Groovy & Grails

Langage de script basé sur Java appliqué dans un environnement JEE

Martin Le Bas

Xposé 2010

Plan de la présentation

- Introduction
- Groovy
  - Du Java au Groovy
  - Le langage Groovy
  - Groovy avancé
- Grails
  - Architecture
  - Créer une application web
- Conclusion
- Références

Pourquoi ce nouveau langage ?

- Caractéristiques d'un langage
  - Typage : Fort ? Faible ?
  - Langage Objet ?
  - Langage de Script ?
  - Flexibilité ? Maintenance ?
  - Productivité ?

- Langages statiques : Java, C#, C++
- Langages dynamiques : Perl, Python, Ruby, PHP
- Plateformes .Net et Java : IronRuby, IronPython, JRuby ...

Groovy : késako ?

- C'est quoi ?
  - Langage de script utilisant la syntaxe de Java
  - Compatibilité du code Java
  - Faiblement typé
  - Vérification des types à l'exécution

- Comment ?
  - Utilise la JVM : Profite du JIT Compiler
  - Compilation et Exécution
    - javac & java
    - groovy `#!/usr/bin/groovy`

HelloJava.java

```
    /* Getters and Setters */
    public String getFirstname() {
        return firstname;
    }
    public String getLastname() {
        return lastname;
    }
    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }
    public void setLastname(String lastname) {
        this.lastname = lastname;
    }

    /* Main method */
    public static void main(String[] args) {
        HelloJava martin = new HelloJava("tinmar","Le Bas");
        martin.setFirstname("Martin");
        System.out.println(martin.sayHello());

        ArrayList<HelloJava> xpose =  new ArrayList<HelloJava>();
        xpose.add(new HelloJava("Etienne", "Duris"));
        xpose.add(new HelloJava("Dominique", "Revuz"));

        for (HelloJava helloMan : xpose) {
            System.out.println(helloMan.sayHello());
        }
    }
}
```

Writable | Smart Insert | 24 : 6

Simplification du code

- Getters et Setters implémentés par défaut.
- Utilisation simplifiée des setters : `martin.firstname="Martin";`

- Typage dynamique : Vérification à l'exécution

```
def firstname
def lastname
```

- Retourne l'objet de la dernière ligne d'une méthode
- Simplification de la concaténation

```
def sayHello(){
      "Hello ${firstname} ${lastname} !!!"
}
```

- Suppression des points virgules
- Suppression du main : Script

HelloGroooovy.groovy

```groovy
package fr.umlv.xpose.groovy;


class HelloGroovy {
    def firstname
    def lastname

    HelloGroovy( firstname,  lastname) {
        setFirstname(firstname)
        setLastname(lastname)
    }

    def sayHello(){
         "Hello ${firstname} ${lastname} !!!"
    }
}

HelloGroovy martin = new HelloGroovy("tinmar","Le Bas");
martin.firstname="Martin";
println(martin.sayHello());

def xpose =  [
              new HelloGroovy("Etienne", "Duris"),
              new HelloGroovy("Dominique", "Revuz")
]
xpose.each {
    println(it.sayHello());
}
```

Writable | Smart Insert | 3 : 1

Le langage Groovy

- Chaines de caractères
- Constructeur
- Méthodes
- Closures
- Collections
- Opérateurs
- Expressions régulières

Chaines de caractères

- Slashy string

```
martin.firstname=/Mar"t"in/;
println(martin.sayHello());
```

```
Console
<terminated> HelloGroooovy (1) [Groovy Application] /Syster
Hello Mar"t"in Le Bas !!!
```

- Multiline string

```
println """ Hello ${firstname}
${lastname} !!! """
```

- GString

```
"Hello ${firstname} ${lastname} !!!"
```

Constructeur

- Utilisation d'une Map pour initialiser un GroovyBean.

```
3  class Person {
4      def firstname;
5      def lastname;
6
7      String toString(){
8          return firstname+" "+lastname;
9      }
10 }
11
12 def me = new Person(firstname:"Martin" , lastname:"Le Bas");
13
14 HashMap<String,String> resquestParameters = new HashMap<String,String>();
15 resquestParameter.putAt("firstname", "Martin");
16 resquestParameter.putAt("lastname", "Le Bas");
17
18 def me2 = new Person(resquestParameters);
19
20 println me;      // Martin Le Bas
21 println me2;     // Martin Le Bas
```

Méthodes

```groovy
public String hello(String firstname, String lastname){
    return "Hello "+firstname+" "+lastname+" !!!";
}
```

```groovy
def hello(firstname, lastname){
     "Hello ${firstname} ${lastname} !!!"
}
```

- public par défaut
- pas de type
- pas de « return »
- Peut être utilisé comme une « fonction »

Closure

- Sans argument

```
def hello = { println "Hello world !" }
hello()
```

- Avec arguments

```
def hello = { name1, name2, name3 -> println "Hello ${name1}, ${name2}, ${name3} !" }
hello("Martin", "Etienne", "DR")    // Hello Martin, Etienne, DR !
hello "Titi","Nono","Clemclem"      // Hello Titi, Nono, Clemclem !
```

- Crée une classe interne avec une méthode « call » qui exécute les instructions.
- Comme pour les méthodes, les parenthèses sont facultatives si il existe des arguments.
- Peut être passé en argument d'une méthode ou d'une autre closure.

Méthode & Closure

- Passage d'une closure à une méthode.

```
3  class Student {
4      def name;
5
6      def doIt(argument){
7          argument(name);
8      }
9  }
10
11 def martin = new Student(name : "Martin");
12
13 martin.doIt {
14     studentName -> println "Hello ${studentName} !"
15 }
16
17 // Hello Martin !
```

Collections

- List

```groovy
def myList = []
println myList.getClass().name // java.util.ArrayList
myList.add "Bob"
myList += "Bob"
myList += "Martin"
myList << "Homer"
myList -= "Martin"
println myList // [Bob, Bob, Homer]
```

- Set

```groovy
def mySet = ["Bob", "Martin"] as Set
println mySet.getClass().name  // java.util.HashSet
mySet.add "Bob"
mySet += "Homer"
println mySet // [Martin, Bob, Homer]
```

Collections

- Map

```groovy
def emptyMap=[:]
def myMap = [
    key1 : "value1",
    "key 2": "value2"
]
println myMap.getClass().name //java.util.LinkedHashMap
myMap.put("key3", "value3")
println myMap.get("key 2")  // value2
println myMap.key1          // value1
println myMap.values();     // [value1, value2]
```

- Range

```groovy
def numRange = 0..9
println numRange.getClass().name // groovy.lang.IntRange
println numRange    // [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
for (i in 5..0){
    print i+", "    // 5, 4, 3, 2, 1,
}
println ""
('z'..'a').each{
    print it+", "   // z, y, x, w, v, u, t, s, r, q, p,
}
```

Opérateurs

- Possibilité de surcharger l'opérateur :

a + b	a.plus(b)
a * b	a.multiply(b)
a << b	a.leftShift(b)
a & b	a.and(b)
a++	a.next()
a[b]	a.getAt(b)
a[b]=c	a.putAt(b,c)

Opérateurs

- Nouveaux opérateurs :
  - Spread operator « *. »

```
class HelloGroovy {
    def firstname
    def lastname
    HelloGroovy( firstname,  lastname) {
        setFirstname(firstname)
        setLastname(lastname)
    }

    def sayHello(){
         println "Hello ${firstname} ${lastname} !"
    }
}
def xpose =  [
    new HelloGroovy("Etienne", "Duris"),
    new HelloGroovy("Dominique", "Revuz")
]
xpose*.sayHello();
//Hello Etienne Duris !
//Hello Dominique Revuz !
```

Opérateurs

- Nouveaux opérateurs :
  - Elvis operator « ?: »

```
HelloGroovy somebody = new HelloGroovy(null, null)
println((somebody.firstname !=null) ? somebody.firstname : "unknown")
println(somebody.firstname ?: "unknown") // unknown
```

  - Safe navigation operator « ?. »

```
somebody=null;
println "Hello ${somebody?.firstname} !"; // Hello null !
println "Hello ${somebody.firstname} !";  // Throws a NullPointerException
```

  - Field operator « .@ »

```
class Demo {
    def field="toto";
    def getField(){
        "tata"
    }
}
def demo = new Demo();
println(demo.field); // tata
println(demo.@field); // toto
```

Expressions régulières

- Match operator : ==~

```
println(/I:\love\blue\screen/ ==~ /^[A-Z]:(\\[a-zA-Z]+)+$/); // true
println('/Mac/is/fabulous' ==~ /^[A-Z]:(\\[a-zA-Z]+)+$/);   //false
```

- Find operator : =~

```
def results = 'lebas.martin@gmail.com' =~ /^([\w._]+)@([\w.]+)\.(fr|ca|net|com|org|info)$/;

if (results){ // if(results!=null)
    println results.getClass().name; // java.util.regex.Matcher

    results[0].eachWithIndex { it, i ->
        println "${i} : ${it}";
    }
    /*
    0 : lebas.martin@gmail.com
    1 : lebas.martin
    2 : gmail
    3 : com
    */
}
```

