
# Grails – In Depth

Graeme Rocher, CTO – G2One Inc

TS-5764

Java

Sun microsystems

Learn how to dramatically simplify web development on the Java™ platform with Groovy & Grails

GOAL

# Your Speaker

- Graeme Rocher
  - Project Lead of Grails
  - CTO of G2One Inc – The Groovy/Grails Company
  - Member of JSR-241 Expert Group
  - Author of “The Definitive Guide to Grails”


**Graeme Rocher**
**Chief Technology Officer**

Phone
**+44(0)7747612883**

Email
**graeme@g2one.com**

Website
**http://www.g2one.com**

# Agenda

- Introduction to Groovy & Grails
- Why Groovy?
- Getting Started
- Simplified ORM
- The Web Layer
- Web Services
- Easy Ajax

# What is Groovy?

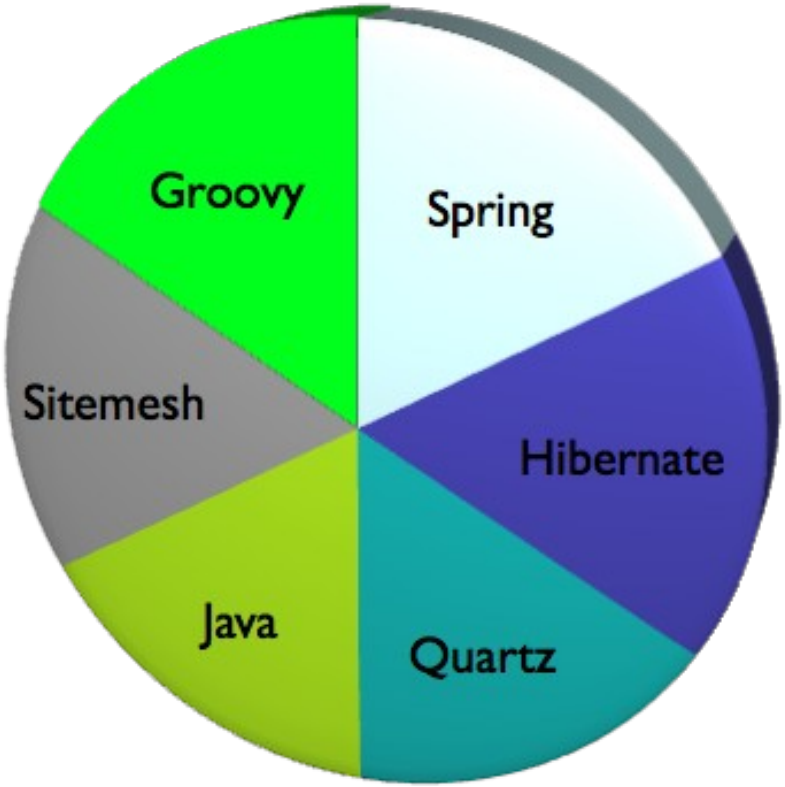
- Groovy is a dynamic language for the Java Virtual Machine (JVM™)
- Takes inspiration from Small Talk, Python and Ruby
- Integrates with the Java language and platform at every level

# What is Grails?

- A Web **platform** that implements the full stack from build system down to ORM layer
- Leverages existing technologies like Spring, Hibernate, Quartz etc. avoiding re-inventing the wheel
- Features and extensible plug-in system and an environment for runtime configuration built on Spring

# Solid Foundations

- Spring
- Hibernate
- Groovy
- Quartz
- SiteMesh
- Jetty
- Apache Ant
- Java Technology



# Why Groovy & Grails?

- **All the power** of frameworks from other platforms like Ruby on Rails, Django and TurboGears
- Built from the ground up on **Java technologies**.
  - Design for Java technology by Java technology developers
- Leverage **existing tools and technologies**
  - Distributed caches, replicating technologies etc.
  - Profiling, monitoring and debugging tools
  - IDEs
- Reduce **context switching**

# Getting Started


- Download from http://grails.org/Download
- Extract zip to disk
- Set `GRAILS_HOME` variable to location on disk
- Add `$GRAILS_HOME/bin` to your environment variables


