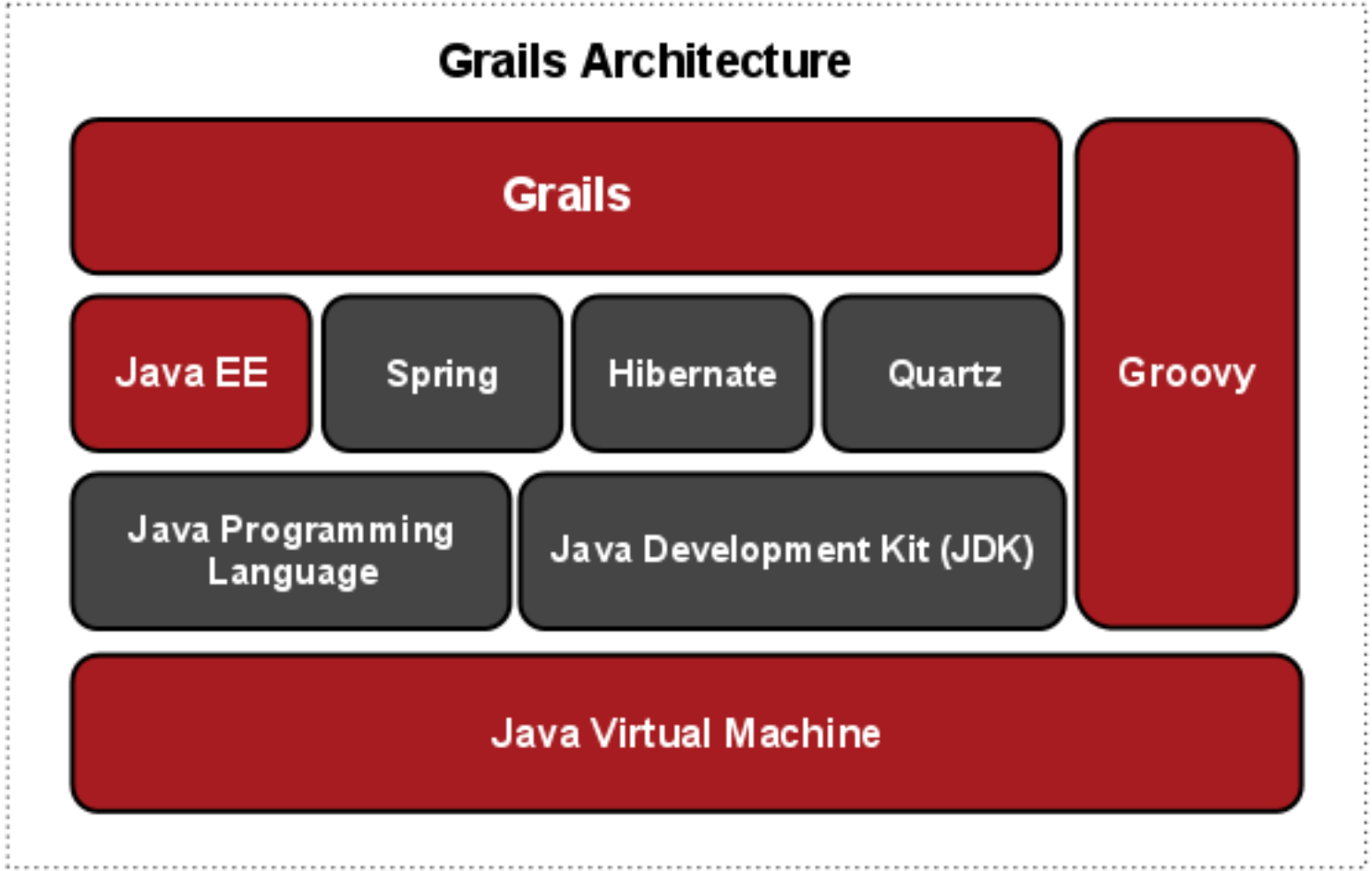
Grails

By: Roger Chen & Kevin Sheu

What is Grails?