Expressions régulières

- Pattern operator : ~string

```
def emails = [
    'lebas.martin@gmail.com',
    'lebas@progiweb.com',
    'mlebas%01@etudiant.univ-mlv.fr'
]
def emailPattern  =  ~/^([\w._]+)@([\w.]+)\.(fr|ca|net|com|org|info)$/
emails.each{
    results=emailPattern.matcher(it);
    if (results){
        def r=results[0];
        println "${it} \t Username : ${r[1]} \t Provider : ${r[2]}"
    }else{
        println it+" is not a valid e-mail address !"
    }
}
/*
lebas.martin@gmail.com  Username : lebas.martin  Provider : gmail
lebas@progiweb.com      Username : lebas         Provider : progiweb
mlebas%01@etudiant.univ-mlv.fr is not a valid e-mail address !
*/
```

Groovy avancé

- Meta Object Protocol (MOP)
- Gestion d'XML
- Templates
- Tests unitaires

Meta Object Protocol

- Exemple simple

```
3 String.metaClass.spellIt = {
4     def chars = delegate.toCharArray();
5     for(char c : chars){
6         println c;
7     }
8 }
9
10 "hello".spellIt();
11 /*
12 h
13 e
14 l
15 l
16 o
17 */
```

Meta Object Protocol

- Autre exemple : Foreach

```
19 // Définition de la "meta-closure"
20 Collection.metaClass.myForEach = { closureArg ->
21     for(Object item : delegate){
22         closureArg(item)
23     }
24 }
25
26 // Une liste de String :
27 def list = ["Hello","!", "How", "are", "you", "?"];
28
29 // Utilisation de la "meta-closure"
30 list.myForEach { item ->
31     print item+" ";
32 }
33 // Hello ! How are you ?
```

XML

- Génération d'XML avec un MarkupBuilder

```groovy
def writer = new StringWriter();
def builder = new MarkupBuilder(writer);
builder.doubleQuotes=true;

builder.students {
    student(id:"1"){
        firstname("Martin")
        lastname("Le Bas")
        extra(type:"email", "lebas.martin@gmail.com")
    }
}
println writer.toString();
```

```xml
<students>
  <student id="1">
    <firstname>Martin</firstname>
    <lastname>Le Bas</lastname>
    <extra type="email">lebas.martin@gmail.com</extra>
  </student>
</students>
```

XML

- Parsing d'XML avec un XmlSlurper

```
input = """
<students>
    <student id="1">
      <firstname>Martin</firstname>
      <lastname>Le Bas</lastname>
      <extra type="email">lebas.martin@gmail.com</extra>
    </student>
    <student id="2">
      <firstname>Etudiant</firstname>
      <lastname>Bidon</lastname>
      <extra type="email">bidon@etudiant.univ-mlv.fr</extra>
    </student>
</students>
"""
//On parse le texte
students = new XmlSlurper().parseText(input)

//Affichage du prénom de l'étudiant ayant l'ID 1
println students.student.find{ it.@id=="1" }.firstname.text()

//Affichage des adresses e-mail des étudiants ayant un nom "Bidon"
students.student.findAll{ it.lastname.text()=="Bidon" }.each{
    println it.extra.text();
}
//Changement de la valeur des attributs type par "courriel"
students."*".extra.@type="courriel";
```

Templates

- Utilisation du SimpleTemplateEngine

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head><title>Fiche étudiant</title></head>
    <body>
        Prénom : ${user?.firstname} <br/>
        Nom : ${user?.lastname} <br/>
        Message : ${message} <br/>
        <!-- Insertion de code groovy -->
        Surnom : ${ if(user?.firstname=="Martin") "tinmar" }
    </body>
</html>
```

```
class User {
    def firstname;
    def lastname;
}
def binding = [ user : new User(firstname: "Martin", lastname:"Le Bas"),
                message : "Bonjour !!"]

def userTemplate= this.class.getResource("user.gtpl");
def templateEngine = new SimpleTemplateEngine();
def template = templateEngine.createTemplate(userTemplate);

println template.make(binding);
```

Templates

- Utilisation du SimpleTemplateEngine

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head><title>Fiche étudiant</title></head>
    <body>
        Prénom : Martin <br/>
        Nom : Le Bas <br/>
        Message : Bonjour !! <br/>
        <!-- Insertion de code groovy -->
        Surnom : tinmar
    </body>
</html>
```

```
class User {
    def firstname;
    def lastname;
}
def binding = [ user : new User(firstname: "Martin", lastname:"Le Bas"),
                message : "Bonjour !!"]

def userTemplate= this.class.getResource("user.gtpl");
def templateEngine = new SimpleTemplateEngine();
def template = templateEngine.createTemplate(userTemplate);

println template.make(binding);
```

Tests Unitaires

- Groovy Unit Testing
  - Primordial
  - Facilite la maintenance
  - Groovy étend JUnit pour l'adapter aux particularités du langage.
    - assertArrayEquals
    - assertContains
    - assertEquals
    - assertInspect
    - assertScript
    - etc...

Java - xpose/src/fr/umlv/xpose/groovy/UnitTest.groovy - Eclipse - /Users/kruder/workspace/groovy

HelloGroooovy.groovy | UnitTest.groovy

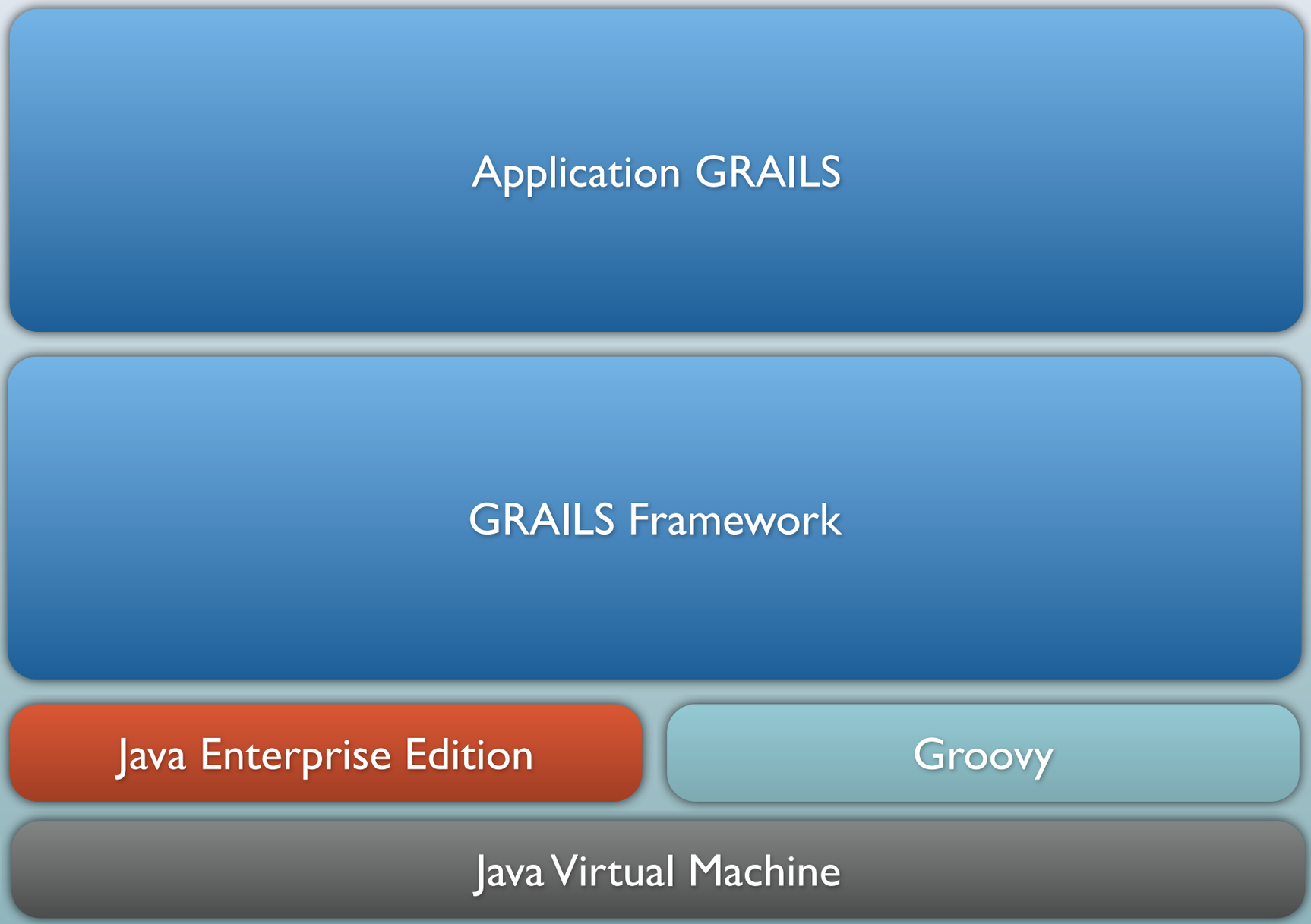
```groovy
package fr.umlv.xpose.groovy;

import groovy.util.GroovyTestCase;

class UnitTest extends GroovyTestCase {

    void testEquals(){
        HelloGroovy hello = new HelloGroovy("anony", "mous");
        assertEquals("Hello anonymous !", hello.sayHello());
    }
    void testScript(){
        assertScript("fr.umlv.xpose.groovy.HelloGroooovy");
    }

}
```

