

# GRAILS

**NOOB TO NINJA IN 3 HOURS**

**[INTRODUCTION WORKSHOP]**

Brian Johnsen / @brianjohnsendk

# WHOAMI

(whocares)

- Brian Johnsen, Gennemtænkt IT A/S
- 12+ years Java
- Groovy & Grails since 2008
- Your Friendly Neighborhood GR8Conf Organizer

# READY?

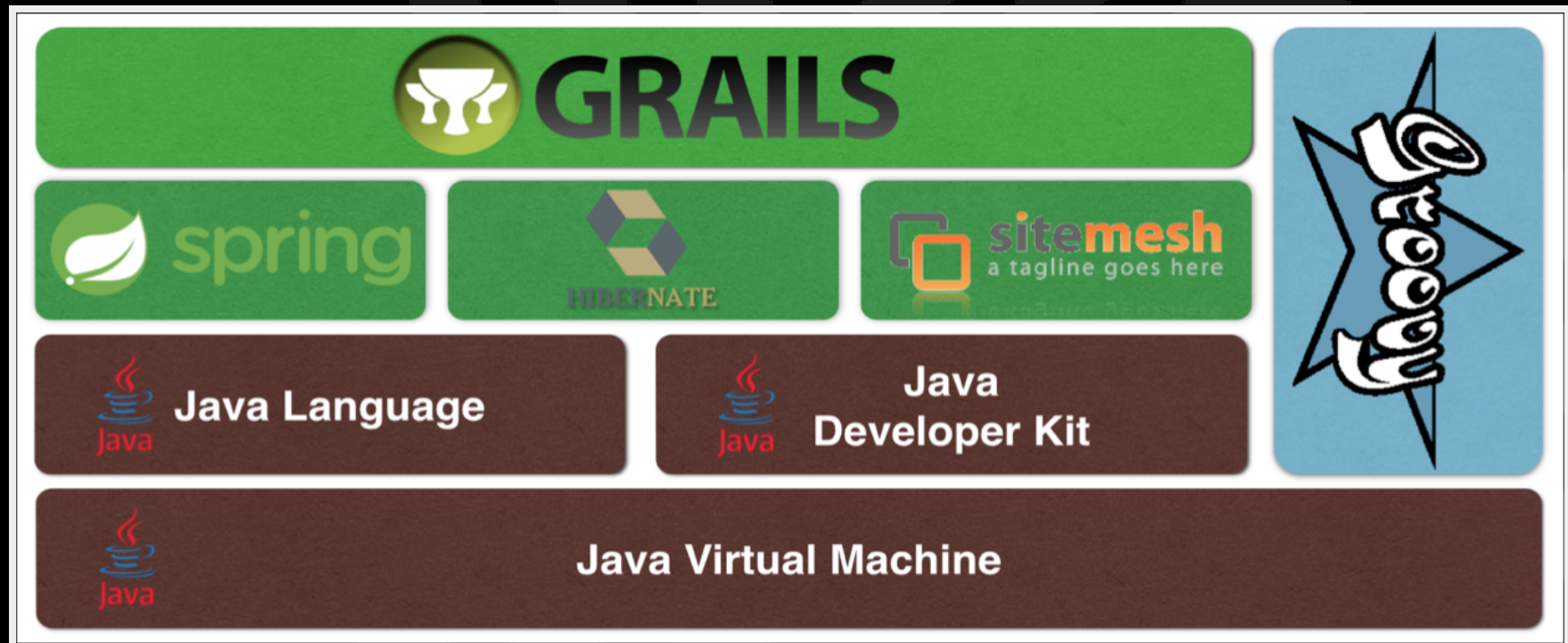
1. Copy rails install to disk and unzip
2. Add to path
3. Voilà

```
$ rails -version  
Rails version: 2.3.8
```

# FULL STACK FRAMEWORK

- ORM (GORM)
- Powerful view technology (GSP) and easy TagLib creation
- Dependency Injection and on-the-fly reloading (Spring)
- Internationalization support (i18n)
- Runtime Container (Tomcat)
- Testing Framework (Spock)
- Plugin System
- Command line tools
- Build Tool (Gant/Gradle)
- and a bunch of other stuff...

