Groovy & Grails

Desenvolvimento ágil para plataforma Java

Luís Bruno P. Nascimento
luisbrunu@gmail.com

Enucomp 2010

- Groovy e Grails – Parte II

Introdução ao Grails

Introdução ao Grails

- Java EE
  - Java Enterprise Edition – Java Edição Empresarial
    - Plataforma de programação para servidores na linguagem Java.
  - Alvo de várias críticas relacionadas à complexidade no desenvolvimento de aplicações web;
    - Muita configuração XML;
    - Muito trabalho repetitivo;
    - Fazer muito para criar pouco;
  - Daí surge o Grails

O que é Grails?

- Uma Web framework baseado em RoR para desenvolvimento de aplicações Java utilizando a linguagem Groovy.
  - Framework de ALTA produtividade;
  - Full Stack web framework;
  - Programação por Convenção;
  - Don’t Repeat Yourself (DRY);
- Lançamento versão 0.1, Março/2006;
- Ultima versão: 1.3.5

Grails

- Framework de alta produtividade:
  - Em questão de minutos você cria toda a estrutura de banco de dados, as telas HTML e CSS, tudo já funcionando.
  - CRUD fácil
    - Com apenas alguns comandos, a framework gera toda a estrutura de um CRUD;

Full Stack web Framework

- Após instalado, o Grails já vem com todos os componentes necessários para que possamos iniciar o trabalho;
  - É completo, pois já vem com diversos componentes pré-configurados para o uso:
    - Spring;
    - Hibernate;
    - Apache;
    - HSQLDB;
    - E muitos outros;

Convenção sobre configuração

- Programar seguindo convenções da framework;
  - Essas convenções definem o padão de comportamento da framework;
  - Principal motivação: programar fazendo configurações de arquivos XML;
  - Se não quisermos configurar nada, basta seguir as convenções da framework;
  - É dessa forma que o Grails pode ajudar na programação, dando mais praticidade no desenvolvimento e também mais agilidade.

DRY

- **DRY - *Don't Repeat Yourself*** – Não se repita
- No desenvolvimento de aplicações web, boa parte do trabalho feito pelo desenvolvedor é repetitiva;
- Agora o framework passa a farzer essas tarefas repetitivas;
- Código mais enxuto;

Instalando

- Faça o download da ultima versão do Grails: http://www.grails.org/Download
- Para instalar, basta descompactar o arquivo grails-1.3.5.zip em qualquer diretório;
- Definir as variáveis de ambiente e o PATH;
- Para testar, basta digitar grails no terminal:

```
>>Welcome to Grails 1.3.5 - http://grails.org/
>>Licensed under Apache Standard License 2.0
```

Ambientes para desenvolvimento

- Grails plugin para Eclipse
  - www.grails.org/Eclipse+IDE+Integration
- Netbeans já vem com suporte total à Groovy e Grails;
- Os editores de texto em geral, já resolvem nosso problema;
  - Gedit;
  - JEdit
  - Notpad++

Grails Comandos

>> **grails** *help*

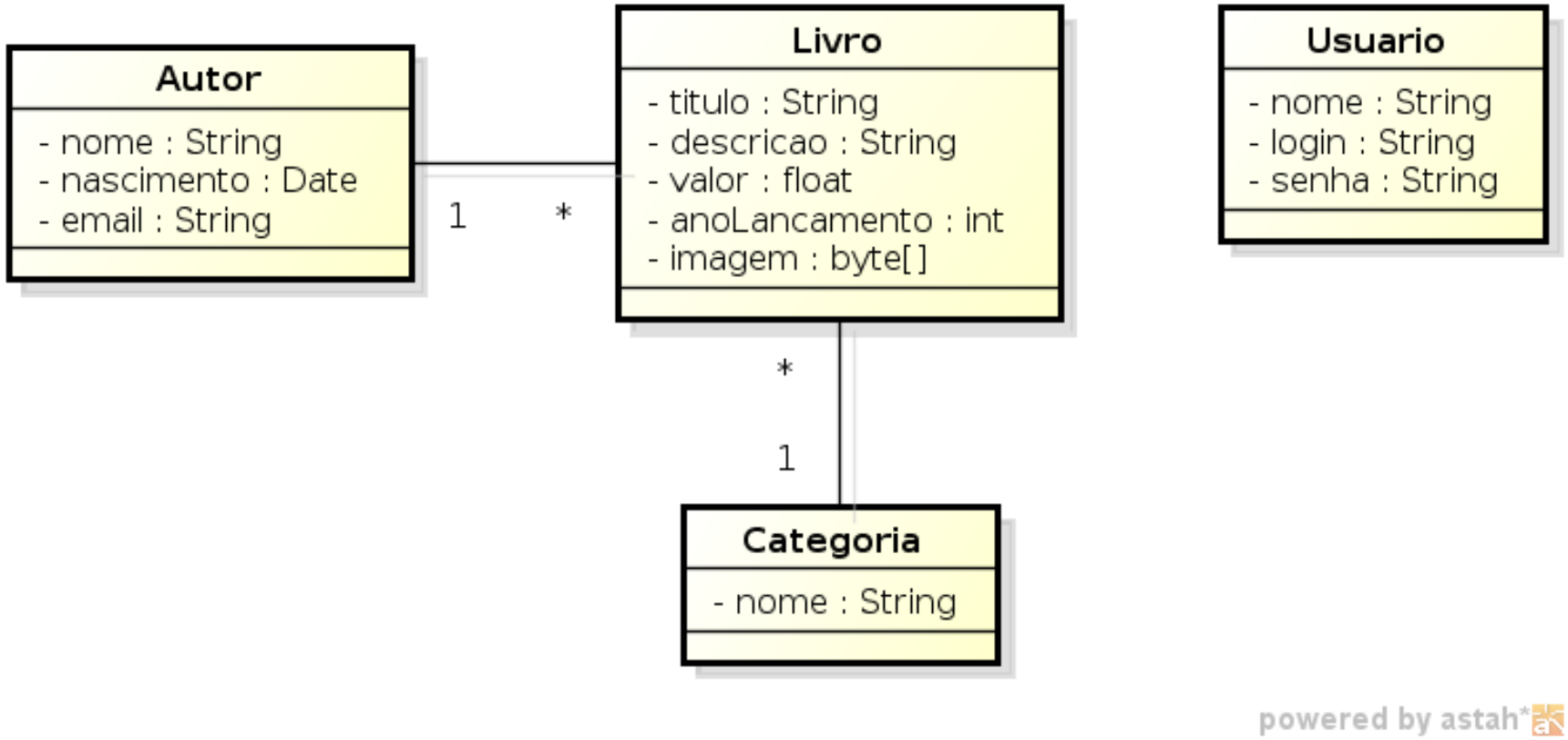
- Lista todos os comandos que podemos executar na linha de comando em Grails:
  - grails create-app
  - grails generate-controller
  - grails generate-views
  - grails install-plugins
  - ...

