Overview

• Background
  ▪ AJAX
  ▪ Google Web Toolkit (GWT)
  ▪ App Engine for Java

• Plugin Design Principles

• Challenges, Solutions, and Lessons
Background

GPE, AJAX, GWT, App Engine
What is the Google Plugin for Eclipse?

- Collection of plugins
- Assists in the creation of Web Apps that use:
  - Google Web Toolkit
  - App Engine for Java
- And by Web App I mean... AJAX
  - Gmail
  - Google Maps
  - Wave
AJAX - The Good

- Update the browser UI without switching pages
  - Relies on JavaScript running in the browser to direct the UI updates
- Fetch data in the background using XHR
- Viewing browsers as smart clients
  - Improves server utilization
  - Applications are more responsive than classic HTML
AJAX - The Bad & Ugly

• But...
  ▪ Dynamically-typed language
    • Runtime-only bugs (e.g. spelling bugs)
      ▪ Did you mean component or compnent -- I'll assume that they are different!
    • Results in limited tool support for JavaScript
  ▪ Browsers are a moving target
    • What bugs does this browser have?
    • How am I supposed to make X happen on this browser?
  ▪ Hard for large teams to work on the same code base
  ▪ AJAX expertise is a limited resource
GWT - The Motivation

• What if we could leverage what developers already know?
• What if we could deal with errors at compile time (static typing)?
• What if we could get better tool support?
  ▪ Breakpoints
  ▪ Variable inspection
  ▪ Auto-completion
  ▪ Refactoring
GWT - The Solution? Use Java.

- Lots of Java developers
- Great tools support: Eclipse, IntelliJ, etc.
  - Auto-completion
  - Refactoring
- Debug your web app code as bytecode in a special browser
  - Breakpoints
  - Variable inspection
- Cross-compile into stand-alone, optimized JavaScript
- No browser plugins / no obligatory server-side machinery
- Developers can still get to raw JavaScript via JavaScript Native Interface (JSNI)
GWT - Putting it all together

• Additional leverage allows you to be more aggressive about the problems you tackle
• Productivity for you, the developer
• Performance for your users
App Engine - What about the server?

- App Engine is a cloud computing platform
- Run your web apps on Google’s infrastructure
- We provide the container and services (Platform-as-a-Service)
  - Hardware connectivity
  - JVM
  - Servlet Container
  - Software services
App Engine - Key Features

• No need to install or maintain your own infrastructure
• We do the scaling for you
• Use Google’s scalable services via standard APIs
• Charge for actual usage
  ▪ Free to get started
• Built-in application management
App Engine - Java Support

• Servlet 2.5 container support
• Software services
  ▪ Authentication
  ▪ Datastore
  ▪ Caching
  ▪ E-mail
  ▪ Fetch URLs
• Sandboxing
  ▪ Restrict JVM Permissions
  ▪ Whitelist JRE classes
• DevAppServer
  ▪ Emulates the production environment
  ▪ Local implementation of software services
• Deployment
  ▪ App lives at <app_id>.appspot.com or custom domain with Google Apps
Plugin Design Principles
Design Principles

- Stability is paramount
- Make it easy to get started
- Reward sophisticated developers
- Control is happiness
- Keep things simple
- Blend naturally into Eclipse
- Developer’s time is valuable - help them maximize it
- Minimize plugin magic
Challenges, Solutions, and Lessons
Installation

Challenge
• Don’t make the user download all of the pieces individually
• Installation is ready for use once the install completes
• People behind firewalls

Solution
• Bundle the App Engine and GWT SDKs as plugins
• Produce stand-alone archives for people behind firewalls

Lessons
• Optional features?
• Doh!... P2 Garbage Collection
• There are a lot of people behind firewalls!
New Web Application

Challenge
• Quickly create web apps that use GWT and/or App Engine
• Use an expanded WAR layout

Solution
• Create a wizard that generates web apps that are ready-for-launch
• Allow users to select which versions of GWT/App Engine to use
• Use project natures to indicate what is being used, allow users to add GWT/App Engine after the fact
• Manage SDK jars in the WEB-INF/lib folder

Lessons
• Made it really, really easy to get started, but can’t:
  ▪ Create empty projects
  ▪ Import SDK samples
Run/Debug Web Apps

Challenge

• Create a simple, customizable launch configuration
• Handle possible combinations of GWT & App Engine
• App Engine and GWT use different development servers

Solution

• Extend Java launch configurations and add some guiding UI
• Classpath provider to add source paths needed by GWT
• RuntimeClasspathEntryResolver to deal with OOPHM

Lessons

• Magic values should expand into the program and VM args for added transparency
• Classpath modifications and reset are imperative
SDKs

Challenge
- Plugin should support multiple versions of the App Engine & GWT SDKs
- Make it easy to switch versions
- Properly configure complex classpaths

