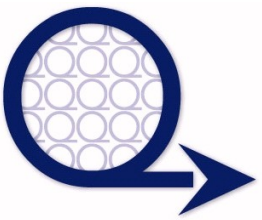


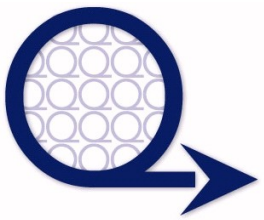
Web Development Framework Comparisons

Scott Fradkin
QWANTify



Many other web development frameworks are available

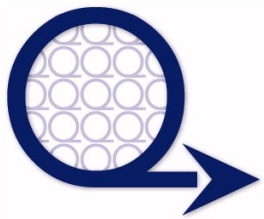
Show development of a modern Java J2EE application and compare it to a couple other frameworks



Which other frameworks?

Ruby on Rails

Grails (Groovy and Java)



Demo Application

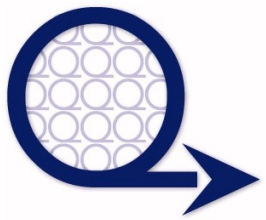
www.qwantify.com

Widget Store

Store page with cart and user account

Set of Administration pages to administer users, store items, and orders

Meant to be non-trivial, yet not too complex

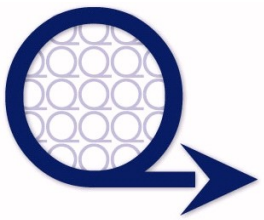


Demo Application

www.qwantify.com

Database schema is not very complex

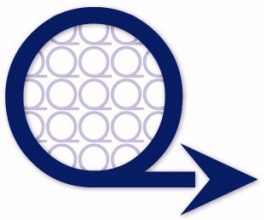
Only 6 tables



Java and J2EE

www.qwantify.com

- Environment
 - Java JDK 1.5
 - Eclipse 3.2 with the most recent JST and WST plugins
 - Tomcat 5.5
 - MySQL 5 (Also need MySQL Connector/J)
 - Struts 1.2.9
 - Spring 2.0
 - Hibernate 3.2 with Annotations 3.2.0



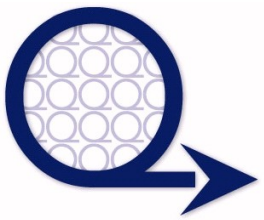
Java and J2EE

www.qwantify.com

Already have a base Eclipse project setup with all of the JAR dependencies and base Struts setup completed

Just to make sure the server is setup correctly, create an index.jsp and start the server

Now, create a set of pages to administer Items



Java and J2EE

www.qwantify.com

Create the Model Entity for Item

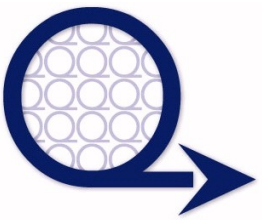
Just a POJO with Hibernate

Rather than using the standard Hibernate mapping files, we can now use Annotations

Annotations require JDK 1.5+

Entity Annotations are EJB 3 standard,
Hibernate has extensions

Basic Annotations to describe the Entity at a class level, and the columns and associations at the method level

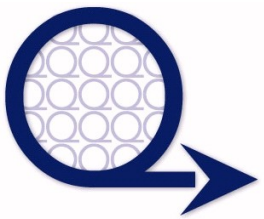


Java and J2EE

www.qwantify.com

Create JSPs to create and edit an Item and to view a list of all the items to administer

Sequence diagrams for searching for Items to display and for editing/creating Items are available on my website



Java and J2EE

www.qwantify.com

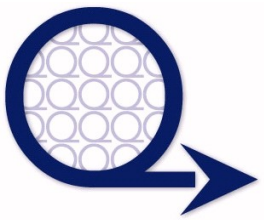
Create the DAO

No need to create multiple DAOs anymore

All DTOs inherit from a BaseDTO and the DAO uses the BaseDTO

DAO also uses Hibernate's DetachedCriteria

The DAO is completely agnostic to the DTO being manipulated or searched for



Java and J2EE

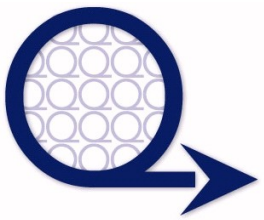
www.qwantify.com

Create an ItemFacade

Contains methods for searching, saving, and deleting

Facade converts input data for the data layer to use

Gives us a point to anchor our Transaction to
Spring 2.0 can use Annotations to demarcate Transactions



