Building Web Applications

Fall Forecast 2009

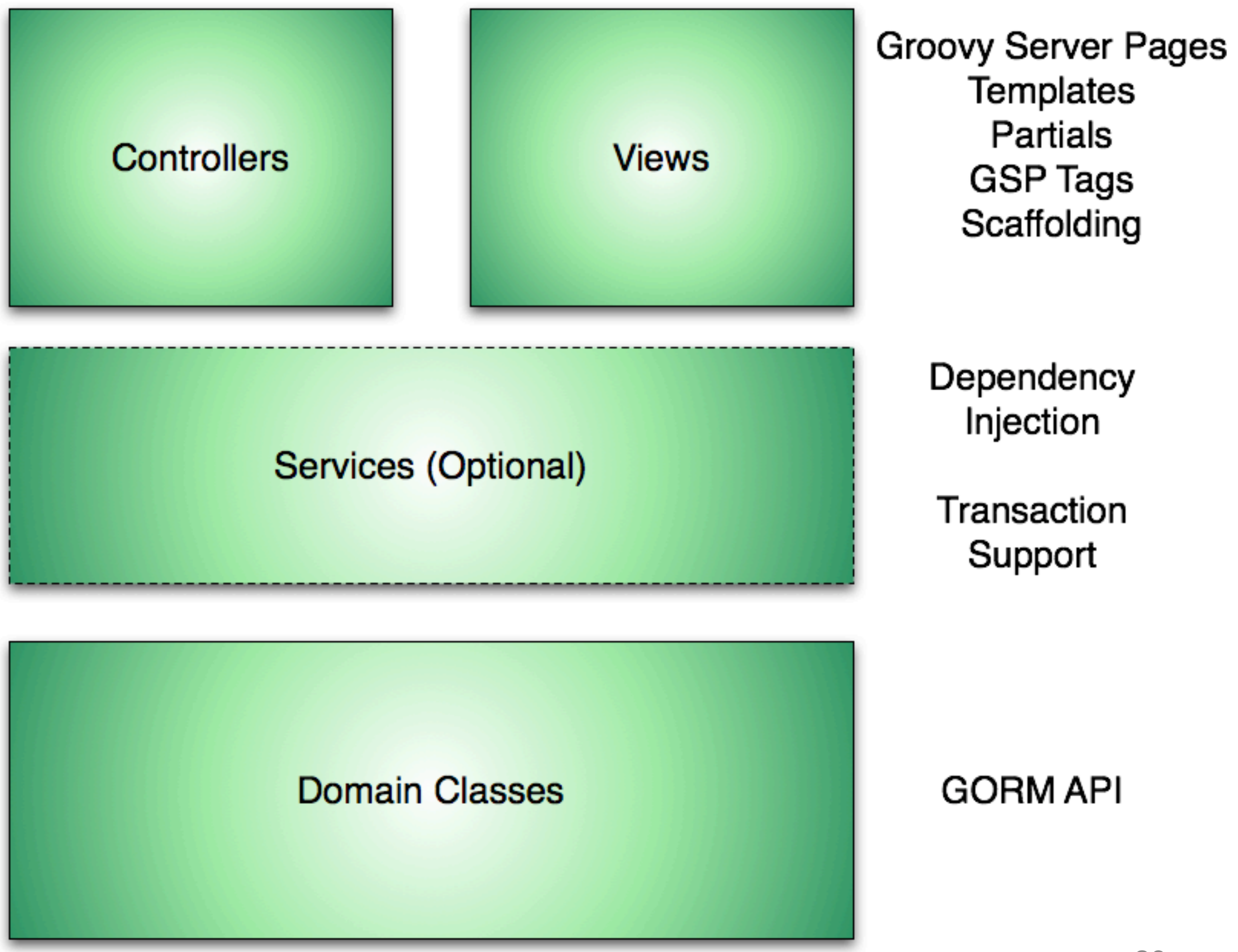
Agenda

- The Grails Framework
- Components
  - Domains
  - Controllers
  - Views - Groovy Server Pages

The Grails Framework

Some embedded frameworks:

Spring
Spring MVC
Spring Webflow
Hibernate
Criteria API
Sitemesh
Quartz



Key Grails Classes

- **Domain Class**
  - A class representing an object in your domain (database)
- **Controller**
  - A class that operates on URLs submitted to the web site
- **View**
  - A Groovy Server Page (GSP) designed to render the content based on a specific request

CHARIOT SOLUTIONS

Grails Controllers

- Analogous to a Struts Action
- Backed by Spring Controllers
- Each method handled by the Controller is a closure

```
grails generate-controller com.chariot.Customer
```

CHARIOT SOLUTIONS

Scaffolding

- Allows quick structure of your application
- Typically used for quick demo or prototype
- Controller code:

```
class CustomerController {
def scaffold = Customer
}
```

CHARIOT SOLUTIONS

Scaffold @ Runtime

- Views Generated
- Actions Generated
  - list
  - show
  - edit
  - delete
  - create
  - save
  - update

