Grails – Fast, Robust and Plugin-based Web Development with Groovy

Christian Wulf and Florian Fittkau

29.01.2013

Outline

Outline

Motivation

- Web development is greatly supported in the Java environment, e.g., by JSF and custom JSF components
- However, you need to write very much code due to Java, XML files and the lack of scaffolding
- Moreover, much configuration has to be done to get your JSF app run as you like

the search is over.

the search is over.

- Web application framework for the JVM
- Open source
- Utilizes Groovy and "convention over configuration"
- Implements concepts of Rails to reduces complexity
- Builds on already established Java technologies like Spring and Hibernate

An Excerpt of Features

- Scaffolding
- Uncluttered source code, e.g., only unchecked exceptions
- Integrated ready-to-use support for AJAX calls, Spring, GAnt, i18n, Hibernate (GORM), ...
- Hundreds of plug-ins: Validation, less css, excel im-/export, authorization, Maven integration, job execution, ...
- Convention over configuration
- Customizable deployment environment (built-in: development, test, and production)

Outline

Language Features

- Dynamic language for the Java Virtual Machine
- Provides the ability to statically type check and statically compile your code
- Has additional power features inspired by languages like Python, Ruby and Smalltalk
- Seamlessly integrates with all existing Java classes and libraries

Syntax & Semantics

- Private attributes, public methods
- Implicit getter and setter definition
- Closures
- Additional convenient methods for standard Java classes such as `Object`, `List` and `Map`
- Multi-line strings
- String evaluation
- ...

Code Examples


```
1  def myList = [ 5, 7, 9, 12 ]
2
3  myList.eachWithIndex{ num, idx -> println "$idx: $num" }
4
5  myList.collect { ++it } // returns [ 6, 8, 10, 13 ]
6
7  def s = '''a multi-line
8  string in groovy printing out the
9  contents of mylist: ${myList}'''
10
11 def scores = ["Brett":100, Pete:"Hello", "Andrew":86.87934]
```

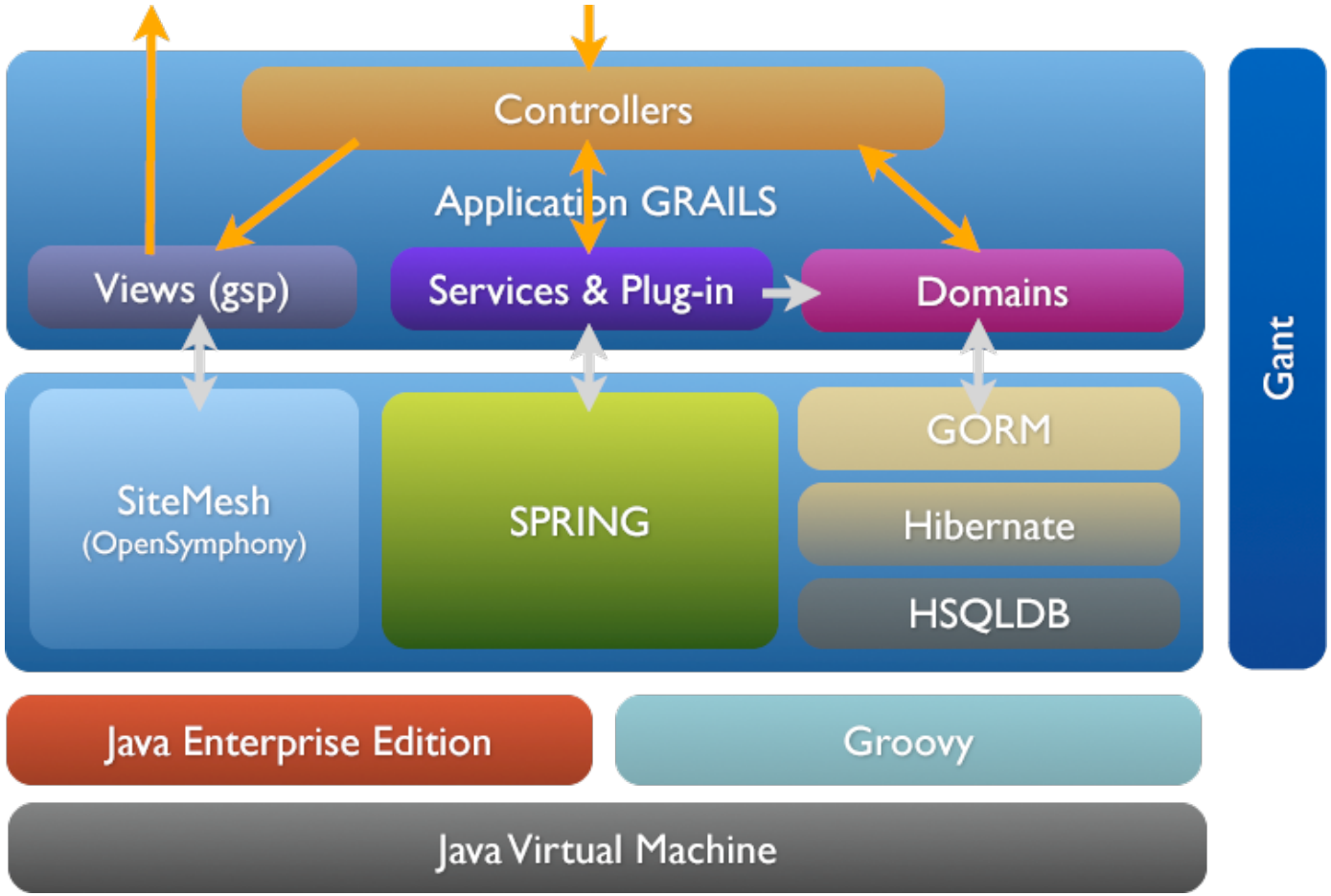
A Groovy Class


```
1  class Person {
2    def firstname
3    String lastname
4    private birthday
5
6    def sayHello() {
7      "Hello $firstname $lastname born in ${birthday.year}"
8    }
9  }
```

```
1  def p = new Person(firstname: "Bart", lastname: "Simpson")
```