- Open source web application framework that uses the Groovy language
- known as “Groovy on Rails”
- like ruby on rails, django for python


Grails Architecture
Grails
Java EE
Spring
Hibernate
Quartz
Groovy
Java Programming Language
Java Development Kit (JDK)
Java Virtual Machine

What is Groovy?

- is an dynamic language for the Java Virtual Machine
- integrates all existing Java classes and libraries - most java code is syntactically valid Groovy
- object oriented, but offers functional programming features
- weak typing

Closures

- Anonymous code blocks that can be passed around code
- piece of code that can be defined and then executed at a later point
- several special properties as like implicit variables and support for free variables

Closure Example

```
def printsum = {a,b -> print a+b}
printsum(3,4) // prints 7

def const = 5
def addfive = {num -> num + const}
println(addfive(10)) // prints 15
```

Closures (cont.)

If you have a closure that takes a single argument, you may omit the parameter definition and use the keyword `it`:

```
def say = {print it}
say(“hi”) // prints hi
```

Operators: == and ===

== evaluates data equality

```
a = true, b = 2
println a == b // prints true
```

=== evaluates object identity

```
object a, b
a = b
println a === b // prints true
```

Safe Navigation Operator: ?.

- used to avoid NullPointerException

instead of using:

```
if (obj != null && obj.val != null)
```

use:

```
if (obj?.val != null)
```

- val is only accessed if obj is not null

Spread-Dot: *.

- used to invoke an action on all items of an aggregate object
  - instead of :

```
parent.collect {child -> child?.action}
```

  - use:

```
parent*.action
```

```
assert [‘cat’, ‘elephant’]*.size() == [3,8]
```

Spaceship Comparisons: <=>

returns a negative number if left side is less than right side, returns 0 if equal, return positive number otherwise

```
public int compare (a,  b) {
   return a <=> b;
}
```

Elvis: ?:

Usage: `<condition> ? <expr1> : <expr2>`

returns the value of the left expression (`expr1`) if condition evaluates to be true

```
String name = (person.getName() != null) ? person.
getName() : “”

String name = person.getName() ?: “”
```

Other features

- Omitting dots and parentheses

Java: `move(left);` Groovy: `move left`

- String interpolation

```
BigDecimal account = 10.0
def text = "Your account shows currently a
balance of $account, if you add 1 dollar,
the balance becomes ${account+1}"
println text
```

Simplifications

- code can be made far more compact than Java

Java:

```
for (String it: new String[] {“Rod”, “Carlos”, “Chris”})
    if (it.length() <= 4)
        System.out.println(it);
```

Groovy:

```
["Rod", "Carlos", "Chris"].findAll{it.size() <= 4}.each{println it}
```

Regular Expression

- =~ looks for the pattern and returns true if exists in string

```
assert “hihihihi” =~ /hi/    //returns true
```

- match: ==~

```
assert “2009” ==~ /\d+/      // returns true
assert “holla” ==~ /\d+/   // returns false
```

Beans

- Groovy's version of JavaBeans
- implicitly generates accessor and mutator methods

```
class AGroovyBean {
  String color
}
def myGroovyBean = new AGroovyBean()
myGroovyBean.setColor('baby blue')
assert myGroovyBean.getColor() == 'baby blue'
myGroovyBean.color = 'pewter'
assert myGroovyBean.color == 'pewter'
```