# ARCHITECTURE



# PHILOSOPHY

**convention**

**configuration**



**LET'S GET STARTED!**

# DEMO TIME!



CLI and create app demo



# YOUR TURN!



# CLI

Try it out

```
$ rails help
```

It just goes on...

# CREATE THE APP

```
$ rails create-app grailsdemo
```

```
$ cd grailsdemo
```


```
$ rails run-app
```

```
| Server running. Browse to http://localhost:8080/grailsdemo
```

Open your favorite browser and check it out!

# RUNNING APPLICATION!

localhost:8080/grails-intro/

 **GRAILS**

### APPLICATION STATUS

App version: 0.1  
Grails version: 2.3.8  
Groovy version: 2.1.9  
JVM version: 1.7.0\_55  
Reloading active: true  
Controllers: 1  
Domains: 0  
Services: 2  
Tag Libraries: 13

### INSTALLED PLUGINS

### Welcome to Grails

Congratulations, you have successfully started your first Grails application! At the moment this is the default page, feel free to modify it to either redirect to a controller or display whatever content you may choose. Below is a list of controllers that are currently deployed in this application, click on each to execute its default action:

**Available Controllers:**

- [grails.plugin.databasemigration.DbdocController](#)

# ANATOMY

```
— application.properties
— grails-app
  — conf
  — controllers
  — domain
  — i18n
  — migrations
  — services
  — taglib
  — utils
  — views
— grailsw
— grailsw.bat
— lib
— scripts
— src
  — groovy
  — java
— target
— test
  — integration
  — unit
— web-app
  — META-INF
  — WEB-INF
  — css
  — images
  — js
— wrapper
```

# ANATOMY

- `grails-app` - Top level directory for Groovy sources
  - `conf` - Configuration sources.
  - `controllers` - Web controllers - The C in MVC.
  - `domain` - The application domain - The M in MVC.
  - `i18n` - Support for internationalization (i18n).
  - `migrations` - Database migration.
  - `services` - The service layer.
  - `taglib` - Tag libraries.
  - `utils` - Grails specific utilities.
  - `views` - Groovy Server Pages - The V in MVC.
- `scripts` - Gant scripts.
- `src` - Supporting sources
  - `groovy` - Other Groovy sources
  - `java` - Other Java sources
- `test` - Unit and integration tests.
- `web-app` - JavaScript and CSS.
- `wrapper` - Wrapper.



**DON'T  
PANIC**



# DON'T LET FRIENDS FORK

Open:

```
/grails-app/conf/BuildConfig.groovy
```

Change:

```
grails.project.fork = [  
    // configure settings for compilation JVM, note that if you alter the Gro  
    ...  
    ...  
    // configure settings for the Console UI JVM  
    console: [maxMemory: 768, minMemory: 64, debug: false, maxPerm: 256]  
]
```

To:

```
grails.project.fork = false
```

# DOMAIN

`/grails-app/domain/`

- The M in MVC
- Gets mapped to the database
- Dynamic finders FTW -> Forget SQL
- `constraints` enforce database consistency
- Defines scaffolded views

# DOMAIN DEMO!



# MY FIRST DOMAIN CLASS

```
$ grails create-domain-class Person
```

```
//grails-app/domain/Person.groovy
class Person {
    String name
    Integer age
    static constraints = {
        name blank: false
        age min: 0
    }
}
```

# SAVE AND VALIDATE

constraints are enforced on `save()` and `validate()`

```
def person = new Person(age: 42, name: 'Douglas Adams')  
  
person.validate() //only checks constraints  
person.save() //saves the instance to the database
```

# WHOLE LOTTA DYNAMIC GOIN' ON

Finders are dynamically created for all properties

```
def everybody = Person.list()
def theOne = Person.findByNameAndAge('Brian', 41)
def caseInsensitive = Person.findAllByNameIlike('graeme')
def underAge = Person.findAllByAgeLessThan(18)
def seniors = Person.findAllByAgeGreaterThanOrEquals(65)
def orderedByName = Person.listOrderByName()
```

