Grails Framework

Modern Web Applications written in Groovy

You…

Who knows what Groovy is?

Who knows what Grails is?

Who knows JAVA?

Summary

- **What is Grails**
- **What is Groovy**
- **Grails Architecture**
- **Grails MVC and other goodies**
- **Build your new grails application**

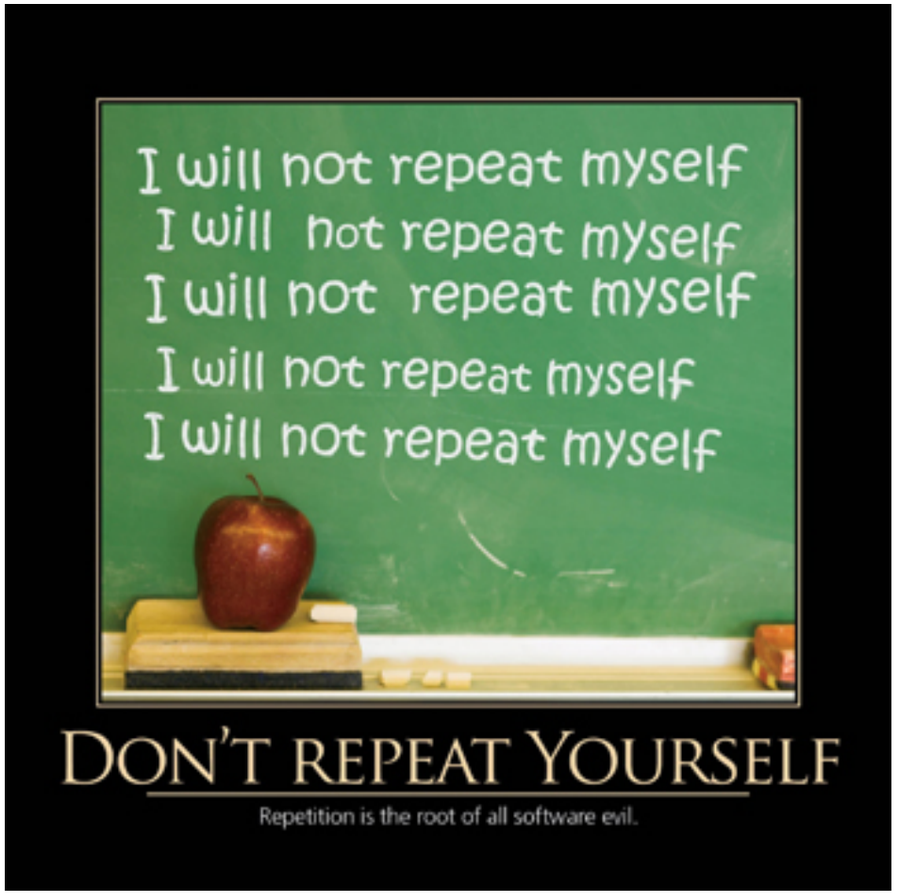
What is Grails?

- **Open Source Framework** http://grails.org
- **A bit of history**
  - Bring **Ruby on Rails** (http://rubyonrails.org) hight productivity to the Java community
  - Was born with the name **Groovy on Rails** in July 2005
  - David Heinemeier - founder of Ruby on Rails requested the change of name → **Grails**
  - First released version (0.1) on March 2006
  - In February 2008 Grails 1.0 is released

What is Grails?

- **Current version 2.3.7**
- **A Full-stack Web Framework**
  - Runs on the Java Virtual Machine (JVM)
  - Rapid development
    - Inspired of Ruby on Rails
    - Re-use existing Java technologies: Spring, Hibernate, …
    - Use a Java-like dynamic language (Groovy)
    - Plugin-in based framework
    - Convention over configuration
      - Many built-in conventions
      - Config files are in Groovy
    - Don´t Repeat Yourself (DRY) principle
      - Formulated by Andy Hunt and Dave Thomas in “The Pragmatic Programmer”, underlies many other well-known software development best practices and design patterns. Duplication is waste, Repetition in process class for automation, repetition in logic calls for abstraction…

JVM



What is Groovy?