Quiz

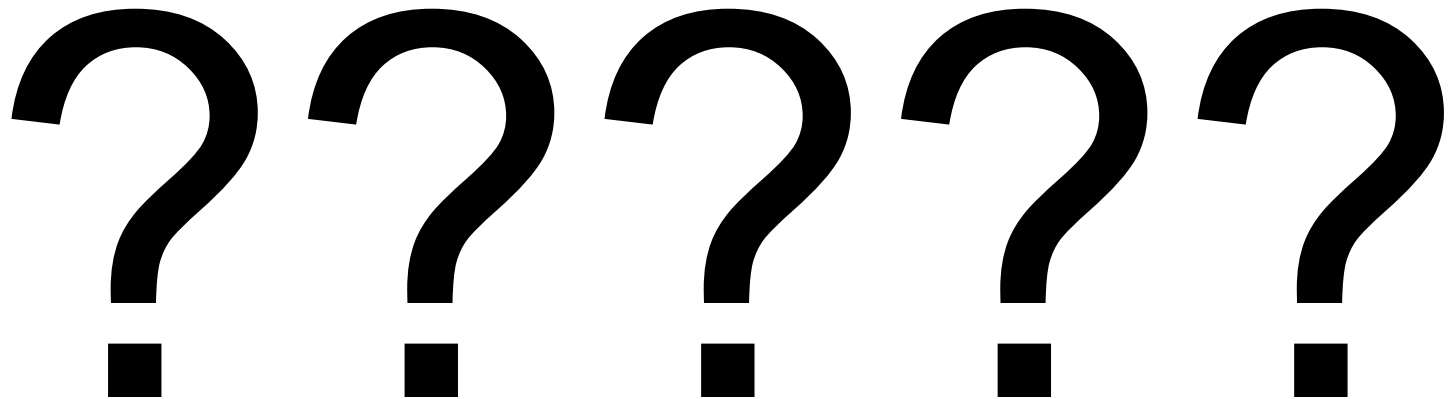
1. Name a difference between Groovy and Java
2. Groovy’s main features as a functional programming language

Groovy Activity: What is printed?

```
class Employee {
       def name, salary
       boolean manager
       String toString() { return name }
}

def emps = [new Employee(name:'Guillaume', manager:
true, salary:200),
       new Employee(name:'Graeme', manager:true, salary:
200),
       new Employee(name:'Dierk', manager:false, salary:
151),
       new Employee(name:'Bernd', manager:false, salary:
50)]

def managers(emps) {
       emps.findAll { e -> e.isManager() }
}

println managers(emps) // [Guillaume, Graeme]

def highPaid(emps) {
       threshold = 150
       emps.findAll { e -> e.salary > threshold }
}
```

```
println highPaid(emps) // [Guillaume, Graeme, Dierk]

def paidMore(amount) {
       { e -> e.salary > amount}
}
def highPaid = paidMore(150)

println highPaid(emps[0]) // true
println emps[0..2] == emps.findAll(highPaid)

def filename = 'test.txt'
new File(filename).withReader{ reader ->
doSomethingWith(reader) }

def readersText
def doSomethingWith(reader) { readersText = reader.text
}

assert new File(filename).text == readersText
```

QUESTIONS?