Writable | Smart Insert | 15 : 2

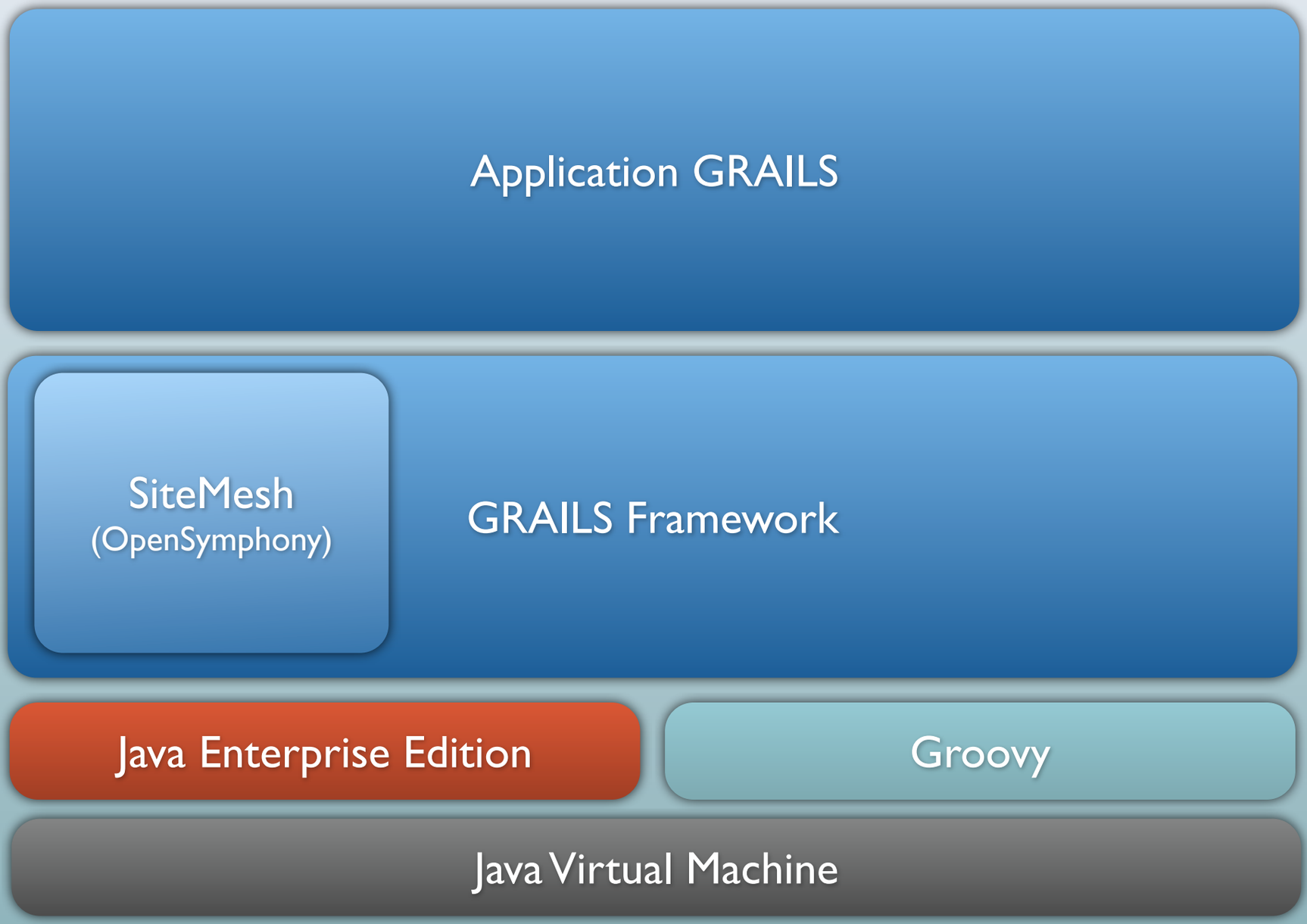
GRAILS

- Architecture
- Créer une application web
  - scaffolding
  - controller
  - domain
  - validation
  - services
  - views
  - taglib
  - plug-in