Atualizando Grails

- Para atualizar devemos fazer o download da versão mais nova, descompactar e trocar as variáveis definidas.
- Dentro do projeto:
  - **>>grails upgrade**
    - Esse comando atualizará o seu projeto para a utilização dos recursos da nova versão do grails;

Projeto

- Livraria Virtual



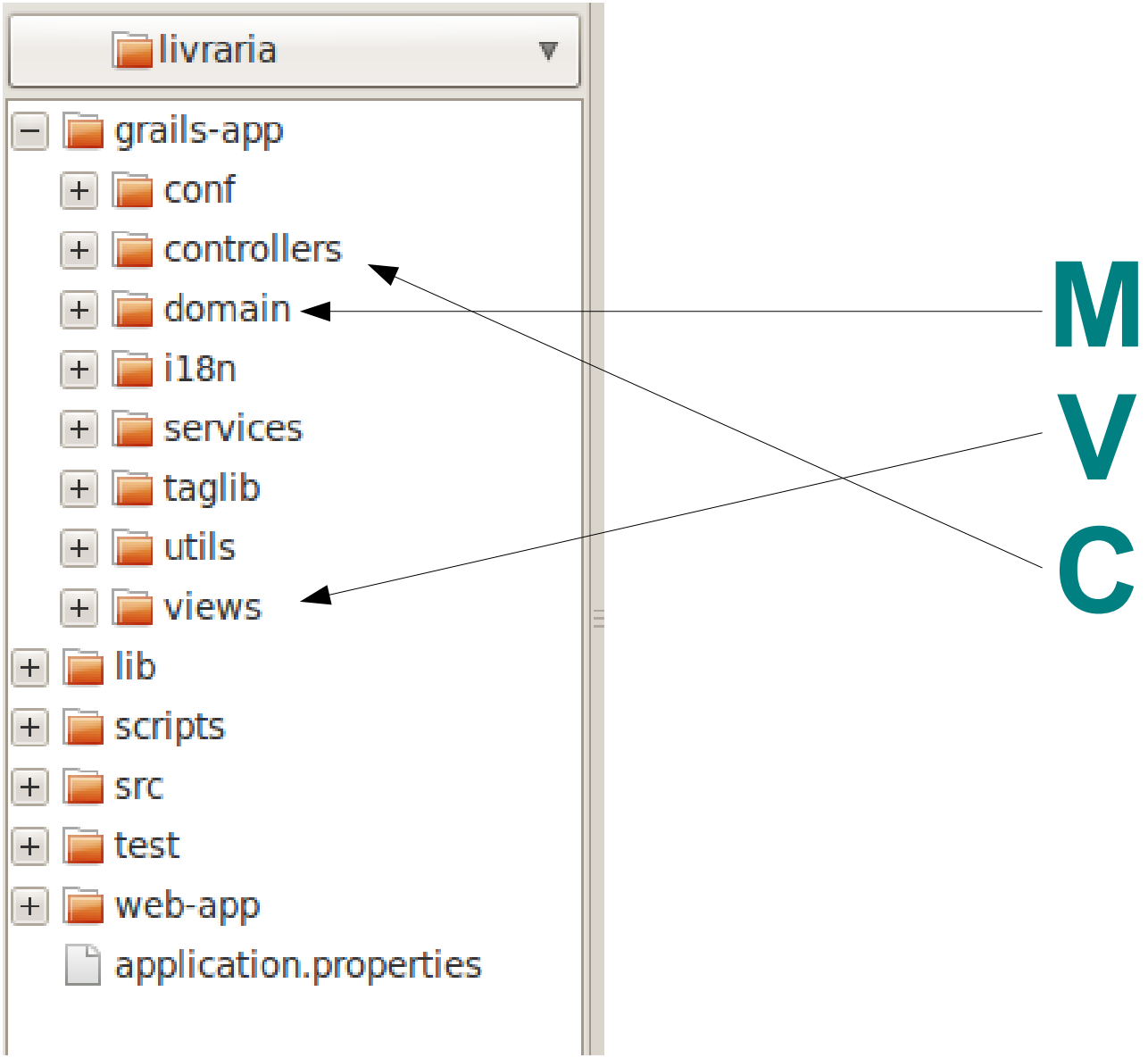
Criando a aplicação

- **>>grails create-app livraria**

```
Welcome to Grails 1.3.4 - http://grails.org/
Licensed under Apache Standard License 2.0
Grails home is set to: /opt/Groovy/grails-1.3.4

Base Directory: /opt/Groovy/web
Resolving dependencies...
Dependencies resolved in 1653ms.
Running script /opt/Groovy/grails-1.3.4/scripts/CreateApp_.groovy
Environment set to development
    [mkdir] Created dir: /opt/Groovy/web/livraria/src
    [mkdir] Created dir: /opt/Groovy/web/livraria/src/java
    [mkdir] Created dir: /opt/Groovy/web/livraria/src/groovy
    [mkdir] Created dir: /opt/Groovy/web/livraria/grails-app
    [mkdir] Created dir: /opt/Groovy/web/livraria/grails-app/controllers
    [mkdir] Created dir: /opt/Groovy/web/livraria/grails-app/services
    [mkdir] Created dir: /opt/Groovy/web/livraria/grails-app/domain
    .......
```

Diretórios da aplicação



Diretórios da aplicação

- Grails-app é o diretório principal. É onde se encontra as classes modelos, os controllers, as views, traduções...
  - **conf** – Possui arquivos de configuração da aplicação
    - *config.groovy* – várias configurações da aplicação como ambientes de execução;
    - *DataSource.groovy* – configurações da base de dados;
    - *Bootstrap.groovy* – habilita códigos a serem executados quando a aplicação é estartada;
    - *URLMappings.groovy* – alterar como as requisições são tratadas através de URLs.

