

RAPID DATABASE DRIVEN WEB DEVELOPMENT WITH  GRAILS

bob brown, transentia pty. ltd.
www.transentia.com.au



'Grails aims to bring the *"coding by convention"* paradigm to Groovy. It's an open-source web application framework that *leverages the Groovy language* and complements Java Web development.'

—www.grails.org



“...Grails is supported by *proven technologies*.

Hibernate, a de facto standard in the software industry, provides the basis for the object-relational mapping (ORM) in Grails.

The *Spring Framework* supplies the core of the Grails Model-View-Controller (MVC) architecture and enables powerful dependency injection.

SiteMesh brings flexible and effective layout management to Grails.

And, let's not forget *Java*. Because of Groovy's excellent Java integration, Grails applications not only have direct access to the multitude of Java libraries, but also to the enterprise services (distributed transactions, messaging, etc.) provided by JEE...”

—http://www.infoq.com/minibooks/grails

'One of the great things about Grails is that it *does not suffer from "Not Invented Here" syndrome*. Instead of reinventing the wheel, it *integrates with best of breed proven open source frameworks*. It then goes on to make the integration seamless by hiding much of the complexities of those frameworks.'

—www.grails.org/Reference

“...Java still has *too much as a platform to give-up on*. There are libraries to do just about anything in Java. You have fantastic things happening in the industry like data grid solutions (Terracotta, Coherence etc.), standards-based CMS (JSR-170), and all the innovation within Spring. These are *things you simply don't get elsewhere*.”

—http://www.indichthreads.com/interviews/1098/grails_web_framework_java_developers_1.html

"Grails is an outstanding framework. ...It is really hard to work with the beautiful technologies groovy and grails at home and then switch again to JSF and Struts at work and see how much time you are wasting in writing boilerplate and repeating code. Every serious Java-Developer should have a look at grails and groovy. It is definitively the future of Java!"

"One thing that I came to value particularly about Grails is that it allows me to create little web applications so quickly..."

"I downloaded Grails a couple months ago. I got really excited as I got into it - every time I went looking for something ('I wish I could do xxx') - it was there! It was also great to know that if I needed to get 'under the covers' of the app it was just a matter of opening the Spring and Hibernate config files and getting to work."

"We wanted to implement a platform for user experiences with alternative medicine. Because we had a limited budget we looked for a way to implement it more efficient than with Java and the usual change-build-deploy-test cycles. First we tested Ruby on Rails than we discovered Groovy and Grails so we could leverage our existing Java knowledge. It took us less than a month to create the first version and to get online. Thanks to the rapid prototyping features we can release an improved and extended version every week. Thanks for enabling us to start our own Web 2.0 adventure."

“So one day several years ago the suits came to me and told me I had to stop using Perl and start using Java. It was hell. I lived in J2EE hell, JSF hell, Portlet hell, Workflow engine hell, Seam hell, then I spent some time with EJB3 and that felt less like hell.



Then the clouds parted, the angels sang, and there before me stood Groovy and Grails.

Groovy is pure joy in a bucket. It was so much less painful to transition to Groovy. Grails made so much more sense than JSF and Seam. Jetty was so much easier to set up and run than a full application server. I was so much happier... and I was able to use any of the Java stuff I wanted and I could even write shell scripts in Groovy. I could use Grails tools to automatically generate so much code I would have had to write in any other framework... Perl included... I was so happy that the pain had stopped. The confusion lifted! The buzzword acronym laden pea soup had stopped. Life began to make sense. I actually began to prefer working in Groovy and Grails to working with Perl, CGI.pm, and Template engines.

This is shocking but Groovy and Grails might actually be better than Perl, Ruby, or Python. No really. I'm sure this will start a flame war. But honestly you need to look at it. You really do.”

—<http://developers.slashdot.org/comments.pl?sid=431506&cid=22199018>

“10 Reasons to Switch from Rails to Grails

After spending a few years really enjoying Rails it was difficult to bring myself to even try groovy and grails. But my latest contract forced me to look for alternatives, and I'm glad I did. Here are some reasons that you may want to switch...

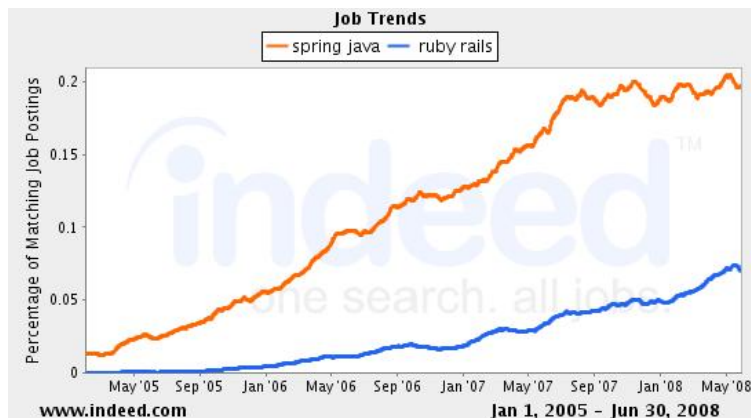
- GORM with hibernate/spring and jpa is much better than ActiveRecord
- No distribution problems; runs in many production ready containers
- Internationalization out of the box (not specifically ignored...)
- Transactions actually work, and they include save-points.
- Not only dynamic finders and counters, but dynamic listOrderBy
- No green threads (although this may be fixed in Ruby 2.0, around 2010?)
- Ability to use pessimistic locking out of the box
- Real-live prepared statements and .withCriteria method
- Production level test reporting with built in Mocking and Stubbing
- Search operations are based on Lucene (with a plugin)”

—<http://raincitysoftware.blogspot.com/2007/12/10-reasons-to-switch-from-rails-to.html>

“...now Graeme Rocher of G2One has added another ten. ...

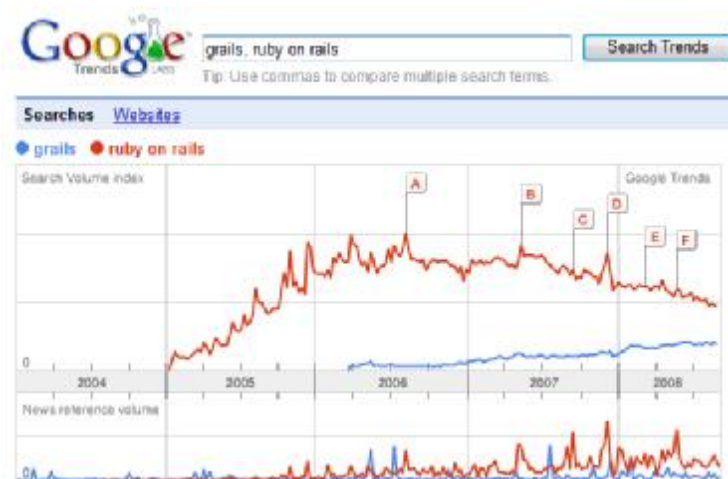
- View technology that doesn't suck
- Mixed source development made easy with the Groovy joint compiler (no falling back to C to solve those performance problems ;-)
- Built in support for rich conversations with Web Flow
- A rich plug-in system that integrates Grails with technologies Java people care about like GWT, DWR, JMS etc.
- A buzzing and growing community with a larger traffic mailing list as opposed to a stagnating one
- Built on Spring, the ultimate Enterprise Application Integration technology
- A Service layer with automatic transaction demarcation and support for scopes
- More books coming and being adopted by enterprise organizations near you”

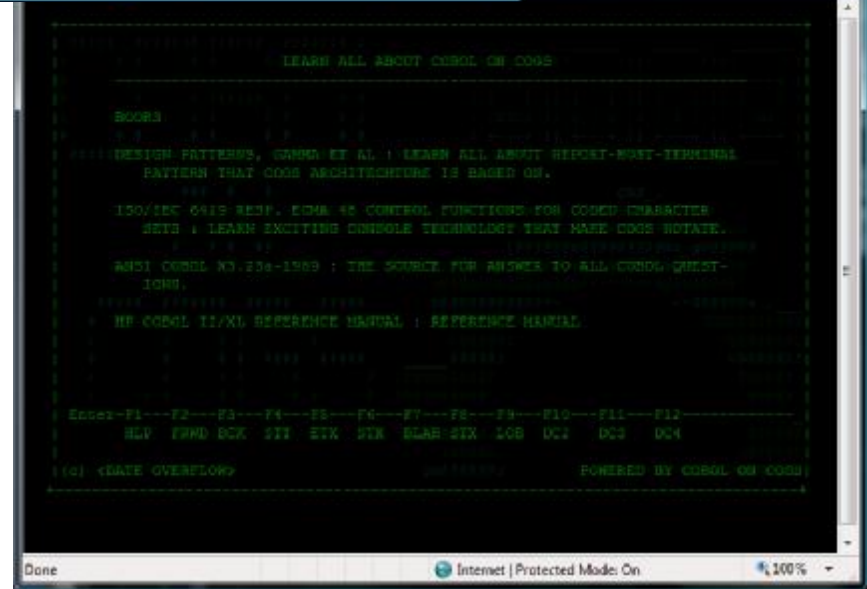
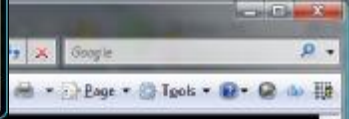
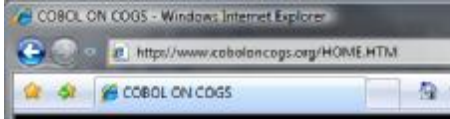
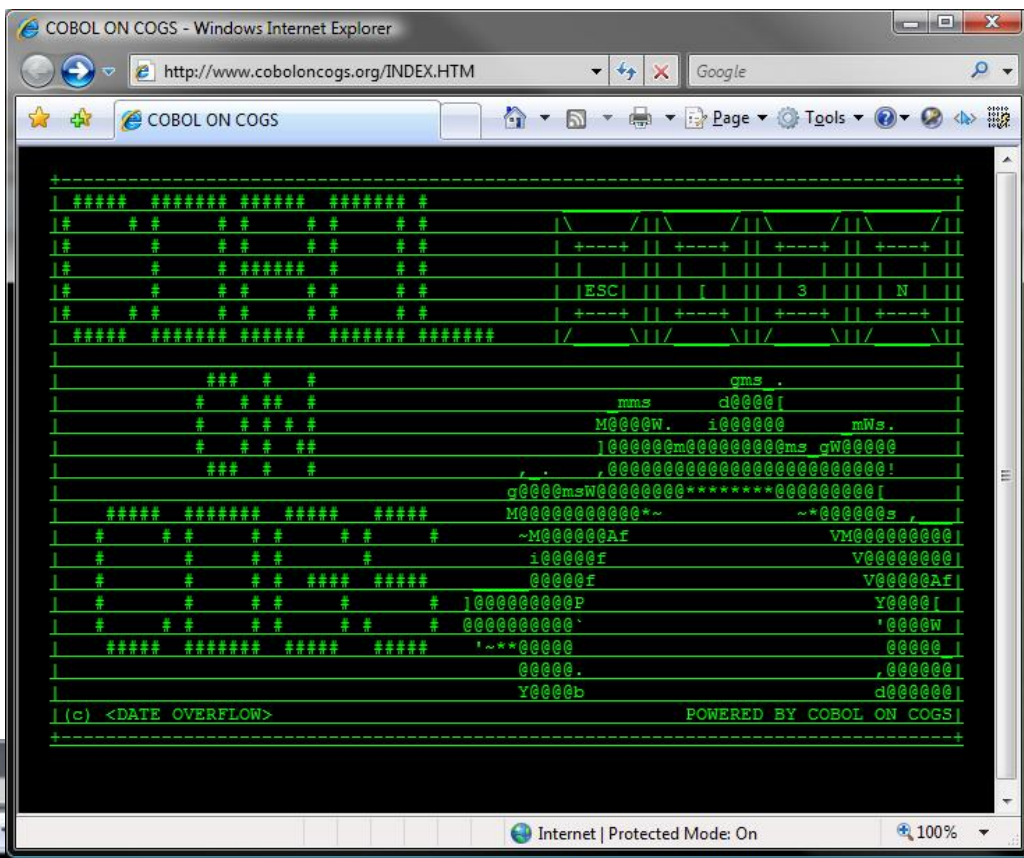
—<http://swik.net/Rails-Ruby/TechKnow+Zenze/The+Great+Grails+Switch/bz95c>