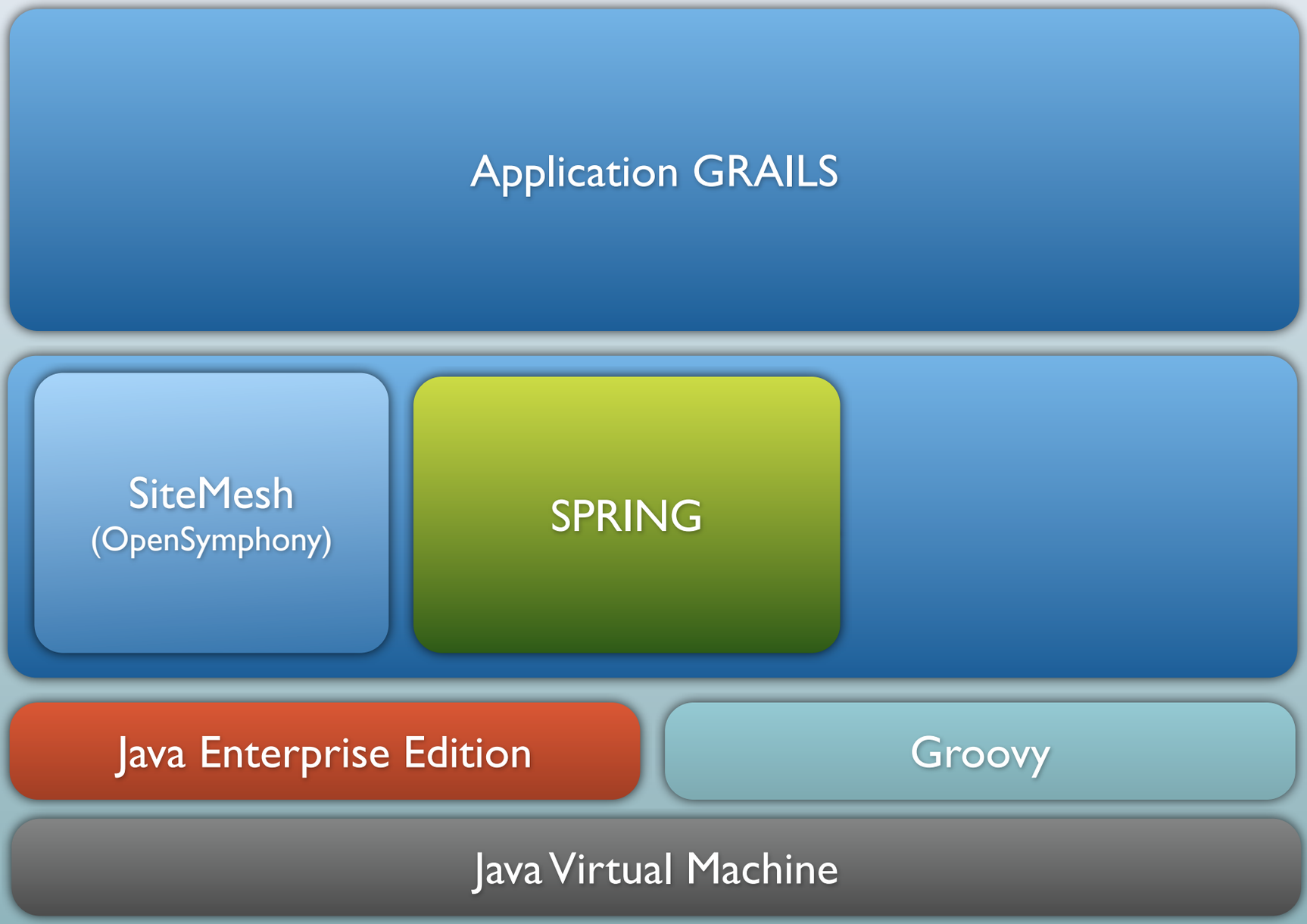
Architecture



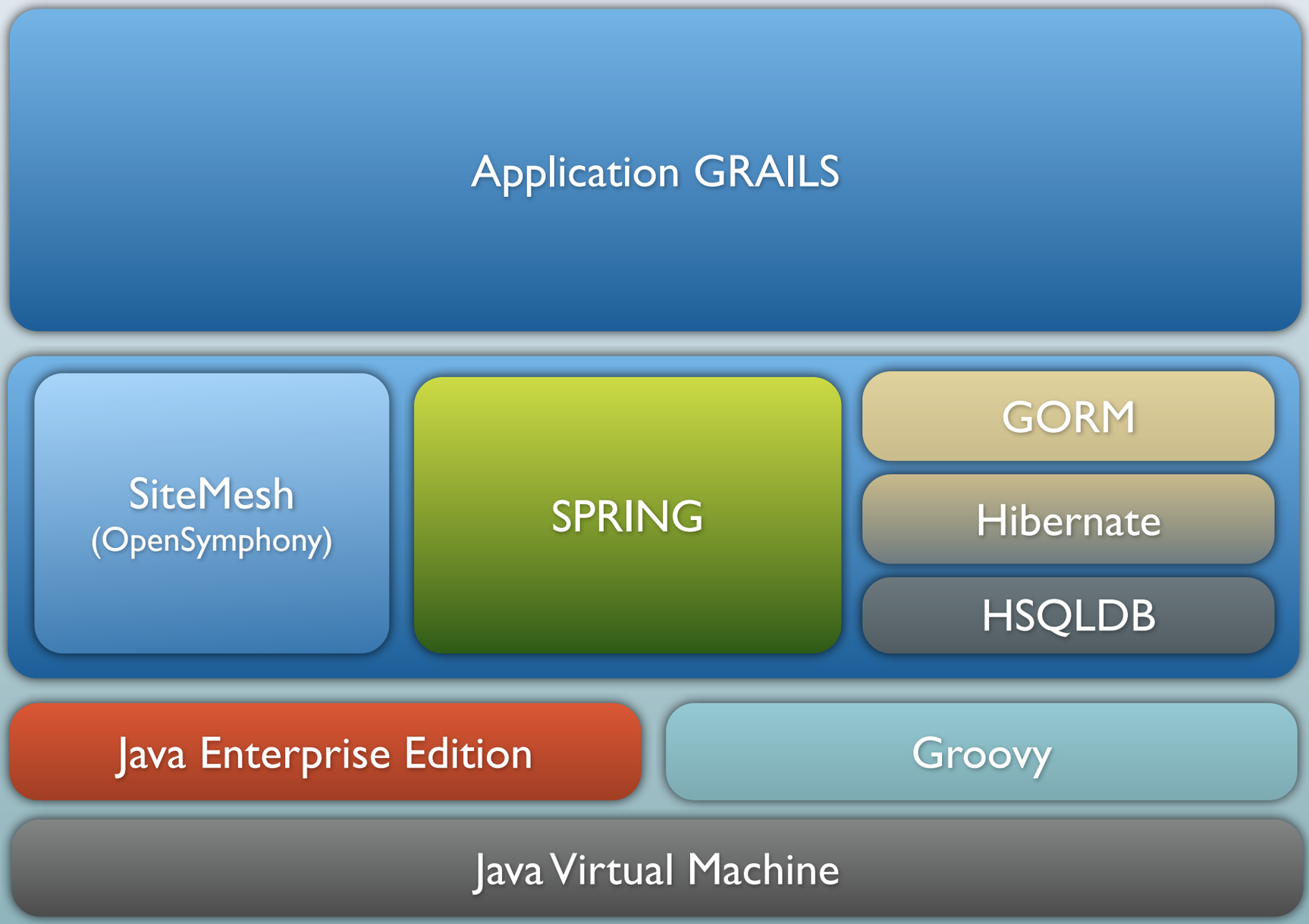
Architecture



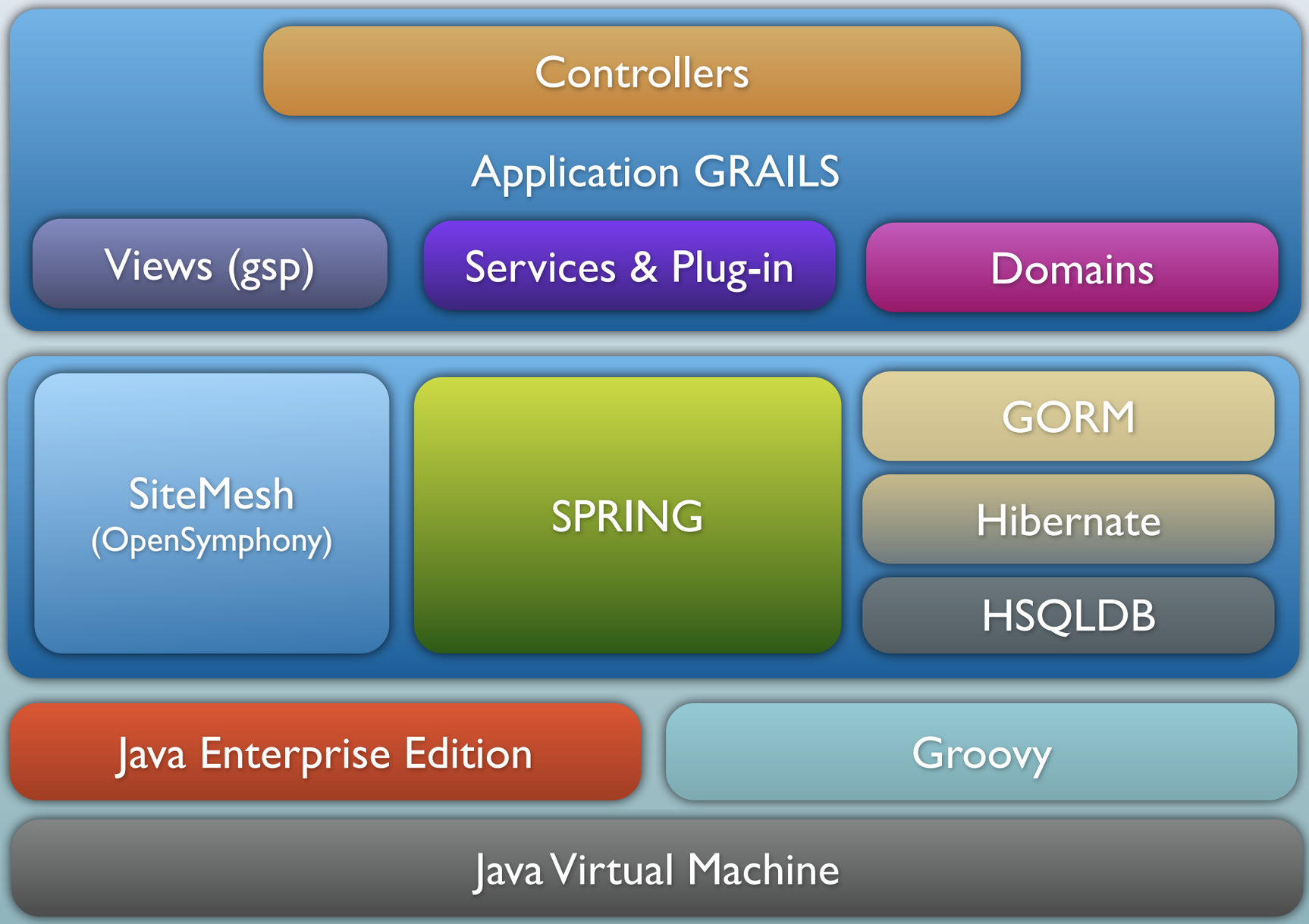
Architecture



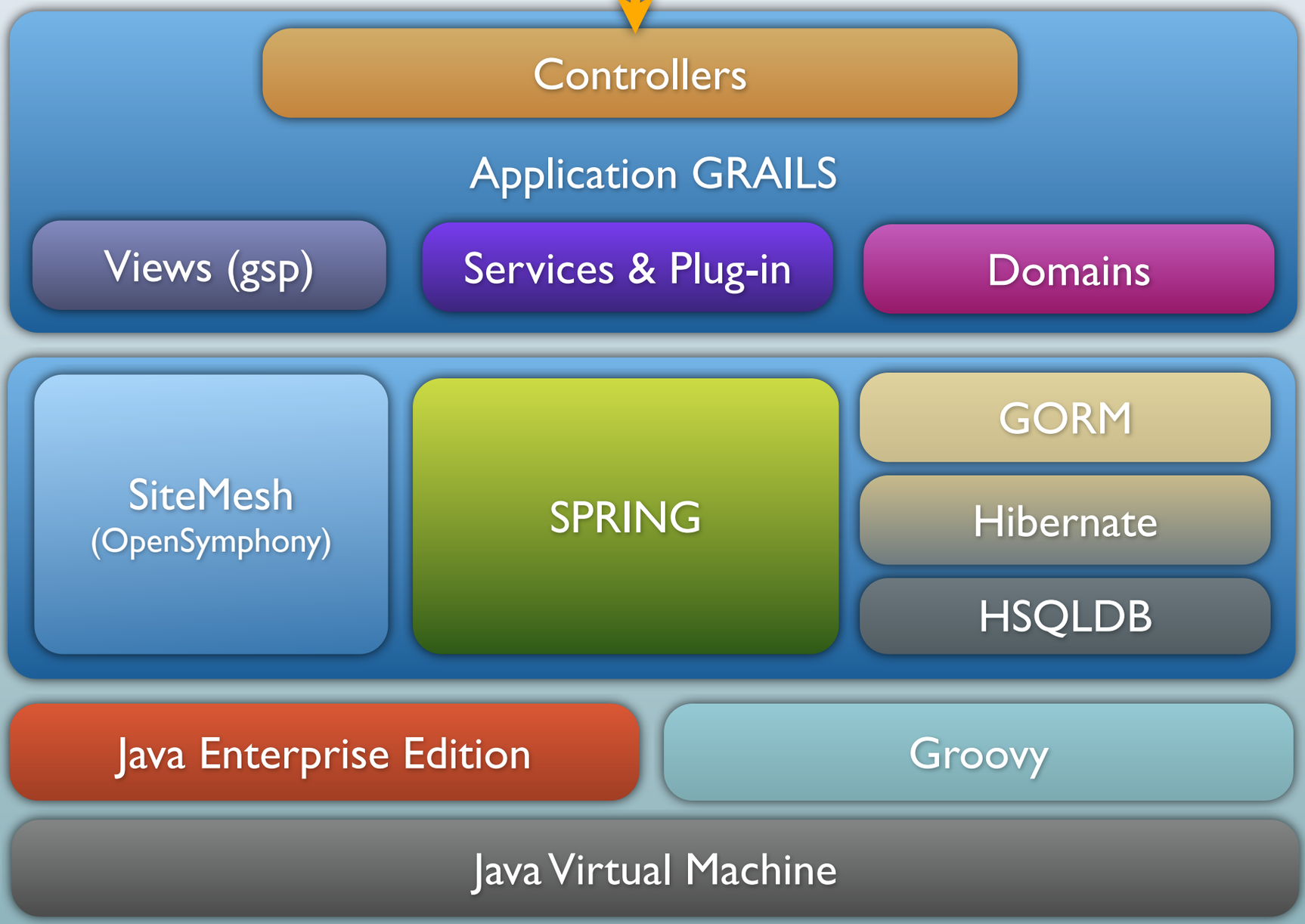
Architecture



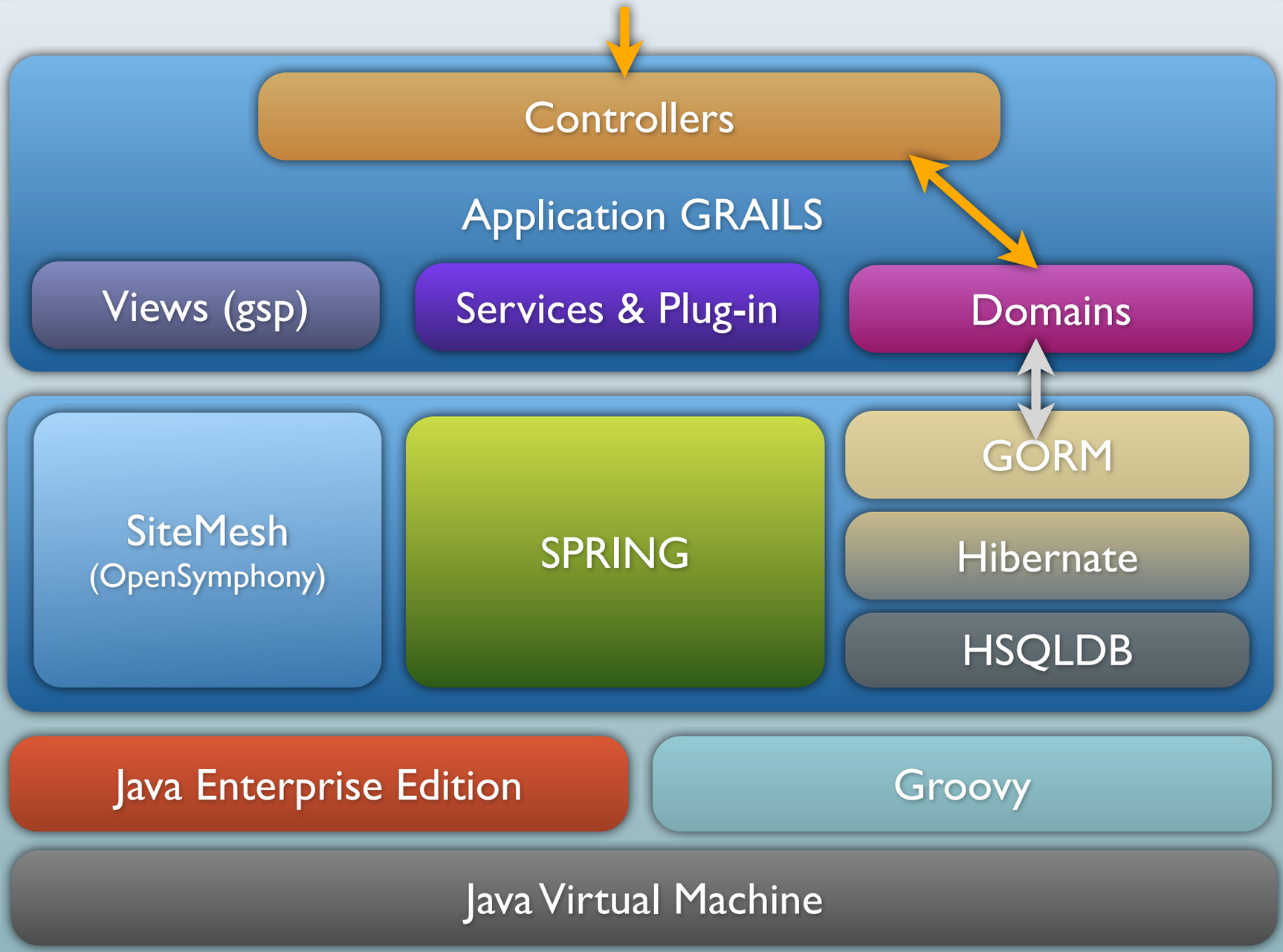
Architecture



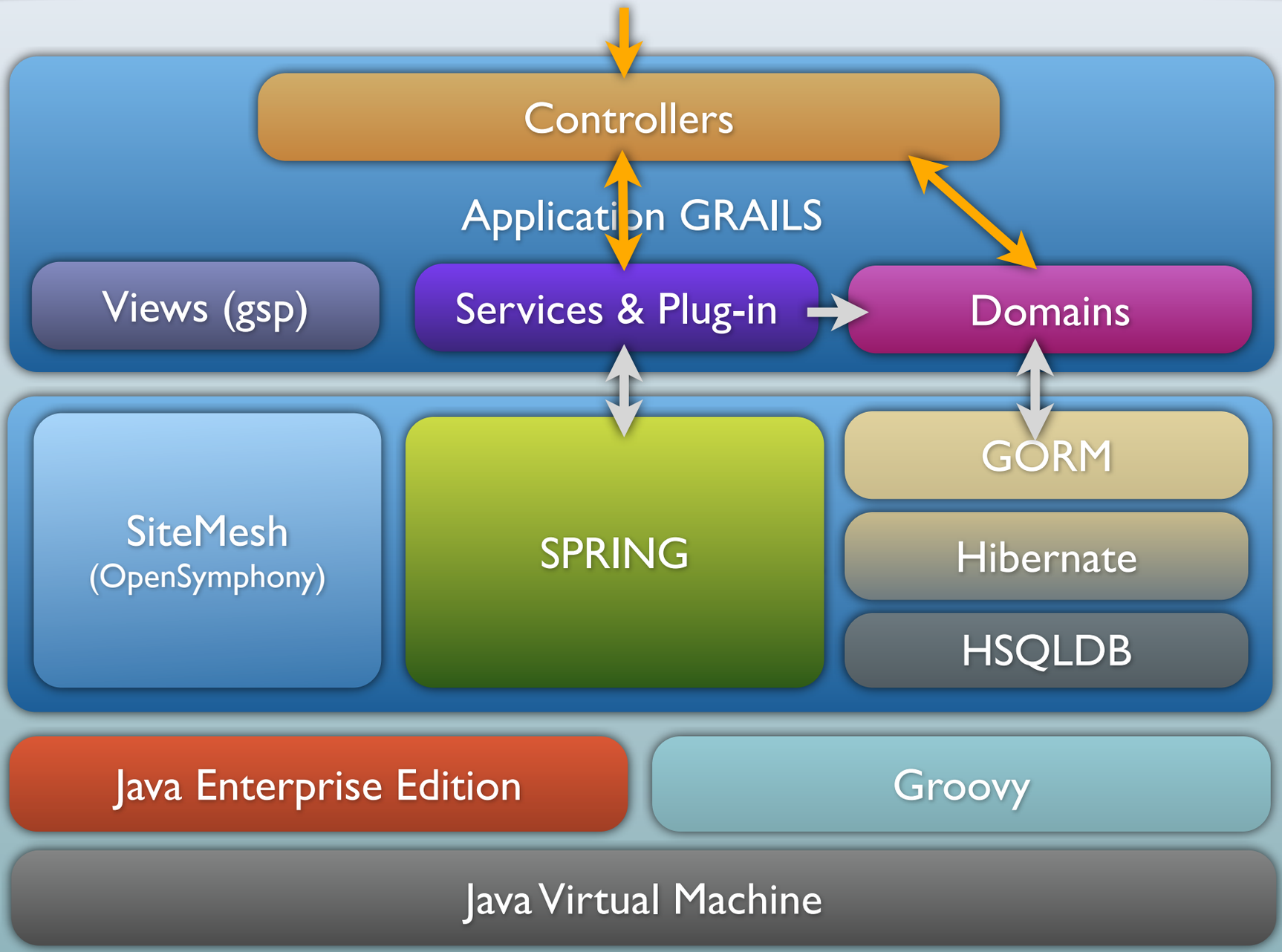
