Grails Application Development

Part 1 - Introduction

Objectives

- To get introduced Grails
- To learn the steps for creating a CRUD application

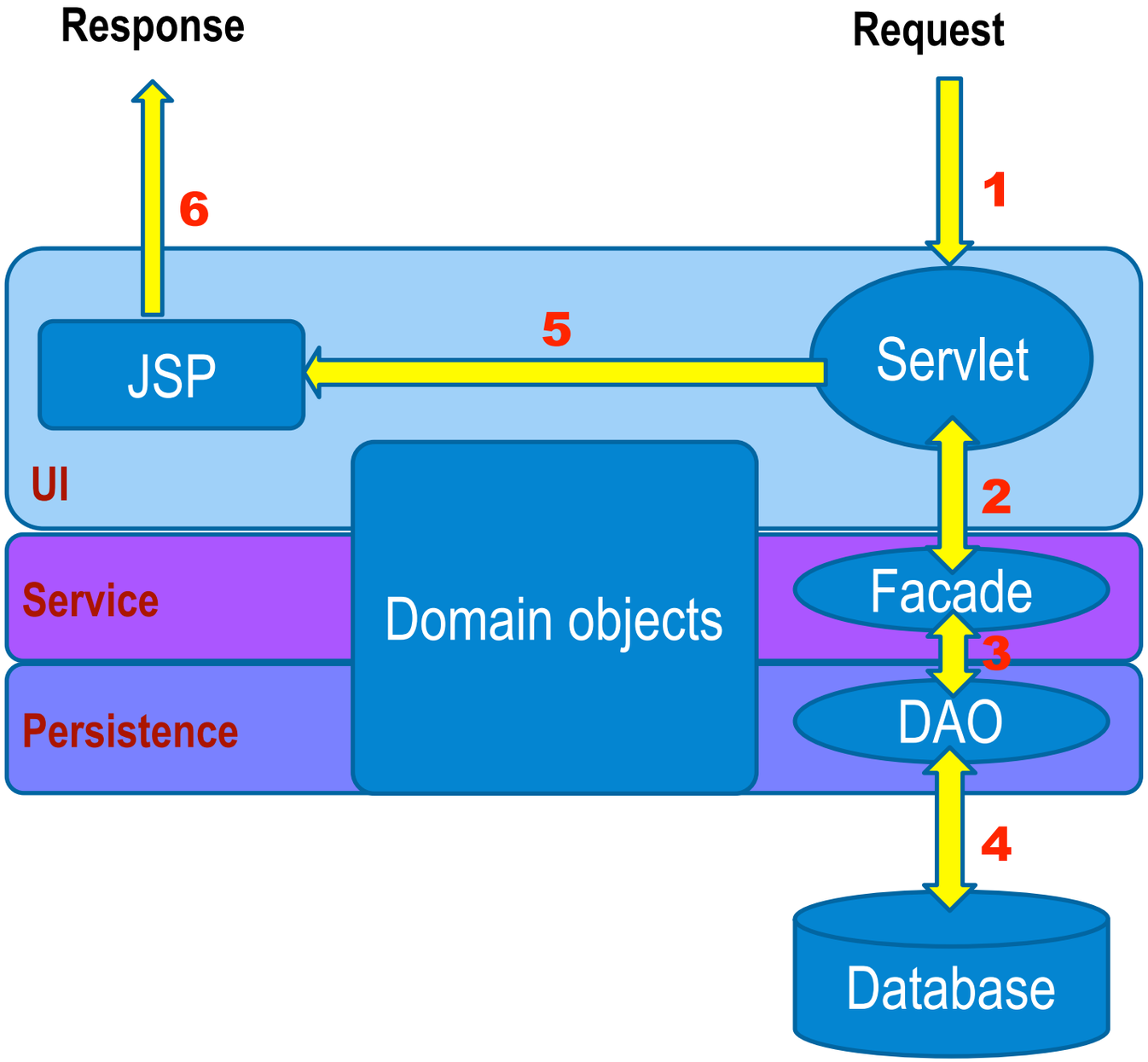
Session Plan

- Introduction to Grails
- Introducing MVC, DAO/Active record
- Installing Grails
- Running Grails
- Installing GGTS
- Create an application
- Running the application

What is Grails?