# TESTING

/test/unit/

- Spock Specifications
- Deeply integrated in Grails
- When creating any artifact a corresponding test is created
  - Creating `Person` also creates a `PersonSpec`

# TEST DEMO!





# SPOCK TEST DEFAULT

```
//test/unit/grailsdemo/PersonSpec.groovy
@TestFor(Person)
class PersonSpec extends Specification {
    def setup() {
    }

    def cleanup() {
    }

    void "test something"() {
    }
}
```

# WRITE A TEST!



"age should be more than 0"

# COULD LOOK LIKE THIS

```
//test/unit/grailsdemo/PersonSpec.groovy
@TestFor(Person)
class PersonSpec extends Specification {
    void "age should be more than 0"() {
        when:
            def person = new Person(age: age, name: 'joe')

        then:
            person.validate() == valid

        where:
            age      | valid
            0         | true
            41        | true
            -1        | false
    }
}
```

# TESTING

- THE EASIEST WAY TO EXECUTE CODE!



Play with dynamic finders

# TAKING CONTROL

`/grails-app/controllers/`

- The C in MVC
- Data binding
- Redirecting requests
- Delegating to business logic

# HIM AGAIN!



Scaffolded UI Demo

# SCAFFOLDED CONTROLLER

```
//grails-app/controllers/grailsdemo/PersonController.groovy
class PersonController {
    static scaffold = true
}
```

Fire it up and give it a spin

```
$ grails run-app
```

# STOP THE PRESSES

## IT'S BACKED BY A REAL DATABASE!

Browse to <http://localhost:8080/demo/dbconsole>

The screenshot displays the H2 Database Console web interface. At the top, there's a toolbar with icons for undo, redo, and a status bar showing 'Auto commit' is checked, 'Max rows' is set to 1000, and 'Auto complete' is set to 'Normal'. Below the toolbar, the left sidebar shows the database structure: 'jdbc:h2:mem:devDb;MVCC=TRUE' (selected), 'PERSON' (table), 'INFORMATION\_SCHEMA' (schema), 'Sequences', 'Users', and 'H2 1.3.173 (2013-07-28)' (version). The main area contains a text input field with the SQL statement 'SELECT \* FROM PERSON'. Below the input field, the results of the query are displayed in a table format. The table has four columns: 'ID', 'VERSION', 'AGE', and 'NAME'. It contains two rows of data: one for 'Brian' with ID 1 and age 41, and another for 'Hector' with ID 2 and age 0. Below the table, it indicates '(2 rows, 2 ms)'.

jdbc:h2:mem:devDb;MVCC=TRUE Run (Ctrl+Enter) Clear SQL statement:

SELECT \* FROM PERSON

SELECT \* FROM PERSON;

ID	VERSION	AGE	NAME
1	0	41	Brian
2	0	0	Hector

(2 rows, 2 ms)



**WANT SOME REST?**

# AUTOMATIC CONTENT NEGOTIATION

<http://localhost:8080/demo/person/show/1.json>

```
{  
  class: "grailsdemo.Person",  
  id: 1,  
  age: 41,  
  name: "Brian"  
}
```

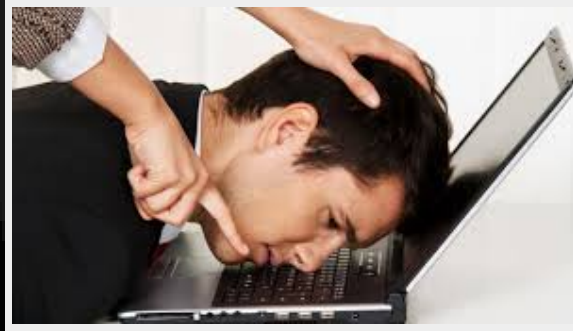
<http://localhost:8080/demo/person/show/1.xml>

```
<person id="1">  
  <age>41</age>  
  <name>Brian</name>  
</person>
```