Diretórios da Aplicação

- **I18n**
  - Recursos para internacionalização e mensagens de erro.
    - Traduções e/ou customização de mensagens
- **taglib**
  - Tags criadas pelo usuário.
    - Tags usadas em páginas GSP

Diretórios da aplicação

- **controllers**
  - Diretórios onde ficam os controllers da aplicação;
  - Tratam de requisições web;
- **doimain**
  - Os modelos da plicação;
  - Classes onde serão mapeados os objetos e propriedades do mundo OO para o mundo relacional (tabelas, colunas);
    - GORM – Grails Object Relational Mapping;

Diretórios da aplicação

- **views**
  - Arquivos **GSP** – Groovy Server Pages;
    - Páginas HTML que suportam código em Groovy;
    - Similar às **JSP**;
  - As visões são responsáveis por mostrar o retorno da das requisições para o browser;

Executando...

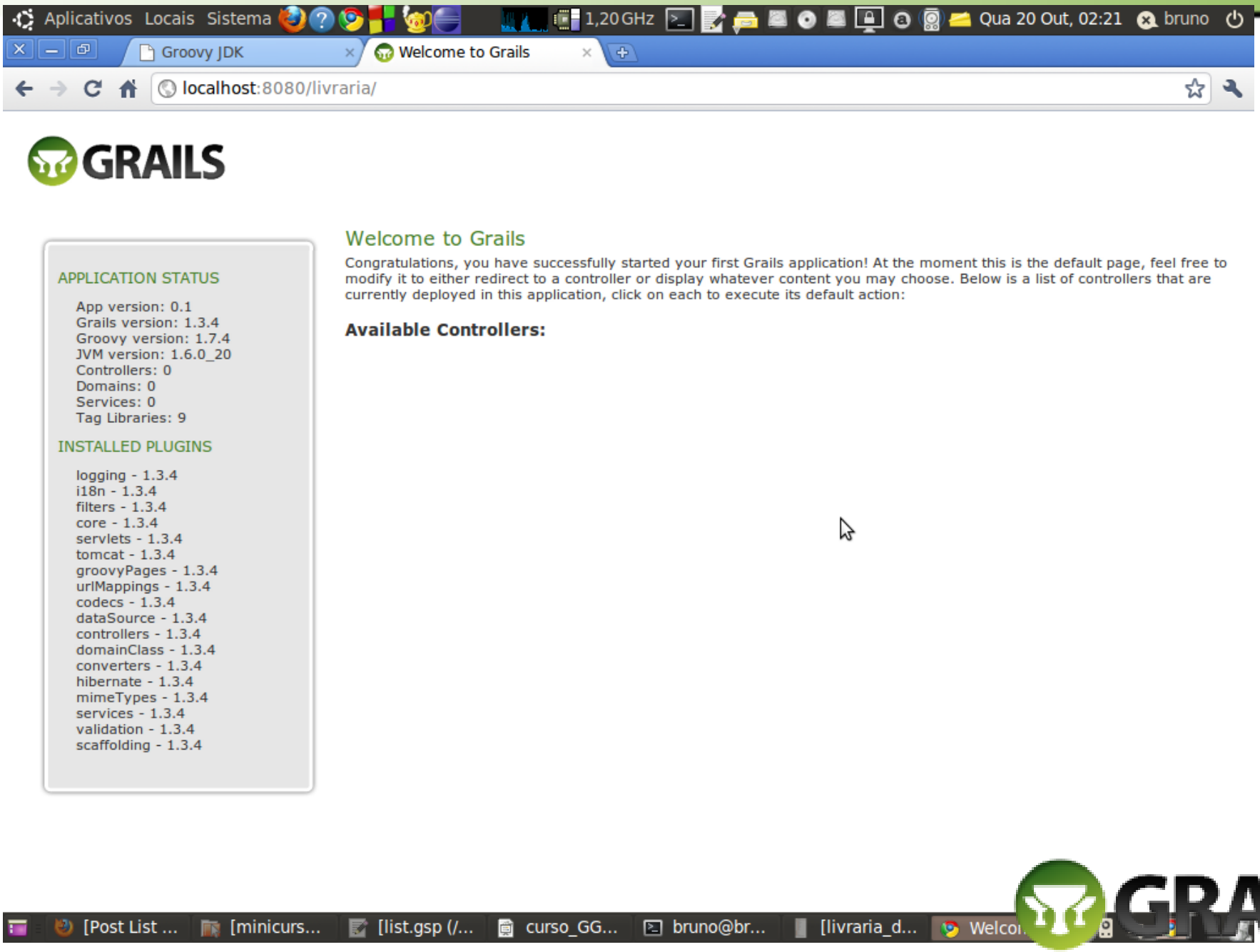
>>cd livraria

>>grails run-app

```
Welcome to Grails 1.3.4 - http://grails.org/
Licensed under Apache Standard License 2.0
Grails home is set to: /opt/Groovy/grails-1.3.4

Base Directory: /opt/Groovy/web/livraria
Resolving dependencies...
Dependencies resolved in 1116ms.
Running script /opt/Groovy/grails-1.3.4/scripts/RunApp.groovy
Environment set to development
    [mkdir] Created dir: /opt/Groovy/web/livraria/target/classes
  [groovyc] Compiling 7 source files to /opt/Groovy/web/livraria/target/classes
     [copy] Copying 1 file to /opt/Groovy/web/livraria/target/classes
   [delete] Deleting directory /home/bruno/.grails/1.3.4/projects/livraria/tomcat
Running Grails application..
Server running. Browse to http://localhost:8080/livraria
```

Executando...