- Next generation web framework
- No! Not YAJWF (Yet Another Java Web Framework)!
- Rails – Web application framework + code generator - for Ruby language – Ruby on Rails
- Similar work using Groovy (dynamic agile language) called Groovy on Rails initially
- Later name changed to Grails – on request by Rails team
- Famous tag – The search is over!
- Before grails a bit of history of Java web application development

World of Servlets & JSPs



- Servlet receives the request
- Passes the info to Façade (validation and assembling data) for storage or retrieval
- Façade uses DAO to persist individual domain objects and retrieve domain objects from the database
- Servlet puts the domain objects in request, session or application scope and invokes JSP
- JSP renders the response to the user Browser

UI improvements

- Many a framework came like struts, Spring MVC
- Instead of developer writing each servlet a servlet was available in the frameworks
- Which called a piece of logic implemented by the developers for each type of request (knowing which implementation to invoke from the info contained in the URL)
- These implemented controller components interacted with the service layer to fetch or store data
- Then the framework servlet forwarded to the JSPs or let the controller components do that
- Developers had to maintain a config file to map URL -> controller component -> JSP

UI improvements

- Struts – the father of all MVC
  - Used inheritance for controller components
  - Components are tightly coupled
- Spring MVC – A lighter alternative
  - Used interfaces as a standard
  - Loosely coupled components with dependency injection
- Both Spring and Struts uses XML heavily for configuration
- Struts 2.0 has overcome a lot of negative aspects of Struts 1.0
- Spring nowadays do a lot of stuff with annotations than XML
- There are other frameworks today
- Wicket, JSF, Stripes etc.

Persistence improvements

- DAO using JDBC with SQL had a lot of boiler plate code
- Spring provided template methods and made developers focus on the main logic
- Object Relation Mapping frameworks gained momentum
- Instead of writing SQLs
  - Classes were designed after tables (attributes after columns)
  - Relationships like 1-M, M-M etc. were done with collection classes
  - Mapping syntax between class – table, attribute – column
  - DAOs used objects directly to save, update and delete and retrieved list of objects with out writing SQL
- Hibernate is the most popular and robust ORM available today

DAO Vs. Active Record

- DAOs were invented to keep domain objects decoupled from DB operations
- DAOs typically have Methods to
  - Retrieve domain objects (return type – List of domain objects)
  - Save a domain object (parameter – domain object)
  - Update a domain object (parameter – domain object)
  - Delete a domain object (parameter – domain object)

- Active records are domain objects which are capable of persistence – coupled with persistence code
- In Active Record pattern domain objects will have
  - Static methods for retrieval returning a list of objects
  - Instance methods for save, update & delete with no parameter

Dependency injection - Why?

- Controllers typically use service layer objects and service layer classes use DAO objects
- For example
  - A controller class will have instance variables to store service objects
  - So, controller must create them in its constructor or some other class responsible for creating controller should create service objects and set them up in controller
  - In either case whoever creates the service object should know the class name of the service object
  - No concurrency issues as most of these are stateless
- This not only increased coupling between components but also resulted in inconsistent object creation
  - Controller can be used only after it got an instance of service object

Dependency injection - How?

- Framework like Spring lets developer configure which class use which other class and how it should be setup (via constructor or setter)
- Spring has factories to construct controllers, Service objects and DAOs
- Developers don’t create them. They only request Spring to provide those objects
- Spring provides those object with dependent objects injected even solving problems like circular dependencies

What is Grails again?