- **Agile and Dynamic language for the JVM**
  - Supports also static typing
  - Compiles directly to byte code
  - Open source
- **Java on steroids**
  - Builds upon the strengths of Java but…
    - with power features borrowed from Smaltalk/Python/Ruby
    - Makes modern programming features available to Java developers with a flat learning curve
- **Seamlessly integrates with all existing Java objects and libraries**
- **Feels natural to Java developers**

Normal Java Program


```java
public class HelloStudent
{
  private String name;

  public void setName(String name)
  {
    this.name = name;
  }

  public String getName(){
    return name;
  }

  public String sayHelloToStudents()
  {
    return "Hello CERN Spring Campus Student: " + name;
  }

  public static void main(String[] args)
  {
    HelloStudent helloStudent = new HelloStudent();
    helloStudent.setName("Eloy");
    System.out.println(helloStudent.sayHelloToStudents());
  }
}
```

Normal Groovy Program


```
public class HelloStudent
{
  private String name;

  public void setName(String name)
  {
    this.name = name;
  }

  public String getName(){
    return name;
  }

  public String sayHelloToStudents()
  {
    return "Hello CERN Spring Campus Student: " + name;
  }

  public static void main(String[] args)
  {
    HelloStudent helloStudent = new HelloStudent();
    helloStudent.setName("Eloy");
    System.out.println(helloStudent.sayHelloToStudents());
  }
}
```

But….

A Groovier programmer

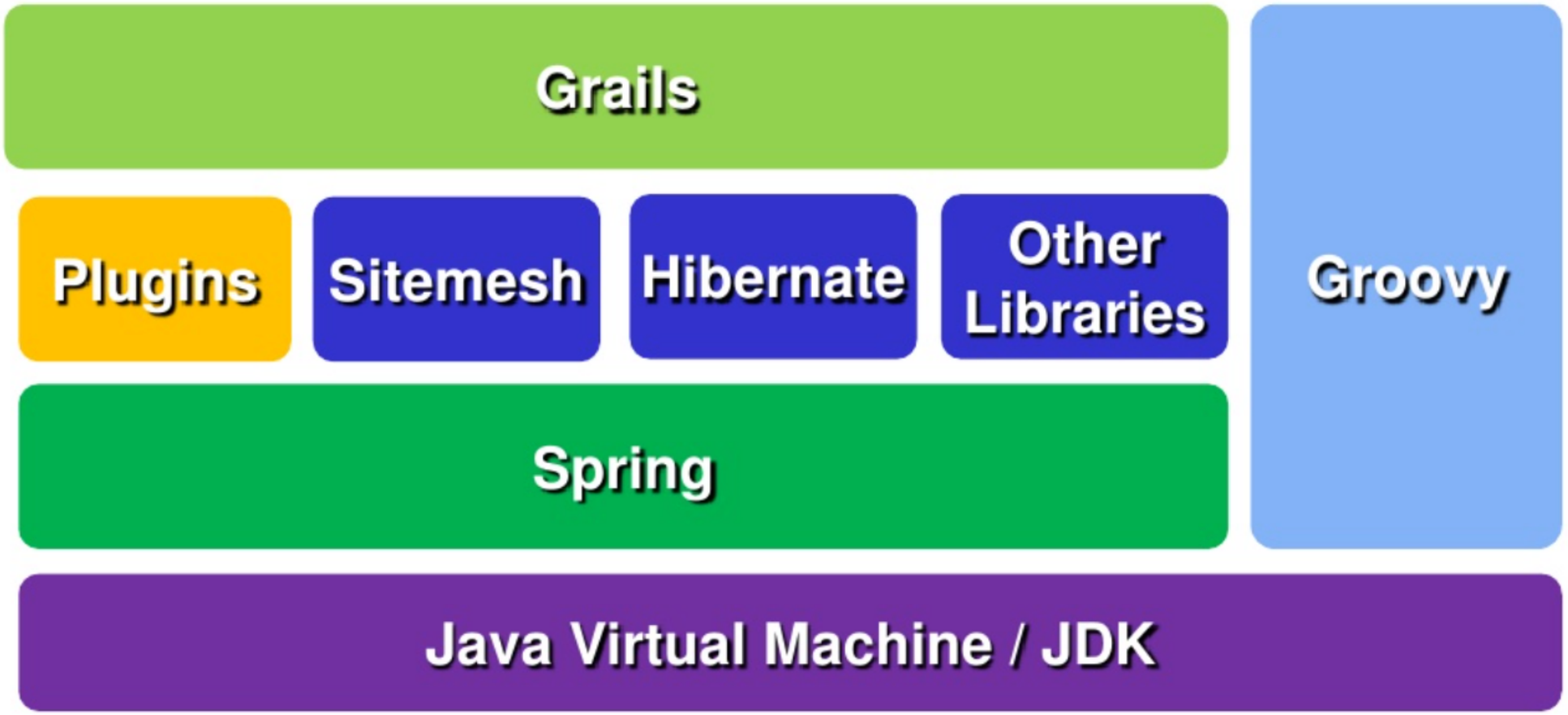

```
class HelloStudent2 {
  def name

  def sayHelloToStudents(){
    "Hello in Groovy CERN Spring Campus Student: ${name}"
  }
}

def helloStudent = new HelloStudent2(name: "Eloy")
println(helloStudent.sayHelloToStudents())
```

