



Road to Grails 3.0

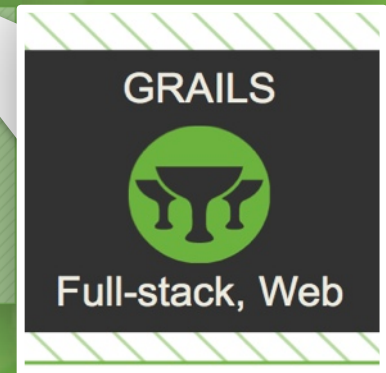
By Graeme Rocher

Agenda

- What's new in Grails 2.3?
- What's planned for Grails 3.0?
- Summary / Q & A



SPRING IO EXECUTION: Grails



What's new in Grails 2.3?



GRAILS

- Forked Execution and Test Daemon
- Dependency Resolution with Aether
- Complete Server-Side REST Support
- Async Programmings APIs
- New Data Binding APIs

2.3

What's new in Grails 2.3?



GRAILS

- XSS Prevention
- Hibernate 4 Support (via Plugin)
- RESTful Scaffolding v2.0
- Controller Namespaces
- ... and much, much more

2.3

GRAILS Demo

Aether & Forked
Execution



Aether & Dependency Management

- Aether == Same dependency resolution engine as Maven
 - No snapshot troubles
 - Same local repo
 - No Ivy bugs
- New dependency-report command to visualize graph

```
build - Dependencies for the build system only (total: 23)
+---- xalan:serializer:2.7.1
+---- org.grails:grails-bootstrap:2.3.0.RC2
|   \--- jline:jline:1.0
|   \--- org.fusesource.jansi:jansi:1.2.1
|   \--- org.slf4j:jcl-over-slf4j:1.7.5
|   \--- org.slf4j:slf4j-api:1.7.5
|   \--- org.apache.ant:ant:1.8.4
|   \--- org.apache.ant:ant-launcher:1.8.4
|   \--- org.apache.ant:ant-trax:1.7.1
|   \--- net.java.dev.jna:jna:3.2.3
|   \--- org.apache.ant:ant-junit:1.8.4
|   \--- org.apache.ivy:ivy:2.3.0
|   \--- org.codehaus.gant:gant-groovy:1.8:1.9.6
+---- org.grails:grails-scripts:2.3.0.RC2
```

Forked Execution

- Build and test / runtime split into separate JVMs
- Daemon used to speed test execution
- Correctly isolates classpaths
- test-app can't break run-app!



GRAILS Demo

REST Support



REST: URL Mappings

- RESTful Mappings
 - Single or Multiple resources
 - Versioning
 - Nested resources
- New url-mappings-report command to mappings

```
url-mappings-report
| URL Mappings Configured for Application
| -----
|
| 10
|
| Dynamic Mappings
| * 11 | /${controller}/${action}?/${id}?(.${f
| * 12 | /
| * 13 | ERROR: 500
|
| Controller: book
| GET | /books
| GET | /books/${id}
```

REST: @Resource Transformation

- Automatically exposes a domain class as a REST resource
- Configurable formats
- Configuration HTTP methods

REST: @Resource Transformation

- Automatically exposes a domain class as a REST resource
- Configurable formats
- Configuration HTTP methods

```
import grails.rest.*  
  
@Resource  
class Person {  
    String name  
}
```

REST: The respond method

- Sends the most appropriate response for the given object
- For HTML calculates a model by convention and delegates to view

REST: The respond method

- Sends the most appropriate response for the given object
- For HTML calculates a model by convention and delegates to view

```
class PersonController
{
  def index() {
    respond Book.list()
  }
}
```

REST: HAL, JSON, XML Renderers

- Rendering customizable in `resources.groovy`
- HAL, Atom, `Vnd.Error` renders included
- Flexible and complete

REST: HAL, JSON, XML Renderers

- Rendering customizable in resources.groovy
- HAL, Atom, Vnd.Error renders included
- Flexible and complete

```
import grails.rest.render.hal.*  
beans = {  
    halBookRenderer(HalJsonRenderer,  
                    rest.test.Book)  
}
```