Solution
- Classpath containers, e.g. JRE containers
- Classpath dialog and project properties enable trivial SDK switching

Lessons
- Be careful what you do inside of ClasspathContainerInitializers
- Multiple ways for containers to be edited
- People ignore warnings -- Problems View
- Default SDKs might not be a great idea
GWT JSNI: Embed JavaScript in Java Files

Challenge

• Embed JavaScript code in Java files
• GWT overloads native methods and Java block comments for JSNI
• Unfortunately, JSNI delimiters /*-{ }-*/ make JSNI blocks into multiline comments so they get auto-formatted

Solution

• Declare a new document partition for JSNI methods
• Color JSNI method bodies as JavaScript
• Fix up “formatting” performed by auto-format

Lessons

• Smokescreen pattern
• Redoing the formatting works but alters undo behaviors
GWT JSNI: Refactoring, Search, and Completion

Challenge
- JSNI uses JNI-like signatures that are subject to typos and can be invalidated by refactoring
- Invisible to Java dependency and Java Search

Solution
- Add completion proposals and quick assist to expand JSNI refs
- Add validation to check the validity of JSNI refs
- Participate in Java refactoring to update JSNI refs
- Participate in Java searches to include refs from JSNI

Lessons
- Indexing
- Java refactoring is not as friendly as we’d like
GWT RPC

Challenge

• A remote service is a pair of interfaces
  ▪ Synchronous interface for use in the server
  ▪ Asynchronous interface for use in the browser

Solution

• JavaCompilationParticipant checks interface pair consistency
• Associate problems and quickfixes bidirectionally
• Java refactoring participant

Lessons

• Problem/Quickfix separation -- pure but impractical
• Let a user address sync problems from either side
• Hooking into Java refactoring is ... tricky
App Engine JRE Whitelist

Challenge

• Inform developers when they are using unsupported JRE APIs

Solution

• Compilation participant
• Load whitelist out of the SDK in the project’s classpath
• Provide quick fixes to flag the containing class as not being used in server code

Lessons

• Caching whitelist can be tricky because the underlying SDK can be changed
• Versioning issues when modifying IDE behavior based on project classpath
App Engine ORM

Challenge

- App Engine DataNucleus-based ORM uses bytecode enhancement
- Flag enhancement problems before the user runs the app

Solution

- Drive the DataNucleus enhancer as part of the build
- Provide ability to select what classes should be enhanced

Lessons

- DataNucleus wants to report output to standard out
- Feedback via the console is less than optimal
- Want feedback as red squiggly in file
- Classfile changes are what really matter
- Do not let exceptions escape from your builders!
App Engine - Deploy to Google

Challenge
• Create a simple mechanism for deploying web app to Google
• Deal with GWT compilations if applicable

Solution
• Add an action which packages the app and uploads it to Google
• Asynchronous status updates

Lessons
• Someone always want to tweak the process
  ▪ Disable GWT-compile
  ▪ Remember my password
Bleeding Edge - GWT ClientBundles

Challenge

- Expose GWT’s ability to bundle resources (text, images, css) into compiled JavaScript
- Not readily discoverable; very flexible system

Solution

- Create a wizard that creates ClientBundles based on a set of resources
- Add validation of ClientBundles to guide developers during post-creation edits

Lessons (Still learning)

- How discoverable is drag-and-drop?
Bleeding Edge - GWT UiBinder

Challenge

• Expose GWT’s ability to create UIs declaratively via special *.ui.xml files bound to Java classes!

Solution

• Create completion proposals and quick assists to aid in the creation and maintenance of ui.xml files

• Add validation of ui.xml files to inform users when something is off

Lessons (Still learning)

• Content describers don’t respect compound extensions (ui.xml)
Bleeding Edge - Web App View

Challenge

• GWT development server (OOPHM): how do you expose the hierarchical logs for browser and server code
• Can you help the user do something useful with the logs?

Solution

• Create an Eclipse view, a model, and a protocol to facilitate log surfacing
• Make the logs searchable and integrate client and server logs
• Update the model based on launch configuration life-cycle and development server events

Lessons (Still learning)

• Protocol buffers are a great way to deal with version skew
• Learn about Viewers, ContentProviders, and LabelProviders; they are worth the investment
Google Plugin for Eclipse

Documentation
- http://code.google.com/eclipse

Update sites
- Eclipse 3.5 (Galileo) - http://dl.google.com/eclipse/plugin/3.5
- Eclipse 3.4 (Ganymede) - http://dl.google.com/eclipse/plugin/3.4
- Eclipse 3.3 (Europa) - http://dl.google.com/eclipse/plugin/3.3

App Engine Feedback
- Group - http://groups.google.com/group/google-appengine-java

Google Web Toolkit Feedback
- Issue Tracker - http://code.google.com/p/google-web-toolkit/issues
Thank You!

Q&A