GWT IN THE REAL WORLD

BUILDING ENTERPRISE GWT APPLICATIONS IN LEGACY APPLICATION FRAMEWORKS

Douglas Bullard

THE APPLICATION

- Existing Nike.net OrderCapture application
    - Struts 1.3
    - JDK 1.4
    - ATG (not a J2EE application)
    - Running under Vignette Portal
        - Forces browser quirks mode
    - Order entry data grid written in JavaScript
        - Based on Nitobi’s grid

PROBLEMS WITH EXISTING APP

- Grid is broken in all browsers but IE
- Slow and buggy even in IE
- Business desired new features and enhancements
- Nitobi is defunct

THE TEAM

- 3 Full time developers
  - 1 had prior GWT experience
  - 1 had light GWT and medium Swing experience
  - 1 had neither Swing nor GWT experience
- 1 tech lead
  - 20% time coding

FIX OR RE-ENGINEER?

- Existing grid was undocumented spaghetti code
  - Several JS experts spent months writing it
  - We're just not that smart
- Business didn't want it rewritten, just fixed
- If we re-engineered, we wanted GWT
  - Allows us to use Java, which we're very good at
  - Allows us to use IDEs with full debuggers
  - Encapsulates browser variability
  - Good widget sets available

REQUIREMENTS

- Has to work across all modern browsers
  - We don’t want to have to rewrite it (much) for iPad, etc.
- IE compatibility back to IE 7 (6 would be nice)
- Order size quantities need to be editable, but nothing else
- Must be fast - 10-key entry speed needed
- Must be able to filter visible entries by multiple criteria
- Keep running retail/wholesale totals per line and for entire cart
  - Japan and other countries can’t see suggested retail prices
- Search within grid and across entire cart
- Cut/paste from within grid
- Cut/paste from spreadsheet
- Freeze panes would be nice
- Currency symbols instead of ISO codes (€ instead of EUR)

WHICH GWT FRAMEWORK?

- Stock GWT
  - Simple
  - Stock GWT tables don’t offer functionality needed
- SmartGWT
  - Fast
  - Tables can have freeze panes
  - Data proxies for live scrolling
  - Extensive framework
  - GWT-wrapped JavaScript

WHICH GWT FRAMEWORK?

- Sencha Ext GWT (GXT)
  - Fast
  - No freeze panes
  - Data proxies for live scrolling
  - Extensive framework
  - Pure Java
- We went with Sencha Ext GWT

BEGINNINGS

- Business just wanted a drop-in replacement
- We wanted a review of the UI
  - Was the "online Excel" paradigm what we needed?
  - GWT consultant didn't like it
  - Internal UI expert review didn't like it
  - RG/A reviewed our needs and suggested new design
- Results - everyone thought the "online Excel" paradigm was a bad idea
  - Large volume data entry should be done via offline spreadsheets and uploaded
- This did not sit well with the business
  - Desire to have all data entry online
  - They really liked the "online Excel" paradigm
  - Users know the UI, don't want to retrain
- Do it anyway

INITIAL DESIGN

- Prototype begin using Google's MVP
- GWT consultant didn't like that
  - M-V-VM (Model-View-ViewModel)
    - More complicated than MVP, but does the same thing
      - View
        - Separation of the view from everything else
        - Different views can be switched in for different devices
        - No state logic or behavior
        - UI behavior OK (size, position, etc)

INITIAL DESIGN

- View Model
  - Fully unit testable w/o needing GWT test framework
  - Careful use of mocks bypassed nasty GWT runtime errors (for example, resource bundle text)
  - All logic for UI goes here - anything that can be tested
  - UI view binds to this class
- Model
  - Base representation of data

INITIAL DESIGN

  - MVP is better
    - Simpler, more examples, less complexity
- We didn’t like the M-V-VM framework, wanted to use MVP
- “We bought it, we have to use it”

