Google Web Tools - Starting Point

GWT and Wicket

Introduction

Wicket and GWT are both component-oriented web application frameworks, but they take opposite approaches to state management. Wicket pages and sessions are composed and maintained entirely on the server, and a round trip takes place when reflecting any significant changes to them in the browser. GWT pages, on the other hand, are composed of JavaScript widgets running on the client-side, where state is maintained and ongoing server communication occurs against a flat RPC layer.

Motivation

Why would you *want* to include GWT widgets as components in your Wicket pages?

Language Consolidation

A compelling aspect of Wicket its lack of any JSP expression language equivalent. Page logic is instead implemented in Java code, allowing full use of standard development tools. GWT offers the same advantages with its Java -> JavaScript compiler. All widgets are implemented, tested, and debugged in Java code before being compiled to JavaScript for deployment. Marrying GWT and Wicket would further language consolidation by removing hand-authored JavaScript from the picture without sacrificing intensive client-side behavior.

Client-side State *Can* be Nice

Server round-trips can be heavy for implementing some types of client-side behavior. For example, what if you're tracking the coordinates of a mouse-over on an image? Doing a round-trip to check the mouse cursor location against some state on the server would be prohibitive. What if you could simply ship the Java code for those checks and the associated state to the client?

Starting Point for Integration

Here is a partial design for wrapping GWT widgets with Wicket components.

Class Diagram

This class diagram captures the design of a potential integration between Wicket and GWT.

blocked URL

Discussion

Create a central abstract class to represent your GWT widget as a Wicket component: GwtWicketComponent. This class implements a Wicket listener for RPC calls from the widget running on the client, and subtypes of GwtWidgetComponent like FooComponent provide and implementation of RemoteServiceServlet for fielding the calls.

GwtWidgetComponent extends Panel, and should have some associated markup that includes the container for your GWT widget. GwtWidgetComponent will automatically add an appropriate callback URI to an attribute of this container, which the client-side code must use to invoke RPC calls.

A special GwtRpcTarget implementation of IRequestTarget transports the RPC response payload back to the client via Wicket's standard RequestCycle.

FooComponent and FooService implement an example GWT-widget-as-wicket-component.

Obstacles

I concluded that the work of massaging GWT into some kind of cohesive development environment with Wicket outweighed the benefit to my project at the time I write this. Getting the GWT development shell to render pages out of their servlet which were also passed through the Wicket servlet filter was trivial, but that's where I stopped. Rectifying the URI schemes for the two frameworks to get them to cooperatively render a page was the next step. A specialized Wicket URICodingStrategy might help here.

Other improvements may come in the form of an incremental Eclipse builder for GWT, or other niftyness spawning from the Googleplex.