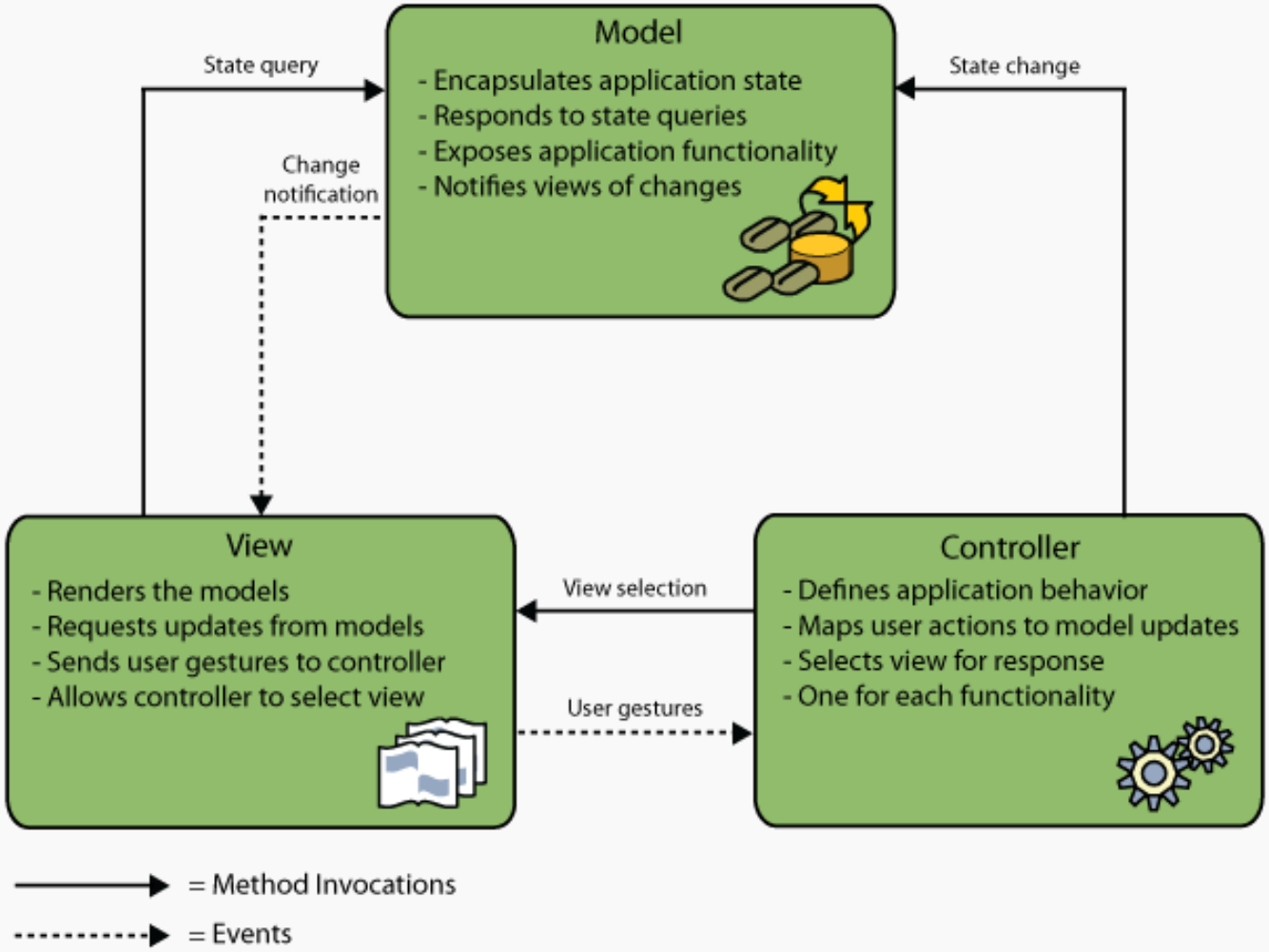


Grails: The Basics

- Built on Spring and Hibernate
- integrates easily with code already written in java
- provides its own testing framework that makes writing unit and integration tests easier
- designed according to the MVC paradigm

Model - View - Controller (MVC)

- Models: data behind an application, reacts to requests from the controller
- View: what the user sees and interacts with, sends requests to the controller
- Controller: reacts to changes in both the model and view


Model
- Encapsulates application state
- Responds to state queries
- Exposes application functionality
- Notifies views of changes
State query
State change
Change notification
View
- Renders the models
- Requests updates from models
- Sends user gestures to controller
- Allows controller to select view
Controller
- Defines application behavior
- Maps user actions to model updates
- Selects view for response
- One for each functionality
View selection
User gestures
= Method Invocations
= Events

Controller

- Heart of every grails application - takes input, interacts with business logic and data model, and routes the user to the correct page

Your first app

- `grails create-app <app-name>`
- `/web-app`: Place static resources, e.g. images, style sheets, JS files
- `/grails-app`: Place grails artifacts, e.g. controllers, domains, views
- `/src`: Place Java/Groovy helper files

Activity: Sample Application

- Initialize grails app:

```
grails create-app msgapp
```

- run the app:

```
cd msgapp
grails run-app
```

- point browser to http://localhost:8080/msgapp
- stop application with Control-C

Activity: Controller

Create a controller: `grails create-controller message`

Create a page: (in `MessageController.groovy`)

```
class MessageController {
    def index = {
       render "<h1>Real programmes never press backspace</h1>"
    }
}
```

Run the app (`grails run-app`) and go to http://localhost:8080/msgapp/quote/index

Activity: Controller

On your own: add another page with a different URL and navigate to that page

Views

- encoding HTML directly in the code is a bad idea, you need to access the source code directly to change the pages
- move display logic to a separate file, called a view