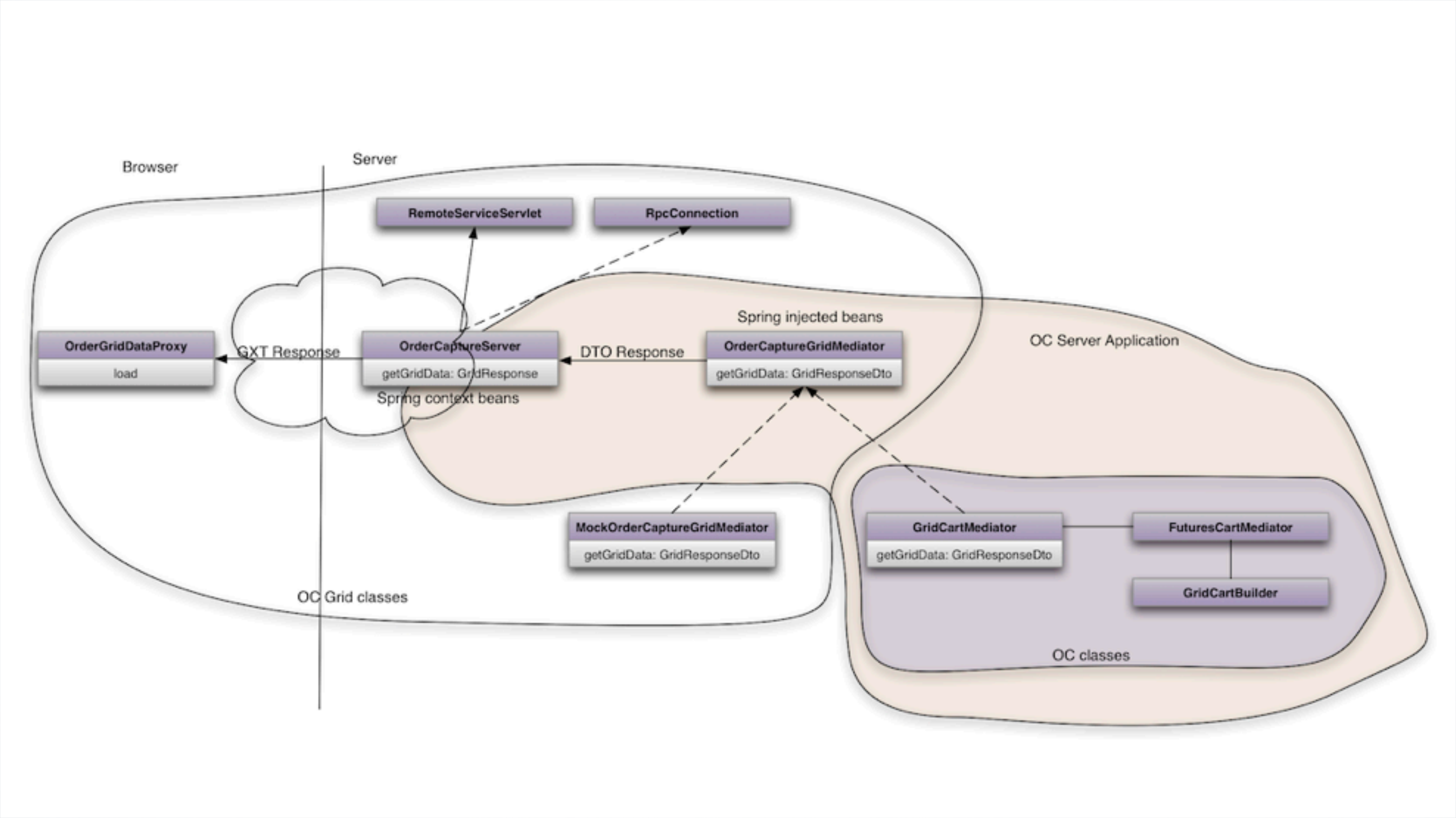
SENCHA

- Initially used Sencha data proxies and framework
- Very complicated design doing it this way
  - Makes your app a Sencha app
- Data transfer time is negligible in initial load
- Slow networks make streaming data problematic
- Better to do load all at once if data size reasonable
  - Give the user a progress bar and they'll tolerate a few seconds to load better than scrolling that freezes
  - Got buy-off from the business to limit number of rows fetched from the server (1,000)
  - We removed Sencha data proxies and went to straight GWT RPC calls
  - Save on command
  - Scrolling now very fast