- Dynamic types using def keyword
- Everything is public unless defined otherwise
- Automatic getters and setters
- Semicolons at end-of-line are optional
- Variable interpolation through GStrings
- Return keyword is optional

```
public class HelloStudent
{
  private String name;

  public void setName(String name)
  {
    this.name = name;
  }

  public String getName(){
    return name;
  }

  public String sayHelloToStudents()
  {
    return "Hello CERN Spring Campus Student: " + name;
  }

  public static void main(String[] args)
  {
    HelloStudent helloStudent = new HelloStudent();
    helloStudent.setName("Eloy");
    System.out.println(helloStudent.sayHelloToStudents());
  }
}
```

Grails Architecture

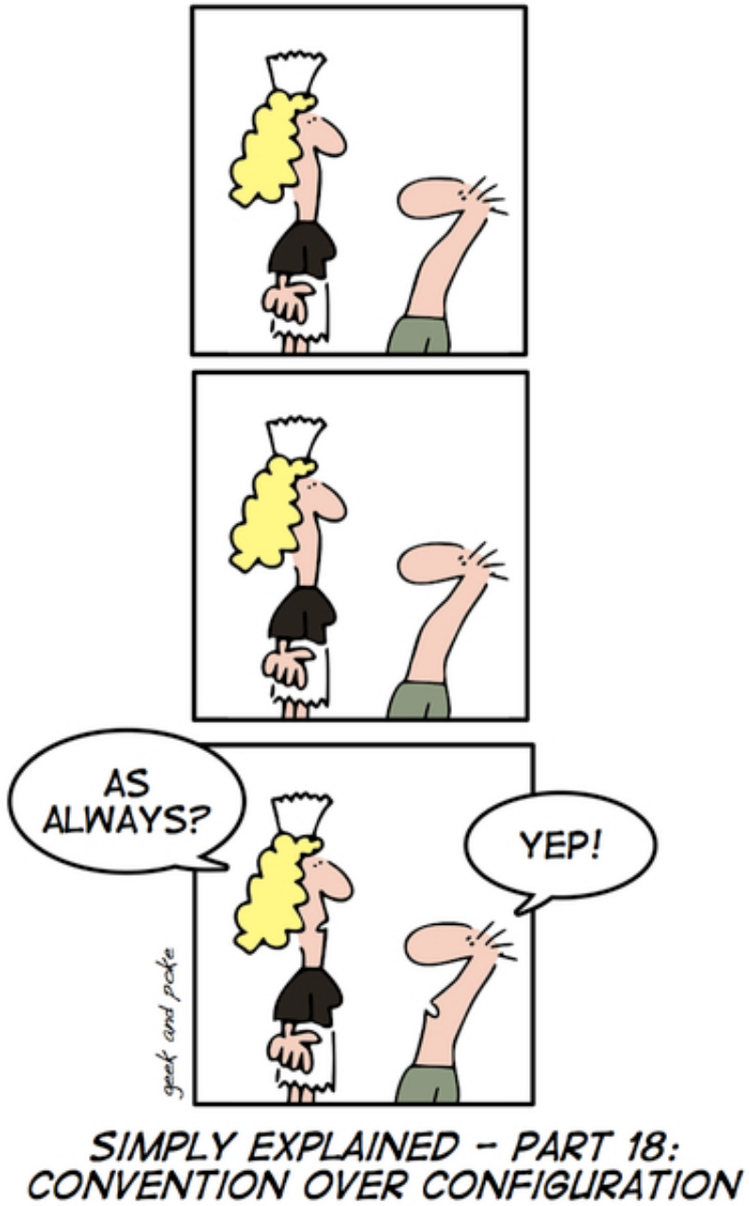
Grails Architecture



Stack of Technologies

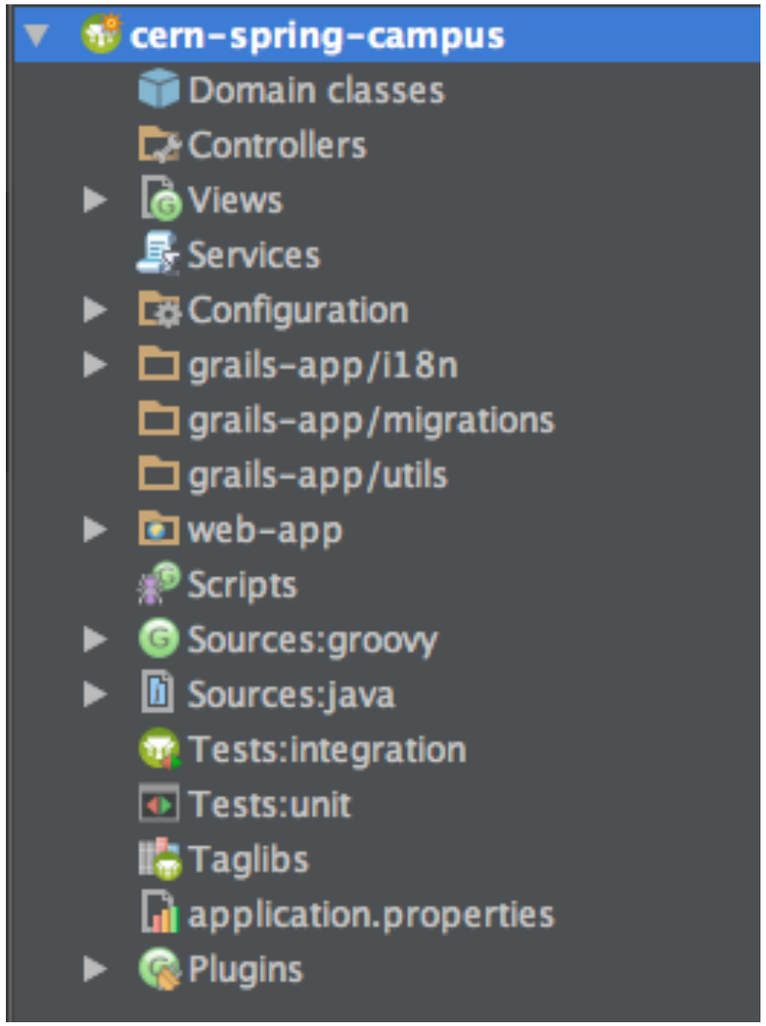
- **Java technologies:**
  - Spring:
    - Plumbing
    - Dependency Injection
    - Transaction management
  - Hibernate:
    - Object-Relational Mapping
  - Sitemesh
    - Layouts
    - Templates
    - Groovy Server Pages
  - jQuery
    - Built into Grails 2.0+
    - Other js libraries available as plugins