- Download from http://groovy.codehaus.org
- Extract zip to disk
- Set `GROOVY_HOME` variable to location on disk
- Add `$GROOVY_HOME/bin` to your environment variables

# Grails – The Basics

- Creating and Running

```
> grails create-app gtunes
> cd gtunes
> grails run-app
```

- Grails will load by default with an in-memory HSQLDB
- The default container is an embedded Jetty Container
- Grails is configured for automatic hot-reloading at runtime

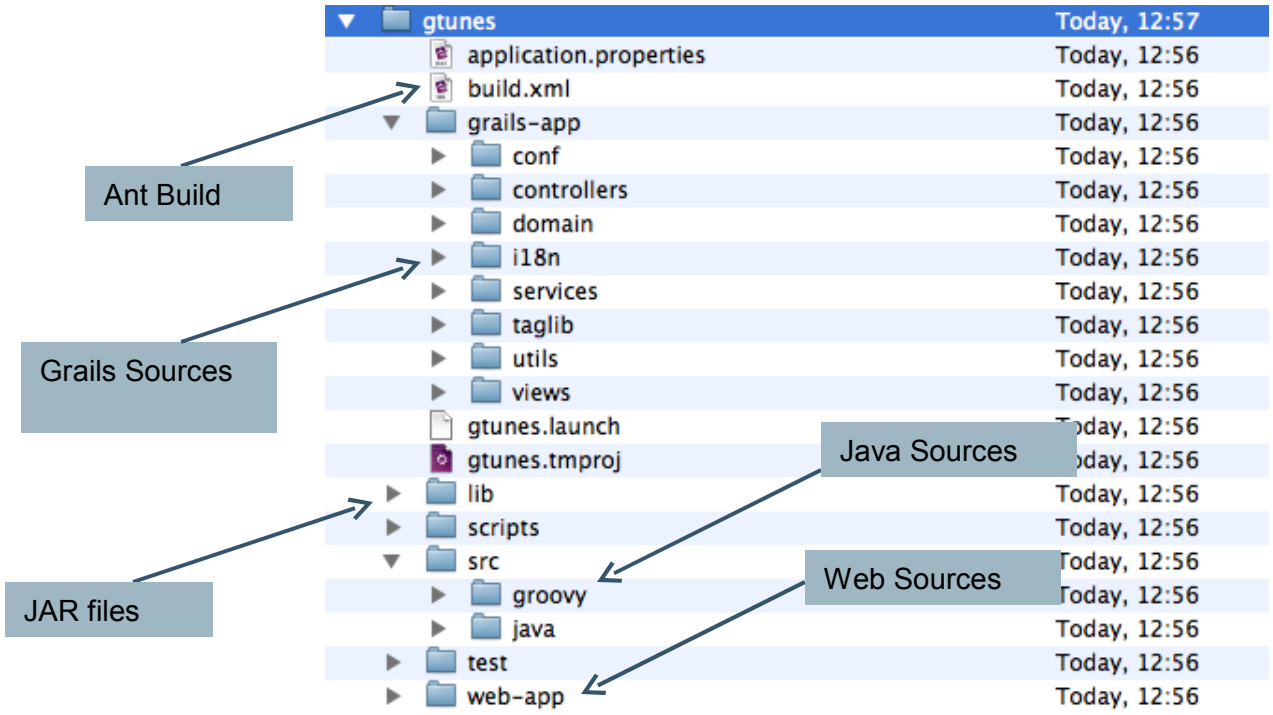
- WAR'ing and Deploying

```
> grails run-war // or
> grails war
```

- Grails deploys as a standard WAR file deployable onto modern Java EE platform containers
- In production mode Grails is fully compiled byte code

# Getting Started with Grails
## Creating and Running

# A Grails Project



# Grails Headline Features

- GORM - ORM Layer built on Hibernate
- Rich conversation support with Web Flow


- Domain Specific Languages
  - Validation Rules
  - ORM Mapping
  - URL Mapping


- Vibrant Plug-in community
  - http://grails.org/Plugins
  - Over 40 user contributed plug-ins
  - Everything from web services to RIA with Flex

