Client Development Setup

**TODO: project still called wave-protocol?**

### Note for Mac

Ensure your jdk is set to 1.7 and you're running Safari 5+

---

### Before you start

For developing and debugging code in an isolated wave panel environment, use the Wave Panel Harness setup. **This is the preferred development environment**, because there is no need to run a wave server.

For developing and debugging code in the full client with end-to-end server communication, use the Full Client setup.

### Setting up Eclipse project (optional)

Eclipse is the recommended development environment, in order to maintain consistent code formatting. However, since the debugging processes are all contained in the ant build file, you can use whichever development environment you like.

1. Follow [WIAB build instructions](#)
2. Generate some code required for the Eclipse project to compile.
   - `ant eclipse`
3. In Eclipse, Import the wave-protocol project into your workspace.
   - File -> Import -> Existing Projects Into Workspace
   - Navigate to the wave-protocol project
   - The 'wave-protocol' project should then appear and be selected in the Projects: list
   - Click Finish.

### Wave Panel Harness

#### GWT Hosted mode

1. Run the wave panel harness using the GWT code server (called OOPHM).
   - `$ ant waveharness-hosted`
2. Using a supported browser*, preferably running on the same machine as the GWT code server, open the URL that the hosted mode server names. The wave panel harness is a page with a wave panel displaying a fake, in-memory wave. There is no communication with a wave server.
3. Attach a debugger to the hosted mode server.
   - In Eclipse, create a launch configuration using Run -> Debug Configuration -> Remote Java Application, and use port 8001. Click Debug.
4. Profit.
   - Set breakpoints, interact in the browser, step through code etc.
   - In Eclipse, code changes you make do not always take effect in the browser immediately (HotCode). However, refreshing the browser will reload the application through the GWT code server, including all your changes.

#### GWT Superdev mode

1. Run the wave panel using the GWT superdev mode.
   - `$ ant waveharness-superdev`
4. Do some code changes (for instance in UndercurrentHarness.java), click in "Dev mode on" in the bookmarks and recompile the module to see the changes.

#### Superdev debugging

In Google Chrome, for debugging and logs:

- Launch Chrome Developer Tools (F12)
- For debugging, you need to Enable JS Source Maps in Chrome Developer Tools preferences.
- Open the Source tab and observe the files, or the Console tab to see the logs.


### Editor Test Harness

#### GWT Hosted mode

1. Run the wave panel harness using the GWT code server (called OOPHM).
1. $ ant editor-hosted
2. Using a supported browser*, preferably running on the same machine as the GWT code server, open the URL that the hosted mode server names. The editor harness is a page with an div for the local editor and another div for the remote editor. There is no communication to the server.
3. Attach a debugger to the hosted mode server.
4. Profit.
   a. Set breakpoints, interact in the browser, step through code etc.
   b. In Eclipse, code changes you make do not always take effect in the browser immediately (HotCode). However, refreshing the browser will reload the application through the GWT code server, including all your changes.

GWT Superdev mode

1. Run the wave panel using the GWT superdev mode
   * $ ant editor-superdev
2. Open and add the bookmarks in: http://localhost:9876/ to your browser.
3. Open the wave panel in http://localhost:9876/org.waveprotocol.wave.client.editor.harness.EditorTest/EditorTest.html
4. Do some code changes, click in "Dev mode on" in the bookmarks and recompile the module to see the changes. See waveharness superdev mode section for debugging tips.

Full Client

GWT Hosted mode

1. Start the server following the instructions at Starting WIAB
   a. Remember to restart the server after any change that would affect the server. In general, this is required for any change outside the box
   /webclient package.
2. Start the GWT code server (called OOPHM)
   * $ ant hosted-gwt
3. Set up an Eclipse debug target to attach to the GWT code server (this only needs to be done once, after that reuse the same target to start debugging)
   a. Run -> Debug Configuration
   b. Select Remote Java Application
   c. Change the port from 8000 to 8001 (this port number should match the console output when 'ant hosted_gwt' was run)
   d. Click Debug
4. Using a supported browser*, preferably running on the same machine as the GWT code server,
   a. Go to http://<your_host_name>:9898
      * Sign in. This only needs to be done once per server restart, in order to get a session cookie. Ignore any popups about the client needing to be GWT compiled.
   b. Go to http://<your_host_name>:9898/?gwt.codesvr=<your_host_name>:9997
      * In order to communicate with the GWT code server, your browser needs a plugin. If you do not already have it installed, you will be directed to install it.
5. You should now be able to make any change to the client code in wave-protocol and see the effects by refreshing the browser
   * You may need to restart "ant hosted_gwt" periodically once the GWT code server runs out of memory (after ~10-20 refreshes).

GWT Superdev mode

In development.

Supported Browsers

The GWT code server requires a plugin to be installed in the browser that you use. The first time you use the GWT code server in a particular browser, you will be prompted to install this plugin.

Not all browser/OS combinations that Wave-In-A-Box supports can be used for debugging; this is limited by the browser/OS combinations for which a GWT code server plugin is available. At the time or writing, this set is: Linux x (Firefox), Windows x (Chrome, Firefox, IE), Mac x (Safari, Firefox).

For the wave panel harness, the full set of combinations is available.

For the full client, the only debugging setups that work out of the box are Safari on Mac, and Chrome on Windows (i.e., webkit browsers).