GSP (Groovy Server Pages)

- Default way to render views to user
- Easily combines static/dynamic content on same page
- Each GSP is associated with a controller
- URL’s correspond to actions found in the controller

```
/appname/controllername/actionname/params.id
```

GSP continued

- GSP tags
- Implicit objects
- *render*
- Databinding
- Command objects

View vs Template

- Template = reusable part of view
- Grails convention: Put _ before name of view to identify as a template
- Use `<g:render>` tag

Apply template to ...	Example of convention or technique
All actions in a controller	Create layout in /layouts/post.gsp
A specific action in a controller	Create layout in /layouts/post/list.gsp
A portion of a target page	Include tag in target page: `<g:applyLayout name="postFragment">Hi</g:applyLayout>`
Override any conventions explicitly for a single page	Include tag in target page: `<meta name="layout" content="vanilla"/>`

Activity: Views

In file `MessageController.groovy`: add another method

```
def hello = {
    def phrase = “Hello World”
    [message: phrase]
}
```

Creating a GSP

create a file `hello.gsp` in msgapp/views/message

```
<html>
<head>
    <title>Message:</title>
</head>
<body>
    <p>${message}</p>
</body>
</html>
```

- `${phrase}` format is called GSP expression language, the `${ }` shows the contents of the variable

Activity

Create another view !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!

ask roger if you need help

QUIZ

1. Explain how model interacts with view.
2. Why is it better to use views????

GORM

- Grails Object Relational Mapping
- Process of getting objects in and out of database (CRUD)
  - Create `save()`
  - Read `get()`
  - Update `save()`
  - Delete `delete()`

Domain Classes

- The M in MVC

```
grails create-domain-class
zynx.Song

class Song {
    String title
    String artist
    Integer duration
}
```

- CRUD operations

```
def song = new Song(title:'
Happy', artist: 'Pharrell',
duration: 353)
song.save()

def s = Song.get(1)
assert 1 == s.id

s.title = "Number One"
s.save()
```

Domain Classes

- Constraints
  - `blank` vs `nullable`
  - Custom Validation
  - `importFrom`
  - `matches`

```
class Song {
    String title
    String artist
    Integer duration

    static constraints = {
        title blank:false
        artist(blank:false,
validator: {singer, song ->
    return singer != song.title)
        duration nullable: true
}
```

Relationships

- One-to-one (1:1)
  - General
  - BelongsTo
  - HasOne

```
class Car {
    static hasOne =
[engine: Engine]
}
class Engine {
    Car car
}
```

```
class Car {
    Engine engine
}
class Engine {
    static belongsTo = [car: Car]
}
```

```
class Car {
    Engine engine
}
class Engine {
    static belongsTo = Car
}
```

Relationships

- One-to-many (1:M)
- `<1>addTo<M>()`
- `<1>removeFrom<M>()`
  - `Album.addToSongs()`
- Keeping “many” side sorted
  - `static mapping = {...}`

```
class Song {
    static belongsTo
= Album
}
```

```
class Artist {
    static hasMany =
[albums: Album]
}
```

```
class Album {
    static hasMany =
[songs: Song]
    static belongsTo
= [artist: Artist]
}
```

Relationships

- Many-to-many (M:N)

```
class Song {
    static hasMany =
[genres: Genre]
}
```

```
class Genre {
    static hasMany =
[songs: Song]
    static belongsTo =
Song
}
```

- Cascading in 1:M and M:N

Relationships

- Self-referencing
  - Just a special case of 1:M
  - `static hasMany = [following: Class]`
- Inheritance
  - table-per-hierarchy
  - table-per-subclass

```
class Person {
    String name
    Integer age
}
class Employee extends Person {
    String employeeNumber
    String companyName
}
```

Querying

- BootStrap
- Dynamic finders
  - `Object.findByProperty(), findAll()`
- Where queries
  - `<domainClass>.where{<criteria>}.<execution>()`
  - `def answer( String title) {`

```
Song.where{title =~ "%${title}%"}.list()
}
```