# Grails – The Platform

- Java 2 Platform, Enterprise Edition (J2EE™ platform) was too complex. Spring and Hibernate reduced that complexity and became very successful
- Grails represents the **next level of abstraction** for the Java EE platform
- Built on Spring and Hibernate, but dramatically simplifying their usage

# Easy ORM with GORM

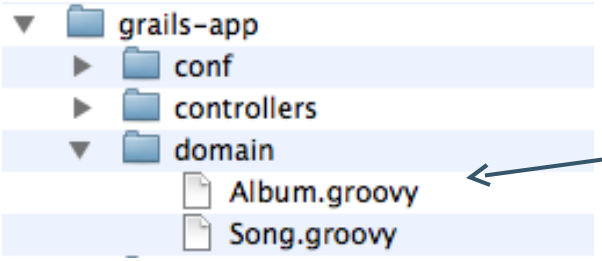
```
class Album {
  String title
  String artist
  Date releaseDate
  static hasMany = [songs:Song]
}
class Song {
    String title
    Double duration
}
```

table - album

| id | title | artist | release_date |
|---|---|---|---|
| | | | |

table - song

| id | title | duration | album_id |
|---|---|---|---|
| | | | |



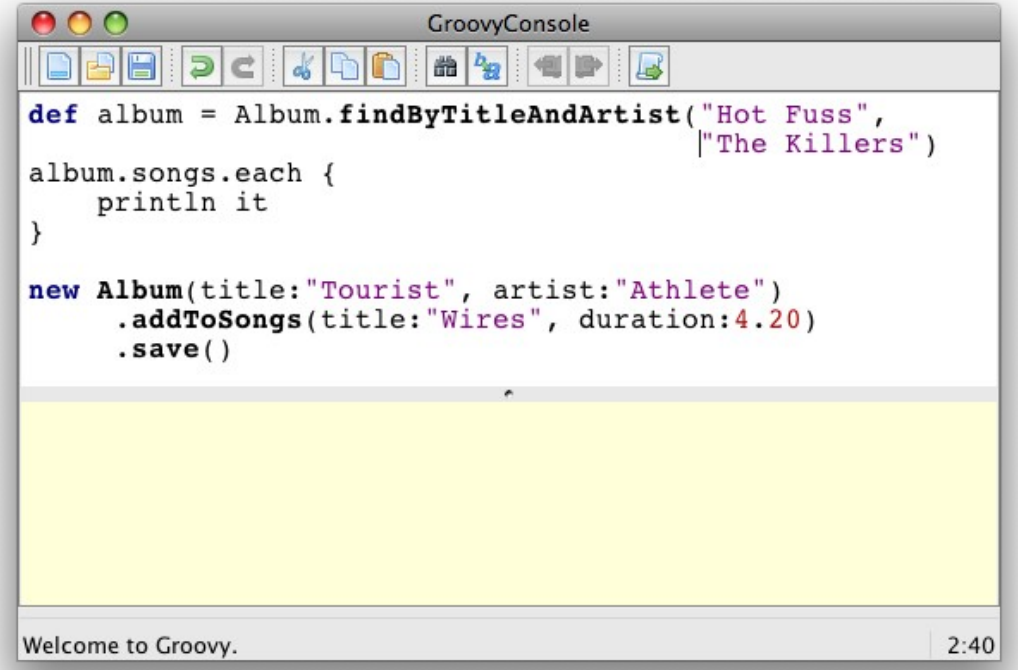

GORM classes, also known as domain classes, go in the domain directory

# GORM in Action

## Querying and Persistence with GORM

# The Grails Console

Activate with: `grails console`



# Persistence Methods

- GORM classes automatically provided with CRUD capabilities:
  - save
  - get
  - delete

```
new Album(title:"Tourist",
          artist:"Athlete")
   .addToSongs(title:"Wires",
               duration:4.20)
         .save()

def album = Album.get(1)

album.title ="Hot Fuss"
album.save()

album.delete()
```