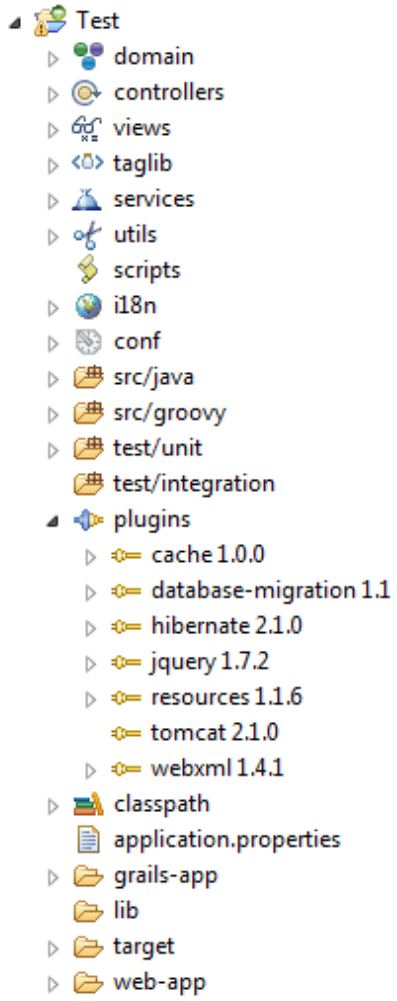
Outline

Schematic Architecture



[0] http://www-igm.univ-mlv.fr/~dr/XPOSE2009/Groovy_and_Grails/grails_architecture.php

Grails' Project Architecture



Model-View-Controller (MVC)

- **Domain** classes represent the model saved in the DB and simultaneously provide the interface to the DB
- **Controller** classes are (only) responsible for rendering and navigation
- **View** files (Groovy Server Page (GSP)) contains GSP-tags, javascript, html, and css

Further Artefacts

- **Services** contain re-usable Groovy/Java code used in controllers
- **Tag Libraries** represents custom GSP-tags with re-usable GSP code used in GSP files
- **Groovy & Java Classes** that should be not stored in the DB, but are used temporarily

Processing a Request

```
1  class UserController {
2    def authorizationService // automatically injected
3
4    def list() {
5      def users = User.findAll()
6      def currentUser = authorizationService.currentUser
7      render template: "usermgmt", model:[users:users,
           currentUsername:currentUser.username]
8    }
9  }
```

This example maps to the /user/list URI.

GORM

- Abstracted hibernate 3 in Groovy (and helper methods)

```
1  class Person {
2    String name
3    Integer age
4  }
5
6  def charlie = new Person(name: "Charlie", age: 20)
7  charlie.save()
8
9  def charlieFromDatabase = Person.findByName("Charlie")
10 charlieFromDatabase.delete()
```

Tests

- Unit tests
- Integrations tests (with Grails environment)
- Functional tests (with HTTP requests against the running application)
- Integrated Mocking API

Grails' Deployment

- `grails run-app`
- Grails listens for code changes (hot deploy)
- `grails war MyApp.war`
- `grails test-app`