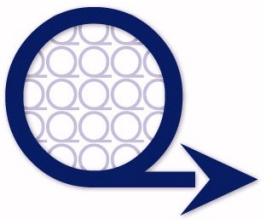
Java and J2EE

www.qwantify.com

Need an InputConverter to convert from a
SearchInput to a DetachedCriteria

Use a Factory

Create specific ItemConverters that are bound
to a specific SearchInput



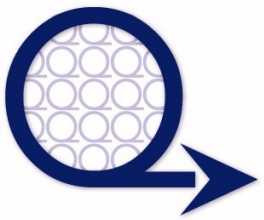
Java and J2EE

www.qwantify.com

Create an Action and ActionForm

ActionForm contains a SearchInput, but could contain anything else we need to pass to the Action or Facade for decision making

Action extends DispatchAction which allows us to group related Action functionality together rather than create multiple different Actions or have if/else statements in a single action

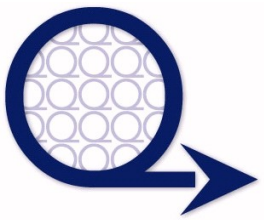


Java and J2EE

www.qwantify.com

Create a custom Servlet to load up Spring configuration

Create a SpringWrapper that will allow for easy retrieval of items from Spring without having to find the ApplicationContext



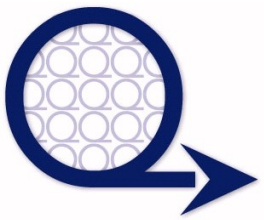
Java and J2EE

www.qwantify.com

Modify the struts-config.xml to include our
ActionForm and Action definition

Create Spring configuration file in which we
define our datasource, Hibernate setup,
transaction setup, and setup all of our beans

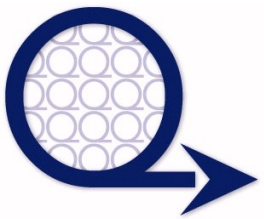
Add Spring ContextListener to the web.xml and
add entries for the ActionServlet



Java and J2EE

www.qwantify.com

Startup the server and see how everything works



Ruby on Rails

Ruby was created by Yukihiro “matz”

Matsumoto and publicly released in 1995

Ruby borrows elements from Perl, Smalltalk, Eiffel, Ada, and Lisp (functional programming)

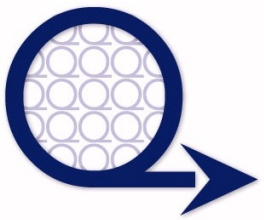
Clean and concise syntax

No semi-colons needed

Allows closures (sort of like anonymous function blocks)

Everything is an object (even primitives)

Modules (similar to interfaces with implementation) can be Mixed-Into any class



Ruby on Rails

www.qwantify.com

Easy Loops -

```
5.times { print "Hello!" }
```

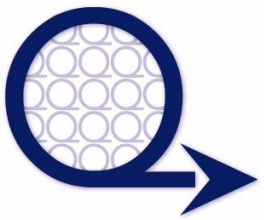
Arrays have nice methods (this returns a total)

-

```
def total
```

```
  @cart_items.inject(0) { |sum, item| sum  
  + item.subtotal }
```

```
end
```



Ruby on Rails

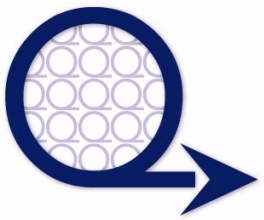
Rails created by David Heinemeier Hansson
and released in mid 2004

Philosophical concepts that Rails is built upon:
DRY (Don't Repeat Yourself) and convention
over configuration

MVC framework

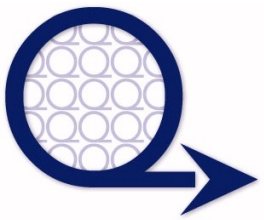
A place for each piece of code

Because it favors convention, it can easily
auto-generate a lot of code



Ruby on Rails

- Environment
 - Ruby 1.8.5
 - Rails 1.1.6
 - Eclipse 3.2 with most recent RDT and RadRails plugins
 - WEBRick webserver built into Rails



Ruby on Rails

Create a new Rails project in Eclipse

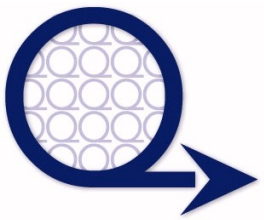
Open up the config/database.yml file

This is in YAML (YAML Ain't A Markup Language)

Rails convention for database names is to be suffixed with “_development”, “_test”, “_production”

Create the database and user

Run a Rake task (Rake is sort of like make or ANT) db:migrate to test connection



Ruby on Rails

www.qwantify.com

Create the Model entity for Item

Rails has a generator: `model item`

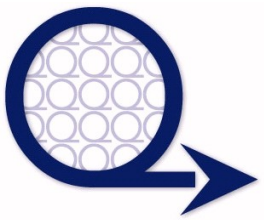
Creates `item.rb` in `models`, some test classes, and a file called `001_create_items.rb` in `db/migrate`