- Grails uses frameworks like Spring and Hibernate
- But it does not mean that a truck load of XML files need to be maintained
- Grails uses convention over configuration
- That means you
  - Name components appropriately
  - Organize them properly and use them as prescribed
- This discipline will help you avoid using XML files

Techniques used by Grails

- Minus XML configuration files Grails uses
  - Spring dependency injection
  - Comes with a default front controller servlet
  - Lets you write controller components
  - Uses Hibernate for persistence
  - Uses Active record, you don't have to write any DB code! Grails injects them based on usage
  - And does not use DAO pattern
- By the time you write a single XML mapping file for persisting a class / object into a table you can create a whole working CRUD application for the class/table

Advantages of Grails

- Convention over configuration – No XML files
- Agile tool - generates test skeletons
- Uses Groovy for increased productivity
- Uses template based code generation for controllers and view pages
- Solid technology foundation - Spring, Hibernate, SiteMesh
- Java integration
- Open source & Lot of support from community
- Support of IDEs like Eclipse, Netbeans and from vmWare Spring Source Tool Suite & recently Groovy & Grails Tool Suite(GGTS)

Installing Grails

- Download grails-x.y.z.zip binary distribution
- Extract the zip file to a folder of your choice
- Setup an environment variable GRAILS_HOME to the folder containing unzipped grails
- Add %GRAILS_HOME%\bin to the “path” environment variable
- Type “grails - version” in the command prompt to check
- Type “grails” in the command prompt you will get a grails command prompt with auto-completion feature

Not needed if you are using GGTS. If you prefer command line then go ahead and install!

Running Grails - command line

- You can type “grails help” in the command prompt to get a help on grails commands
- You have commands to execute to
  - Create an application
  - Create a domain object
  - Create a controller
  - Create views
  - Running the application etc.
- We will use Groovy & Grails Tool Suite (GGTS) IDE to create and run Grails applications

Installing GGTS

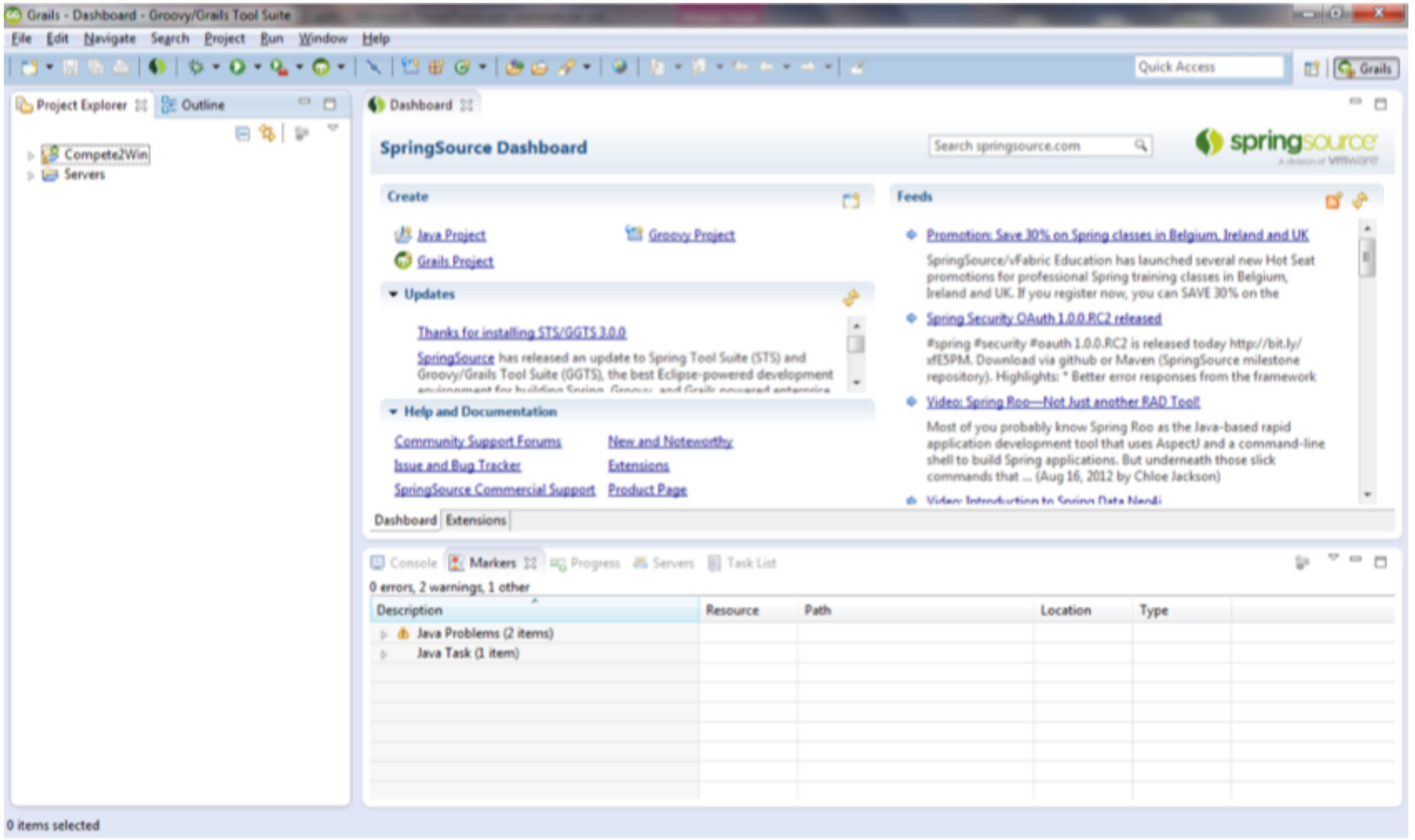
- GGTS is a tool built on top of eclipse
- You can download a latest version from Springsource web site
- Unzip to a folder of your choice
- GGTS comes with a version of Grails and Tomcat Server
- Go to “ggts-x.y.z.RELEASE” folder and create a shortcut to the GGTS.exe found in that folder on the desktop
- Open GGTS.ini file for edit and change the lines highlighted to suite your configuration