Aplicativos Locais Sistema
1,20 GHz
Qua 20 Out, 02:21
bruno
Groovy JDK
Welcome to Grails
localhost:8080/livraria/
GRAILS
APPLICATION STATUS
App version: 0.1
Grails version: 1.3.4
Groovy version: 1.7.4
JVM version: 1.6.0_20
Controllers: 0
Domains: 0
Services: 0
Tag Libraries: 9
INSTALLED PLUGINS
logging - 1.3.4
i18n - 1.3.4
filters - 1.3.4
core - 1.3.4
servlets - 1.3.4
tomcat - 1.3.4
groovyPages - 1.3.4
urlMappings - 1.3.4
codecs - 1.3.4
dataSource - 1.3.4
controllers - 1.3.4
domainClass - 1.3.4
converters - 1.3.4
hibernate - 1.3.4
mimeTypes - 1.3.4
services - 1.3.4
validation - 1.3.4
scaffolding - 1.3.4
Welcome to Grails
Congratulations, you have successfully started your first Grails application! At the moment this is the default page, feel free to modify it to either redirect to a controller or display whatever content you may choose. Below is a list of controllers that are currently deployed in this application, click on each to execute its default action:
Available Controllers:
[Post List ...
[minicurs...
[list.gsp (/...
curso_GG...
bruno@br...
[livraria_d...
GRAILS

Executando...

- Para remover o nome do projeto do fim da URL, como normalmente deve ser no modo de produção, você seta a propriedade **app.nome** em *application.properties*, o diretório raiz da nossa aplicação;
  - app.nome=/
- Para rodar a aplicação em outra porta:
  - » grails run-app -Dserver.port=9090
- Para parar a aplicação:
  - Ctrl + C

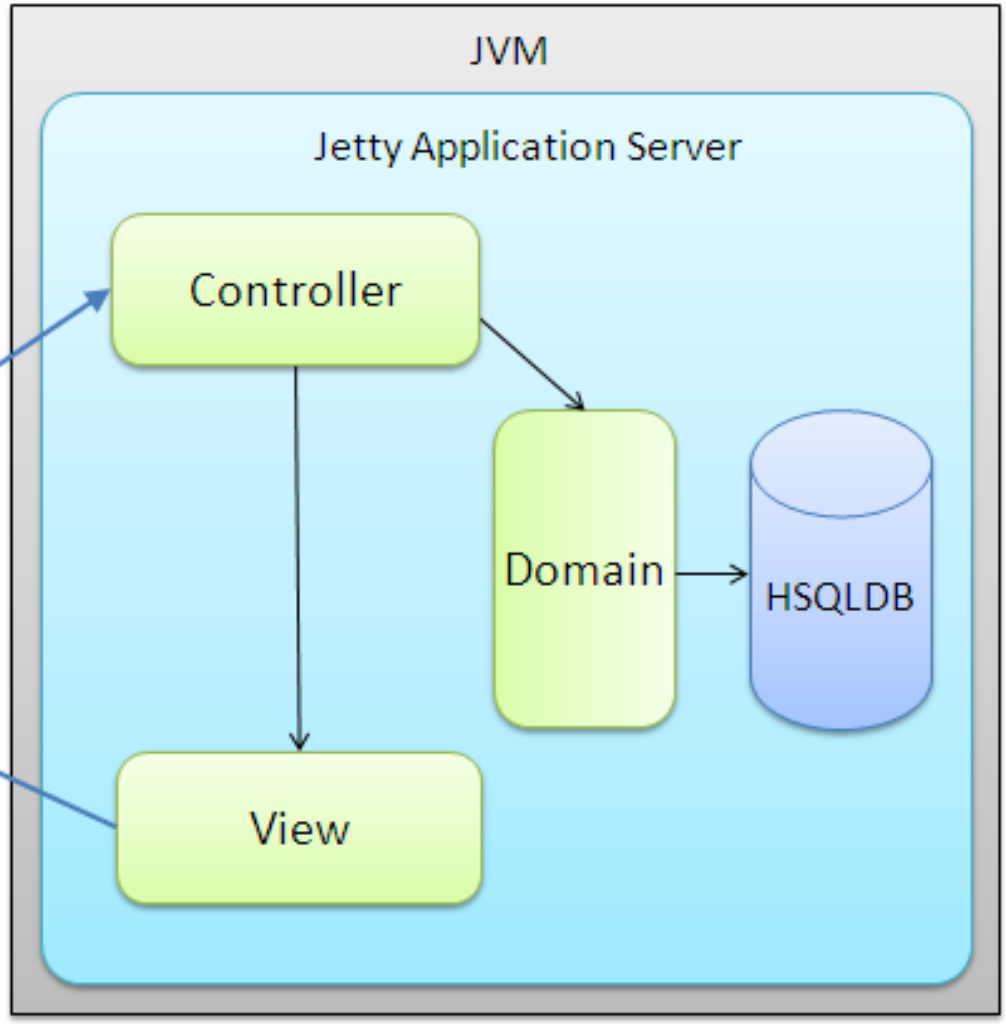
Ambientes de execução

- **development**(dev)
  - Em modo de desenvolvimento;
  - Padrão;
- **production**(prod)
  - Em modo de produção;
  - As alterações são salvas no banco de dados;
  - » **grails prod run-app**
- **test**(test)
  - Ambiente para testes

MVC em Grails

- MVC é um padrão de projeto, para outros, uma arquitetura de software no qual a aplicação web é dividida em camadas para separar a lógica de negócio da apresentação dos dados;
  - **M**odel – Modelo → domains;
  - **V**iew – Visão → views;
  - **C**ontroller – Controlador → controllers;

MVC em Grails



GRAILS

GORM

- GORM é um componente do Grails para implementar o ORM – Mapeamento Objeto Relacional;
- O GORM que cuida da parte do mapeamento das classes -domains- para se tornarem tabelas;
  - As classes de domínio são tabelas;
  - Os atributos os campos;

Classes de domínio

- Classes de domínio são a força vital de uma aplicação grails. Elas definem as “coisas” que você está controlando.
  - As tabelas do banco de dados são criadas para as classes de domínio correspondente.
- » **grails create-domain-class** <Nome do Modelo>
  - Gerar uma classe de domínio em /grails-app/domain/;
- No contexto da nossa aplicação, os domínios existentes são:
  - Livro;
  - Categoria;
  - Autor;
  - Usuário;

Classes de domínio

- » **grails create-domain-class** liv.enucomp.Livro
  - O Grails cria a classe de domínio Post.groovy. Abra-o em seu editor de texto.

```
class Livro {
    static constraints = {    }
}
```