Conventions in Grails



Conventions in Grails

- **Every Project Layout in Grails is the same**



- **Default URL mappings:**

  http://<host>:<port>
  /appName
  /controller
  /action
  /id

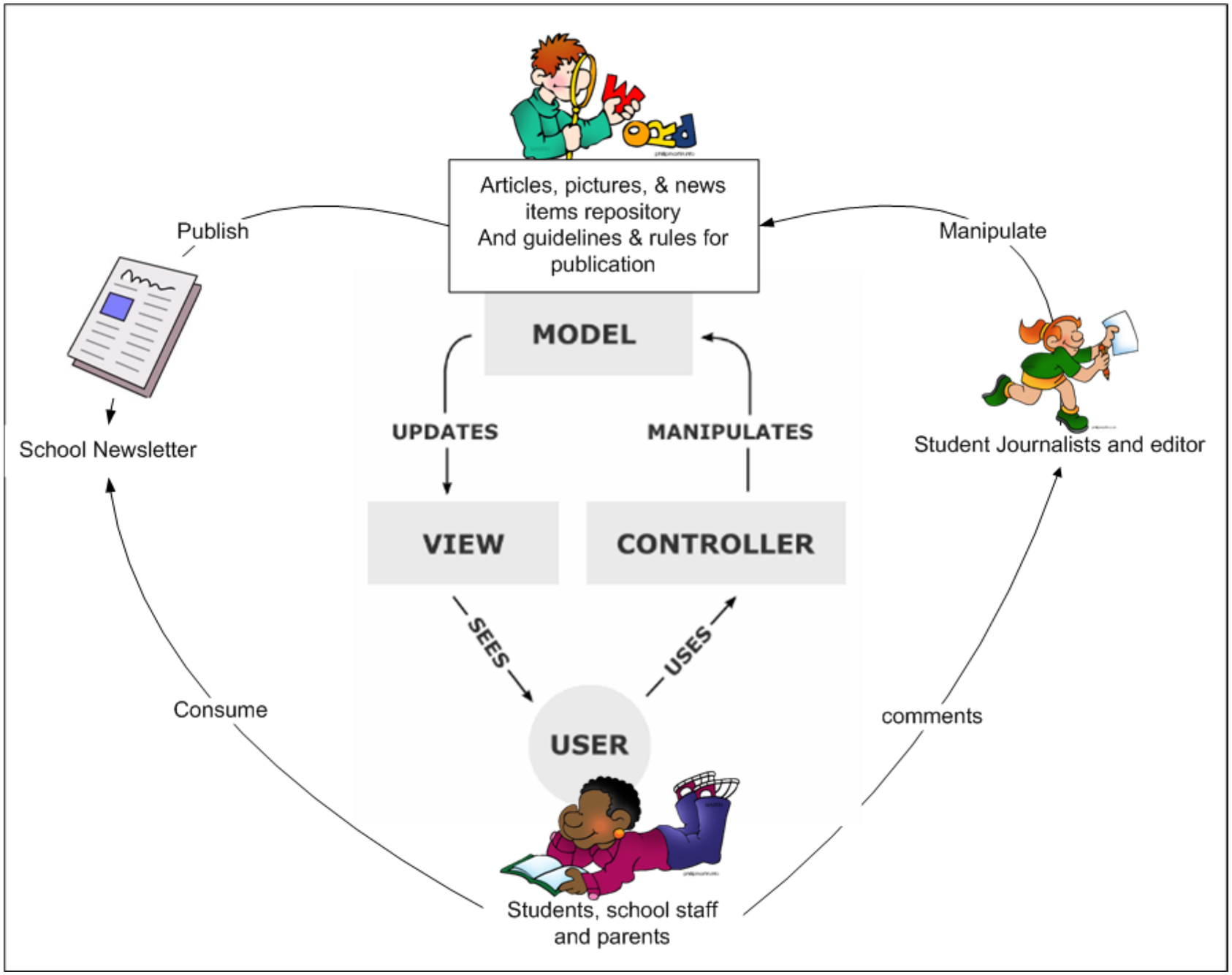
- **Input request parameters**

  map called "params"

Grails MVC

- **MVC pattern is a software architecture pattern that encourages separation of concerns, such as:**
  - Model – handles data representation and operations. For example, Person
  - View – handles how data is viewed by the user. For example, the HTML code for the list of customers.
  - Controller – handle the code that links a user to the system. It receives request or events from the user, prepare data and invoke proper business logic, and then return the view that represents the result of user actions.

