# Testando o Controle





## ControllerUnitTestCase

- ControllerUnitTestCase
  - Herda de GrailsUnitTestCase
  - Capaz de simular o comportamento de classes de domínio e de controles
  - Pode-se testar entre outras coisas o retorno dos métodos
    - O modelo





## AlbumController

```
...
def list = {
  params.max =
  Math.min(params.max ? params.int('max') : 10, 100)

  [albumInstanceList: Album.list(params),
   albumInstanceTotal: Album.count()]
}
...
```





## AlbumControllerTests
## testando a ação list

```
void testList() {
  mockDomain(Album,
             [ new Album(titulo: "RATM"),
               new Album(titulo: "Abbey Road")])
  def model = controller.list()
  assertEquals 2, model.albumInstanceList.size()
}
```






Faça você mesmo!

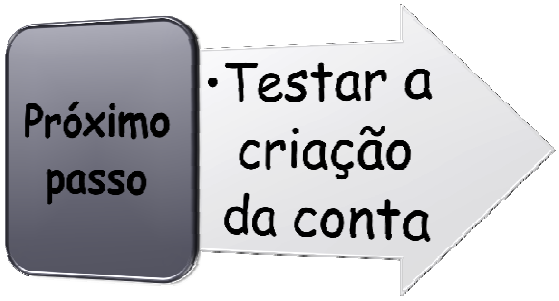
- Teste a outra propriedade do modelo

## AlbumControllerTests
## testando a ação list

```
void testList() {
  mockDomain(Album,
             [ new Album(titulo: "RATM"),
               new Album(titulo: "Abbey Road")])
  def model = controller.list()
  assertEquals 2, model.albumInstanceList.size()
  assertEquals 2, model.albumInstanceTotal
}
```












## Testando o criarConta

```
void testCriarContaFalha() {
    mockRequest.method = 'POST'
    mockDomain(Usuario)
    mockParams.login = ""
    def model = controller.criarConta()
    assertNull mockSession.usuario
    assert model

    def usuario = model.usuario
    assert usuario.hasErrors()
    assertEquals "blank", usuario.errors.login
    assertEquals "nullable", usuario.errors.senha
    assertEquals "nullable", usuario.errors.primeiroNome
    assertEquals "nullable", usuario.errors.sobreNome
}
```



## Entendendo as Visões



## GSPs

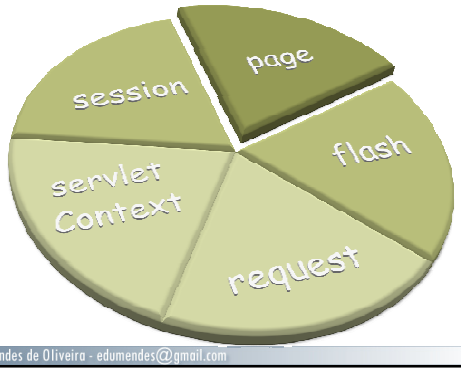
São bem próximas das sintaxes de JSPs

O Groovy possui uma linguagem de expressão mais poderosa





## Escopos das visões



## Objetos implícitos nas GSPs

| | |
|---|---|
| application | Instância de ServletContext |
| flash | Instância de flash |
| out | Instância do writer da resposta |
| params | Mapa com parâmetros de requisição |
| request | Instância de HttpServletRequest |
| response | Instância deHttpServletResponse |
| session | Instância de HttpSession |





## Lembrando o modelo

```
class LojaController {
    def loja = {
        def generoList = Album.withCriteria {
            projections {
                distinct "genero"
            }
        }
        [generos:generoList.sort()]
    }
}
```





## Diretivas





## Diretivas de páginas <%@ page

- A diretiva page aparece no topo da página, entre as instruções possíveis
  - Tipo de conteúdo
    - <%@ page contentType="text/xml; charset=UTF-8" %>
  - Import
    - <%@ page import="java.sql.Time" %>
  - java.lang, java.io, java.util, java.net, groovy.lang, groovy.util
    - IMPORTADOS POR DEFAULT





## Scriplets Groovy

- Como no java: <% ... %>

```
<html>
  <body>
    <% 3.times { %>
      <p>Serei impresso 3 vezes!</p>
    <% } %>
  </body>
</html>
```





Prefira

## Taglibs e Expression Language





## Taglibs GSP

- Operações Básicas
  - Loop
  - Switch
  - Condições lógicas
  - Atribuição de valores
- O prefixo padrão é **g:**
  - Não há necessidade de da diretiva taglib





## Expression Language

- Como nas JSPs
  - Entre Chaves
  - Pré-fixadas com um cifrão






Atribuindo Valores com Tags e EL

## <g:set>

```
<g:set
        var="tituloAlbum"
        value="${album.titulo}" />
```

- Por default: escopo de página
- Como mudar o escopo?

```
<g:set scope="session"
   var="usuario" value="${usuario}" />
```





Condicionais

## <g:if>, <g:elseif> e <g:else>

```
<g:if test="${album?.ano < 1980
              && album?.genero == 'Rock'}">
  Rock Clássico
</g:if>
<g:elseif test="${album?.year >= 1980
              && album?.genre == 'Rock'}">
  Rock Moderno
</g:elseif>
<g:else>
   Outro
</g:else>
```