REST: HAL, JSON, XML Renderers

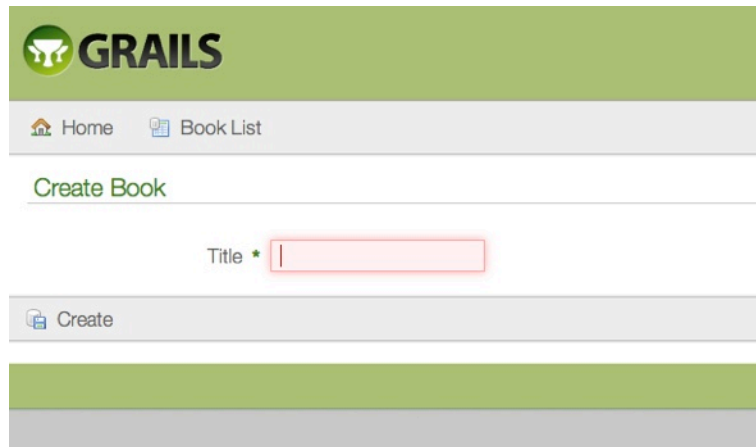
- Rendering customizable in resources.groovy
- HAL, Atom, Vnd.Error renders included
- Flexible and complete

```
import grails.rest.render.hal.*
beans = {
    halBookRenderer(HalJsonRenderer,
                    rest.test.Book)
}
```

```
import grails.rest.render.xml.*
beans = {
    bookRenderer(XmlRenderer,
                  Book) {
        includes = ['title']
    }
}
```

REST: Scaffolding 2.0

- Scaffolding support now a plugin
 - <http://grails.org/plugin/scaffolding>
- Generated controllers now RESTful
- New `generate-async-controller` command for Async support



The screenshot shows a web application interface for a Grails project. At the top is a green header with the Grails logo and the word 'GRAILS'. Below this is a navigation bar with links for 'Home' and 'Book List'. The main content area is titled 'Create Book' in green. It features a form with a label 'Title *' and a corresponding text input field. Below the form is a 'Create' button with a document icon. The page has a green footer bar at the bottom.

REST: Misc Goodies

- `grails.rest.RestController`
 - Controller base class for inheriting RESTful actions
- The `respond` method
 - Chooses most appropriate response format
- Versioning support via URL mapping or Accept-Version header
- Find out more <http://grails.org/doc/2.3.x/guide/webServices.html#REST>

GRAILS Demo

Async



Async: Promise Primitives

- Foundational API for Async programming
- Includes notion of Promise
- Ability to combine / chain promises

Async: Promise Primitives

- Foundational API for Async programming
- Includes notion of Promise
- Ability to combine / chain promises

```
import static
grails.async.Promises.*

def p1 = task { 2 * 2 }
def p2 = task { 4 * 4 }
def p3 = task { 8 * 8 }
assert [ 4, 16, 64 ] ==
waitAll(p1, p2, p3)
```

Async: @DelegateAsync Transform

- Transform any synchronous API into an asynchronous one
- Takes each method and returns a promise

Async: @DelegateAsync Transform

- Transform any synchronous API into an asynchronous one
- Takes each method and returns a promise

```
import grails.async.*  
  
class AsyncBookService {  
    @DelegateAsync  
    BookService bookService  
}
```


Async: GORM

- Makes all GORM methods async
- Deals with binding session to background thread
- Works across all datastores (MongoDB, GORM for REST etc.)

Async: GORM

- Makes all GORM methods async
- Deals with binding session to background thread
- Works across all datastores (MongoDB, GORM for REST etc.)

```
def p1 = Person.async.get(1)
def p2 = Person.async.get(2)
def p3 = Person.async.get(3)

def results =
    waitAll(p1, p2, p3)
```

Async: Request Processing

- Handle requests asynchronously
- Uses Servlet 3.0 async under the covers
- Create asynchronous models

Async: Request Processing

- Handle requests asynchronously
- Uses Servlet 3.0 async under the covers
- Create asynchronous models

```
import static
grails.async.Promises.*

class PersonController {
    def index() {
        tasks books: Book.async.list(),
              total: Book.async.count(),
              otherValue: {
                // do hard work
              }
    }
}
```

GRAILS Demo

Data Binding



Data Binder: @BindingFormat

- Per property formatting
- Extensible

Data Binder: @BindingFormat

- Per property formatting
- Extensible

```
class Person {  
    @BindingFormat('MMddyyyy')  
    Date birthDate  
    @BindingFormat('Uppercase')  
    String salutation  
}
```

Data Binder: @BindUsing

- For cases where more flexibility is needed
- Can be applied to class or property
- Takes a closure or a class that implements `BindingHelper`

Data Binder: @BindUsing

- For cases where more flexibility is needed
- Can be applied to class or property
- Takes a closure or a class that implements `BindingHelper`

```
class Person {  
    @BindUsing({  
        obj, source ->  
            source['name']?  
                .toUpperCase()  
    })  
    String salutation  
}
```

Data Binding: XML/JSON Command Objects

- Can now bind XML/JSON data to command objects
- Binding can be customized using the `BindingSource` and `BindingSourceCreator` interfaces