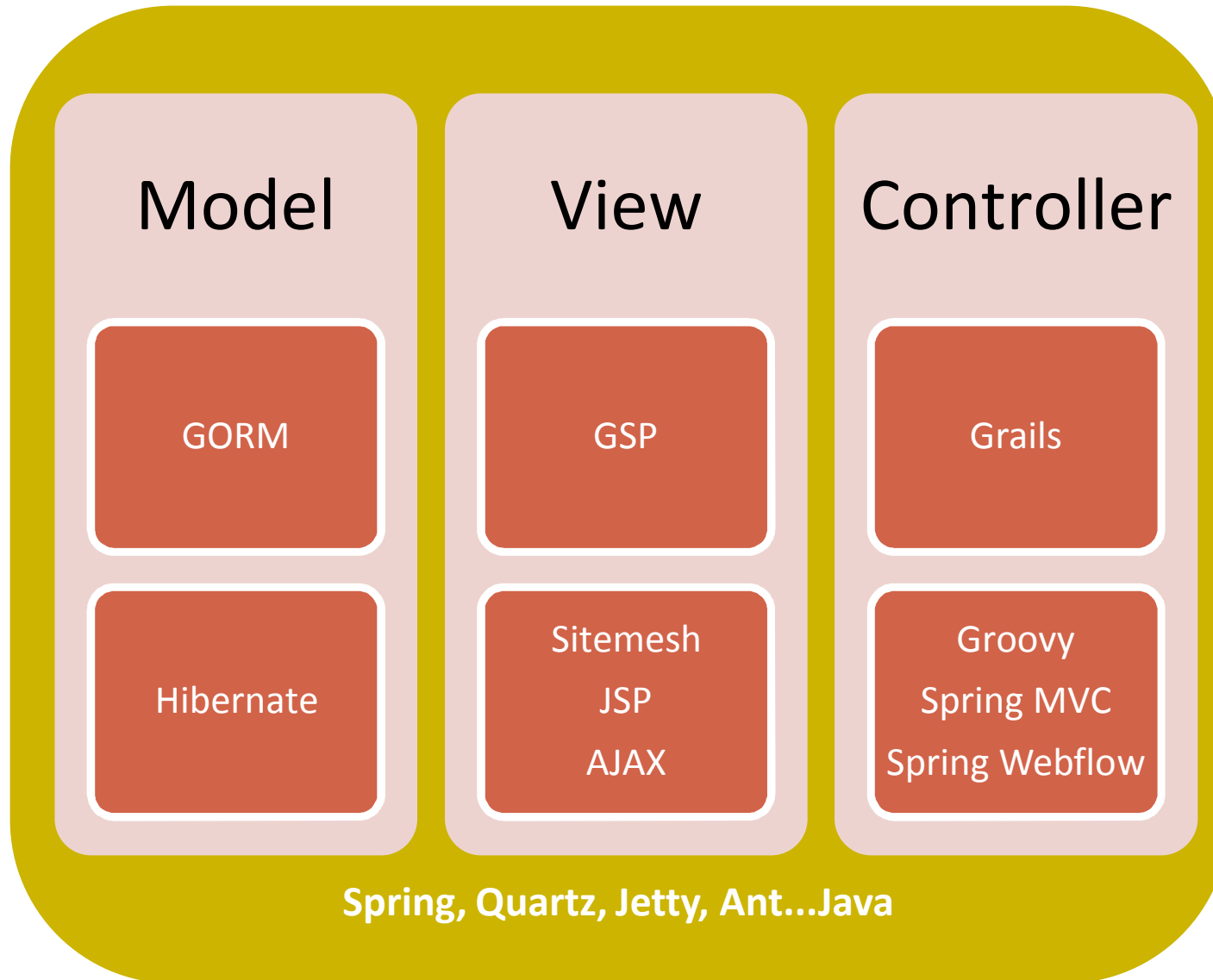
\$\$ or \$\$

...that is
the *real*
question...





- best of breed stuff



- can still get “under the covers” when necessary



- convention over configuration
 - lots of little Domain Specific Languages
- scaffolding
 - command-line generator/tool
 - generate full CRUD webapp with minimal effort

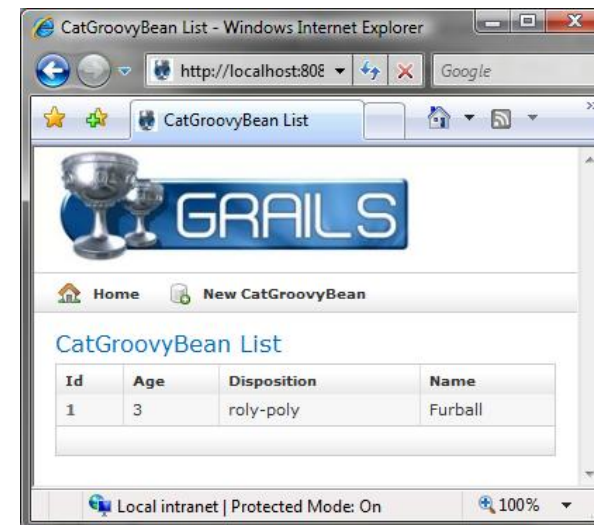
persistent domain
class

```
class CatGroovyBean {
    String name
    short age
    String disposition
}
```

scaffolding

```
class CatGroovyBeanController {
    static scaffold = CatGroovyBean
}
```

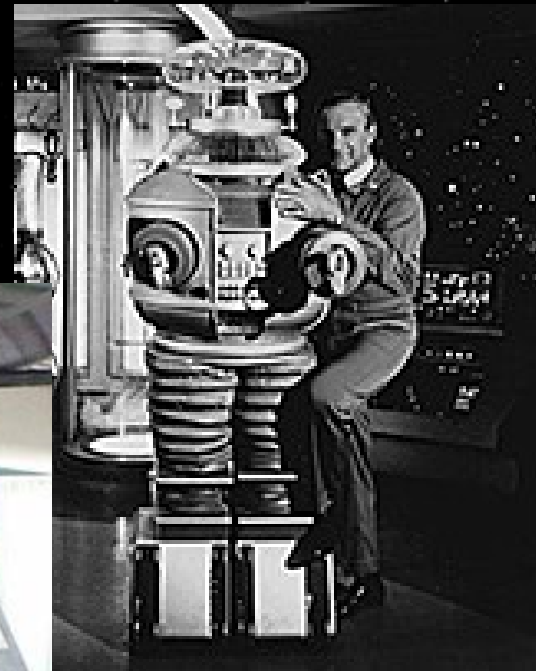
controller class



```
C:\Windows\system32\cmd.exe
C:\Users\Bob Brown\Desktop>grails create-app Cats
C:\Users\Bob Brown\Desktop>cd Cats
C:\Users\Bob Brown\Desktop>grails create-domain-class CatGroovyBean
C:\Users\Bob Brown\Desktop>grails create-controller CatGroovyBean
C:\Users\Bob Brown\Desktop>vim grails-app\domain\CatGroovyBean.groovy
C:\Users\Bob Brown\Desktop>vim grails-app\controllers\CatGroovyBeanController.groovy
C:\Users\Bob Brown\Desktop>grails run-app Cats
```

Danger, Will Robinson!

My sensors detect a large techno-nerd-fest coming this way...



- DRY & KISS
- work on live systems
 - no more build-wait-deploy-wait-test-wait-wait-wait...
- common/central configuration
 - conf/BootStrap.groovy
- project artifacts
 - domain classes, controllers, services, taglibs, jobs, bootstraps, datasources, etc.
 - conventional naming
 - Fred, FredController, FredService, FredFilter, FredTagLib, etc.
- project/filesystem layout
- logging
 - all artifacts have a dynamically added “log” property
 - configured via log4J DSL in conf/Config.groovy

```

C:\temp\Temp>tree /a
Folder PATH listing for volume OS
Volume serial number is 0007F208 98D8:A37A
C:
+---.settings
+---grails-app
+---conf
|   +---hibernate
|   \---spring
+---controllers
+---domain
+---i18n
+---services
+---taglib
+---utils
|   \---views
|       \---layouts
+---lib
+---scripts
+---src
|   +---groovy
|   \---java
+---test
|   +---integration
|   \---unit
\---web-app
    +---css
    |   \---tree
    |       +---check
    |       +---default
    |       +---folders
    |       \---menu
    +---images
    |   +---skin
    |   \---tree
    |       +---check
    |       +---default
    |       +---folders
    |       \---menu
    +---js
    |   \---prototype
    +---META-INF
    \---WEB-INF
        \---tld
C:\temp\Temp>_
  
```


- command line

- grails

- where everything starts

- grails console

- live investigation of an app
 - *extremely important*, IMHO

- IDE support

- Eclipse, Netbeans

- IntelliJ

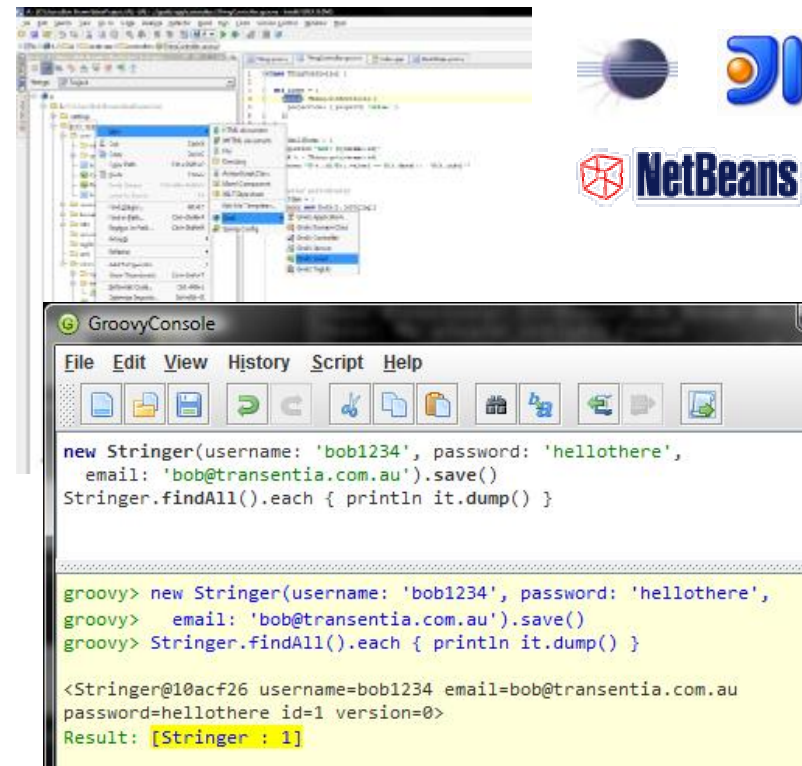
- best (so far ...)

