

Grails

GORM

Aula 2 e 3

Professor: Bruno Gama Catão

Agenda

- Classes domínio
- Tipos de atributos
- Relacionamentos
- Criando e Alterando Registros
- Consultas
 - get
 - dynamic finders
 - list
- criteria
- Controladores
 - mapeamento de URLs
 - parâmetros
 - render e redirect
 - exibindo uma view

Classes de Domínio

- Comando:
 - `grails create-domain-class NomeClasse`
- Exemplo:
 - `grails create-domain-class Filme`

Tipos de Atributos

- Texto
 - String
- Números Inteiros
 - int, long, Integer, Long, BigInteger
- Números Decimais
 - float, double, Float, Double, BigDecimal
- Booleanos
 - boolean
- Data e hora
 - Date
- Arquivos
 - byte[]

Exemplo

```
1 package locadora-  
2 -  
3 class Filme {  
4     String titulo-  
5     String sinopse-  
6     int numeroDeOscars-  
7     Date dataLancamento-  
8 }
```

Tipos Grails x SQL

Grails	SQL
String	VARCHAR
int	INTEGER
double	NUMBER
byte[]	BLOB
boolean	BYTE

Modificando o mapeamento

- Closure **mapping**
- Opções
 - **table**
 - atributo **column:** 'nomeDaColuna'
 - atributo **sqlType:** 'tipoSql'

Exemplo

```
1 package locadora
2
3 class Filme {
4     String titulo
5     String sinopse
6     int numeroDeOscars
7     Date dataLancamento
8
9     static mapping = {
10         sinopse sqlType: 'longtext'
11     }
12 }
```


Validações - Constraints

- blank
- creditCard
- email
- inList
- matches
- max
- min
- notEqual
- nullable
- range
- scale
- size
- unique
- url
- validator
- widget

Exemplo

```
1 package locadora-  
2 -  
3 class Filme {  
4     String titulo-  
5     String sinopse-  
6     int numeroDeOscars-  
7     Date dataLancamento-  
8     -  
9     static constraints = {  
10         titulo(blank: false, unique: true)-  
11         sinopse(blank: false, widget: 'textarea')-  
12         numeroDeOscars(min: 0)-  
13         dataLancamento(max: new Date())-  
14     }-  
15     -  
16     static mapping = {  
17         sinopse sqlType: 'longtext'-  
18     }-  
19 }
```

Configurando o Banco de Dados

- Arquivo
 - `grails-app/conf/DataSource.groovy`
- Ambientes (environments)
 - development
 - test
 - production

```
1  ▼ dataSource {
2      pooled = true
3      driverClassName = "org.hsqldb.jdbcDriver"
4      username = "sa"
5      password = ""
6  ▲ }
7  ▼ hibernate {
8      cache.use_second_level_cache = true
9      cache.use_query_cache = true
10     cache.provider_class = 'net.sf.ehcache.hibernate.EhCacheProvider'
11 ▲ }
12 // environment specific settings
13 ▼ environments {
14     ▼ development {
15         ▼ dataSource {
16             dbCreate = "create-drop" // one of 'create', 'create-drop', 'update'
17             url = "jdbc:hsqldb:mem:devDB"
18         ▲ }
19     ▲ }
20     ▼ test {
21         ▼ dataSource {
22             dbCreate = "update"
23             url = "jdbc:hsqldb:mem:testDb"
24         ▲ }
25     ▲ }
26     ▼ production {
27         ▼ dataSource {
28             dbCreate = "update"
29             url = "jdbc:hsqldb:file:prodDb;shutdown=true"
30         ▲ }
31     ▲ }
32 ▲ }
```