Grails MVC


Articles, pictures, & news items repository And guidelines & rules for publication
Publish
Manipulate
MODEL
UPDATES
MANIPULATES
School Newsletter
Student Journalists and editor
VIEW
CONTROLLER
SEES
USES
Consume
comments
USER
Students, school staff and parents

Controllers

- **Responsible for handing cliente request (web requests)** ***http://grails.org/doc/latest/ref/Controllers/Usage.html***
- **Controller Actions**
  - **Input request parameters: "params" variable**
  - **3 ways to exit a controller action (The Three R´s):**
    - render
      - write to output stream: *REST Services…*

        render(text: "<xml>some xml</xml>", contentType: "text/xml", encoding: "UTF-8")
      - call write to template (partial GSP)

        def theShining = new Book(title: 'The Shining', author: 'Stephen King')
        render(template: "book", model: [book: theShining])
    - redirect
      - tell browser to go to new URL
      - original parameters are lost

        redirect(controller: "book", action: "list")

        redirect(url: "http://www.uniovi.es")
    - return
      - in groovy is optional
      - add map entries to the request
      - forward to *grails-app/views/controller/action.gsp*

Views

- **Groovy Server Pages (*.gsp)**
  - Similar to Java Server Pages but using Groovy
  - custom tag library
- **They are other possibilities using Plugins**

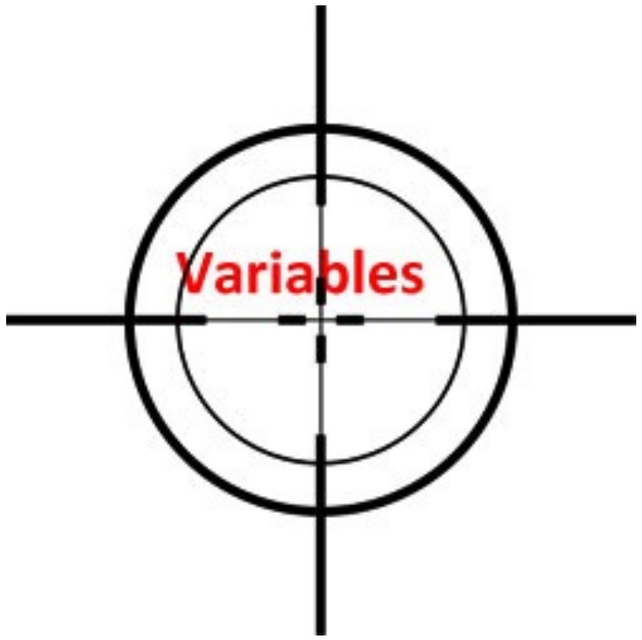
Modeling with GORM

- Grails Object Relational Mapping
  - DSL for Managing ORM - Hibernate
- Update, save, delete…
- Dynamic finders generated
  - using Groovy metaprogramming
- Define constraints

Person.findByNameAndSurname("Eloy", "Reguero Fuentes")

Scope

- **Objects can live at scopes:**
  - page
    - declared in a .gsp and we can only use it there
  - request
    - objects life during the whole round-trip of the request
  - flash
    - coming from Ruvy and Rails
    - variable lives in the current request plus one more
    - great for information messages and errors
  - session
    - objects can be in session: for example shopping carts
  - application
    - not so common, entirely life of the application