- not JDeveloper (yet!)

“Although Oracle JDeveloper does not yet come with special features for Groovy and Grails, Oracle JDeveloper’s external tools capability enables us to quite easily set up Oracle JDeveloper for Grails development.”

—<http://www.oracle.com/technology/pub/articles/oak-grails.html>

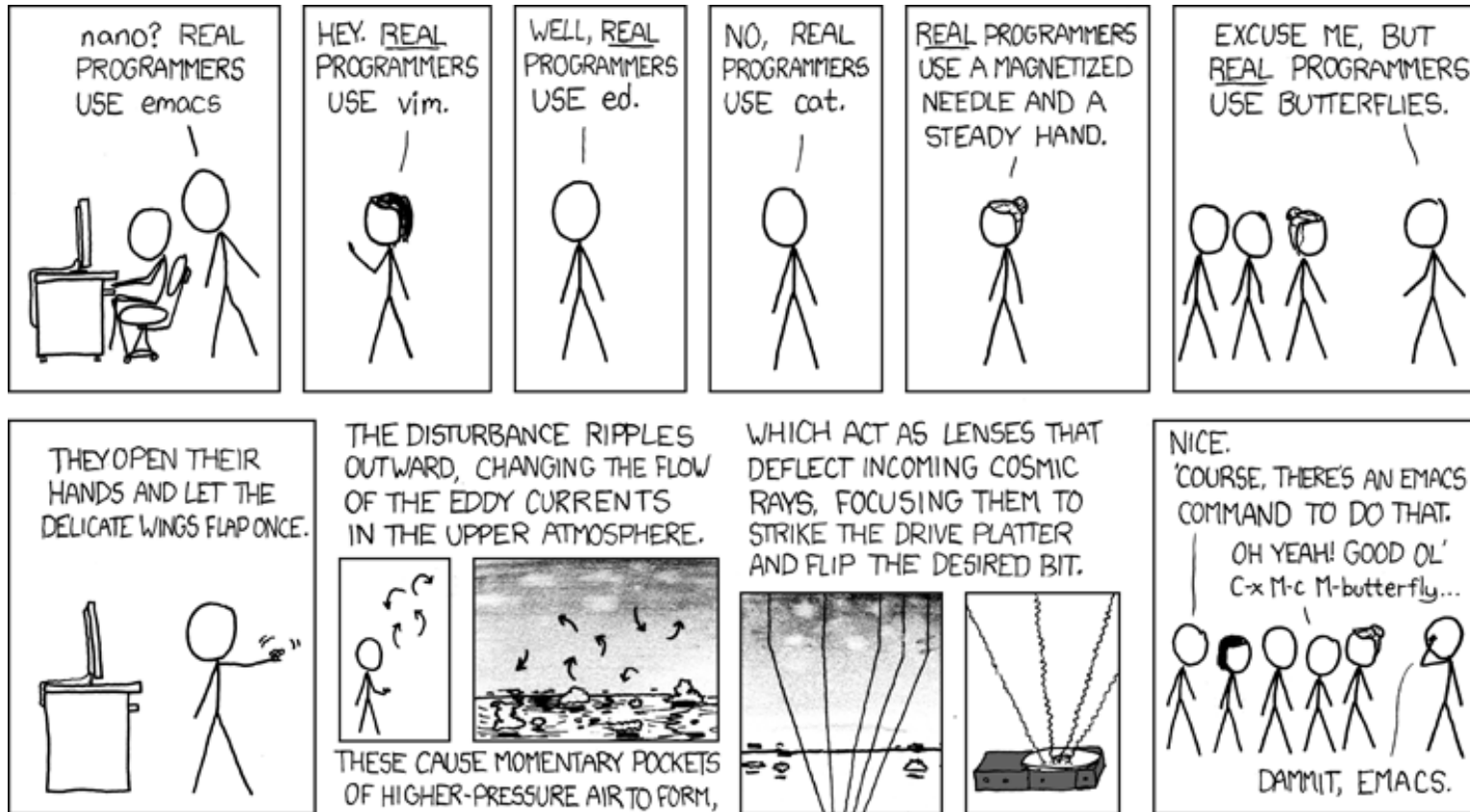
- currently often need to drop into the command line



“...other important drawback I see is about tool support, and especially IDE support. Current plugins mostly just support syntax highlighting, compilation, running. But they don’t offer the nice features available in standard Java IDEs.”


—<http://www.indichthreads.com/content/view/429/0/1/3/>

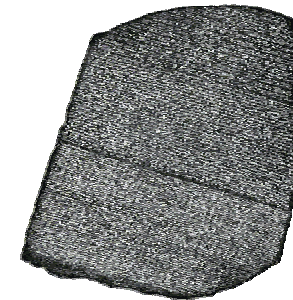
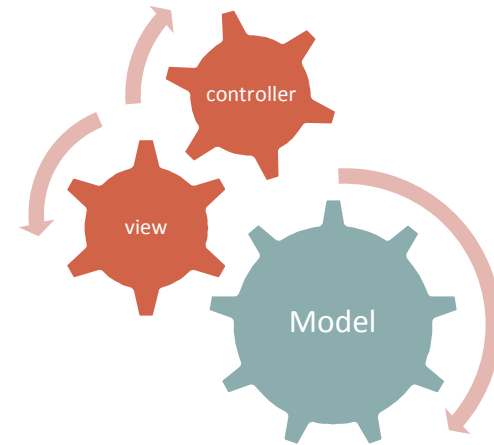
- IDEs are over-rated...



<http://xkcd.com/378/>

- makes it easy to create an application domain model

- based on  HIBERNATE
 - open-source, mature, powerful
 - (the JPA spec. is basically a formalisation of Hibernate)
- maps POGOs to Relational DBs
- Bob's rules...
 - *"a database is just a bit-bucket"*
 - *"thou shalt not SQL"*



"Usually the model will live (persist) in a database, but it's important not to think of the model as the database. The domain of an application, in an object oriented design, is the set of object definitions for any object that will play a part in the application. ...it's important to note that the domain model does not have to be consistent with the underlying storage schema."

—<http://etweb.wordpress.com/category/uml-diagraming/>

- datasources
 - caching/pooling
 - environmentally-aware
 - can defer to the container via JNDI
 - conf/DataSource.groovy

```
dataSource {
    pooled = true
    driverClassName = "org.hsqldb.jdbcDriver"
    username = "sa"
    password = ""
    //logSql = true
}
hibernate {
    cache.use_second_level_cache = true
    cache.use_query_cache = true
    cache.provider_class = 'com.opensymphony.oscache.hibernate.OSCacheProvider'
}
environments {
    development {
        dataSource {
            dbCreate = "create-drop" // one of 'create', 'create-drop', 'update'
            url = "jdbc:hsqldb:mem:devDB"
        }
    }
    production {
        dataSource {
            dbCreate = "update"
            url = "jdbc:hsqldb:file:prodDb;shutdown=true"
        }
    }
}
```


- command line example

```

c:\ Command Prompt

C:\temp\Temp>grails create-domain-class Catastrophe

Welcome to Grails 1.0.3 - http://grails.org/
Licensed under Apache Standard License 2.0
Grails home is set to: C:\DEUTOOLS\grails-1.0.3

Base Directory: C:\temp\Temp
Note: No plugin scripts found
Running script C:\DEUTOOLS\grails-1.0.3\scripts\CreateDomainClass.groovy
Environment set to development
  [copy] Copying 1 file to C:\temp\Temp\grails-app\domain
Created Domain Class for Catastrophe
  [copy] Copying 1 file to C:\temp\Temp\test\integration
Created Tests for Catastrophe
C:\temp\Temp>grails create-domain-class Citation

Welcome to Grails 1.0.3 - http://grails.org/
Licensed under Apache Standard License 2.0
Grails home is set to: C:\DEUTOOLS\grails-1.0.3

Base Directory: C:\temp\Temp
Note: No plugin scripts found
Running script C:\DEUTOOLS\grails-1.0.3\scripts\CreateDomainClass.groovy
Environment set to development
  [copy] Copying 1 file to C:\temp\Temp\grails-app\domain
Created Domain Class for Citation
  [copy] Copying 1 file to C:\temp\Temp\test\integration
Created Tests for Citation
C:\temp\Temp>grails create-domain-class Stringer

Welcome to Grails 1.0.3 - http://grails.org/
Licensed under Apache Standard License 2.0
Grails home is set to: C:\DEUTOOLS\grails-1.0.3

Base Directory: C:\temp\Temp
Note: No plugin scripts found
Running script C:\DEUTOOLS\grails-1.0.3\scripts\CreateDomainClass.groovy
Environment set to development
  [copy] Copying 1 file to C:\temp\Temp\grails-app\domain
Created Domain Class for Stringer
  [copy] Copying 1 file to C:\temp\Temp\test\integration
Created Tests for Stringer
C:\temp\Temp>

```

```

class Catastrophe {
}
...

```



```

class CatastropheTests
  extends GroovyTestCase {

    void testSomething() {

    }

}
...