Data Binding: XML/JSON Command Objects

- Can now bind XML/JSON data to command objects
- Binding can be customized using the `BindingSource` and `BindingSourceCreator` interfaces

```
class PersonController
{
    def save(PersonCmd p)
    {
        ...
    }
}
```

Data Binding: Auto Query if Domain Object

- If command object is a domain class query on id
- Passes null if domain class not found
- If found binds XML, JSON or params

Data Binding: Auto Query if Domain Object

- If command object is a domain class query on id
- Passes null if domain class not found
- If found binds XML, JSON or params

```
class PersonController
{
    def save(Person p) {
        ...
    }
}
```

GRAILS Demo

XSS Prevention



XSS Prevention

- All GSP expressions and scriptlets now escaped by default
- Double-encoding protection
- New `raw` method to disable escaping



XSS Prevention

- All GSP expressions and scriptlets now escaped by default
- Double-encoding protection
- New `raw` method to bypass escaping

```
// Inside GSP  
<section>${raw(request.getHeader('Content-Type'))}</section>
```


Other Goodies: Hibernate 4 Support

- Hibernate 3 and 4 binary incompatible, so new plugin
- Not all plugins that work with Hibernate 3 will work with Hibernate 4

Other Goodies: Hibernate 4 Support

- Hibernate 3 and 4 binary incompatible, so new plugin
- Not all plugins that work with Hibernate 3 will work with Hibernate 4

```
// add to BuildConfig  
compile ':hibernate:hibernate4:4.1.11.1'
```

Other Goodies: @Transactional Transform

- Can be applied to any class
- No proxies required!
- Works with controllers
- Replacement for Spring's @Transactional annotation

Other Goodies: @Transactional Transform

- Can be applied to any class
- No proxies required!
- Works with controllers
- Replacement for Spring's @Transactional annotation

```
import
grails.transactions.*

@Transactional
class PersonService {
    ...
}
```

Other Goodies: @DirtyCheck Transform

- Can be applied to any class to add dirty checking
- Tracks calls to setters that modify state
- Classes implement the `DirtyCheckable` interface

Other Goodies: @DirtyCheck Transform

- Can be applied to any class to add dirty checking
- Tracks calls to setters that modify state
- Classes implement the `DirtyCheckable` interface

```
@DirtyCheck
class Person {
    ...
}
def p =
    new Person(name: 'Bob')
p.trackChanges()
assert !p.hasChanged()
p.name = 'Fred'
assert p.hasChanged()
```

Other Goodies: Action Error Handlers

- Actions that define an exception as the first argument get invoked if the exception is thrown from any other action

Other Goodies: Action Error Handlers

- Actions that define an exception as the first argument get invoked if the exception is thrown from any other action

```
class PersonController {  
  ...  
  def tooOld(TooOldException e) {  
    render view: 'tooOld'  
  }  
}
```


Other Goodies: Controller Namespaces

- Controllers can now define a namespace
- URL Mappings can map to a specific namespace

Other Goodies: Controller Namespaces

- Controllers can now define a namespace
- URL Mappings can map to a specific namespace

```
class PersonController {  
    ...  
    static namespace = 'admins'  
}
```

Q & A

Grails 2.3

2.3



The Future

Grails 3.0



3.0



Grails 3.0: Goals



GRAILS



- Embrace Gradle
- Abstract Packaging / Deployment
- Reach outside the Servlet Container
- App Profiles: Netty, Servlet, Batch, Hadoop
- Deployment with run-app
- Extend the reach of Grails

Grails 3.0: Gradle

- Deprecate existing build system
- Replace with Gradle build
- Easier now everything is forked in 2.3.x
- All the flexibility of a powerful build system
- Legacy bridge layer for running old



Grails 3.0: App Profiles

- Each profile will have distinct plugins, runtimes and packaging (JAR, WAR etc.)

Grails 3.0: App Profiles

- Each profile will have distinct plugins, runtimes and packaging (JAR, WAR etc.)

```
$ grails create-app myapp --profile=netty  
$ cd myapp  
$ grails package  
$ java -jar target/myapp.jar
```


Grails 3.0: Challenges



- **Compatibility**
 - Plugins
 - Build System
- **Modularization**
 - Servlet API independence
 - Refactoring



Q & A

Grails 3.0



Friday, September 13, 13

Learn More. Stay Connected.



Also by me

“RESTfully Async with Grails 2.3” - Wednesday 2:30pm

Web: grails.org

Twitter: twitter.com/grailsframework

LinkedIn: <http://linkedin.com/groups/Grails-User-Group-39757>

Google +: <https://plus.google.com/communities/109558563916416343008>