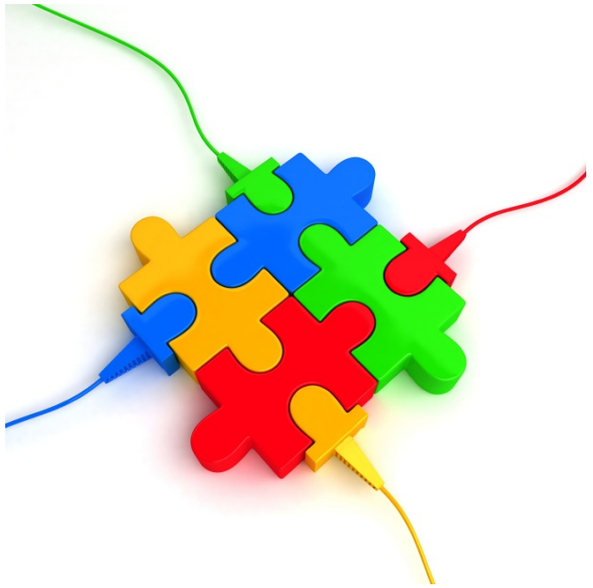


Testing

- **Unit tests**
  - No grails infrastructure (no dependency injection, no database,…)
    - Lots of mock objects are available to simulate Grails
    - HashMap to simulate Database
  - Very fast!
- **Integration tests**
  - Grails infrastructure available
    - Database access, Web Server, Dependency Injection…

Plugins

http://grails.org/plugins/



- **1079 Plug-ins available (different qualities…)**
- **Some good ones**
  - Quartz schedulers
  - Neo4J GORM
  - Resources
  - Spring security
  - Searchable
  - Google visualisation
  - …

Demo

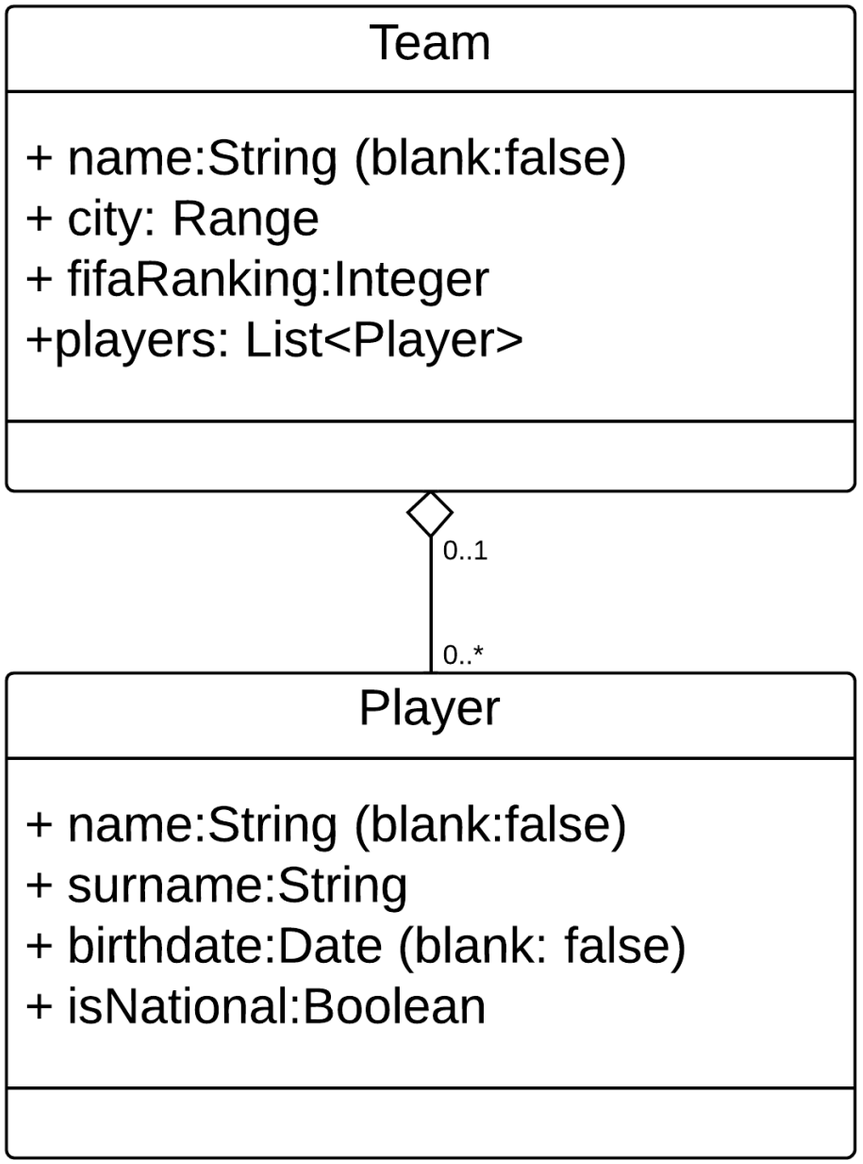
Part 1:

Player
+ name:String (blank:false) + surname:String + birthdate:Date (blank: false) + isNational:Boolean

Demo

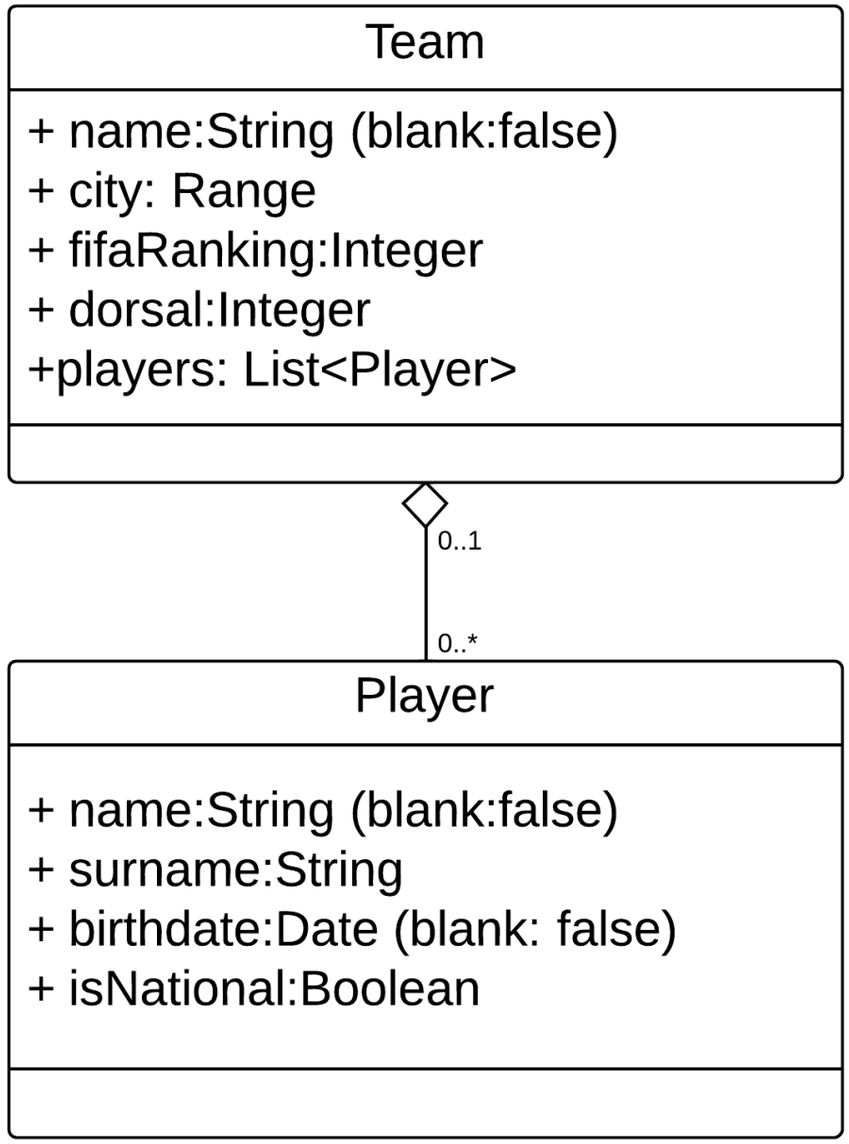
Part 2:


PRODUCT DEMOS


Team
+ name:String (blank:false)
+ city: Range
+ fifaRanking:Integer
+players: List<Player>
0..1
0..*
Player
+ name:String (blank:false)
+ surname:String
+ birthdate:Date (blank: false)
+ isNational:Boolean

Demo

Part 3:



New player will have the max number of dorsal +1

Your turn …