```



all the requisite, essentially boilerplate, configuration

```

class Citation {
    static belongsTo = Catastrophe

    Catastrophe catastrophe
    Date publicationDate = new Date()
    String details
    dnd.CitationType citationType
    Stringer creator

    static constraints = {
        details blank: false, maxLength: 1024
        creator nullable: false
        catastrophe nullable: false
        citationType nullable: false
        publicationDate nullable: false
    }

    // dealing with legacy database yukky-ness...
    static mapping = {
        table 'THE_CITATION_TABLE'
        version false
        columns {
            id column: 'CITATION_PK',
                generator:'hilo',
                params:[table:'hi_value',
                    column:'next_value',
                    max_lo:100]
            citationType column: 'CITATION_TYPE_COLUMN'
            publicationDate index: 'PUB_DATE_INDEX'
            details type: 'text'
        }
    }
}

```

note the absence of
unwanted 'fluff'

```

class Catastrophe {
    Date deadline
    String description
    static hasMany = [citations: Citation]
    Stringer creator

    static constraints = {
        description blank: false
        deadline validator: {return (it > new Date())}
        creator nullable: false
    }
}

```

```

class Stringer {
    String username
    String email
    String password

    static constraints = {
        username unique:true, nullable:false,
            blank:false, size:6..12
        email unique:true, email:true
        password nullable:false,
            blank:false, size:8..80
    }
}

```

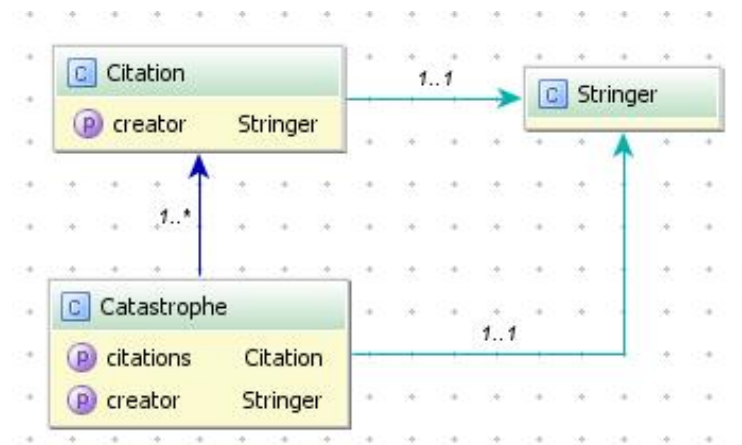
- all the standard database goodness
 - cardinality, directed relationships, cascading, compound keys, indices, composition, inheritance, locking, versioning, lazy fetch, ...

```
class Citation {
  static belongsTo = Catastrophe
  Stringer creator
  ...
}

class Catastrophe {
  ...
  static hasMany = [citations: Citation]
  Stringer creator
  ...
}

class Stringer {
  ...
}

new Catastrophe (creator: new Stringer(...))
  .addToCitations(new Citation(...)).save()
```



```
class Person {
  String firstName
  String lastName
  static mapping = {
    id composite:['firstName', 'lastName'] }
}

def p = Person.get(new Person(firstName:"Fred",
                              lastName:"Flintstone"))

println p.firstName
```

- dynamically generated methods
 - excellent example of convention over configuration

- CRUD

```
Stringer bob = new Stringer(username:'bob1234', password:'bob987654321',  
    email:/bob@transentia.com.au/).save()
```

```
if (Stringer.exists(1)) {  
    Stringer firstOne = Stringer.get(1)  
    firstOne.email = 'bob@gmail.com'  
    firstOne.save()  
}
```

```
bob.delete()
```

- listing, sorting, counting

```
Stringer.list(offset:0, max:10, sort:'username',  
    order:'asc').each { s -> println s }
```

```
Stringer.listOrderByUsername (order:'asc')*.delete()
```

```
println Stringer.count()
```

```
println Stringer.countByUsernameLike ('bob%')
```

“save the bean, save the world”

—http://jroller.com/page/ie?entry=java_on_grails_save_the



```
Book.metaClass.static.methodMissing = { String name, args ->
    if(name.startsWith("findBy") && args) {
        def prop = name[6..-1]
        prop = prop[0].toLowerCase() + prop[1..-1]
        def callable = { Object[] varArgs ->
            def results = []
            def sql = Sql.newInstance( url, user, pass, driver)
            sql.eachRow("""select * from ${Book.name} where $prop=?""",
                [varArgs[0]]) {
                results << new Book(title:it.title, author:it.author)
            }
            return results
        }
        Book.metaClass."$name" = callable
        return callable.call(args)
    }
}
```

GroovyConsole

```
File Edit View History Script Help
[Icons]

new Stringer(username: 'bob1234', password: 'hellothere',
    email: 'bob@transentia.com.au').save()

Stringer.findAll().each { println it.dump() }

Stringer.findByUsernameLikeAndPasswordLike('bob%', 'hell%').each { println it.dump() }

Stringer.findByVersionNotEqual(0).each { println it.dump() }

Stringer.find(new Stringer(username: 'bob1234')).each { println it.dump() }

groovy> new Stringer(username: 'bob1234', password: 'hellothere',
groovy>     email: 'bob@transentia.com.au').save()
groovy> Stringer.findAll().each { println it.dump() }
groovy> Stringer.findByUsernameLikeAndPasswordLike('bob%', 'hell%').each { println
it.dump() }
groovy> Stringer.findByVersionNotEqual(0).each { println it.dump() }
groovy> Stringer.find(new Stringer(username: 'bob1234')).each { println it.dump() }

<Stringer@be5beb username=bob1234 email=bob@transentia.com.au password=hellothere
id=1 version=0>
<Stringer@be5beb username=bob1234 email=bob@transentia.com.au password=hellothere
id=1 version=0>
<Stringer@be5beb username=bob1234 email=bob@transentia.com.au password=hellothere
id=1 version=0>
Result: Stringer : 1

Execution complete. 10:1
```

- events

```
class Stringer {
    Date dateCreated
    Date lastUpdated

    def beforeInsert = { dateCreated = new Date() }
    def beforeUpdate = { lastUpdated = new Date() }
    def onLoad = { println 'I am a new Stringer' }
}
```

```
class Stringer {

    Date dateCreated
    Date lastUpdated

    static mapping = {
        autoTimestamp true
    }
    ...
}
```

- HQL

- ‘native’ Hibernate

```
Stringer.executeQuery('select s.username, s.password from Stringer s where s.username like :u',
    [u:'bob%'])
```

- criteria builder

```
Stringer.withCriteria {
    projections { property 'email' }
    and {
        like "username", "bob%"
        between "id", 0L, 1000L
    }
    maxResults 10
}
```

“Hibernate projections are uber-powerful for grouping and reporting operations.”

—<http://blogs.bytecode.com.au/glen/2008/09/03/groovyawards-org-source-code-now-on-github.html>

- tuning

- still have access to the raw hibernate underneath

- transactions
 - simple
 - can be tuned via the underlying Spring infrastructure

```
class StringerService {  
  static transactional = true  
  def modifyLotsOfData(id) {  
    def s = Stringer.get(id)  
    s.save()  
    // .. etc.  
  }  
}
```

```
def id = 1  
Stringer.withTransaction { status ->  
  def s = Stringer.get(id)  
  
  // something goes horribly wrong  
  
  status.setRollbackOnly()  
}
```

- domain 'drives' validation throughout the application

```
class Stringer {
    String username
    String email
    String password

    static constraints = {
        username unique:true, nullable:false,
            blank:false, size:6..12
        email unique:true, email:true
        password nullable:false,
            blank:false, size:8..80
    }
}
```

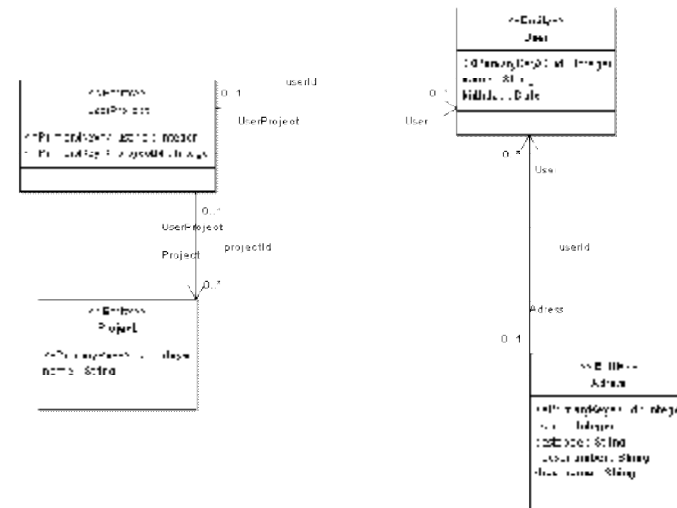
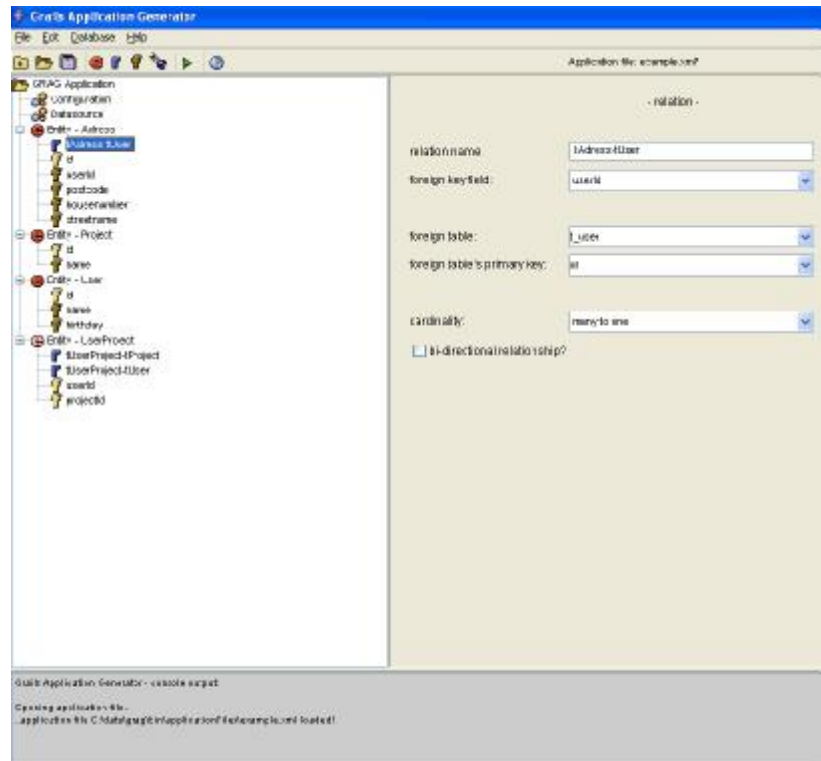


- i18n is catered for
 - messages files

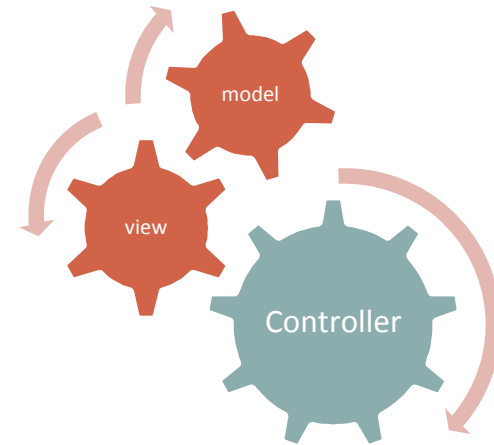
```
...
default.invalid.email.message=Property [{0}] of class [{1}] with value [{2}] is not a valid e-mail address
default.invalid.range.message=\
    Property [{0}] of class [{1}] with value [{2}] does not fall within the valid range from [{3}] to [{4}]
default.invalid.size.message=\
    Property [{0}] of class [{1}] with value [{2}] does not fall within the valid size range from [{3}] to [{4}]
default.invalid.max.message=Property [{0}] of class [{1}] with value [{2}] exceeds maximum value [{3}]
default.invalid.min.message=Property [{0}] of class [{1}] with value [{2}] is less than minimum value [{3}]
...
```


- GRails Application Generator
 - <http://grag.sourceforge.net>
 - 3rd party tool to reverse-engineer a database into GORM
 - with some UML ‘goodness’

“UML...one of the most successful scams in our industry.”
 —http://developers.slashdot.org/comments.pl?sid=569941&no_d2=1&cid=23616705



