Grails In The Enterprise

Ryan Vanderwerf

Chief Architect @ ReachForce

www.reachforce.com

My Background

- Currently building a Grails and Cloud based infrastructure for ReachForce
- Architected and built a Grails solution for Developerprogram.com that allows rapid deployment of Developer Program portals for all kinds of companies, specializing in the mobile industry.
- Built Java and Linux based webcasting for events such as SXSW, built telecom software, and ASP's for the financial sector
- Worked with Java since 1996, and built server-side applications ever since
- Enticed into the Groovy and Grails space by speakers at the early NFJS conferences

What we will cover

- My stories of introducing Grails into 2 organizations
- Tips to help create a plan to start a project or introduct Grails in your company
- Tips to convince management or project managers
- Tips for selecting a candidate application
- Deploying a plugin to a maven repository
- A little tour of Jenkins
- Plugins to help keep things running smoothly
- Plugins to help you get started and get Grails 'in the door' such as using an old Struts 1.x application

How do I get Grails in the door?

- Looking to introduce Grails into your Company or Enterprise? Make sure you can answer these questions:
  - Do you have a plan? (Please work without one, there it will hurt)
  - How will you sell it to management? (What's the upside? Risk?)
  - How will you know it is effective? (Have a way to benchmark progress)
  - How will you keep things running smoothly? (Call on your trusty plugins!)
  - What is a good kind of low-risk project to migrate? (This varies, I'll show a struts example)

My Story With Grails

- Employer had very antiquated Struts/OJB application
- Because of technical debt work orders from customers were taking too long to do to be profitable
- Wanted a fresh start, but could not afford to re-write all old code
- Need a framework that allows for rapid development
- Need a solution that Java developers can quickly grasp and understand (and enjoy!)
- Can be deployed to existing infrastructure as a WAR file

In Comes Grails – What Will I Run Into?

- Barriers to entry from non-technical staff or managers:
  - Fear of change
  - Skepticism it can deliver
- A little different way of thinking to leverage Grails strengths like scaffolding
- Patience: People will resist it if it doesn't give immediate magical results, so don't promise them
- Difficult for project managers to understand defining the domain model up front saves time later (i.e. regenerating scaffolding repeatedly loses saved time)
- Difficult for project managers/managers to understand why putting work into scaffolding templates up front saves time and money going forward (new process)

Picking a legacy application

- Picking a good candidate application
  - Greenfield projects are always easier but can be risky if you are new to the framework – Start with some kind of internal tool
  - Old Struts 1.x projects can be blended easily without a lot of risk
  - Easy win because people will have low expectations on such an old application
  - Write unit tests for existing code in Groovy to 'wet' developer appetites (and practice)

Creating a plan

- Create a strategy of migrating your application into the Grails application file layout
- Plan out how you will integrate library dependencies (Maven, Ivy, etc?) If you had just a lib folder, you will need to work through dependencies by adding them to BuildConfig.groovy (or POM file on Grails 2.1)
- Plan and talk with your QA team – what are they looking for during testing? How are they going to test it? (Hint, often JSPs work on Tomcat during 'run-app' but break when deployed to containers like Weblogic, Websphere, Jboss, etc)
- Are you going to try to add functionality and migrate it to Grails (I wouldn't recommend it!) - make migration its own project/sprint before new project work begins

Selling to Management

- Gain development efficiencies, no constant server restarts during development
- Plugin ecosystem – won't have to re-invent the wheel so many great useful plugins to save time (and most of the source, you can contribute most of them to make them better)
- Easy for Java developers to understand
- Still deploys as a WAR file to the container, no significant infrastructure changes
- Can build a hybrid product to bridge the gap between new a old, giving a long term plan to modernize as you go

Items to put in place

- CI Server – Hudson/Jenkins most popular free choice, works great with Grails.
  - setup jobs on CI Server to run tests
  - trigger builds by watching SCM for changes
  - use coverage tools like Cobertura or Emma
  - use code analysis tools like FindBugs or CodeNarc
  - Make your CI server fun, little things like the chuck norris plugin or integration game keep developers entergized and interested (A little competition never hurts!)
  - Use a service like cloudbees if you can't run it yourself
- Some kind of deployment tool. Roll your own in Grails, or use artifact deployers in Hudson, or tools like Cargo, Chef, or Puppet to help you

Struts 1 Plugin

- Lets you merge Struts applications into a Grails application
- Often overlooked option that works quite well to get Grails 'in the door' to help aging applications
- You can even write Struts actions in Groovy
- Easy win to run old legacy code and new Grails functionality in parallel
- Version on Grails Portal currently only works with Grails 1.3.x. For Grails 2.x, go to https://github.com/rvanderwerf/grails-struts1

Struts 1 Plugin

- Handle exceptions well for legacy code. In Grails 1.3.x, exceptions on a 500 server error page in your legacy application will be wrapped in a Grails stack. This will frequently cause blame on Grails for legacy bugs and give opponents ammunition to not go forward with Grails. Stack traces in Grails 2 are much cleaner and also avoid this error
- If your target container is NOT tomcat, tests a war file on it to make sure your JSPs function BEFORE you let anyone see it.