  - Adicione os atributos de acordo com o diagrama de classes.

```
class Livro {
  String titulo
  String descricao
  Float valor
  int anoLancamento
  byte[] imagem
  ...
```

Classes de domínio

» **grails create-domain-class** liv.enucomp.Categoria

» **grails create-domain-class** liv.enucomp.Autor

» **grails create-domain-class** liv.enucomp.seg.Usuario

- Inserir os atributos de acordo com o diagrama de classes;

Controllers

- Comando para criar um controller:

» ***grails create-controller*** *<Classe de domínio>*

- » **grails create-controller** liv.enucomp.Livro
  - Gera uma classe chamada LivroController.groovy;
  - O controller, a partir do método index vazio, chama um arquivo chamado index.gsp;
  - Podemos simular uma página dentro .gsp através do método render, ou chamar um outro arquivo;

```
class LivroController {
    def index = { render 'Hello World' }
      ou
    def chama = {render(view:'teste.gsp')  }
}
```

Scaffolding

- Estrutura de CRUD em MCV já conectada com banco de dados gerada pela mágica da meta-programação.
- Em grails, para tornar isso realidade, basta adicionar uma linha de código em seu controller:

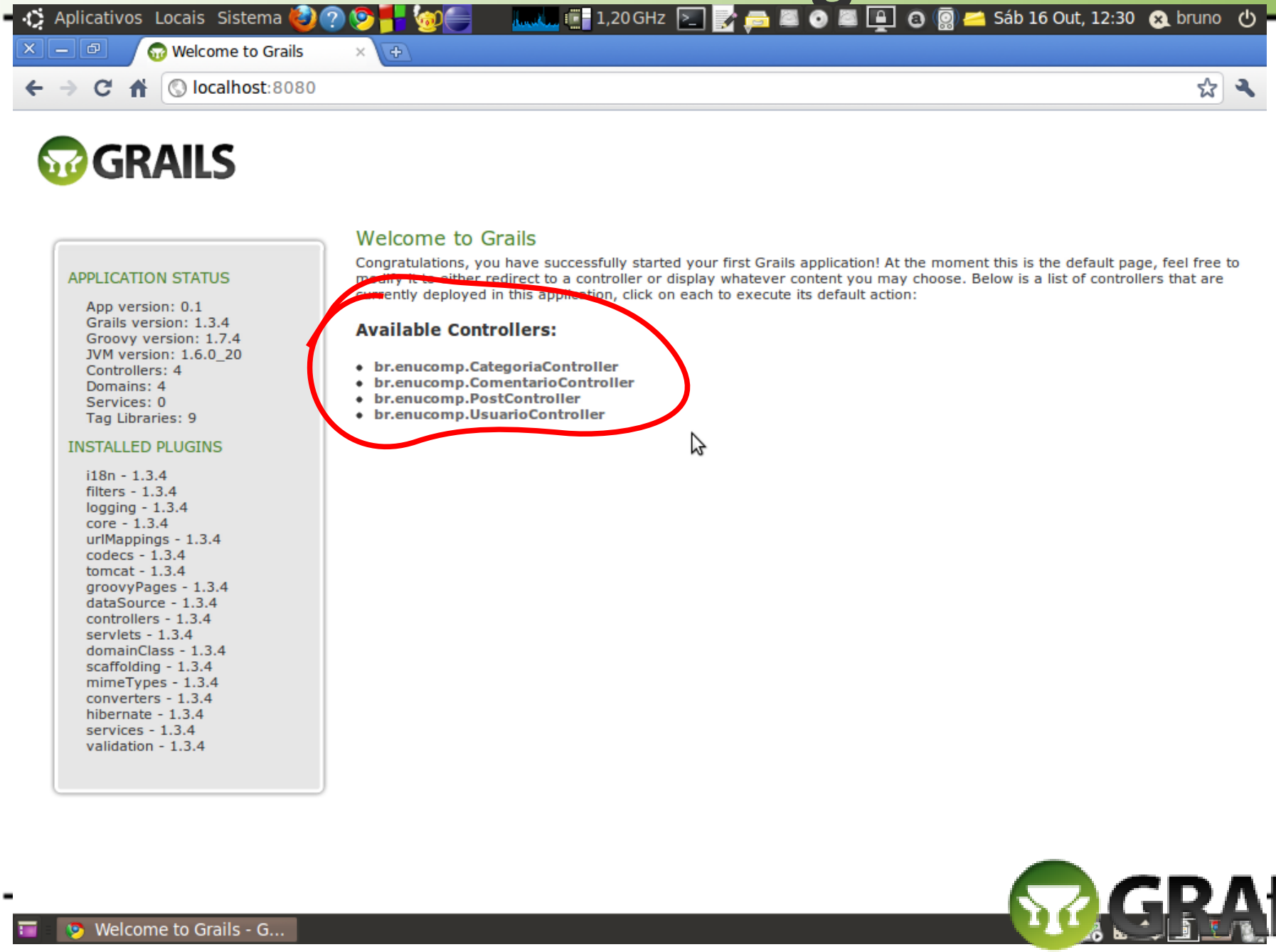
```
class LivroController {
   def scaffold = Livro
}
```

  - Agora rode novamente a aplicação;