Grails' Environments

- `grails prod run-app`
- `grails prod war MyApp.war`

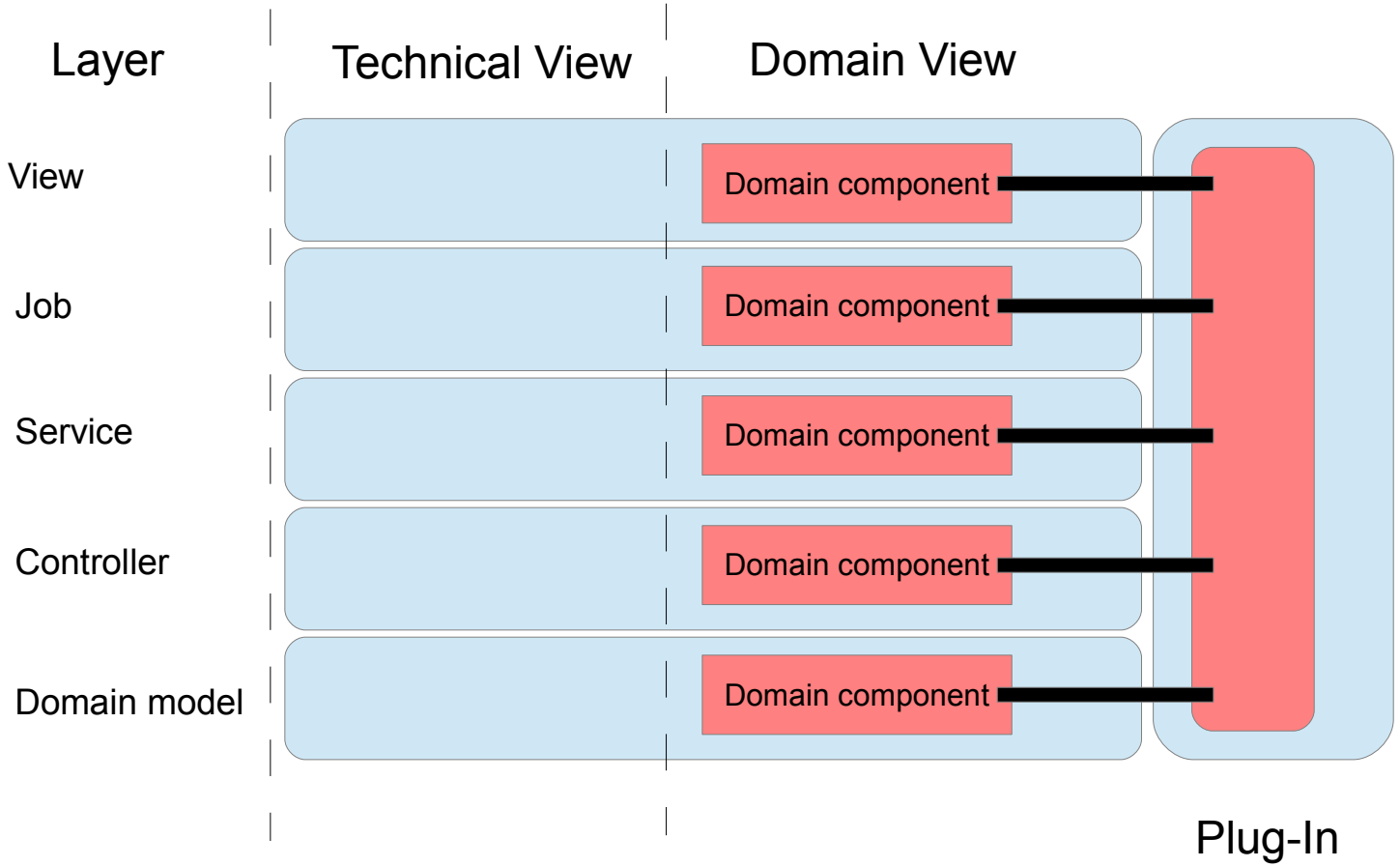
```
1 switch (Environment.current) {
2   case Environment.DEVELOPMENT:
3     configureForDevelopment()
4     break
5   case Environment.PRODUCTION:
6     configureForProduction()
7     break
8 }
```

Outline

Grails' Plug-in Concept

- A plug-in is a standard Grails project
- Integrated support for installing from and publishing to a Maven or SVN repository
- Built-in versioning
- Artifact API

Grails' Plug-in Concept



Grails' Dependency Resolution

- Integrated dependency resolution
- Maven, SVN

```
1 repositories {
2   mavenRepo name: "repo-releases", root: "http://localhost
        :8080/nexus/content/repositories/releases"
3 }
4
5 plugins {
6   compile ":mail:1.0.1"
7 }
```

A Selection of Freely Available Plug-ins

- quartz2 (job definition and scheduling)
- shiro (authorization etc.)
- excel-import
- mail
- wslite (REST and JSON)
- cached-resources
- browser-detection

Authorization – Shiro

- `grails shiro-quick-start`
- Generates User and Role classes

```
1 <shiro:hasPermission permission="printer:query">
2   ...
3 </shiro:hasPermission>
```

```
1 if (SecurityUtils.subject.isPermitted("printer:query")) {
2   ...
3 }
```

Resources – cached-resources

- Automatically generates hash for each resource (image, css, js...)
- Automatically sets parameter for caching on the client side

```
1 <img src="${resource(dir: 'images', file: 'logo.png')} />"
```

Outline

Tool Support

- Groovy & Grails Tool Suite (GGTS)
- CodeNarc[1]
- Jenkins plug-in for Grails commands

[1] http://codenarc.sourceforge.net/

Conclusions

- Combination of concepts from Rails with Groovy
- All Java libraries can be re-used
- Rapid web development (bunch of plug-ins)

References