- “...the dictator of your application...”
 - according to Grails’ Graeme Rocher
- POGOs
 - thus easily testable
- ‘prototypes’
 - no need to deal with threading



```

import grails.converters.XML

class MyController {

    def index = { redirect action: 'text' }

    def text = { render 'Hello, Grails World!' }

    def xml = {
        render text: '<greeting>Hello, Grails World!</greeting>',
              contentType: 'text/xml',
              encoding: 'UTF-8'
    }

    def htm = {
        render(contentType: 'text/html', encoding: 'UTF-8') {
            html {
                head {
                    title 'Hello'
                }
                body {
                    h1 'Hello'
                    p 'world'
                }
            }
        }
    }

    def goodstuff = {
        def ex = new Expando()
        ex.f0 = 'hello'
        ex.f1 = 'world'

        render ex as XML
    }
}

```

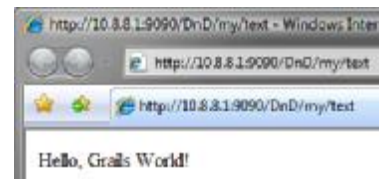
```

C:\temp\Temp>grails create-controller My

Welcome to Grails 1.0.3 - http://grails.org/
Licensed under Apache Standard License 2.0
Grails home is set to: C:\DEU00LS\grails-1.0.3

Base Directory: C:\temp\Temp
Note: No plugin scripts found
Running script C:\DEU00LS\grails-1.0.3\scripts\CreateController.groovy
Environment set to development
  [copy] Copying 1 file to C:\temp\Temp\grails-app\controllers
Created Controller for My
  [mkdir] Created dir: C:\temp\Temp\grails-app\views\my
  [copy] Copying 1 file to C:\temp\Temp\test\integration
Created ControllerTests for My
C:\temp\Temp>

```



- **parameter validation/binding**
 - driven by the domain object's constraints
 - also i18n capable
- **standard properties**
 - `actionName`, `controllerName`, `params`, etc.
- **scopes**
 - `request`,
`session`,
`response`
 - all enhanced
 - `flash`
- **models**
 - data passed to
the view for
rendering

```
class StringerController {  
  
  def show = {  
    def stringer = Stringer.get( params.id )  
  
    if(!stringer) {  
      flash.message = "Stringer not found with id ${params.id}"  
      redirect(action:list)  
    }  
    else { return [ stringer : stringer ] }  
  }  
  
  def save = {  
    def stringer = new Stringer(params)  
    if(!stringer.hasErrors() && stringer.save()) {  
      flash.message = "Stringer ${stringer.id} created"  
      redirect(action:show,id:stringer.id)  
    }  
    else {  
      render(view:'create',model:[stringer:stringer])  
    }  
  }  
}  
  
def title = request.XML?.book?.title  
render "The Title is $title"
```

- interceptors

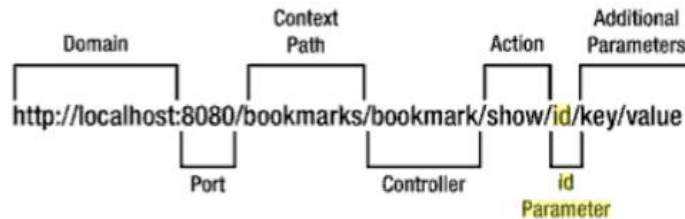
```
class SomeController {  
  
  def auth = { /* authorization stuff; return false to prevent execution */ }  
  def beforeInterceptor [action:this.&auth, except:['login','register']]  
  
  def afterInterceptor = { model, modelAndView ->  
    log.debug "Current view is ${modelAndView.viewName}"  
    if(model.someVar)  
      modelAndView.viewName = "/mycontroller/someotherview"  
    log.debug "View is now ${modelAndView.viewName}"  
  }  
}
```

- filters

- useful for intercepting across multiple controllers
- easier to plug-in and maintain

```
class SecurityFilters {  
  def filters = {  
    allURIs(uri:'/**') {  
      log.debug request.requestURI  
    }  
  
    loginCheck(controller:'*', action:'*') {  
      before = {  
        if(!session.user && !actionName.equals('login')) {  
          redirect(action:'login')  
          return false  
        }  
      }  
    }  
  }  
}
```

- URI format and mapping
 - (REpresentational State Transfer)-like format



- `UrlMappings.groovy`
 - allows for interesting declarative mappings/configuration

```
class UrlMappings {
    static mappings = {
        "/*controller/*action?/*id?" {
            constraints {
                // apply constraints here
            }
        }
        "/*blog/*year/*month/*day/*id" (controller:"blog", action:"show")
        "/*holiday/win" {
            id = { params.id }
            isEligible = { session.user != null } // must be logged in
        }
        "500" (view: '/error')
    }
}
```

• webflow

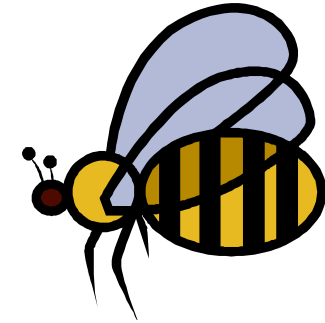
- a *conversation* that spans multiple requests and retains state for the scope of the flow...also has a defined start and end state
- clean DSL
- driven via GSPs

```
<!-- win.gsp -->
<html>
  <body>
    <h1>You win</h1>
    <g:link
      action="game"
      event="playAgain">Play again?</g:link>
  </body>
</html>
```

```
class NumberGuessController {

  def index = { render(view: "game") }

  def gameFlow = {
    def numberOfGuesses
    def numberToGuess
    start() {
      action {
        numberOfGuesses = 5
        numberToGuess = new Random().nextInt(100)
        log.debug("number to guess ${numberToGuess}")
      }
      on("success").to "guess"
    }
    guess {
      on("submit") {
        flow.guessedNumber = Long.parseLong(params.number)
      }.to "evaluateGuess"
    }
    evaluateGuess {
      action {
        return (numberToGuess == flow.guessedNumber) ? win() :
          evalNbrOfGuesses()
      }
      on("win").to "win"
      on("evaluateNumberOfGuesses").to "evalNbrOfGuesses"
    }
    evalNbrOfGuesses{
      action {
        return (--numberOfGuesses > 0) ? tryAgain() : loose()
      }
      on("tryAgain").to "guess"
      on("loose").to "loose"
    }
    win() { on("playAgain").to "start" }
    loose() { on("tryAgain").to "start" }
  }
}
```



- **service == worker bee**
 - POGOs, again
 - designed to be injected into controllers
 - shared, so not thread-safe
 - c.f. controllers, which are thread safe by default
 - can be transactional
 - central place for ‘real’ business logic
 - easily decorable for security, etc.

```
class MyController {
  def myService

  def theAction = {
    [ model: myService.doStuff(params) ]
  }
}

class MyService {
  static transactional = false

  def doStuff(params) {
    ...
  }
}
```


- Grails integrates the Quartz Enterprise Job Scheduler
 - for scheduled executions
 - can have services/datasources, etc. injected

```
class MyJob
{
  def startDelay = 60000
  def timeout = 1000

  def group = "MyGroup"

  def execute() {
    print "MyJob is running!"
  }
}

class MyOtherJob {
  def myService // injected

  def cronExpression = "0 0 6 * * ?"

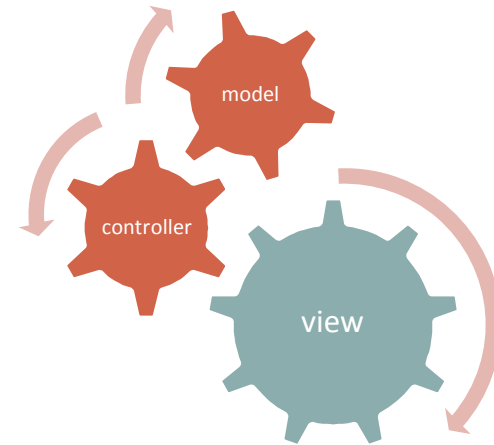
  def group = "MyGroup"

  def execute() {
    print "MyOtherJob is running!"
    myService.doOtherJobStuff()
  }
}
```



- Groovy Server Pages

- no surprises here for JEE people
 - A Good Thing...



- “...essentially one big GString”

- all Groovy’s “\${...}” goodness applies
 - no more fragmented EL syntaxes
- also supports scriptlets and taglibs
- also a sophisticated template/layout
 - courtesy of the underlying sitemesh framework

```
<html>
  <body>
    <h1>Welcome to Grails</h1>
    <p>
      <g:if test="${Calendar.getInstance().get(Calendar.DAY_OF_WEEK) in 1..2}">
        It's a weekend! Why are you working?
      </g:if>
      <g:else>
        <% out << /it's nice to meet you!/ %>
      </g:else>
    </p>
  </body>
</html>
```

- some useful functionality

- `<g:if...<g:elseif...<g:else`
- `<g:while, <g:each`
- `<g:grep, <g:collect, <g:findAll`

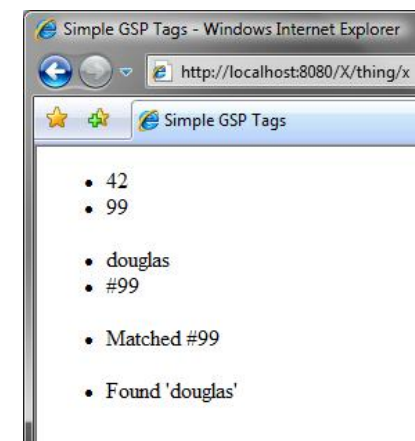
```
class ThingController {

    def index = { redirect(action: x) }

    def x = {
        [model: [new Thing(name:'douglas', value: 42),
                 new Thing(name:"#99", value: 99) ]]
    }
}

class Thing {
    String name
    Integer value
}
```

```
<%@ page contentType="text/html; charset=UTF-8" %>
<html>
<head><title>Simple GSP Tags</title></head>
<body>
<ul>
<g:each in="${model}"><li>${it.value}</li></g:each>
</ul>
<ul>
<g:collect in="${model}"
  expr="${it.name}"><li>${it}</li></g:collect>
</ul>
<ul>
<g:grep in="${model.name}"
  filter="~/.../"><li>Matched ${it}</li></g:grep>
</ul>
<ul>
<g:findAll in="${model}" expr="${it.name =~ 'd'}"
  var="found"><li>Found '${found.name}'</li></g:findAll>
</ul>
</body>
</html>
```



```

class FormController {
    static defaultAction = 'form'

    def form = {}

    def process = {
        User user, Token token ->

        if (user.hasErrors() ||
            token.hasErrors())
            render view: 'form',
                    model: [user: user, token: token]
        else
            render view: 'successView',
                    model: [user: user, token: token]
        }
    }

class Token {
    Integer t

    static constraints = {
        t nullable: false, range: 1..42
    }
}

class User {
    String name
    String password

    static constraints = {
        name blank: false, size: 6..12
        password blank: false, size: 8..80
    }
}

```

```

<%@ page contentType="text/html; charset=UTF-8" %>
<html>
<head><title>A Simple GSP Form View</title></head>
<body>
    <h1>Form View</h1>

    <g:hasErrors bean="${user}"> User! </g:hasErrors>
    <g:hasErrors bean="${token}"> Token! </g:hasErrors>
    <g:form controller="form" action="process">
        Name: <g:textField name="name"
                        value="${user?.name}" />
        Password: <g:passwordField
                name="password" value="${user?.password}" />
        <br />
        Token: <g:textField name="t"
                        value="${token?.t}" />
        <br />
        <g:submitButton name="Submit" value="Submit" />
    </g:form>

</body>
</html>