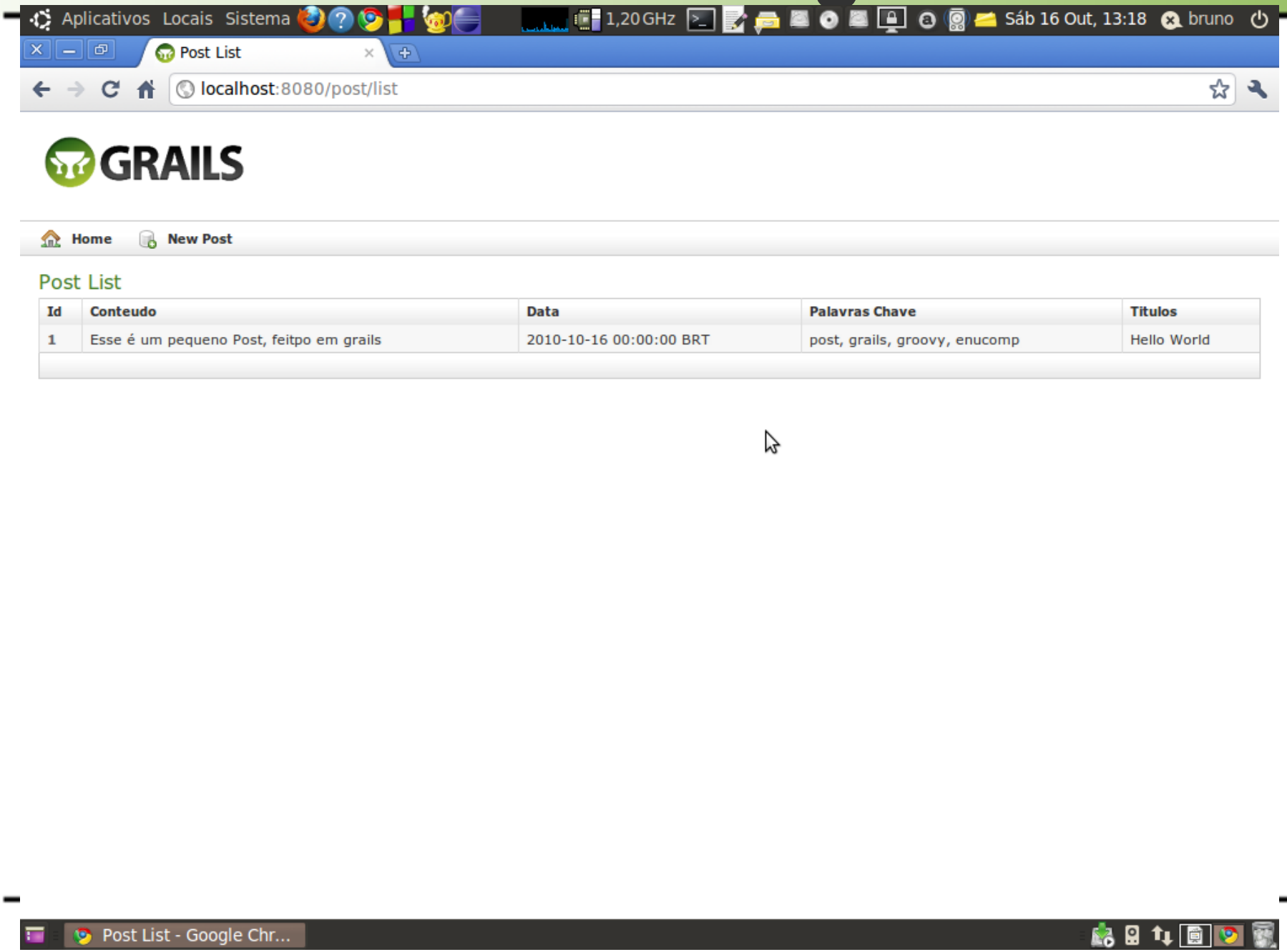
Scaffolding

- Faça o mesmo agora para as outras classes de domínio:
  - » **grails create-controller** liv.enucomp.Livro
  - » **grails create-controller** liv.enucomp.Categoria
  - » **grails create-controller** liv.enucomp.Autor
  - » **grails create-controller** liv.enucomp.seg.Usuario
- Adicione o código do scaffolding para todos os domínios, exceto para Usuario, pois vamos fazer de forma diferente;
- Restarte o servidor;

Scaffolding



Scaffolding



Mudar a ordem dos campos

- Adicionar os campos na ordem específica dentro do bloco **static** constraints;

```
class Livro {
  static constraints = {
    imagem()
    titulo()
    descricao()
    anoLancamento()
    valor()
  }
  String titulo
  String descricao
  Float valor
  int anoLancamento
  byte[] imagem
```

Tratando as Imagens

- Controller em questão:

```
def imagem = {
      def livro = Livro.get(params.id)
      byte[] imagem = livro.imagem
      response.outputStream << imagem
    }
```

- Visões:

```
<img src="${createLink(controller:'livro', action:'imagem', id: livroInstance.id)}" width="80px"
height="100px"/>
```

Validações

```
class Livro {
     static constraints = {
          imagem()
          titulo(blank:false)
          descricao(blank:false, maxSize:1000)
          anoLancamento(blank: false)
          valor()
     }
     …
}
```

- Opções de validação em Grails:

Validações

Constraint	Usage	Description
blank, nullable	blank:true nullable:true	blank traps for blanks at the web tier; nullable traps for blanks at the database tier
creditCard	creditCard:true	Based on the Apache Commons CreditCardValidator, this guarantees that credit card numbers are internally consistent
display	display:false	Hides the field in create.gsp and edit.gsp. Note: be sure to couple this with blank:true and nullable:true or populate these values in the controller; otherwise the default validation will fail
email	email:true	Ensures that values match the basic pattern "user@somewhere.com"
password	password:true	Changes the view from <input type="text"> to <input type="password">
inList	inList:["A", "B", "C"]	Creates a combo box of possible values
matches	matches:"[a-zA-Z]+"	Allows you to provide your own regular expression

Validações

min, max	min:0, max:100 min:0.0, max:100.0	Used for numbers and classes that implement java.lang.Comparable
minSize, maxSize, size	minSize:0, maxSize:100, size:0..100	Used to limit the length of Strings and arrays
notEqual	notEqual:"Foo"	As the name suggests, the value cannot equal the constraint
range	range:0..10	Creates a combo box with an element for each value in the range
scale	scale:2	Sets the number of decimal points
unique	unique:true	Ensures that the value doesn't already exist in the database table. (This is perfect for creating new logins.)
url	url:true	Ensures that the value matches the basic pattern "http://www.somewhere.com"
validator	validator: {return(it%2)==0}	Allows you to create your own custom validation closure

Validações

- Para alterar uma validação, edite o arquivo *grails-app/i18n/messages_pt_BR.properties*

```
default.blank.message = Por favor, digite um
valor para [{0}]. O campo não pode estar vazio.
```

- Mais sobre validações:
http://grails.org/doc/latest/guide/7.%20Validation.html

Traduzindo o Scaffold

- No mesmo arquivo adicione as seguintes linhas:

default.home.label=Principal

default.list.label=Listar {0}

default.add.label=Adicionar {0}

default.new.label=Cadastrar {0}

default.create.label=Cadastrar {0}

default.show.label=Exibir {0}

default.edit.labe=Editar {0}

default.button.create.label=Gravar

default.button.edit.label=Editar

default.button.update.label=Alterar

default.button.delete.label=Excluir

default.button.delete.confirm.message=Você tem certeza?