Setting up GGTS

```
-startup
plugins/
org.eclipse.equinox.launcher_1.3.0.v20120522-1813.
jar
--launcher.library
plugins/
org.eclipse.equinox.launcher.win32.win32.x86_1.1.2
00.v20120522-1813
-data
D:\Dev\Grails\GGTS3.0
-vm
C:\Software\Java\jdk1.7.0_03\bin\javaw
...
```

Running GGTS

- Double click the shortcut to GGTS.exe on your desktop
- GGTS will open!



Demo – Grails Project

- Create a new Grails project (LearnAround)
- Explore the folder structure
- Create a domain class
- Enter code for Domain class – in the next slide
- Generate controllers
- Scaffold controller
- Run the application
- Provide validation constraints for fields – In the next slide
- Run the application to show error messages

Demo – Domain class

- Create-domain-class com.ardhika.learn.User
- User.groovy

```
package com.ardhika.learn
class User {
      static constraints = {
      }
}
```

- Edit this file to include fields

Demo – Domain class

- Changed code of domain class User.groovy

```
package com.ardhika.learn

class User {

    static constraints = {
    }

  String firstName
  String lastName
  String userName
  String password
  String email
  String homePage
  String bio

  String toString() {
      "$lastName, $firstName"
  }
}
```

Controller

- Create controller - <package>.User - Specify only domain class

```
package com.ardhika.learn

class UserController {

    def index() { }
}
```

- Edit this to

```
package com.ardhika.learn

class UserController {

    def scaffold = User
}
```

Run ...

- Fields are not in proper order
- No validation
- Beautiful interface
- Standard navigation
- Predefined CSS

Demo – Domain constraints

- User.groovy

```
class User {

    static constraints = {
     firstName blank:false, size:3..20
     lastName blank:false, size:3..20
     userName blank:false, size:6..12
     password blank:false, password:true
     email blank:false, email:true
     homePage url:true, nullable:true
     bio maxSize:150, nullable:true
    }

  String firstName //and other fields ...

}
```

Run it Now!

- How easy?

Thank You!

Bala Sundarasamy
bala@ardhika.com