```

```

<%@ page contentType="text/html; charset=UTF-8" %>
<html>
<head><title>A Simple GSP Success View</title></head>
<body>
    Submitted:<br/>
    Name: ${user?.name}<br/>
    Pass: ${user?.password}<br/>
    Token: ${token?.t}
</body>
</html>

```

- tabulation/pagination of large datasets

```

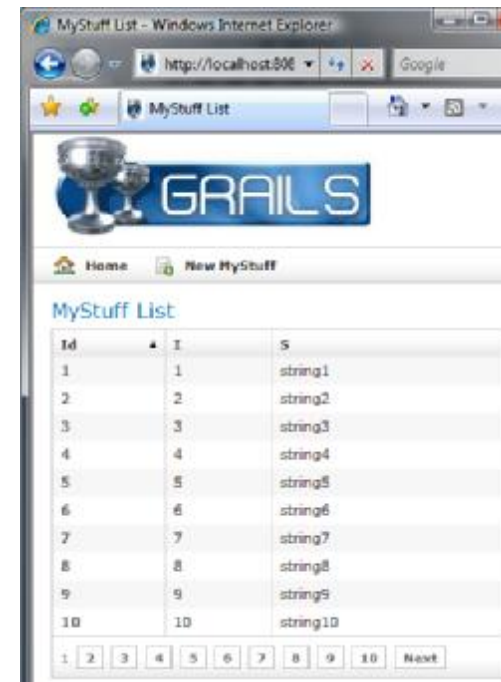
...
<div class="list">
  <table>
    <thead>
      <tr>
        <g:sortableColumn property="id" title="Id" />
        <g:sortableColumn property="i" title="I" />
        <g:sortableColumn property="s" title="S" />
      </tr>
    </thead>
    <tbody>
      <g:each in="${myStuffList}" status="i" var="myStuff">
        <tr class="${(i % 2) == 0 ? 'odd' : 'even'}">
          <td><g:link action="show"
            id="${myStuff.id}">${fieldValue(bean:myStuff, field:'id')}
          </g:link></td>
          <td>${fieldValue(bean:myStuff, field:'i')}</td>
          <td>${fieldValue(bean:myStuff, field:'s')}</td>
        </tr>
      </g:each>
    </tbody>
  </table>
</div>
<div class="paginateButtons">
  <g:paginate total="${MyStuff.count()}" />
</div>
...

```

```

class MyStuffController {
  def list = {
    if(!params.max) params.max = 10
    [ myStuffList: MyStuff.list( params ) ]
  }
}

```



- enables DRY
 - simplify the page designer's life
 - reload on change
 - just like the rest of Grails
 - POGO useable from controller/services as well
 - easy testing!
 - have been too complex, so people don't bother
 - grails changes that...!

```
C:\Windows\system32\cmd.exe
C:\DEVELOPMENT>grails create-taglib Bobz
Welcome to Grails 1.0.3 - http://grails.org/
Licensed under Apache Standard License 2.0
Grails home is set to C:\DEVELOPMENT\grails-1.0.3

Base Directory: C:\DEVELOPMENT
Note: No plugin scripts found
Running script: C:\DEVELOPMENT\grails-1.0.3\script\CreateTagLib.groovy
Environment set to development
[copy] Copying 1 file to C:\DEVELOPMENT\grails-app\taglib
Created TagLib for Bobz
[copy] Copying 1 file to C:\DEVELOPMENT\test\integration
Created TagLibs for Bobz
C:\DEVELOPMENT>
```

```
class BobzTagLib {

    def formatDate = {attrs, body ->
        def date = attrs.get('date')
        if (!date) {
            date = new Date()
        }

        def format = attrs.get('format')
        if (!format) {
            format = "yyyy-MM-dd HH:mm:ss z"
        }

        out << new java.text.SimpleDateFormat(format).format(date)
    }
}
```

```
<g:dateFormat value="${new Date()}"
              format="dd-MM-yyyy" />

<%= formatDate(value: new Date(),
              format: "dd-MM-yyyy") %>
```

<http://grails.org/Contribute+a+tag>

- give multiple views a 'standard' L&F
 - another bit of DRY goodness

```
class XController {
    def x = { }
}
```

```
<!-- file: views/x/x.gsp -->
<meta name="layout" content="main" />

<title>Welcome to My Application</title>

<content tag="leftMenu">
  <ul>
    <li>one</li>
    <li>two</li>
  </ul>
</content>

<content tag="rightContent">
  <p>Right hand side stuff</p>
</content>
```

```
<!-- file: layouts/main.gsp -->
<html>
  <head>
    <title><g:layoutTitle /></title>
    <link rel="stylesheet" href="{createLinkTo(dir:'css',file:'main.css')}" />
  </head>
  <body>
    <table>
      <tr>
        <td width="25%"><g:pageProperty name="page.leftMenu" /></td>
        <td width="2%">&nbsp;</td>
        <td><g:pageProperty name="page.rightContent" /></td>
      </tr>
    </table>
  </body>
</html>
```

- reusable snippets of markup

```
class XController {  
  def x = { [model: [new Stuff(i:99, s: 'hello'), new Stuff(i:999, s: 'world')] ] }  
}
```

```
class Stuff {  
  Integer i  
  String s  
}
```

```
<!-- file: x.gsp -->  
<html>  
<head>  
  <title>Stuff!</title>  
</head>  
<body>  
<ul>  
  <g:each var="s" in="${model}">  
    <li><g:render template="stuffTemplate" model="[stuff:s]"/></li>  
  </g:each>  
</ul>  
</body>  
</html>
```

```
<!-- file: views/x/_stuffTemplate.gsp -->  
Integer: <em>${stuff.i}</em>; String: <strong>${stuff.s}</strong>
```


- all good web 2.0 apps ‘need’ AJAX...*
- Grails delivers
 - ships with the popular Prototype library
 - with ‘adaptive tags’
 - cater for other frameworks such as Dojo, YUI, richui, openRico and GWT
- supports content negotiation
 - allows JSON, XML, etc.



*so all those bits of JavaScript can mess up in myriad different ways; *thou shalt not JavaScript!*
(don't get me started on CSS positioning...!)

```

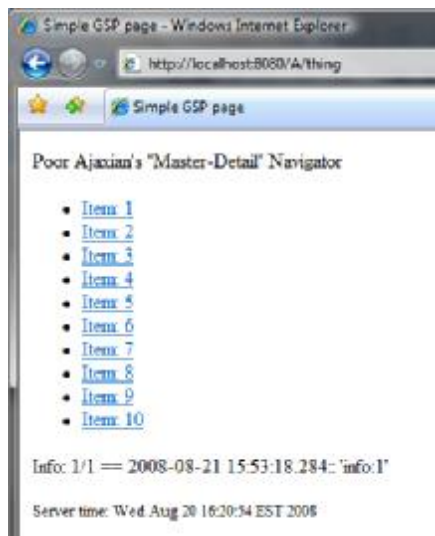
<%@ page contentType="text/html; charset=UTF-8" %>
<html>
<head><title>Simple GSP page</title></head>
<g:javascript library="prototype"/>

<script type="text/javascript">
  new Ajax.PeriodicalUpdater('timeInfo',
    '<g:createLink action="time" />', { frequency: 1});
</script>

<body>
<p>Poor Ajaxian's "Master-Detail" Navigator</p>
<ul>
  <g:each in="${model}" var="thing">
    <li><g:remoteLink action="drillDown"
      update="moreInfo"
      id="${thing}">
      Item: ${thing}</g:remoteLink></li>
  </g:each>
</ul>
<p>Info: <span id="moreInfo">&nbsp;&nbsp;&nbsp;</span></p>

<p style="font-size:x-small;">
  Server time:
  <span id="timeInfo">&nbsp;&nbsp;&nbsp;</span></p>
</body>
</html>

```



```

class ThingController {

  def index = {
    [model: Thing.withCriteria {
      projections { property 'value' }
    }]
  }

  // ajax
  def drillDown = {
    log.error "Got: ${params.id}"
    def t = Thing.get(params.id)
    render
      "${t.id}/${t.value} == ${t.date}:: '${t.info}'"
  }

  // ajax; called periodically
  def time = {
    render new Date().toString()
  }
}

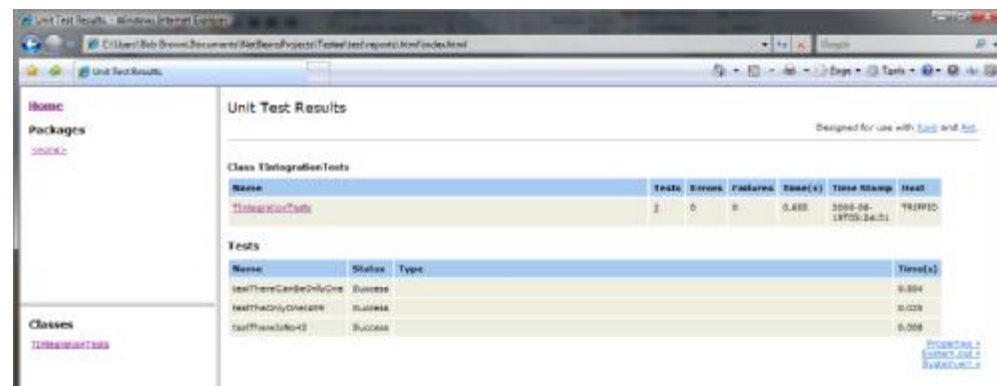
class Thing {
  Integer value
  Date date
  String info
}

```

- even more vital for dynamic languages
- Groovy contains support for basic stubbing and mocking
 - MockFor and StubFor classes
- encourages testing
 - dependency injection makes life easier
 - generates test harnesses alongside ‘proper’ code
- some ‘groovy’ tools out there...
 - Canoo WebTest
 - SoapUI
 - Hudson
 - TestNG (“Test’N’Groove”)
 - Cobertura
 - etc.