Rails understands pluralization

Translated `item` into `items`

Migration classes are used to make database changes

Add the code to define the Item data and run `Rake db:migrate`



Ruby on Rails

The items table has now been created

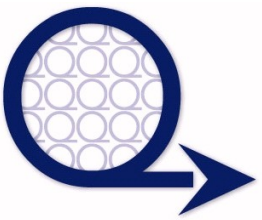
Look at item.rb

The model class is empty

Base ActiveRecord class provides everything dynamically

Data retrieval is based on database field names

Supports dynamic finder methods also



Ruby on Rails

www.qwantify.com

Generate controller admin/item

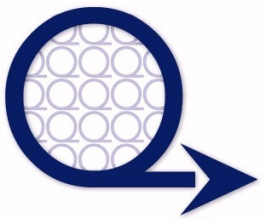
Creates controllers/admin/item_controller.rb

Add one line to the item_controller.rb file

Scaffold :item

Browse to <http://localhost:3000/admin/item>

Rails is dynamically generating an entire administrative interface



Ruby on Rails

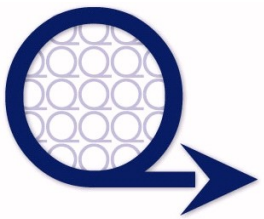
www.qwantify.com

Tough to customize this, though

Generate scaffold item 'admin/item'

Generates all the code that the scaffold :item was doing behind the scenes

All the various pieces of the MVC architecture are in very specific locations according to convention



Groovy and Grails

www.qwantify.com

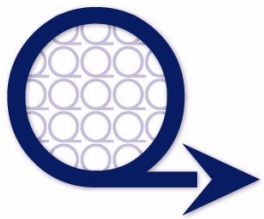
Groovy is a language similar to Ruby

It is built with Java and compiles to Java
bytecode to run in a JVM

Grails was created by some Groovy
enthusiasts who wanted a Rails-like web
framework

Prefers convention over configuration like Rails

Grails integrates with Hibernate 3, Spring 1.2,
and espouses domain-driven development

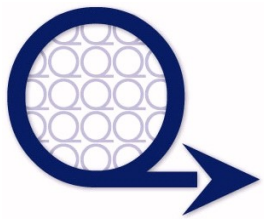


Groovy and Grails

www.qwantify.com

- Environment
 - JDK 1.5
 - Eclipse 3.2 with JST plugin, WST plugin, and most recent Groovy plugin
 - Groovy runtime optional as Eclipse Groovy plugin has an embedded Groovy interpreter
 - Grails 0.3.1

The newest version of Grails integrates with Eclipse. Check the Grails website for the details.



Groovy and Grails

www.qwantify.com

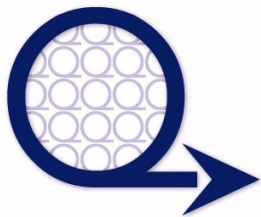
Create a new Java project in Eclipse

At a command prompt navigate to the workspace directory and run “grails create-app” using the Java project name as the application name

ANT script will prompt for name

Refresh the Eclipse workspace

The result is a directory structure similar to a Rails application with Groovy files, GSP files, and J2EE setup files



Groovy and Grails

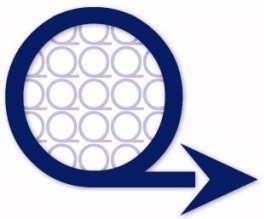
www.qwantify.com

At the command prompt, run “grails run-app”

This will start the Grails server (Jetty)

Browse to <http://localhost:8080/app>

Default Grails page will be displayed



Groovy and Grails

www.qwantify.com

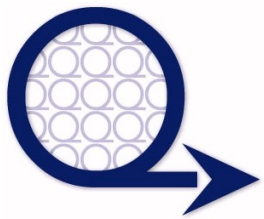
Copy the MySQL Connector into the Grails application lib directory

Create a database along with user

Open `conf/DevelopmentDataSource.groovy`

Grails uses a different Groovy file for each tier's data source configuration

Change the properties to work with the local database



Groovy and Grails

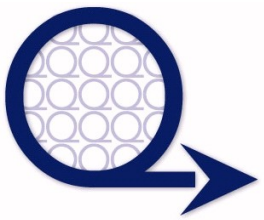
www.quantify.com

Ctrl+C in the command line to shut down the Jetty server

Run “grails create-domain-class”

When it asks for the class name type in “Item”

Grails will generate the domain class as well as some test classes



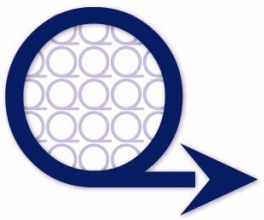
Groovy and Grails

www.qwantify.com

Refresh the Eclipse workspace and open up the file domain/Item.groovy

By default Grails creates an id field, version field, and overloaded equals() and hashCode() methods