ARCHITECTURE

- OrderCapture is a JDK 1.4 app
- Runs under ATG - non J2EE app server
- ShoppingCart is the main object with all items in cart
- GridCartMediator class interfaced between the RPC servlet and the ShoppingCart
    - Although servlet is called via RPC, it shared session with legacy code
    - No special code needed to retrieve cart for user
    - Only one class (+ servlet) needed!
- OrderCapture being migrated to JDK 1.6 & JBoss/J2EE
    - New code base
    - Just need to move our one class over - ShoppingCart API didn't change

GRID & MAIN APP CLASSES



DEPLOYMENT

- Completely separate project from OrderCapture
- Grid developed in JDK 1.6
- Deployment artifacts generated into Ivy repository
    - ocgrid.jar - DTOs, server-side classes and servlet needed for RPC
        - Byte code retroweaved for JDK 1.4
            - Sencha and GWT as well
    - ocgrid-gwt.zip - JS, images, etc - just gets expanded into OC WAR dir
    - Snapshots used for development
        - Used a slimmed down gwt.xml file for development
            - Too many language/browser combinations - 195!
            - Stick to just one for most dev

USEFUL TOOLS

- GWT Developer mode is great for rapid change
  - Change code and just refresh the browser
  - Eclipse & IDEA
- Mocked data
  - Serialize real data from shopping cart - save to grid project
  - Use real data in grid w/o having to run and deploy in app
- Jetty - lets us test "real" compiled GWT without the container
- gwt-log (http://code.google.com/p/gwt-log/)
  - Many types of log output
  - RemoteLogger
    - Sends log output to server
    - Wrap the whole UI to log uncaught exceptions
    - We found stuff going on we never knew about

EXPERIENCES

- No direct JavaScript coding was needed - everything was done in Java
- Some optimization of Sencha code for performant repainting required
- Didn't use GWT event bus
- Unit test coverage of 80% (non-UI)
- Minor cross-browser issues
  - Display tweaks
  - Paste from spreadsheet
    - Text after paste event has different line endings in IE
- Entire team loved working with GWT more than any other UI framework

LESSONS LEARNED

- Translations
  - GWT provides default values for translations
  - Annotations in interface
  - Great for development, resource bundles aren't required
  - As soon as resource bundles are ready, turn them off!
  - Compiler won't report resource bundle errors if defaults exist

LESSONS LEARNED

```
import com.google.gwt.i18n.client.Messages;

/** Interface for resource bundle properties. Note that we provide defaults until
we actually get the resource bundles done. */

public interface Translations extends Messages
{
  //@DefaultMessage("Total Qty")
  @DefaultMessage("Total Qty")
  String totalQty();

  //@DefaultMessage("Total Wholesale")
  @DefaultMessage("Total Wholesale")
  String totalWholesale();

  //@DefaultMessage("Wholesale")
  @DefaultMessage("Wholesale")
  @Key("wholesaleColon")
  String wholesale();

  //@DefaultMessage("Total Retail")
  @DefaultMessage("Total Retail")
  String totalRetail();

  //@DefaultMessage("Suggested Retail")
  @DefaultMessage("Suggested Retail")
  @Key("suggestedRetailColon")
  String suggestedRetail();

  //@DefaultMessage("Style Color")
  @DefaultMessage("Style Color")
  String styleColor();
```

LESSONS LEARNED

- Use two gwt.xml files
  - One for full deploy - all languages, all browsers
  - One for dev - only language/browser of interest
  - OC Grid takes 20 minutes to build on my Mac - 2 hours on our build boxes!

LESSONS LEARNED

- GIN/Guice
  - Not ready for prime time
  - Circular dependency injection is a real pain
  - Spring does this right
  - Use an object factory instead
- GWT RPC vs JSON
  - JSON may have been a better choice
    - Load testing of server difficult with our existing toolset
    - Would have been pretty easy to simulate users with JSON
- KISS - keep the design simple
- Push back on imposed designs
  - If it doesn’t smell right, it isn’t

REFERENCES

- GWT
  - http://code.google.com/webtoolkit/
  - MVP
    - http://code.google.com/webtoolkit/articles/mvp-architecture.html
- M-V-VM
  - http://msdn.microsoft.com/en-us/magazine/dd419663.aspx
- gwt-log
  - http://code.google.com/p/gwt-log/
- Sencha
  - http://www.sencha.com/products/extgwt/
- SmartGWT
  - http://www.smartclient.com/product/

DEMO