

GWT

Google Web Toolkit

For usage and examples, see [GWT RPC Examples](#)

Setup

1. Add the jetty-gwt.jar to your lib/dependencies (if using jetty7, its available in 7.0.0pre3 and up)
2. Edit your class to extend AsyncRemoteServiceServlet
3. Clear your gwt-cache(only the first time)
4. Recompile your gwt-project

Explanation

The Google Web Toolkit allows Ajax applications to be developed in java code using the traditional UI widget paradigm. The toolkit includes support for RPC, but not for comet style Ajax push.

Unfortunately GWT has not made it easy to use continuations within their RPC mechanism. Firstly they catch Throwable, so the Jetty RetryException is caught. Secondly they have made most of the methods on the GWT servlet final, so you cannot fix this by extension.

GWT 1.5.x and below

Luckily GWT is open sourced under the apache 2.0 license, so it was possible to do a cut/paste/edit job to fix this. The [OpenRemoteServiceServlet](#) recently added to Jetty is a version of GWTs RemoteServiceServlet without the final methods and a protected method for extending exception handling.

Once the GWT remote service servlet has been opened up, it is trivial to extend it to support Continuations, which has been done in [AsyncRemoteServiceServlet](#).

Because GWT RPC uses POSTS, the body of the request is consumed when the request is first handled and is not available when the request is retried. To handle this, the parsed contents of the POST are stored as a request attribute so they are available to retried requests without reparsing:

```

protected String
readPayloadAsUtf8(HttpServletRequest
request)
    throws IOException, ServletException
{
    String
payload=(String)request.getAttribute(
PAYLOAD );
    if ( payload==null )
    {
        payload=super.readPayloadAsUtf8(
request );
        request.setAttribute(
PAYLOAD,payload );
    }
    return payload;
}

```

GWT 1.6.x+

The [OpenRemoteServiceServlet](#) is no longer needed. The RemoteServiceServlet has been updated and the limitations mentioned above are no more. You can either patch a copy of [AsyncRemoteServiceServlet](#) in your project, until a new jetty-gwt.jar release, or override the readContent(HttpServletRequest request) and doUnexpectedFailure(Throwable caught) in your subclass. Here is the override for the replaced/renamed readPayloadAsUtf8 (GWT 1.6.x), which is mentioned above.

```

protected String
readContent(HttpServletRequest request)
    throws IOException, ServletException
{
    String
payload=(String)request.getAttribute(
PAYLOAD );
    if ( payload==null )
    {
        payload=super.readContent( request
);
        request.setAttribute(
PAYLOAD,payload );
    }
    return payload;
}

```

The exception handling is also extended to allow the continuation `RetryException` to propagate to the container. This has been done without any hard dependencies on Jetty code. With these extensions, the `AsyncRemoteServiceServlet` allows any GWT RCP method to use continuations to suspend/resume processing. All you need to do is to have your servlet implementation extend the `org.mortbay.gwt.AsyncRemoteServiceServlet`:

```

package com.acme.example;

import
org.mortbay.gwt.AsyncRemoteServiceServlet;
import org.mortbay.util.ajax.Continuation;
import
org.mortbay.util.ajax.ContinuationSupport;

public class MyServiceImpl extends
AsyncRemoteServiceServlet implements
MyService
{
    //MyService method implementations go
here and can use Continuations
}

```

For example below is a Table class from <http://www.gpokr.com> where a continuation is used by the service implementation to wait for an event to be available for a player:

```

class Table
{
    Set waiters = new HashSet();

    public Events getEvents( Context c )
    {

        Player p = getPlayer( c );
        // the player has no events.
        if( p.events.size() == 0 )
        {

```

```

        synchronized( this )
        {
            // suspend the continuation
waiting for events
            Continuation continuation =
ContinuationSupport.getContinuation
                ( c.getRequest(),this );
            waiters.add( continuation );
            continuation.suspend( 30000
);
        }
    }

    return p.events;
}

protected void addEvent( Event e )
{
    // give the event to all players
    Iterator it =
players.values().iterator();
    while( it.hasNext() )
    {
        Player p = (Player)it.next();
        player.events.add( event );
    }

    // resume continuations waiting for
events
    synchronized( this )

```

```
    {  
        Iterator iter =  
waiters.iterator();  
        while ( iter.hasNext() )  
            ( (Continuation)iter.next()  
).resume();  
        waiters.clear();  
    }
```

```
}  
}  
}
```

GWT 1.5.x and below

You will need to build and then include the `jetty-gwt-<version>.jar` in your webapp. The source is included with every [download](#) of jetty. To build:

```
> cd $jetty.home/extras/gwt  
> mvn install
```

Note that `OpenRemoteServiceServlet` was created due to the fact that it was not possible to subclass google's `RemoteServiceServlet` (unless we use reflections)

Details below:

```
@Override  
public final void  
doPost(HttpServletRequest request,  
        HttpServletResponse response) {  
    try {  
  
        perThreadRequest.set(request);  
        perThreadResponse.set(response);  
  
        // This needs to be delegated to a  
protected method  
        String requestPayload =  
RPCServletUtils.readContentAsUtf8(request);  
  
onBeforeRequestDeserialized(requestPayload);
```

```
        String responsePayload =  
processCall(requestPayload);  
  
onAfterResponseSerialized(responsePayload);  
  
        writeResponse(request, response,  
responsePayload);  
        return;  
    } catch (Throwable e) {  
  
        doUnexpectedFailure(e);  
    } finally {  
  
        perThreadRequest.set(null);  
    }  
}
```

```
        perThreadResponse.set(null);
    }
}
```

```
// Accessor needs to be protected
private final
ThreadLocal<HttpServletRequest>
perThreadRequest = new
ThreadLocal<HttpServletRequest>();

private final
ThreadLocal<HttpServletResponse>
perThreadResponse = new
ThreadLocal<HttpServletResponse>();
```

GWT 1.6.x+

Just extend RemoteServiceServlet and override readContent(HttpServletRequest request), shown above, and doUnexpectedFailure(Throwable caught).

```
private static final String
JETTY_RETRY_REQUEST_EXCEPTION =
"org.mortbay.jetty.RetryRequest";

/**
 * Overridden to really throw Jetty
 * RetryRequest Exception (as opposed to
 * sending failure to client).
 *
 * @param caught the exception
 */
```

```
protected void doUnexpectedFailure(Throwable
caught)
{
    throwIfRetryRequest(caught);
    super.doUnexpectedFailure(caught);
}

/**
 * Throws the Jetty RetryRequest if found.
 *
 * @param caught the exception
 */
protected void throwIfRetryRequest(Throwable
caught)
{
    if (caught instanceof
UnexpectedException)
    {
        caught = caught.getCause();
    }

    if
(JETTY_RETRY_REQUEST_EXCEPTION.equals(caught
.getClass().getName()))
    {
```

```
        throw (RuntimeException) caught;
    }
}
```