# BACK ON TRACK



# GENERATE CONTROLLER



```
$ rails generate-controller grailsdemo.Person
```

# ALL THAT CODE

Open:

```
grails-app/controllers/grailsdemo/PersonController.groovy
```

# SOMETHING LIKE THIS?

```
//grails-app/controllers/grailsdemo/PersonController.groovy
@Transactional(readOnly = true)
class PersonController {

    static allowedMethods = [save: "POST", update: "PUT", delete: "DELETE"]

    def index(Integer max) {
        params.max = Math.min(max ?: 10, 100)
        respond Person.list(params), model: [personInstanceCount: Person.count]
    }

    def show(Person personInstance) {
        respond personInstance
    }
    ...
    ...
}
```

# TESTING CONTROLLERS

Run:

```
$ grails test-app unit: PersonController
```

What happens?

Open:

```
/test/unit/grailsdemo/PersonControllerSpec.groovy
```

# TESTING CONTROLLERS

Change:

```
def populateValidParams(params) {  
    assert params != null  
    // TODO: Populate valid properties like...  
    //params["name"] = 'someValidName'  
}
```

To:

```
def populateValidParams(params) {  
    assert params != null  
    params["name"] = 'joe' //or whatever  
    params["age"] = 42 //or whatever  
}
```



# MORE TESTING



Do we have the time?

# VIEWS

`/grails-app/views/`

- The V in MVC
- GSP

# GENERATE VIEWS



```
$ rails generate-all grailsdemo.Person
```

# THE INDEX VIEW

Open:

```
grails-app/views/person/index.gsp
```

# THE SHOW VIEW

Open:

```
grails-app/views/person/show.gsp
```

# CREATE AND EDIT VIEWS - TEMPLATES

Open:

```
grails-app/views/person/create.gsp
```

```
grails-app/views/person/edit.gsp
```

```
grails-app/views/person/_form.gsp
```

# SERVICES

`/grails-app/services/`

- Should contain all business logic
- Transactional
- Injected into any artefact

# CREATE THE SERVICE



```
$ grails create-service Person
```

Open:

```
grails-app/services/demo/PersonService.groovy
```



# UPDATE THE SERVICE

```
@Transactional
class PersonService {
    def getAll() {
        Person.list()
    }
}
```

# UPDATE THE CONTROLLER

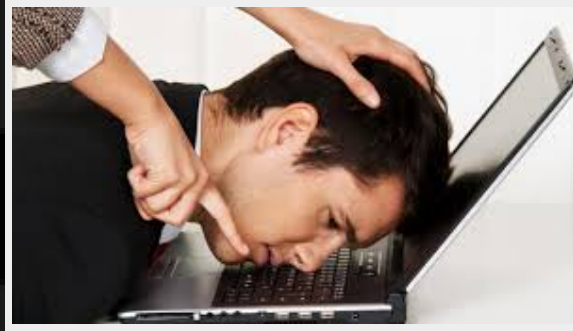
```
class PersonController {  
  
    PersonService personService //injected by rails  
  
    def index(Integer max) {  
        respond personService.getAll()  
    }  
    ...  
}
```

# LETS ADD A NEW FEATURE

Person should have country

```
//grails-app/domain/Person.groovy
class Person {
    String name
    Integer age
    String country
    static constraints = {
        name blank: false
        age min: 0
        country nullable: true
    }
}
```

# UPDATE THE TESTS



Open:

```
test/unit/grailsdemo/PersonSpec.groovy
```

# UPDATE THE VIEW

Open:

```
grails-app/views/person/index.gsp
```

and:

```
grails-app/views/person/_form.gsp
```

# ANOTHER NEW FEATURE

Creating a new person should send you to the list view (index)

Open:

```
grails-app/controllers/grailsdemo/PersonController.groovy
```

# PLUGINS

<https://grails.org/plugins/>

- Build Test Data
- Searchable
- Quartz
- Spring Security
- MongoDB
- and it goes on and on...