Architecture



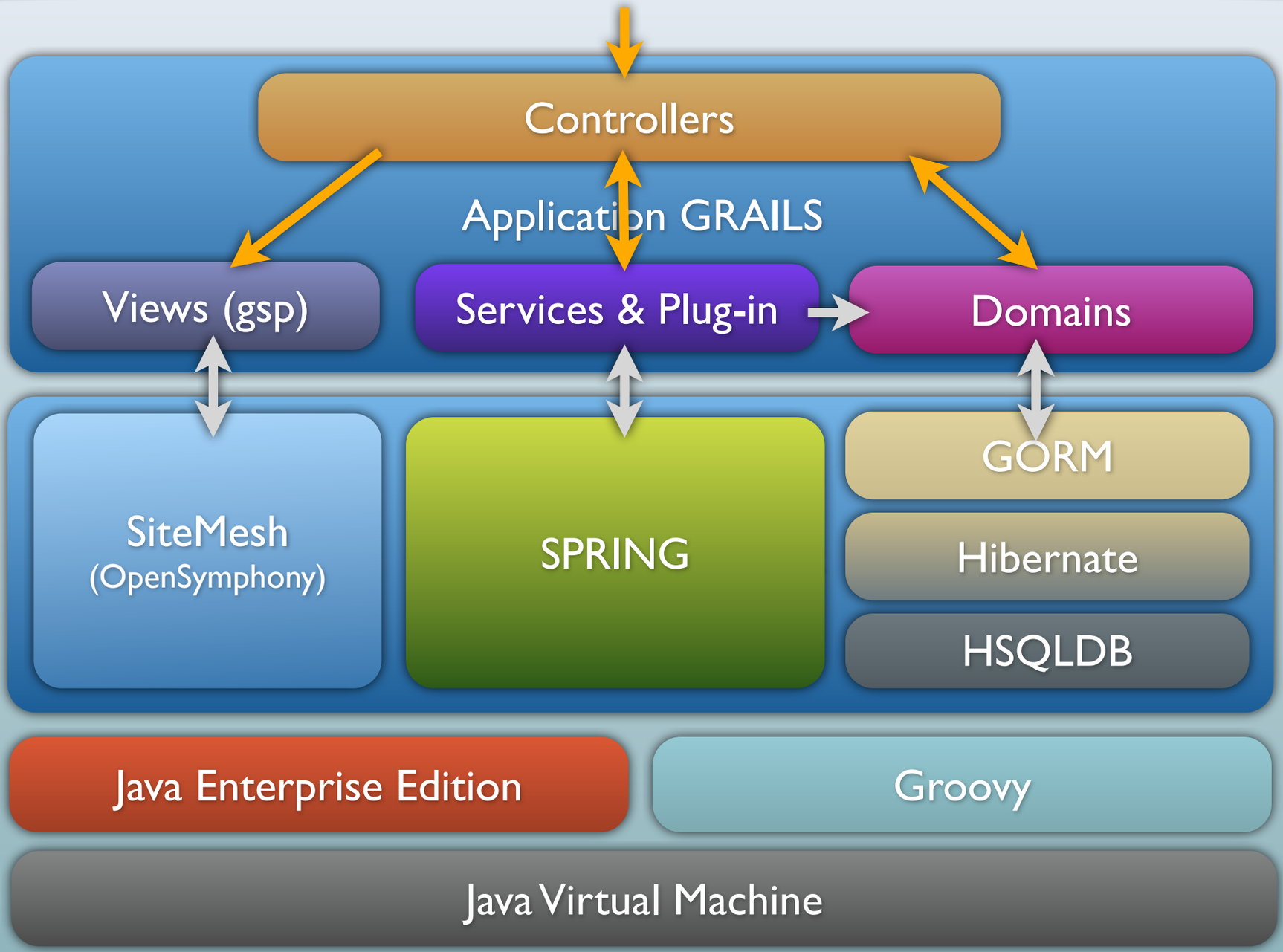
Architecture



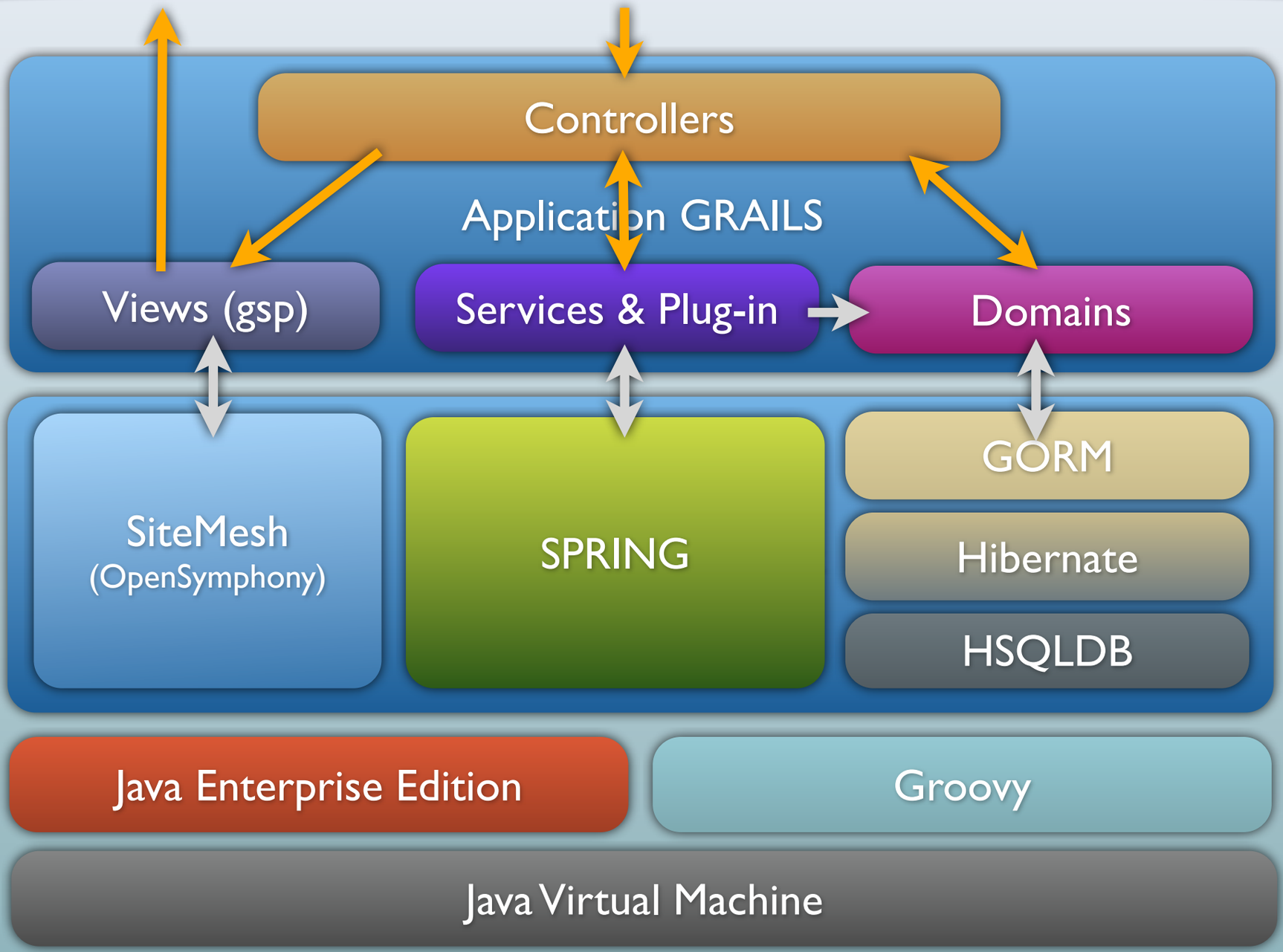
Architecture



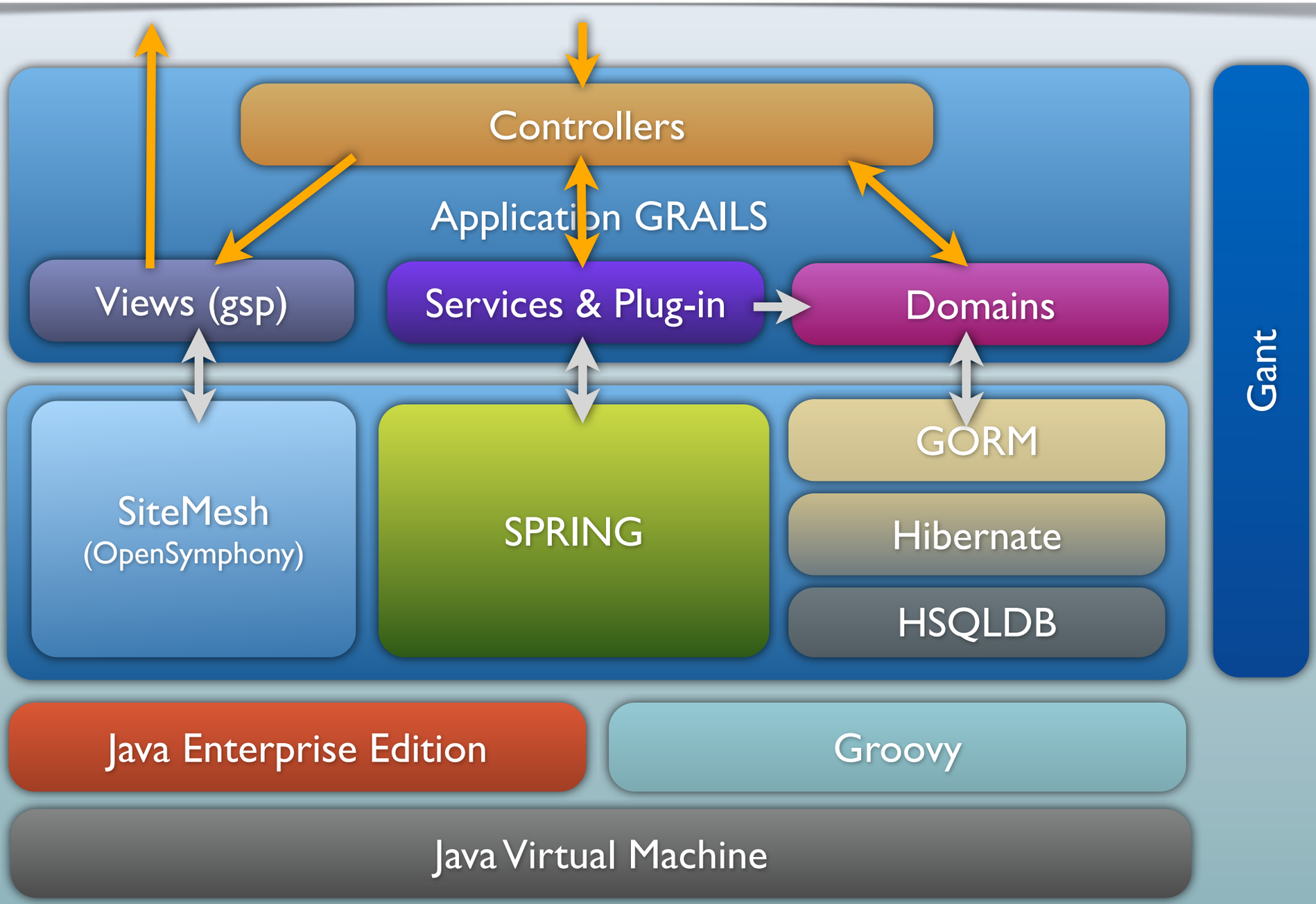
Architecture



Architecture



Architecture



Les commandes grails

- `grails` **`create-app`** `myApp`
- `grails` **`create-domain-class`** `my.pckg.DomainClass`
- `grails` **`generate-controller`** `my.pckg.DomainClass`
- `grails` **`generate-views`** `my.pckg.DomainClass`
- `grails` **`create-service`** `my.pckg.ServiceName`
- `grails` **`clean`**
- `grails` **`run-app`**

Scaffolding

- Génération des vues et des «Controllers» en fonction de la définition des «Domains».