Views

- Para obter as visões usamos o comando:
  - » **grails generate-view liv.enucomp.Livro**
    - O comando irá gerar as visões para a classe Post
- Gerar para todos os domínios:
  - » **grails generate-all**
- As visões geradas ficam dentro do diretório grails-app/views/liv/enucomp/

Relacionamentos

- **One-to-One**
  - belongsTo
  - belongsTo
- **One-to-Many**
  - hasMany
  - belongsTo
- **Many-to-Many**
  - hasMany
  - hasMany

Relacionamentos

- Um Livro tem uma Categoria;
- Uma Categoria pode ser de vários Livros
- O Livro pertence a um Autor
- Um Autor pode ter vários Livros

Relacionamentos

- Livro.groovy
  - `static belongsTo = [categoria:Categoria, autor:Autor]`
- Categoria.groovy
  - `static has_many = [livros:Livro]`
- Autor.groovy
  - `static has_many = [livros:Livro]`

BootStrap.groovy

- Abra o arquivo grails-app/conf/BootStrap.groovy
-
```
class BootStrap {
  def init = { servletContext ->
```
-
```
  }
```
-
```
  def destroy = {}
```
-
```
}
```
- Dentro do bloco **init**, instancie as classes de domínio.
  - Tudo que esta dentro do init será carregado no momento que a aplicação for iniciada.

BootStrap.groovy

```
import liv.enucomp.*
class BootStrap {
  def init = { servletContext ->
    def fernando = new Autor(
                    nome:'Fernando Anselmo',
                    nascimento:new Date(),
                    email:'autor@empresa.com',
                   )
    fernando.save()
    if(fernando.hasErrors()){
      println fernando.errors
    }

   def destroy = {  }

}
```

DataSource.groovy

- **datasource** – Configurações gerais do banco de dados;
- **hibernate** – configurações específicas do Hibernate, como performance...
- **environments** – Configurações específicas para cada ambiente de desenvolvimento.

Alterar a Base de Dados

- Criar a base de dados;
- Copie o driver JDBC para o diretorio grails-app/lib;
- Ajustar as configurações em DataSource.groovy.

- Em nossa aplicação, vamos usar o banco de dados MySQL.
  - O ideal é criar base de dados distintas para cada um dos ambientes(`development`, `production`, `test`), mas em nosso exemplo será criada apenas uma geral.

Alterar a Base de Dados

```
$ mysql -u root -p
Enter password: ********
Welcome to the MySQL monitor. Commands end with ; or \g.
...
mysql> create database livraria_dev;
...
mysql> grant all on livraria_dev.* to grails@localhost
identified by 'server';
...
mysql> exit
```

- O penultimo comando está concedendo todos os privilégios(criar, apagar...) ao usuário grails que se loga através de localhost, cuja senha é server;

Alterar a Base de Dados

```
$ mysql --user=grails -p --database=livraria_dev
Welcome to the MySQL monitor.

Mysql> show tables;
Empty set (0.00 sec)
```

- Próximo passo é copiar o driver MySQL para o diretório `/livraria/lib`
  - O driver pode ser baixado no site: http://dev.mysql.com/downloads/connector/j/
  - Descompacte o arquivo .zip e copie o JAR mysql-connector-java-x.x.x-bin.jar
  - Agora é só ajustar as configurações no DataSource.groovy

Alterar a Base de Dados

```
dataSource {
    pooled = true
    driverClassName = "com.mysql.jdbc.Driver"
    username = "grails"
    password = "server"
}

environments {
    development {
        dataSource {
            dbCreate = "create-drop"
            //Vamos definir essa mesma url para todos os ambientes
            url = "jdbc:mysql://localhost:3306/livraria_dev?autoreconnect=true"
        }
    }
    ...
}
```

Página inicial

- Criando uma página para listar os livros cadastrados;
  - Crie um controller chamado Home;
    - Adicione o seguinte código:
      - def home = { [livroList:Livro.list()]}
  - Copie:
    - A imagem logo.jpg e cole em /web-app/imagens/;
    - O arquivo style.css para /web-app/css
    - O arquivo index.html para /grails_apps/views/layout com o nome de index.gsp

Página inicial

- Abra o arquivo index.gsp e modifique a linha do css para:
  - <link rel="stylesheet" href="${createLinkTo(dir:'css', file:'style.css')}" type="text/css" />
  - Na ultima linha do <head>, a dicione a tag <g:layoutHead/>
  - Apague os textos do html e adicione a tag <g:layoutBody/>

Página inicial

- Crie um controller chamado Home;
  - Modifique para isso:

```
class HomeController {

  def home = {
    [livroList:Livro.list()]

  }

}
```

Página inicial

- Crie um arquivo home.gsp dentro de /views/home/

```
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <meta name="layout" content="index" />
  <title> Meu blogão </title>
</head>
<body>
  <g:each in="${postList}" status="i" var="post">
    <div id="${ (i % 2) == 0 ? 'left' : 'right'}">
      <h2>${post.titulo}</h2>
      <p align="justify"> ${post.conteudo}</p>
      <div id="notice">
        <p>Coloque aqui uma chamada em destaque em seu blog</p>
      </div>
    </div>
  </g:each>
</body>
```

Templates

- Alterando as templates geradas por padrão no grails:
  - `grails install-templates`
  - Arquivos gerados no diretório:
    - `.../livraria/src/templates`

Autorização

- Um usuário eterno não poderá ter acesso às funcionalidades de administração do sistema, somente um usuário cadastrado poderá ter acesso a tais funções;
  - Criar;
  - Editar;
  - Excluir;
  - Listagem;