## ?. Operador de referência seguro

- Só exibe as informações de um objeto, se todos os elementos navegados **não forem avaliados** como nulos (null)
- `${album.titulo.toUpperCase()}`
- `${album?.titulo?.toUpperCase()}`





## Laços <g:each>

```
<g:each in="${album.musicas?}">
    ${it.titulo}
</g:each>

<g:each var="musica"
    in="${album.musicas?}">
    ${musica.titulo}
</g:each>
```





## <g:while>

```
<g:set var="i"
  expr="${album.musicas?.size()}" />

<g:while test="${i > 0}">
      <g:set var="i" expr="${i-1}" />
</g:while>
```





## Filtrando uma iteração

```
<ol>
 <g:collect in="${albuns}"
   expr="${it.titulo}">

      <li>${it}</li>

    </g:collect>
</ol>
```





## findAll

```
<g:findAll
  in="${albuns}"
  expr="${it.musicas?.titulo.contains('Amor')}">
    <li>
       ${it.titulo}
    </li>
</g:findAll>
```





## Links <g:link>

- controller*
  - o controle do link
- action*
  - a ação do link
- id
  - o identificador para adicionar ao final
- params
  - quaisquer parâmetros para ser passado com um mapa

## <g:link>

```
<g:link
  controller="album" action="list">
  Lista de Albums
</g:link>

<g:link
  action="show" id="1">
  Exibir album com id 1
</g:link>
```





## <g:link> Passando parâmetros

```
<g:link
 controller="album"
 action="list"
 params="[max:10,order:'titulo']">
  Exibir os 10 primeiros ordenados pelo
  titulo
</g:link>
```





## <g:createLink>

```
<a href="<g:createLink action="list" />">
  Um link dinâmico
</a>

<a href="${createLink(action:'list')}">
  Mesmo link dinâmico
</a>
```





## Formulários





## <g:form>

```
<g:form action="criarConta"
  name="criarContaForm">
        ...
</g:form>

<g:form url="[controller:'usuario',
  action:'criarContaForm']">
      ...
</g:form>
```





## Inputs

- Caixa de texto

```
<g:textField name="login"
  value="${usuario?.login}"></g:textField>
```

- Checkbox

```
<g:checkBox name="valorBooleano"
  value="${true}" />
```

- Radio

```
<g:radio name="meuGrupo" value="1"
  checked="${algumValor== 1}" /> Radio
```

## Select com valores na página

```
<g:select
   name="genero"
   from="${['Rock', 'Blues', 'Jazz']}"
   value="${album.genero}" />
```

```
<select name="genero">
   <option value="Rock" selected="selected">Rock</option>
   <option value="Blues">Blues</option>
   <option value="Jazz">Jazz</option>
</select>
```





## Select com valores da classe de domínio

```
<g:select name="album.id"

  from="${Album.list()}"

  optionKey="id"

  optionValue="titulo"/>
```





## Datas

```
<g:datePicker
   name="myDate"
   value="${new Date()}" />

<g:datePicker
   name="myDate"
   value="${new Date()}"
   precision="day" />
```

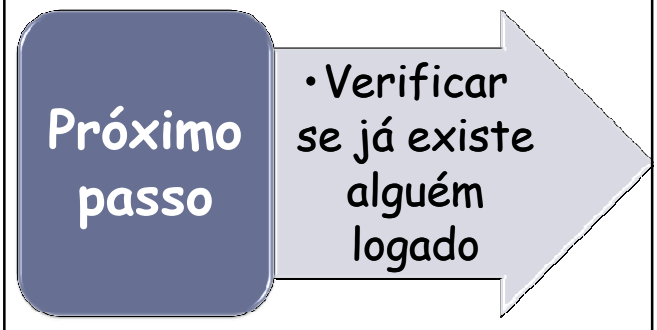




## Adicionando o login em todo canto












## Alterando o layout principal

- Sitemesh
  - Framework de renderização de layout robusto
- Layout principal do Grails
  - grails-app\views\layouts\main.gsp
- Vamos adicionar uma região nova ao layout
  - <g:render template="qualquerCoisa.gsp" />

## Passo-a-passo

- No diretório grails-app\views
  - Crie um novo diretório
    - includes
- Dentro de includes crie um arquivo chamado
  - _confereLogin.gsp
- Adicione em grails-app\views\main.gsp, após o <body>

```
<g:render template="/includes/confereLogin" />
```





## _confereLogin.gsp

```
<div id="loginBox" class="loginBox">
  <g:if test="${session?.usuario}">
            <a href="#">Perfil</a> |
             <g:link controller="usuario"
action="logout">Sair</g:link><br>
        Benvindo,
            ${session?.usuario?.primeiroNome}!
            <br/><br/>
            Você já comprou
(${session.usuario.musicasCompradas?.size() ?: 0})
musicas.<br>

  </g:if>
```





## _confereLogin.gsp

```
<g:else>
        <g:form name="loginForm"
                url="[controller:'usuario',action:'login']">
                <div>Login:</div>
                <g:textField name="login" ></g:textField>
                <div>Senha:</div>
                <g:passwordField name="senha" />
                <input type="submit" value="Login" />
        </g:form>
        <g:renderErrors bean="${loginCmd}"></g:renderErrors>
   </g:else>
</div>
```