  - Vues & Actions générées :
    - Création
    - Edition
    - Suppression
    - Affichage
    - Liste

  - Le scaffolding peut être dynamique ou statique.

progiweb@libratg: ~ — bash

progiweb@...amo — bash | progiweb@...us: ~ — bash | progiweb@l...tg: ~ — bash | bash

```
Mactin-2:scaffold kruder$ grails create-domain-class Student
```

HTTP ERROR: 404

NOT_FOUND

RequestURI=/

(default package) - scaffold/grails-app/controllers

Controller

- Reçoit la requête et effectue le traitement en fonction de l'action demandée.
- Mappage des URLs : http://<... >/controller/action/
- Exemple :

```
32    def register = { } // La vue par défaut est appelée : student/register.gsp
33
34    def handleRegister = {
35        def studentInstance = new Student(params)
36        if(!studentInstance.hasErrors() && studentInstance.save()) {
37
38            emailConfirmationService.send(studentInstance);
39
40            redirect(action:'emailSent',id:studentInstance.id);
41        }else{
42            render(view:'register',model:[studentInstance:studentInstance])
43        }
44    }
```

Domain

- Modélise les données
- Permet de mapper les objets dans une base de données.
- Définit le comportement de l'application.
- Facilite la validation des formulaires.
- Query caching
- MOP : get(), save(), list(), findByFirstname(), etc...

```
5  class SimpleStudent {
6      // Types de base
7      String firstname;          String lastname;
8      String emailAddress;       String postalCode;
9      String password;           String password2;
10     Date createdDate;
11     // Relation one-to-one
12     Bus bus;
13     Room room;
14     // Relation one-to-many
15     static hasMany = [transactions : Transaction];
16
17     // Le champ password2 ne sera pas persisté.
18     static transients = ["password2"];
19     // Actions a effectuer avant de persister l'objet.
20     def beforeInsert = {
21         createdDate = new Date();
22         password=DigestUtils.md5Hex(password);
23     }
24     def beforeUpdate ={
25         password=DigestUtils.md5Hex(password);
26     }
27
```

Validation

```
28    static constraints = {
29        firstname(blank:false)
30        lastname(blank:false)
31        emailAddress(blank:false, unique:true)
32        password(blank:false)
33
34        password2(
35            blank:false,
36            validator: { val,obj->
37                return val==obj.password;
38            }
39        )
40
41        postalCode(matches:/^([1-8][0-9]|9[0-5]|0[1-9])[0-9]{3}$/)
42
43        bus()
44        room()
45    }
46 }
```

Validation

```
28    static constraints = {
29        firstname(blank:false)
30        lastname(blank:false)
31        emailAddress(blank:false, unique:true)
32
33
34
35
36
37
38
39
40
41
42
43
44
45    }
46}
```

Inscription au WEI 2010 (1/3)

- La propriété [emailAddress] de la classe [class org.fadai2000.wei.Student] ne peut pas être vide
- La propriété [firstname] de la classe [class org.fadai2000.wei.Student] ne peut pas être vide
- La propriété [lastname] de la classe [class org.fadai2000.wei.Student] ne peut pas être vide

Prénom :

Nom :

Adresse e-mail :

Envoyer

Services

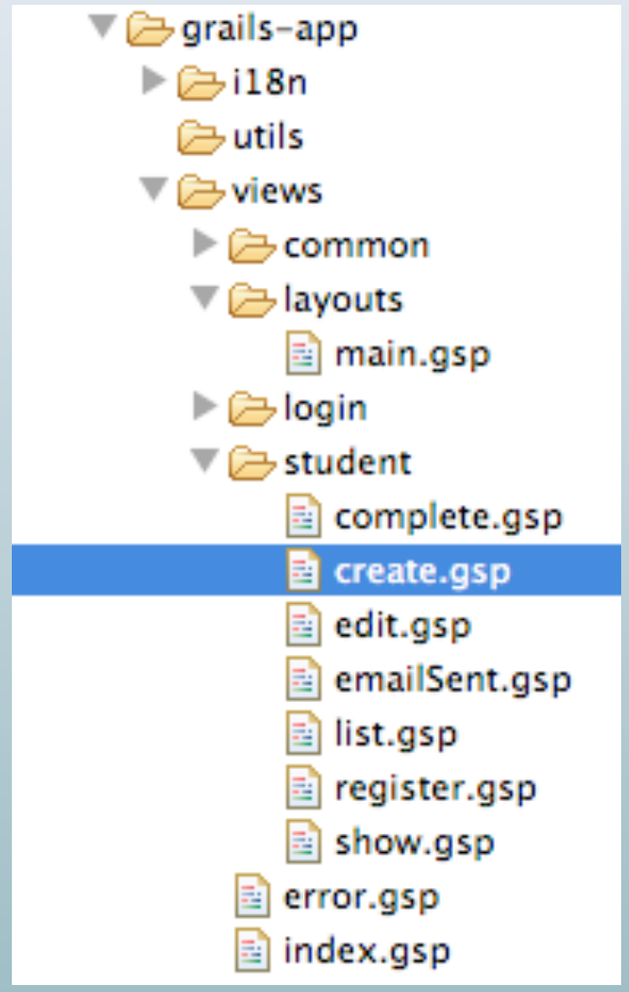
- Scope (durée de vie)
  - singleton
  - session
  - conversation
  - flow
  - flash
  - request
  - prototype

- Utilisation d'un service par « injection »

```
13 class SimpleService implements InitializingBean {
14
15     static scope="singleton"
16
17     def mailService; // Injection du service.
18
19     void afterPropertiesSet(){
20         // Les actions à effectuer
21         // au démarrage du service ...
22     }
23
24     void send(Student student){
25         mailService.sendMail {
26             to "${student?.emailAddress}"
27             subject "Inscription au WEI !"
28             html "Hello";
29         }
30     }
31 }
```

Vues

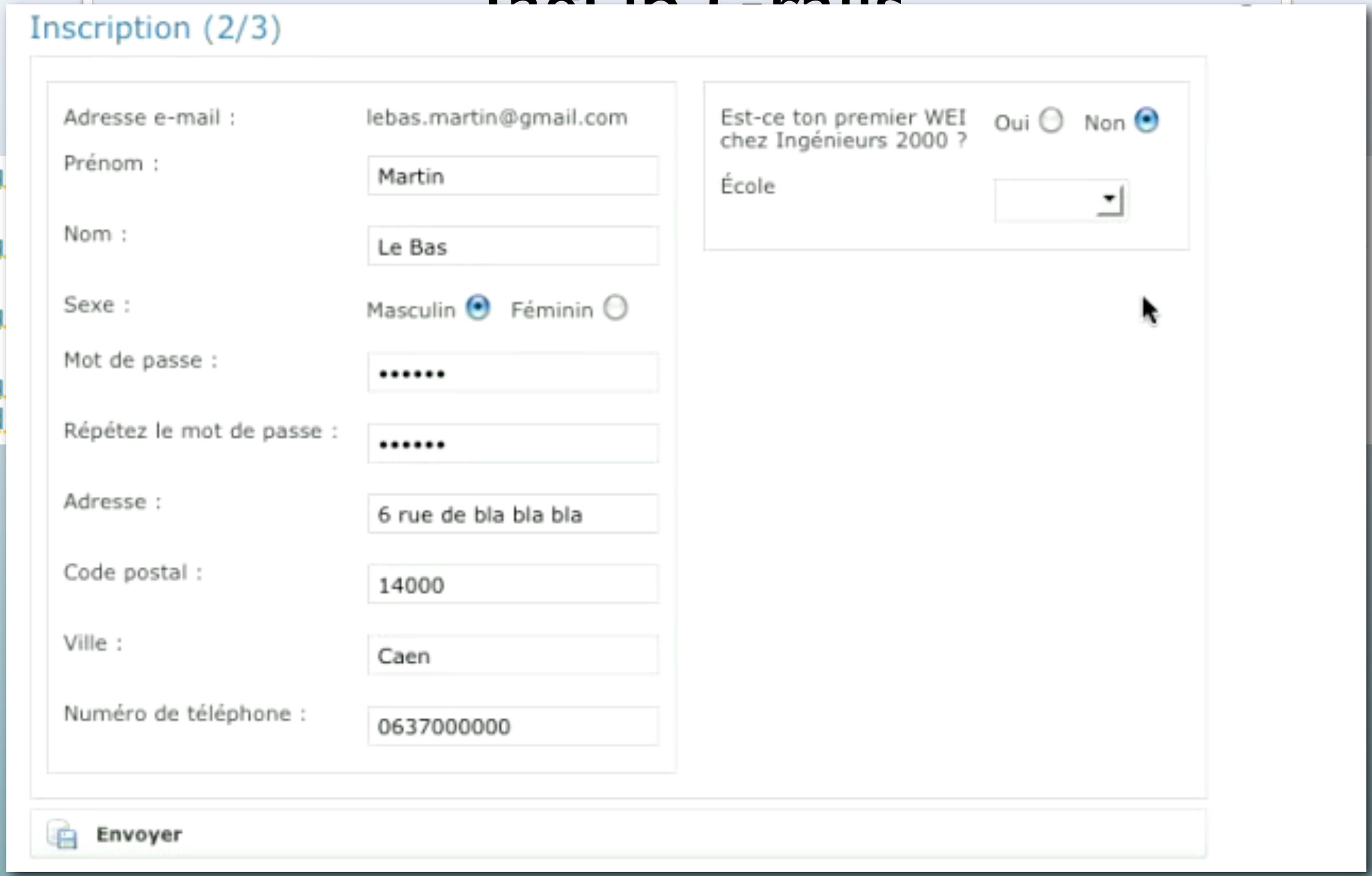
- GSP : Groovy Server Page
- Utilisation de SiteMesh pour «décorer» les pages.
- Intégration de librairies AJAX dans le framework. (taglib)