Querying

- Criteria queries
  - <domainClass>.withCriteria{<criteria>}
  - def fetchSongs(String title) {

```
def songs = Song.withCriteria {
    and {
        ilike “title”, “%${title}%”
    }}}
```

Filters

- Allows you to intercept requests
- Allows you to perform business logic before or after controller action fires
- Body: `before, after, afterView`

```
class SecurityFilters {
    def filters = {
        loginCheck(controller: '*', action: '*') {
            before = {
                if (!session.user && !actionName.equals('login')) {
                    redirect(action: 'login')
                    return false
                }
            }
        }
    }
}
```

Scaffolding

- Easily generate CRUD interfaces

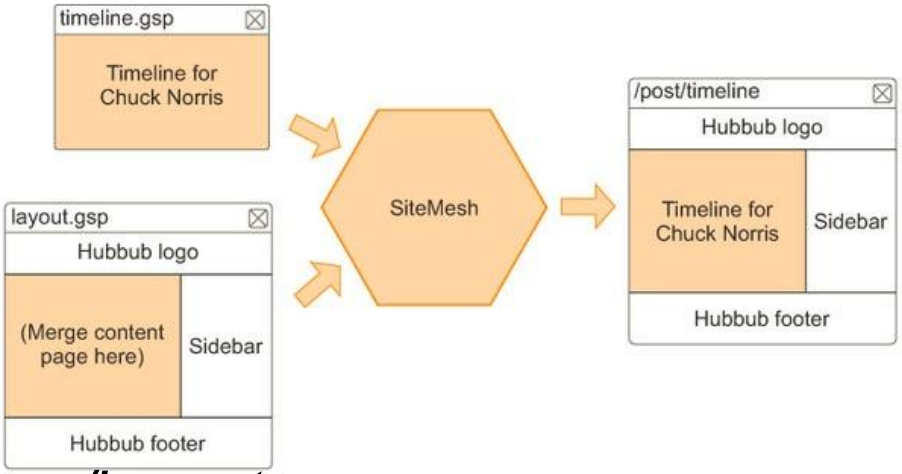
  ```
  grails create-scaffold-controller <scaffold>
  ```

- Validation errors

```
song.title.blank.message = {0}
cannot be empty

{className}.{field}.
{errorCode} = Error message
```

Scaffolding

- Adding CSS
  - Stored in */webapp*
  - SiteMesh
  - Skins
- Adding Layouts
  - Stored in */grails-app/views/layouts*
  - Layout filename should match controller name