Struts 1 Plugin

- If using the Grails 1.3.x, and you have file uploads, set the following in spring.groovy (See JIRA GP-STRUTS1-1):

```
multipartResolver(org.codehaus.grails.struts.StrutsAwareMultipartResolver) {
        strutsActionExtension = ".do"

}
```

- (If using Grails 2.x version from Github, the plugin does this for you)

Struts 1 Plugin

- If using the Grails 1.3.x, and you have file uploads, set the following in spring.groovy (See JIRA GP-STRUTS1-1):

```
multipartResolver(org.codehaus.grails.struts.StrutsAwareMultipartResolver) {
                    strutsActionExtension = ".do"

}
```

- (If using Grails 2.x version from Github, the plugin does this for you)
- If Using Grails 2, the ControllerActionProxy (TODO to fix this)

Struts 1 Plugin Demo

Continuous Integration Server

- Do you have a server to run it on?
  - If yes, you have lots of options:
    - Jenkins / Hudson (recommended for Grails)
    - Cruise Control
    - Continuum
  - If no, there are cloud options:
    - http://www.cloudbees.com/
    - Elastic Bamboo (Atlassian)
    - Run your own instance of Jenkins/Hudson on a EC2 or other cloud provider OS image (and maintain yourself)

Database Reverse Engineering

- 2 Common Options to generate domain model from Database
  - Grails Reverse Engineering Plugin (Only works properly on a separate 1.3.x project due to Hibernate version issues)
  - The GRails Application Generator (GRAG) Standalone application

- Mirror domain objects to legacy tables is key to implementing new features in Grails and leaving legacy code alone (If you won't have hbm files to import to Grails).
- When you have a domain object and a legacy bean make sure you handle cache consistency when writing objects.

Database Management

- Use Grails Database Migration Plugin (wraps Liquibase)
- Liquibase Directly
- Most other plugins like liquibase and autobase are deprecated in favor of the Database Migration Plugin
- Do NOT use the 'dbCreate' option in Grails for any kind of production system – it is not smart enough to handle field renaming of columns.

- If using the Grails Database Migration Plugin, use the changelog.groovy format and not the xml format, due to bugs in the functionality that handles xml changelogs in the plugin
- If you must use the xml format, use Liquibase directly
- Create separate project and install the migration plugin just for it (Seems to work better with a 1.3.x project due to Hibernate versioning issues)

Modularity

- Split up major functional areas of the legacy applications into separate plugins, but keep some things in mind:
  - JSPs do not serve well from plugins, start with just the java code, then work your way to converting JSPs to GSPs called from the plugins
  - Beware of cyclic dependencies, Grails does not tolerate them (Good design should avoid this, but sometimes it's hard to break of legacy spaghetti code)

Dependency Management

- When migrating legacy app to Grails format, and you use Ant and not Maven, follow these tips:
  - don't just copy jars into the grails/lib folder, set up dependencies in BuildConfig.groovy as much as possible
  - run dependency-report to help work out conflicts

- If your legacy project is Maven based, Grails 2.1 has excellent Maven support built in via 'grails create-pom' command (POM demo struts-demo21

Maven Project Management

- **Grails 2.1** required:
grails create-pom <company.group>
- Uses created pom.xml to build and manage depenencies instead of BuildConfig.groovy
- Advantage is easier for developers used to pure Maven build and depenency management instead of Ivy
- Very new so there may be dragons ahead, but so far works well
- Could work around some build/packaging bugs on Ivy (Concurrent building, etc)
- Available goals by default:
validate, initialize, generate-sources, process-sources, generate-resources, process-resources, compile, process-classes, generate-test-sources, process-test-sources, generate-test-resources, process-test-resources, test-compile, process-test-classes, test, prepare-package, package, pre-integration-test, integration-test, post-integration-test, verify, install, deploy, pre-clean, clean, post-clean, pre-site, site, post-site, site-deploy

Maven Project Management

- Multi-module support (From docs):
Create-multi-project-build

```
grails create-app myapp
grails create-plugin plugin-a
grails create-plugin plugin-b
grails create-multi-project-build
com.mycompany:parent:1.0-SNAPSHOT
mvn install
```

Maven Repositories

You will need a repository to service up your private dependencies whether it be jars or plugins. Some free options include:
- Aritfactory (my favorite, but hobbled with commercial options now)
- Nexus
- Archiva

Use release plugin to publish your plugins to your Maven repository

Release Plugin

- Used to push plugins to maven repository (or your own plugins to the public!)

- Add the following to your BuildConfig.groovy:

```
        grails.project.repos.default = "PluginSnapShots"
        grails.project.repos.PluginSnapShots.url =
"http://127.0.0.1:8081/artifactory/plugins-snapshot-local"
        grails.project.repos.PluginSnapShots.type = "maven"
        grails.project.repos.PluginSnapShots.username = "admin"
        grails.project.repos.PluginSnapShots.password = "password"
```

Localization Plugin

- Store your i18n message bundles in the database
- can Import legacy i18n property files from most systems
- provides caching
- allows changes to labels on the app without a new build
- lets non-technical users fill in missing labels for you(and a value add to sell when switching to Grails)
- Combine with the filterpane plugin to add searching ability

Filterpane Plugin

- Great for adding search ability for larger number of legacy reverse engineered domain objects (and rows within those)
- Simple to install and implement: add a new action to your controller and add some parameters to your list view page

Clustering

- Ehcache or new Spring Cache
- Use your servlet containers http session clustering or use Terracotta
- Use Terracotta open source edition for visibiity and cache management of level2 and general cache management

More Information

http://grails.org/plugin/filterpane
http://grails.org/plugin/localizations
http://grails-plugins.github.com/grails-database-migration/
http://grails.org/plugin/struts1
https://github.com/rvanderwerf/grails-struts1
http://terracotta.org/downloads/open-source/catalog
http://grails.org/plugin/release
http://grails.org/plugin/db-reverse-engineer
http://grag.sourceforge.net/

Contact Me

Via twitter: https://twitter.com/RyanVanderwerf
Google+/email: rvanderwerf@gmail.com
Blog: http://rvanderwerf.blogspot.com