Autorização

- Modificar o UsuarioController:

```
def login = {
    if (params.cNome)
      return [cNome:params.cNome, aNome:params.aNome]
    redirect(url:resource(dir:'' ))
  }
  def logout = {
    session.usuario = null
    redirect(url:resource(dir:'' ))
  }
  def validate = {
    def usuario = Usuario.findByLogin(params.login)
    if (usuario && usuario.senha == params.senha) {
      session.usuario = usuario
      if (params.cNome) {
        redirect(controller:params.cNome, action:params.aNome)
      }
      redirect(url:resource(dir:'' ))
    } else {
      flash.message = "Usuário ou Senha inválidos."
      render(view:'home' )
    }
  }
```

Autorização

- Criar uma tagLib:
  - `grails create-tag-lib Protege`

```
class ProtegeTagLib {
  def loginLogout = {
    out << "<div>"
    if (session.usuario) {
      out << "<span style='float:left'>"
      out << " Bem Vindo <b>${session.usuario.nome}</b>."
      out << "</span><span style='float:right;margin-right:10px'>"
      out << "<a href='${createLink(controller:'usuario', action:'logout')}'>"
      out << "Logout </a></span>"
    } else {
      out << "<span style='float:right;margin-right:10px'>"
      out << "<a href='${createLink(controller:'usuario', action:'login')}'>"
      out << "Login </a></span>"
    }
    out << "</div><br/><br/>"
  }
}
```

Autorização

- Agora adicione a tag no arquivo app/views/layouts/main.gsp na segunte linha:

```
<html>
 ...
  </div>
    <g:loginLogout />
    <g:layoutBody />
 ...
</html>
```

Autorização

- Criar um Filtro:
  - Um filtro age em uma determinada ação ou em todas elas. E determina se é antes, depois de ter acontecido. No nosso caso, agiremos sobre todas e antes que elas acontecem.
  - `grails create-filters liv.enucomp.seg.Seguranca`
    - Altere o conteudo do arquivo *`grails-app/conf/liv/.../SegurancaFilters.groovy`*

Autorização

```
. . .
def filters = {
    all(controller:'*', action:'*') {
      before = {
            if (!controllerName)
              return true
            def allowedActions = [ 'index' , 'login', 'logout' , 'validate', 'home',
'entrar']
            if (!session.usuario && !allowedActions.contains(actionName)){
              redirect(controller:'usuario' , action:'entrar.gsp' ,
                params:['cNome': controllerName, 'aNome': actionName])
              return false
            }
        }
. . .
```

Autorização

- Agora iremos construirum formulário de login. Crie um arquivo chamado entrar.jsp e, *grails-app/views/usuario/* com os seguintes códigos:

```
<html>
  <head>
    <meta name="layout" content="main" />
    <title>Login</title>
  </head>
  <body>
    <g:if test="${flash.message}" >
      <div class="message" >${flash.message}</div>
    </g:if>
    <h1 style="margin-left:20px;">Bem Vindo a Livraria Virtual</h1>
    <p style="margin-left:20px;width:80%">Para modificar os dados é necessário
proceder o acesso ao sistema:</p>
    <br />
    <g:form action="validate" >
      <input type="hidden" name="cNome" value="${cNome}" >
      <input type="hidden" name="aNome" value="${aNome}" >
      <table>
```

cont. entrar.gsp

```
        <tr class="prop" >
            <td class="name" >
              <label for="login" >Usuário:</label>
            </td>
            <td class="value" >
              <input type="text" id="login" name="login" value="" >
            </td>
          </tr>
          <tr class="prop" >
            <td class="name" >
              <label for="senha" >Senha:</label>
            </td>
            <td class="value" >
              <input type="password" id="senha" name="senha" value="" >
            </td>
          </tr>
          <tr>
            <td></td>
            <td><input type="submit" value="Acessar" /></td>
          </tr>
        </table>
      </g:form>
    </body>
</html>
```

Grails Deployment

- Container
  - É a parte do servidor destinado a alojar a tecnologia Java
    - Tomcat: Container Web para Servlet e JSP
    - Apache: Servidor web
- Deploy → Implantar
  - Instalar a sua aplicação em um servidor de aplicações, mais especificamente, em um container.

Grails Deployment

- `grails war` → gera um arquivo .war, o servidor retornará para o browser a aplicação rodando, já em modo de produção.

```
>>grails war
```

  - Copie o arquivo .war gerado pelo comando para o diretório dentro do container, no servidor. No nosso caso é o Apache-Tomcat.
    - `…/apache-tomcat-x.x.xx/webapps/`

GRAILS

Grails Plugins

- Instalando Plugins
  - `grails install-plugin [URL/File]`
  - `grails install-plugin [name] [version]*`
    - A versão é opcional
  - `grails install-plugin http://foo.com/grails-bar-1.0.zip`
  - `grails install-plugin ../grails-bar-1.0.zip`
  - `grails install-plugin jsecurity`
  - `grails install-plugin jsecurity 0.1`

Grails Plugins

- http://www.grails.org/plugin/category/all
  - Listagem de plugins disponíveis;

O plugin RichUI

- Esse plugin gera componentes gráficos para realçar as nossas visões;
  - http://www.grails.org/RichUI+Plugin
  - grails install-plugin .../grails-richui-0.8.zip
    - Auto Complete;
    - Calendários;
    - Tabs “Guias”;
    - Editor de texto;
    - ...

O plugin RichUI

- Adicione `<resource:richTextEditor type="advanced" />` dentro da tag <head> nas views onde há fomulário com campos do tipo conteúdo;
- Substitua a tag `<input type='textArea'.../>` do formulário para `<richui:richTextEditor name="conteudo" value="${fieldValue(bean: postInstance,field:'conteudo')}"/>`

Plugins Interessantes

- Spring Security Core;
- Correios-br;
- Searchable;
- Commentable;
- Mail;
- Avatar;
- Twitter

Referências

- Getting Started with Grails Second Edition
  - Scott Davis e Jason Rudolph
- Introduction_to_Groovy_and_Grails
  - Mohamed Seifeddine
- Grails: do Groovy à Web
  - Artigo DevMedia - Henrique Lobo Weissmann
- Em Busca do Grails
  - Fernando Anselmo
- Site oficial do Grails
  - http://www.grails.org/Quick+Start

Referências

- Criando um blog em cinco minutos com Grails
  - http://www.michelsilva.net/index.php?option=com_content&task=vie
- Grails: um guia rápido e indireto
  - http://www.itexto.net/devkico/?page_id=220