Controller Methods

```
def save = {
    def customerInstance = new Customer(params)
    if (customerInstance.save(flush: true)) {
        flash.message = "${message(code: 'default.created.message', args: [message(code: 'customer.label', default: 'Customer'), customerInstance.id])}"
        redirect(action: "show", id: customerInstance.id)
    } else {
        render(view: "create", model: [customerInstance: customerInstance])
    }
}
```

3 R's of Controller Methods

- Return (not required)
  - returns to view with the same name as closure
- Render
  - Renders specified content
  - Simple text, views or templates
- Redirect
  - Redirects one action to another
  - via HTTP redirect

CHARIOT SOLUTIONS

Returning from a Controller

- “return” keyword is optional

- Returns a map of data to the page

```
return [ customer : cust ]
```

- No return map?
  - Grails returns controller properties

```
class MyController {
   List myVar1
   List myVar2

   def list = {
      myVar1  = MyObj1.list()
      myVar2 = MyObj2.list()
   }
}
```

CHARIOT SOLUTIONS

View

- A Groovy Server Page rendered in response to a Controller action
- Use Groovy tag libraries to handle form input, iteration, conditions, etc...
- Expected View Locations
  - /grails-app/views/*domainname*/*viewname*

Generating Views

- Provides a starting point for you

```
grails generate-views com.chariot.Customer
```

- For both controller & all views

```
grails generate-all com.chariot.Customer
```

CHARIOT SOLUTIONS

Grails Workflow - Domain Driven

- Build domain class
  - apply constraints & relationships
- Build a controller
- Build views
- Code tests
- Customize controllers & views
- Focus on the data first! (not the UI)

CHARIOT SOLUTIONS

Groovy Server Pages

- Grails GSP Features
  - Expression Syntax (EL)
  - Grails Tag Libraries for forms, links, iteration, etc.
  - Support for Templates

Grails GSP and the GSP EL

- You may refer to variables using EL Syntax
- Example using g:each GSP tag:

```
<html>
  <body>
    <ul>
      <g:each in="${myValues}" var="value">
        <li>${value}</li>
      </g:each>
    </ul>
  </body>
</html>
```

CHARIOT SOLUTIONS

GSP Tags

Tag	Use	Examples
<g:if> <g:elseif> <g:else>	Conditional Logic	**<g:if test="{editAllowed == true}">** <!-- render form --> **</g:if>** **<g:elseif test="{viewAllowed == true}">** <!-- render view mode --> </**g:elseif**> **<g:else>** <!-- render access denied --> **</g:else>**

<g:each>	Iterating a collection.
	
	Default var name is 'it'
	
	status : the iteration #
		<g:each in="${orders}"
		        var="order" status="i">
		  Order #**${i}**: Quantity: **${order.quantity}**
		  Cost: **${order.cost}**<br/>
		</g:each>

CHARIOT SOLUTIONS

Tags: Formatting Text/Numbers

GSP Tag	Use	Examples
<g:formatNumber>	Replaces DecimalFormat	<g:formatNumber number="${salary}" format="\$###,##0.00"/>
<g:formatDate>	Replaces SimpleDate Format	<g:formatDate date="${new Date()}"> format="MM/dd/yyyy hh:mm:ss a"/>

CHARIOT SOLUTIONS

GSP Form Tags

- <g:form> tag defines a form
  - action – the controller action (optional)
  - controller – the controller base name (optional)
  - name – the id and name to set in the form tag
  - id – the ID to use in the link (if saving an existing domain element, for example)

The Form Itself

- Expurgated from the generate-all code
- Notice the multiple buttons via **g:actionSubmit** and the embedded javascript validate
- One calls update(), the other delete()

```
<g:form method="post">

   ... form fields here ...

   <g:actionSubmit value="Update" /><br/
>

   <g:actionSubmit onclick=
          "return confirm('Are you sure?');"
      value="Delete" />

</g:form>
```

CHARIOT SOLUTIONS

Customizing Templates

- You can install the templates used for scaffolding
  - Customize them
  - Your scaffold classes use the modified templates

- Install templates with:

```
grails install-templates
```

- Installs templates in

```
src/templates
```

CHARIOT SOLUTIONS

Scaffolded GSPs

- `grails-app/views/domain/`
  - `create.gsp`
  - `edit.gsp`
  - `list.gsp`
  - `show.gsp`

- References the base page via meta tag:

```
<head>
  <meta name="layout" content="main" />
…
```

Name of the layout gsp

Customizing Templates

- Modify the main gsp that is used:

```
grails-app/views/layouts/main.gsp
```

- Weaves in your page content at runtime

```
<g:layoutHead/>
<g:layoutBody/>
```

CHARIOT SOLUTIONS

Navigation Plugin

- `grails install-plugin navigation`
- In `main.gsp`

```
<head>
…
<nav:resources/>
```

```
<body>
…
<div id="menu">
    <nav:render/>
</div>
<g:layoutBody/>
```

- In Controller configure menu options:

```
static navigation = [
    [action:'list', order: 0, isVisible: true],
    [action:'create', order: 1, isVisible: true]
  ]
```

Summary

- Grails is a web MVC platform (Domains, Controllers, Views)
- Grails is backed by Hibernate, Spring MVC, and other technologies
- Controllers handle the flow of the application
- GSPs provide the view technology with customizable templates