# Dynamic Finders & Criteria

```
def albums = Album.list()

def recentAlbums =
       Album.findAllByReleaseDateGreaterThan(new Date()-7)

def albumsStartingWithA =
        Album.findAllByTitleLike("A%")

def albumsWithSongsAboutSummer =
     Album.withCriteria {
         songs {
            like("title", "%Summmer%")
         }
     }
```

List all records

Form method expressions

Use “like” queries

Construct criteria on the fly to query associations

# GORM Features

- Dynamic finder and persistence methods
- Criteria with a Groovy builder
- Object-relational Mapping DSL
  - Caching
  - Legacy mapping
  - Locking strategy (optimistic/pessimistic)
- Built on Hibernate

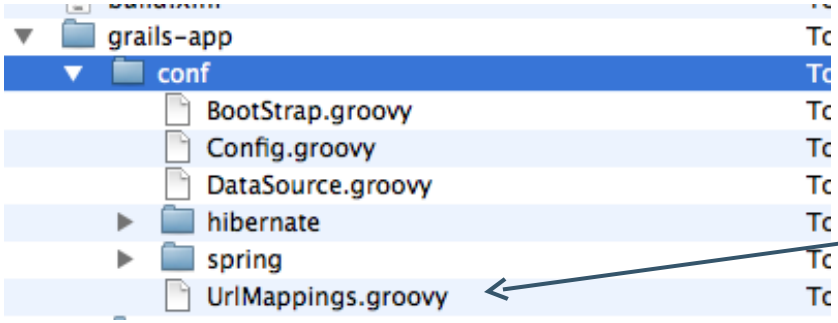
# Easy MVC with Grails

- Comprehensive MVC framework built on Spring
  - Command objects
  - Data binding
  - Zero configuration
  - URL Mappings
  - Groovy Server Pages (GSP)
  - Tag libraries
  - CRUD (Scaffolding)
  - Internationalization (i18n)

# Mapping URLs to Controllers

- Default URL Mapping:

```
"/$controller/$action?/$id?"()
```




UrlMappings file defines the default mapping, but is completely configurable

- Maps patterns like `/album/list` to `list` action of `AlbumController`

# Controllers in Action
## Easy MVC with Grails

# Grails Controllers

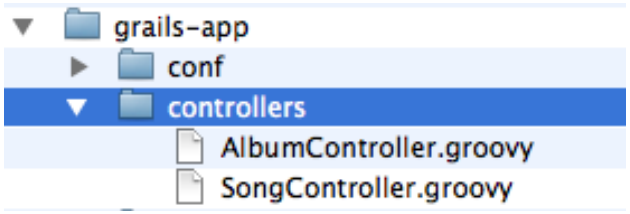
```
class AlbumController {

  def search = {
     def albums= Album.findByTitleLike("%${params.q}%")

     [albumResults:albums]
  }
}
```

Actions are properties assigned a block or closure

Return a model as a map




Controllers go in the "controllers" directory

# Trivial Tag Libraries

## Tag Code

```
def cache = {attrs,body->
  def content=
      store[attrs.id]
  if(!content)
     content = body()
     store[attrs.id]=
           content
  }
  out << content
}
```