- **GroovyTestCase**
 - follows groovy's KISS path
 - enhanced JUnit facilities
 - also avoids the JUnit restriction of requiring all test* methods to be void

```
class TIntegrationTests extends GroovyTestCase {  
  
    void setUp() {  
        new T(i:99, s:'this is a new T').save()  
    }  
  
    void testThereCanBeOnlyOne() {  
        assert T.count() == 1  
    }  
  
    void testTheOnlyOneIs99() {  
        assert T.findByI(99).i == 99  
    }  
  
    void testThereIsNo42() {  
        shouldFail(NullPointerException) {  
            T.findByI(42).delete()  
        }  
    }  
}
```



Unit Test Results

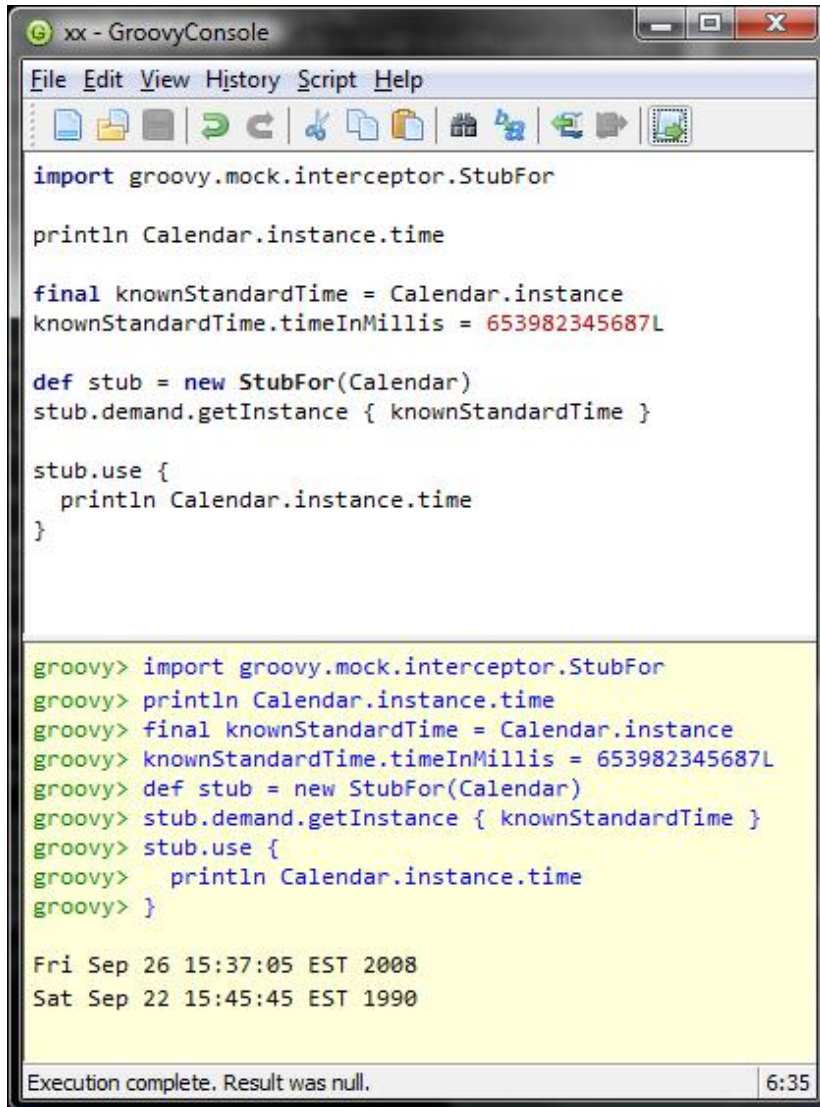
Class TIntegrationTests

Name	Tests	Errors	Failures	Time(s)	Time Stamp	Next
TIntegrationTests	1	0	0	0.489	2006-06-19T05:34:31	

Tests

Name	Status	Type	Time(s)
testThereCanBeOnlyOne	Success		0.304
testTheOnlyOneIs99	Success		0.028
testThereIsNo42	Success		0.008

- enable a Class Under Test (CUT) to *run in isolation* and lets you make assertions about state changes of the CUT



```
xx - GroovyConsole
File Edit View History Script Help
import groovy.mock.interceptor.StubFor
println Calendar.instance.time
final knownStandardTime = Calendar.instance
knownStandardTime.timeInMillis = 653982345687L
def stub = new StubFor(Calendar)
stub.demand.getInstance { knownStandardTime }
stub.use {
  println Calendar.instance.time
}
groovy> import groovy.mock.interceptor.StubFor
groovy> println Calendar.instance.time
groovy> final knownStandardTime = Calendar.instance
groovy> knownStandardTime.timeInMillis = 653982345687L
groovy> def stub = new StubFor(Calendar)
groovy> stub.demand.getInstance { knownStandardTime }
groovy> stub.use {
groovy>   println Calendar.instance.time
groovy> }
Fri Sep 26 15:37:05 EST 2008
Sat Sep 22 15:45:45 EST 1990
Execution complete. Result was null. 6:35
```

“One of the most amazing things I’ve ever seen done in software, something only a geek can appreciate...”

How often do you see software that acts that much like real magic? You’ve got to admit, that is beyond your expectations, really clean, and very cool.”

—<http://leegrey.com/hmm/?m=200708>

- testing to ensure that the *protocol for use of a class is correct*

```
import groovy.mock.interceptor.MockFor

class GoodStuffClass {
    def open() { }
    def process() { }
    def close() { }
}

class GoodStuffTester {
    static main(args) {
        def mocked = new MockFor(GoodStuffClass)

        mocked.demand.open(1..1) { println 'opening'; true }
        mocked.demand.process(1..Integer.MAX_VALUE) { println 'processing'; true }
        mocked.demand.close(1..1) { println 'closing'; true }

        mocked.use {
            def systemUnderTest = new GoodStuffClass()
            systemUnderTest.open()
            (1..5).each { systemUnderTest.process() }
            // forgotten: systemUnderTest.close()
        }
    }
}
```

“...you can test even a rotten design...”

—p.525, GINA



```
Run ▶ GoodStuffTester
"C:\Program Files\Java\jdk1.6.0_07\bin\java" -Didea.launcher.port=7537 "-Didea.launcher.bin.path=C:\Program Files\JetBrains\IntelliJ IDEA
opening
processing
processing
processing
processing
processing
Exception in thread "main" junit.framework.AssertionFailedError: verify[2]: expected 1..1 call(s) to 'close' but was called 0 time(s).
    at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
    at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:99)
```

- very nice plugin for functional testing
 - also has a firefox plugin for recording

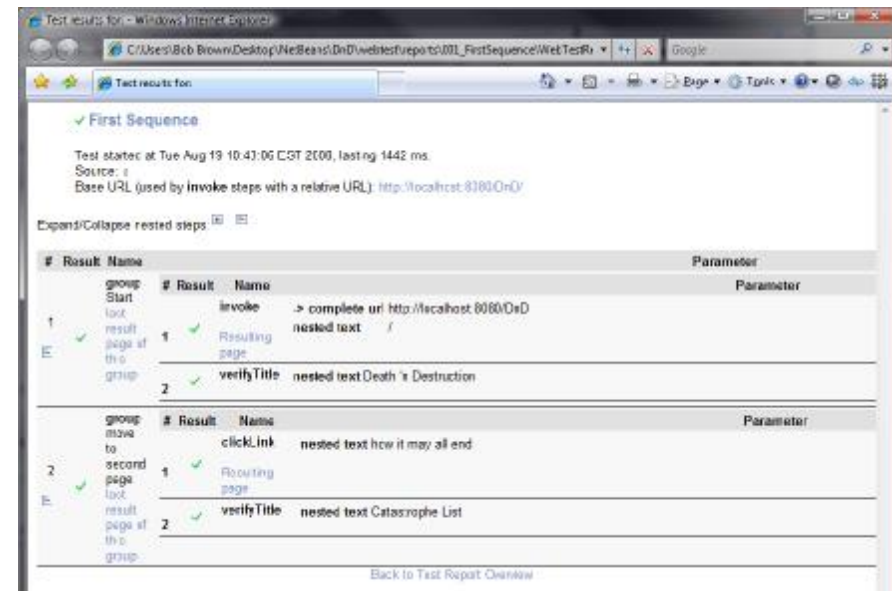
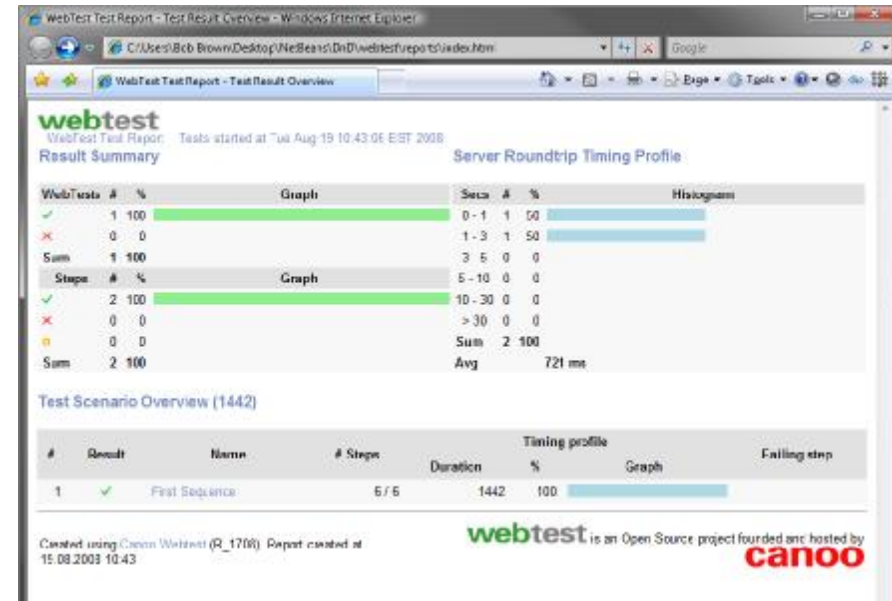
```
class WTest extends grails.util.WebTest {

    void suite() {
        testFirstSequence()
    }

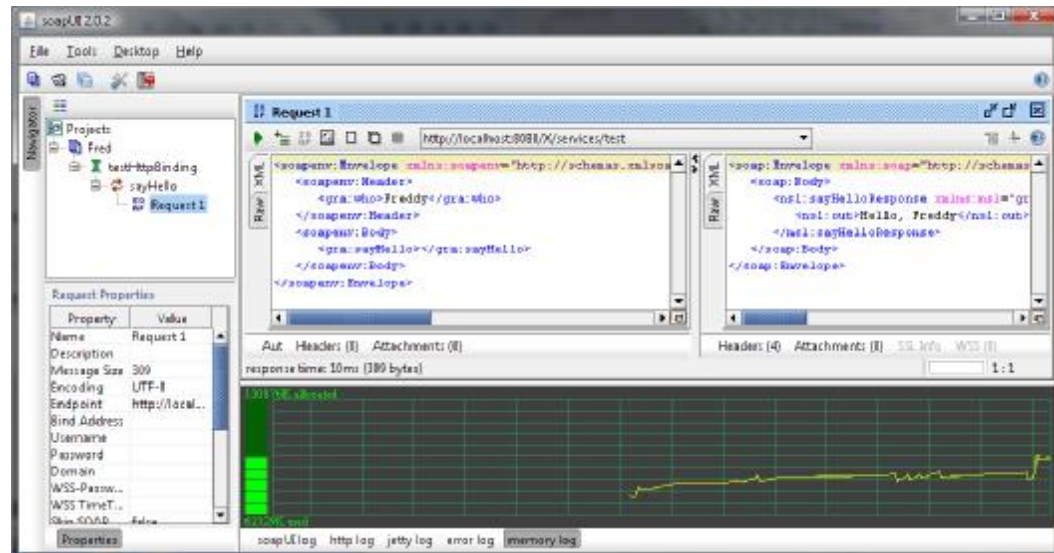
    def testFirstSequence() {
        webtest('First Sequence') {

            group(description: 'Start') {
                invoke '/'
                verifyTitle "Death 'n Destruction"
            }

            group(description: 'Move to second page') {
                clickLink 'how it may all end'
                verifyTitle 'Catastrophe List'
            }
        }
    }
}
```



