... ...
  public void initializeFeature(SetTitleFeature feature) {
    this.setTitle = feature;
  }
    ... ...
  protected void init(final HelloPreferences prefs) {
```

```
public interface HelloPreferences extends UserPreferences {
  @PreferenceAttributes(display_name = "Alert prompt",
      default_value = "Hello, Gadgets!", options = Options.REQUIRED)
  StringPreference promptSomethingElse();
}
```

The Generator creates a stub manifest.xml file, into which the Linker injects the bootstrap script.

Generators + Linkers

Integrating with a Gadgets container

The GadgetLinker is simply a specialization of the built-in cross-site Linker.

```
public final class GadgetLinker extends XSLinker {
    ... ...
  @Override
  protected EmittedArtifact emitSelectionScript(TreeLogger logger,
      LinkerContext context, ArtifactSet artifacts)
      throws UnableToCompleteException {
    ... ...
  @Override
  protected String generateScriptInjector(String scriptUrl) {
    ... ...
  @Override
  protected String generateStylesheetInjector(String stylesheetUrl) {
```

Additional reading

- GWT4Gadgets: gwt-google-apis Google Code project
- ImmutableResourceBundle: GWT Incubator project
- Linkers: In the core GWT project

Questions / Comments?

- "Google Web Toolkit" Google Group
- irc://irc.freenode.net#gwt