TagLib Grails

```
<g:renderErrors bean="${studentInstance}" as="list" />

<g:select  id="schoolID"  name="school.id"  from="${School.list()}" optionKey="id"/>

<g:richTextEditor name="content"  toolbar="Basic" height="150" />

<g:remoteField action="changeValue" update="valueDiv" name="value" />
<div id="valueDiv">Je suis mis à jour avec la nouvelle valeur !</div>
```

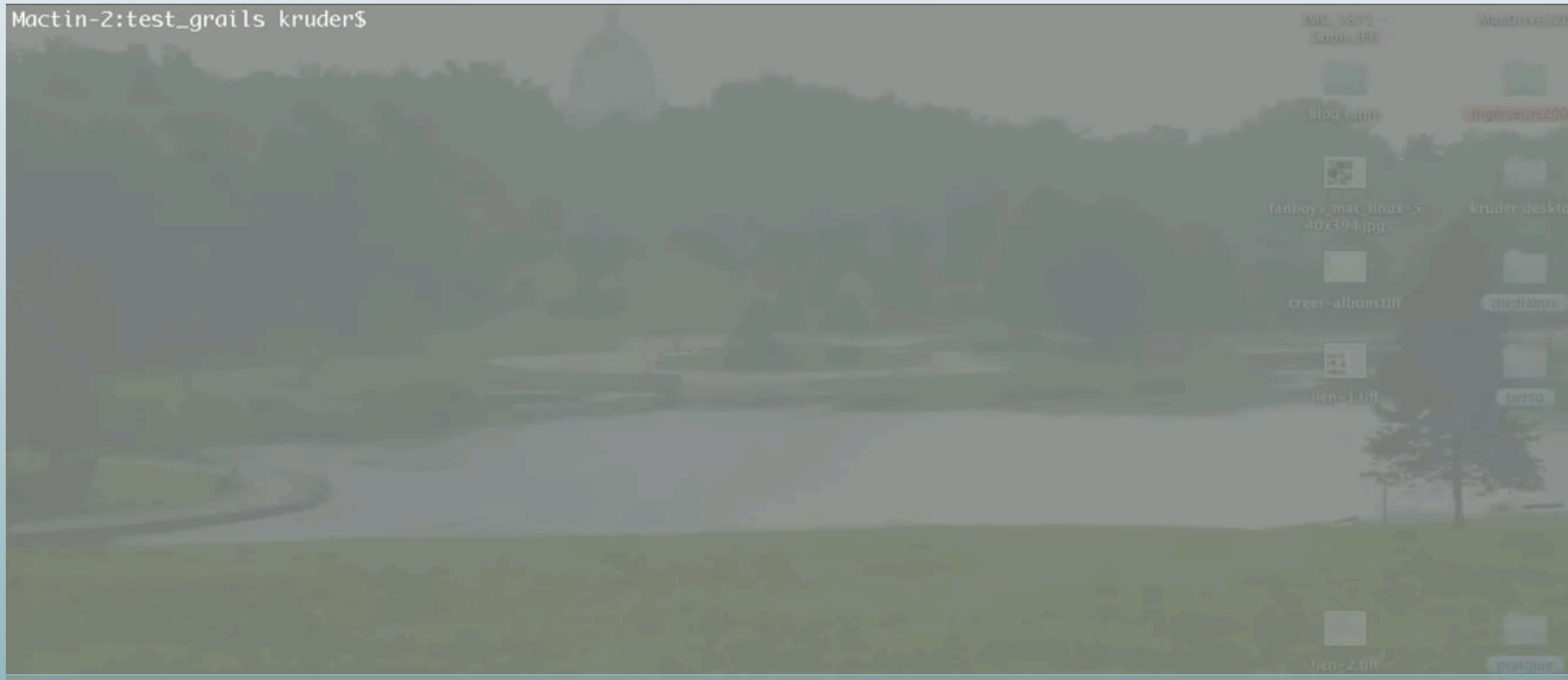
TagLib Grails



Plug-in

- Une liste d'extensions est disponible sur le site de Grails.
  - Service de mail
  - Authentification
  - Moteur de recherche
  - RSS
  - Paiement en ligne
  - etc...
- Possibilité de créer sa propre extension.
- Installation en utilisant un système repository. (Similaire à la commande apt-get de Debian)

Plug-in

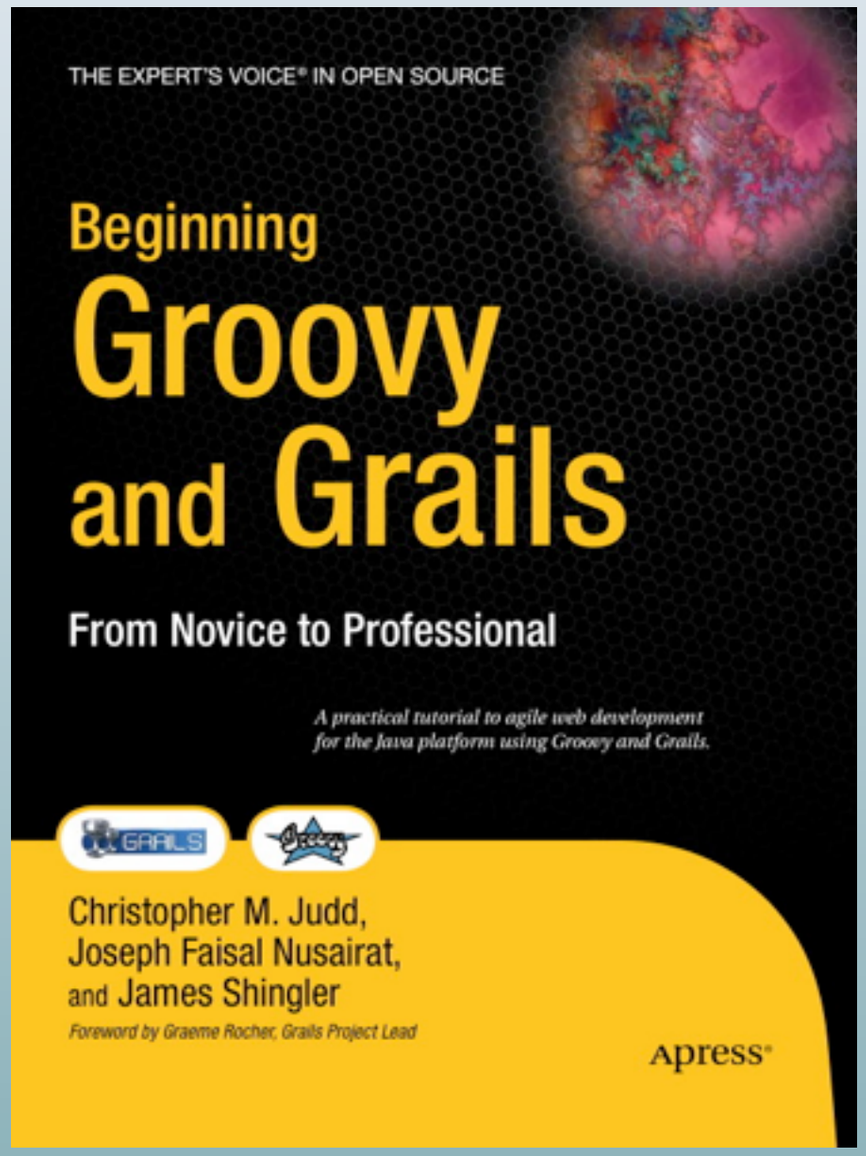


Conclusion

- Groovy
  - Apporte plus de flexibilité au développeur.
  - Gain de productivité ?
  - Maintenance ?
- Grails
  - Permet de développer rapidement une application.
  - Facilité à mettre en oeuvre.
  - Travail en équipe ?
  - Taille du projet ?
- Mon opinion

Références

- «Beginning Groovy and Grails», Apress (disponible sur Amazon)



- Site officiel de Groovy http://groovy.codehaus.org/

- Site officiel de Grails http://www.grails.org/

- La documentation indispensable : http://grails.org/doc/latest/