- **painless to expose and use remote services**
 - xfire plugin
 - Grails Remoting plugin
 - rmi, hessian, burlap, httpinvoker



```
class MinimalService {
    static expose=[ 'xfire' ]

    String sayHello(String who) {
        "Hello, ${who}"
    }
}
```

```
import javax.jws.*

@WebService(targetNamespace = "grails.test")
class FancyService {
    static expose=[ 'xfire' ]

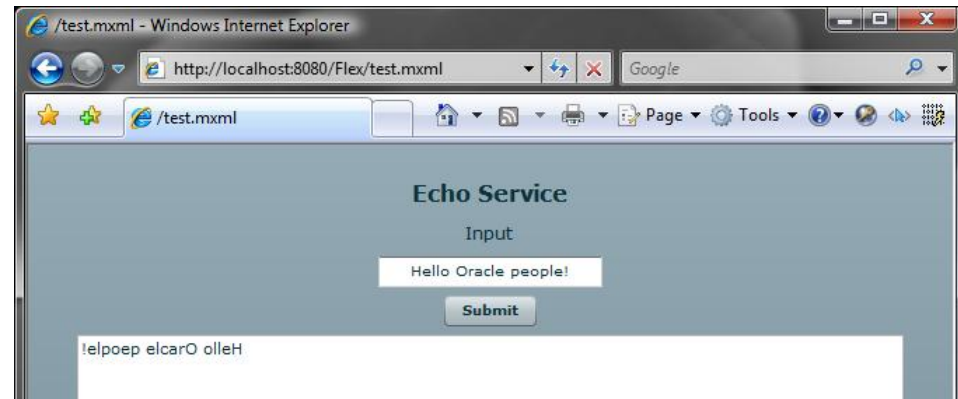
    @WebResult(name="helloReturn")
    String sayHello(@WebParam(name="who", header=true) String who) {
        "Hello, ${who}"
    }
}
```


Adobe Flex 3

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
  xmlns:mx="http://www.adobe.com/2006/mxml">

  <mx:RemoteObject
    id="helloService"
    destination="helloService" />

  <mx:Text text="Echo Service"
    fontSize="16" fontWeight="bold"/>
  <mx:Text text="Input" fontSize="12"/>
  <mx:TextInput id="lookupPsg"/>
  <mx:Button
    label="Submit"
    click="helloService.hello(lookupPsg.text)"/>
  <mx:TextArea height="32" width="320"
    text="{helloService.hello.lastResult}"
    editable="false" fontSize="11"/>
</mx:Application>
```



```
class HelloService {
  static expose=
    [ 'flex-remoting' ]

  def hello(String echo) {
    return echo.reverse()
  }
}
```

“It's a trivial app but it showed just how incredibly seamless the integration is. It uses the Web Tier Flex compiler for on-demand MXML compilation and it automatically manages the destinations for you.”

—http://corfield.org/blog/index.cfm/do/blog.entry/entry/Grails_and_Flex

“Having no clue about Flex, a little understandig of Web Services and some minor understanding of Grails, it took me no more than 3 hours, including googling for web service usage in Flex and hunting of really stupid beginners bug.

This shows the real muscles of Grails and its really awesome plugin ecosystem.”

— <http://n1ceone.blogspot.com/2008/07/grails-flex-soap.html>

• Not ready yet...but a neatly packaged idea

```

class HolidayRequestProcess {
public String requesterName
public String requesterMail
public String message
public String resolution
public String logMessage
public Boolean isApproved
public Date holidaysStart
public Date holidaysEnd

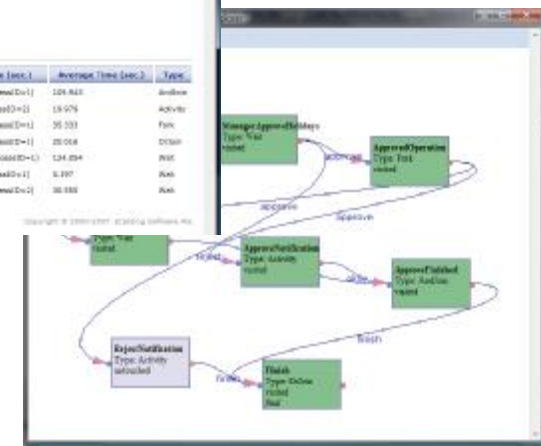
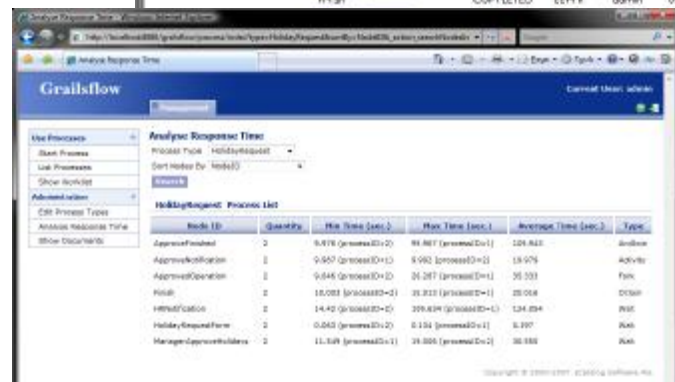
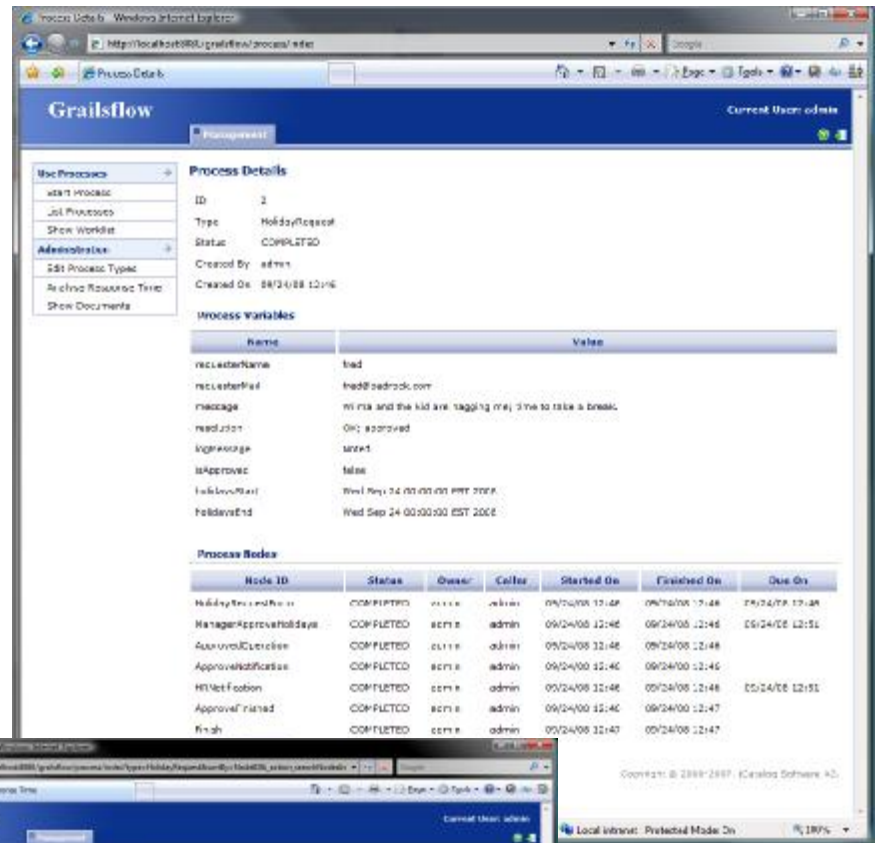
def HolidayRequestProcess = {

    HolidayRequestFormWait(isStart: true) {
        variable(requesterMail: ConstantUtils.WRITE_READ,
            requesterName: ConstantUtils.WRITE_READ,
            holidaysStart: ConstantUtils.WRITE_READ,
            holidaysEnd: ConstantUtils.WRITE_READ,
            message: ConstantUtils.WRITE_READ)
        action {
            SendMail(mailFrom:"requesterMail",
                mailTo: "manager@test.com",
                subject:"Holiday Request",
                message: "message")
        }
        on("submit").to(["ManagerApproveHolidays"])
    }

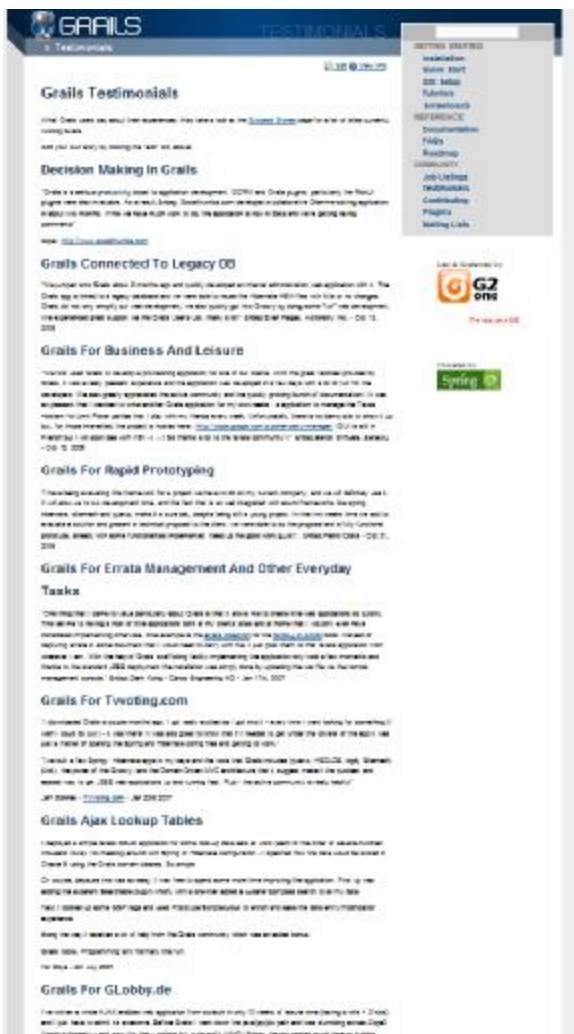
    ...

    ApprovedOperationFork() {
        action {
            AssignTo(var:"isApproved") {
                SetBooleanValue(value: "true")
            }
            return approve()
        }
        on("approve").to(["HRNotification", "ApproveNotification"])
    }

    ...
}
    
```



- <http://www.grails.org/Testimonials>
- <http://www.grails.org/Success+Stories>
- IntelliJIDEA, http://www.jetbrains.com/idea/features/groovy_grails.html
- Netbeans, Eclipse IDEs
- Gravl, <http://code.google.com/p/gravl/>
- jmesa table renderer, <http://code.google.com/p/jmesa/>



- <http://www.grails.org>
- <http://www.oracle.com/technology/pub/articles/oak-grails.html>
- Getting Started with Grails, <http://www.infoq.com/minibooks/grails>
- Bash completion for Grails, http://www.jroller.com/oburn/entry/bash_completion_for_grails
- <http://www.oracle.com/technology/pub/articles/grall-grails.html>
- A guide to mapping databases in Grails with GORM, <http://www.thecoderscorner.com/articleGen/show/8?page=1>
- SAP Composition on Grails (login required), <https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/e09aaf88-2d4f-2a10-ab9c-f49181d7c87c>
- LinkedIn, <http://linkedin.com>, <http://blog.linkedin.com/blog/2008/06/grails-at-linke.html>
- Grails eXchange, <http://grails-exchange.com/>
- Grails in Action, <http://www.manning.com/gsmith/>