Add the rest of the fields an Item needs



Groovy and Grails

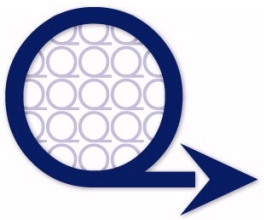
www.qwantify.com

At a command prompt run “grails generate-all”

When asked for a domain class use “Item”

Refresh the Eclipse workspace

Grails has generated a controller in the controllers directory, and various GSP files in the views directory, along with some test classes



Groovy and Grails

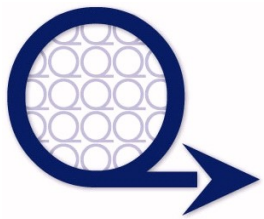
www.qwantify.com

Run “grails run-app” to start the server

Browse to <http://localhost:8080/app>

The default Grails page will show with a link to the generated Item controller admin page

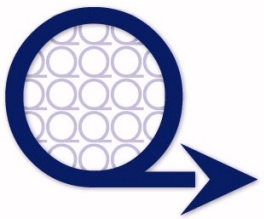
Go to the admin page and enter some widgets



Groovy and Grails

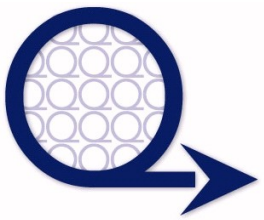
www.qwantify.com

- Even though Grails uses Hibernate and Spring we did not have to set anything up
- Grails generates the Hibernate and Spring definitions dynamically
- Hibernate and Spring configurations *can* be pre-configured and used with Grails



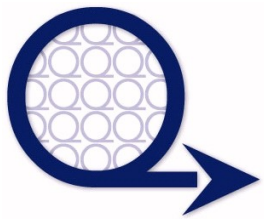
Conclusions

- J2EE development is very flexible due to the large number of frameworks and libraries, but the flexibility comes with a large learning curve
- Hibernate and Spring make a good combination but require a fair amount of configuration
- Simple applications are very easy to create using Rails
- Rails generates the majority of code for you



Conclusions

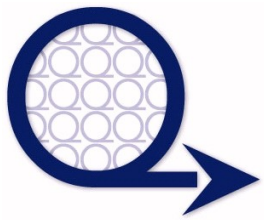
- By choosing convention over configuration, Rails makes it easy to figure out where everything is in an application
- RHTML pages require a fair amount of “coding” inside the pages, seems like a throwback to days before custom tags
- Grails leverages the benefits of Groovy and Rails-like development with the full power of Java and J2EE
- Grails doesn't feel mature at this point



Demo App Statistics

www.qwantify.com

- Java/J2EE
 - 43 Classes
 - XX JSPs
 - XX Configuration files
 - 1178 Non-comment LOC
- Ruby on Rails
 - 15 Classes
 - 33 Page fragments (each scaffold generated controller has 5 fragments)
 - 7 Configuration files (including migrations)
 - 464 Non-comment LOC (with autogenerated code)
- Groovy/Grails
 - XX Classes
 - XX GSPs
 - XX Configuration files



Java Resources

www.qwantify.com

<http://java.sun.com>

<http://www.eclipse.org>

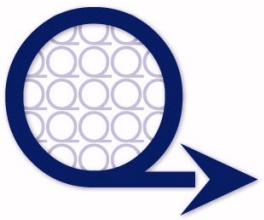
<http://www.mysql.com>

<http://tomcat.apache.org>

<http://struts.apache.org>

<http://www.springframework.org>

<http://www.hibernate.org>



Ruby and Rails Resources

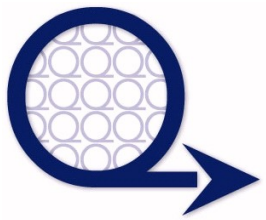
www.qwantify.com

<http://updatesite.rubypeople.org> - RDT Eclipse Plugin

<http://radrails.sourceforge.net/update> - RadRails Eclipse Plugin

<http://www.ruby-lang.org>

<http://www.rubyonrails.org>



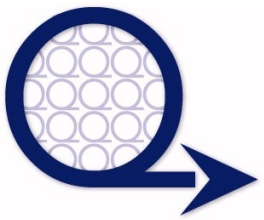
Groovy and Grails Resources

www.qwantify.com

<http://groovy.codehaus.org>

<http://grails.codehaus.org>

<http://dist.codehaus.org/groovy/distributions/update> -
Groovy Eclipse Plugin



Demo Application Resources

www.qwantify.com

Available at <http://tech.fradkin.com> are the complete applications, base Eclipse projects, UML files, SQL scripts, etc.

UML diagrams generated using ArgoUML -
<http://argouml.tigris.org>