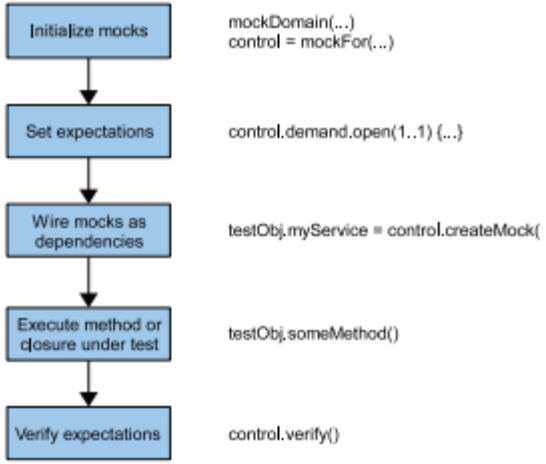


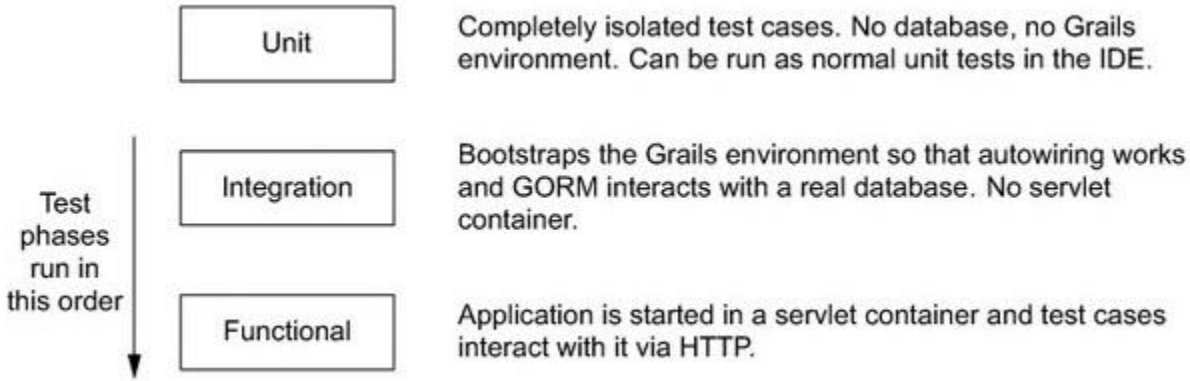
Ajax/Javascript

- `<g:javascript library="jquery"/>`

Testing

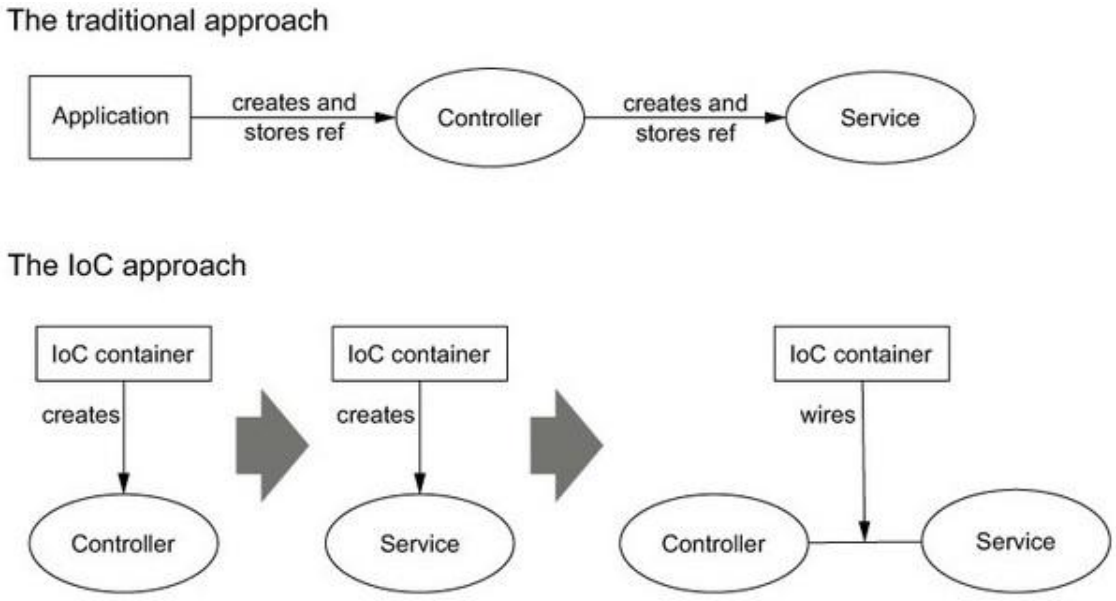
- Mocking
- Unit Testing
- Integration Testing
- Functional Testing





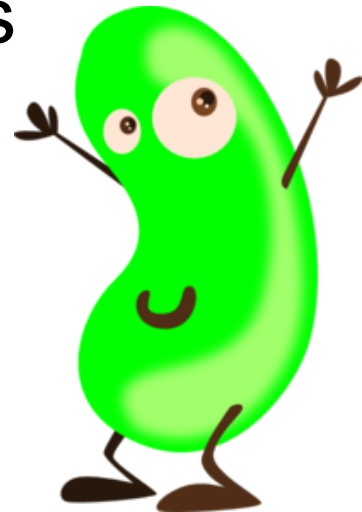
Security/Spring

- Basic security practices
  - Validating user input
  - Data binding
  - Escape output
- Spring
  - Access control
  - Dependency Injection



Spring

- Beans
  - Standard artifacts become Beans, e.g. controllers
  - Autowiring: objects wired together by name
- Make most important singleton objects Spring beans