Read tag attributes

Invoke the body

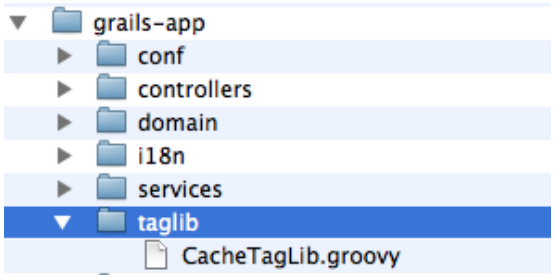
Write to the response

## View

```
<g:cache id="test">
  expensive lazy loaded
  content goes here
</g:cache>
```

Pass tag attributes

Tag name matches closure name



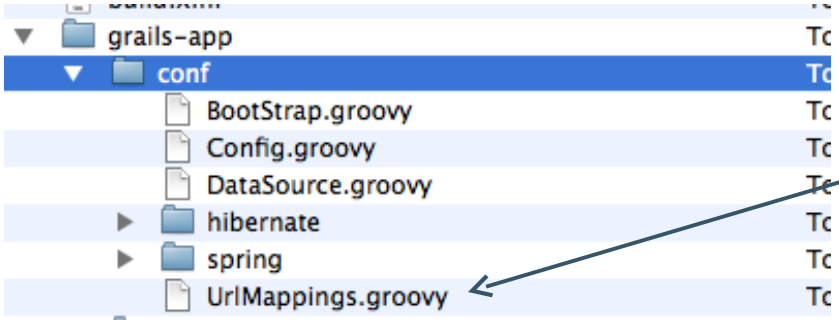

Tag libraries go in the taglib directory

# Web Services

- Grails features built in support for REST
  - REST = Representational State Transfer
- Optional support for SOAP via plug-ins (another example of not re-inventing the wheel!)
  - Xfire – http://grails.org/XFire+plugin
  - Axis 2 – http://grails.org/Apache+Axis2+Plugin

# RESTful URL Mappings

```
"/$controller/$id?" {
    action = [GET:"show",
              POST:"update",
              PUT:"save"]
}
```




Defined in URLMappings.groovy file

- Maps to actions based on HTTP method

# RESTful Marshalling
## Easy Web Services with Grails

# RESTful Marshalling

- To Marshall:

```
import grails.converters.*
…
def album = Album.get(params.id)
render album as XML
```

send XML to the response

- To Unmarshall:

```
def album = new Album(params.album)
```

Read XML from the request when the root node is <album>

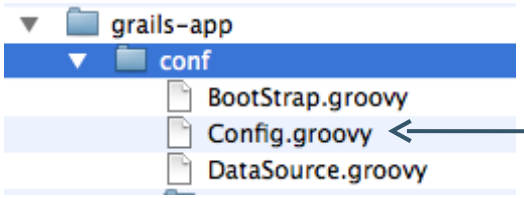
# Content Negotiation

- Process different request types based on `ACCEPT` / `CONTENT_TYPE` header:

```
def a = Album.get(params.id)
withFormat {
    html album:a
    xml { render a as XML }
    json { render a as JSON }
}
```

Request format obtained from ACCEPT header

Different response sent based on content type




Supported content types configurable in Config.groovy

# Easy Ajax with Adaptive Tags

- Prototype enabled by default
  - Yahoo UI and Dojo available as plug-ins!

```
<g:remoteLink action="showAlbum" />
<g:formRemote action="updateAlbum" />
<g:remoteField action="changeTitle"
               update="titleDiv"
               name="title"
               value="${album?.title}"/>
```

# Ajax Plug-ins a Plenty

- Echo 2: http://grails.org/Echo2+Plugin
- GWT: http://grails.org/GWT+Plugin
- OpenLaszlo: http://grails.org/OpenLaszlo+plugin
- ZK: http://grails.org/ZK+Plugin
- Flex: http://grails.org/Flex+Plugin
- ULC: http://grails.org/ULC+Plugin
- Yahoo UI: http://grails.org/YUI+Plugin
- DWR: http://grails.org/DWR+Plugin

# Summary

- Grails is not just a web framework, but a web platform
- Grails is the next generation abstraction of the Java EE platform built on Spring and Hibernate
- Groovy and Grails integrate tightly with Java technology, the Java Virtual Machine at every level providing an easy migration path
- Grails == Simplifying Java EE platform on the web

# For More Information

- Grails
  - TS-5793 – Groovy and Grails: Changing the Landscape of Java Platform, Enterprise Edition (Java EE Platform) Patterns, TS-6457 - Choosing Your Java Technology-Based Web Framework: A Comparison
  - Web Site: http://grails.org
  - User Guide: http://grails.org/doc/1.0.x/
- Groovy
  - Sessions, Panel Discussions, BOFs: PAN-5435, TS-5815, TS-6050, TS-5274, BOF-5102, BOF-5110, TS-5693, BOF-5101
  - http://groovy.codehaus.org

# THANK YOU

Grails – in Depth
Graeme Rocher

TS-5764

Java

Sun microsystems