Spring

- Protecting URLs

```
grails.plugin.springsecurity.securityConfigType = "InterceptUrlMap"

grails.plugin.springsecurity.interceptUrlMap = [
    '/': ['permitAll'],
    '/post/global': ['permitAll']
    '/user/**': ['permitAll']
    '/login/auth': ['permitAll'],
    '/**/js/**': ['permitAll'],
    '/**/css/**': ['permitAll'],
    '/**/images/**': ['permitAll'],
    '/**': ['isFullyAuthenticated()']
]
```

Tells Spring Security to use static URL rules

Unrestricted access

Open access to static resources

Everything else requires authenticated user

- Useful methods
- Tightening access

```
ROLE_USER,
ROLE_ADMIN
```

REST/API

- GET: Retrieves a resource
- POST: Creates a new resource
- PUT: Updates an existing resource

or creates a new one with known ID

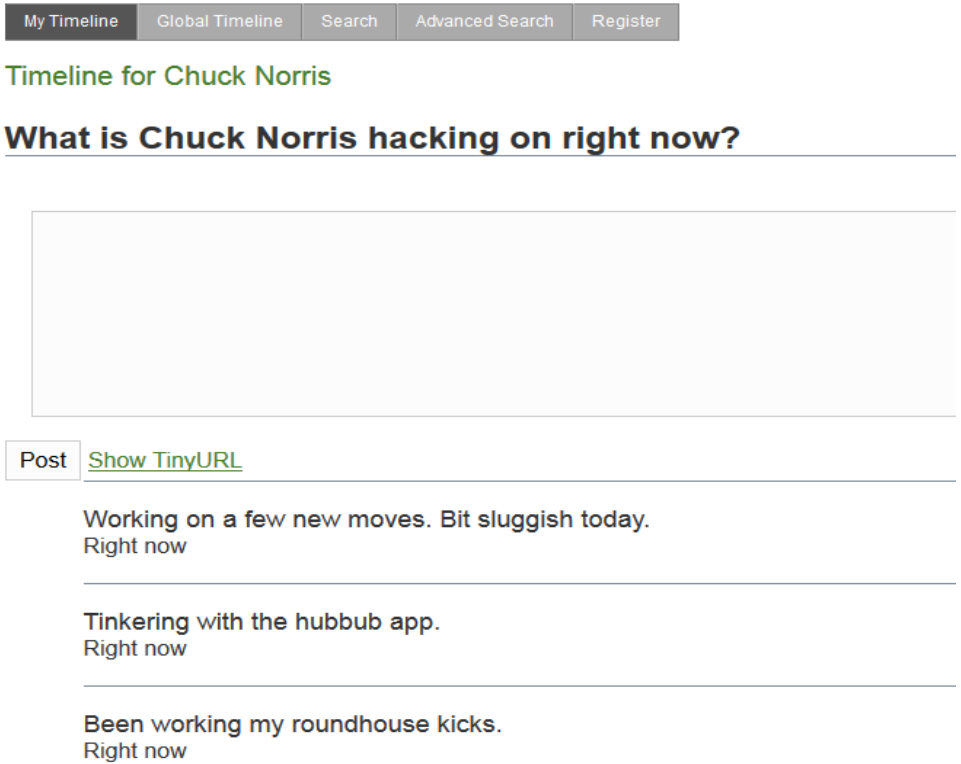
- DELETE: Removes a resource

REST/API

```
"/product" {
    controller = "product"
    action = "list"
}
```

- URL Mappings
- Each URL should represent only one resource and always point to same one
- What makes a good API?
- Relation to AngularJS

Final Activity



A problem has been detected and windows has been shut down to prevent damage to your computer.

The problem seems to be caused by the following file: SPCMDCON.SYS

PAGE_FAULT_IN_NON_PAGED_AREA

If this is the first time you've seen this Stop error screen, restart your computer. If this screen appears again, follow these steps:

Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any windows updates you might need.

If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup Options, and then select Safe Mode.

Technical information:

*** Stop 0x000000050 (OxFD3989C2, 0X0000000001,)xFBFB7658, 0x00000000)

*** SPCMDCON.SYS - Address FBFB7658 base at FBFE50000, DataStamp 3d6dd67c