Utilizando o MySQL

- Instalando o plugin (terminal):
 - `grails install-plugin mysql-connectorj`
- Criando o banco de dados (MySQL):
 - `create schema Locadora_development`

```
1 ▼ dataSource {
2     pooled = true
3     driverClassName = "com.mysql.jdbc.Driver"
4     username = "root"
5     password = "12345"
6 ▲ }
7 ▼ hibernate {
8     cache.use_second_level_cache = true
9     cache.use_query_cache = true
10    cache.provider_class = "net.sf.ehcache.hibernate.EhCacheProvider"
11 ▲ }
12
13 // environment specific settings
14 ▼ environments {
15 ▼     development {
16 ▼         dataSource {
17             dbCreate = "update" // one of 'create', 'create-drop', 'update'
18             url = "jdbc:mysql://localhost:3306/Locadora_development"
19 ▲         }
20 ▲     }
21 ▼     test {
22 ▼         dataSource {
23             dbCreate = "update"
24             url = "jdbc:mysql://localhost:3306/Locadora_test"
25 ▲         }
26 ▲     }
27 ▼     production {
28 ▼         dataSource {
29             dbCreate = "update"
30             url = "jdbc:mysql://localhost:3306/Locadora_production"
31 ▲         }
32 ▲     }
33 ▲ }
```

Criando as tabelas

- Basta executar a aplicação
 - `grails run-app`

Verificando as tabelas

```
+-----+
| Tables_in_locadora_development |
+-----+
| filme                            |
+-----+
1 row in set (0.00 sec)
```

- use Locadora_development;
- show tables;
- desc filme;

```
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | bigint(20) | NO | PRI | NULL | auto_increment |
| version | bigint(20) | NO | | NULL | |
| data_lancamento | datetime | NO | | NULL | |
| numero_de_oscars | int(11) | NO | | NULL | |
| sinopse | longtext | NO | | NULL | |
| titulo | varchar(255) | NO | UNI | NULL | |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)
```


Relacionamentos

- Direto
- belongsTo
- hasMany

Relacionamento Direto

- Você pode ter um atributo que referencie outra classe de domínio;
- Por exemplo, um filme pode ter um atributo **estudio**.

Exemplo

- Criando o Estúdio
 - `grails create-domain-class Estudio`

Estudio

```
1 package locadora
2
3 class Estudio {
4     String nome
5     String email
6     String telefone
7     String endereco
8
9     static constraints = {
10         nome(blank: false)
11         email(blank: false, email: true)
12         telefone(blank: false, matches: /^(\d{2})\d{4}-\d{4}$/)
13         endereco(blank: false)
14     }
15 }
```

Relacionamento Filme - Estúdio

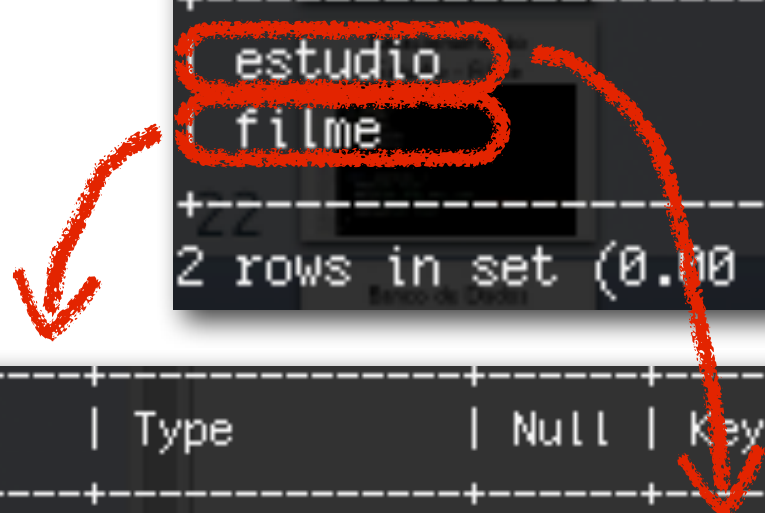
```
1 package locadora
2
3 class Filme {
4     String titulo
5     String sinopse
6     int numeroDeOscars
7     Date dataLancamento
8     Estudio estudio
9
10    static constraints = {
11        titulo(blank: false, unique: true)
12        sinopse(blank: false, widget: 'textarea')
13        numeroDeOscars(min: 0)
14        dataLancamento(max: new Date())
15    }
16
17    static mapping = {
18        sinopse sqlType: 'longtext'
19    }
20 }
```

Relacionamento Estudio - Filme

```
1 package locadora
2
3 class Estudio {
4     String nome
5     String email
6     String telefone
7     String endereco
8
9     static hasMany = [filmes: Filme]
10
11     static constraints = {
12         nome(blank: false)
13         email(blank: false, email: true)
14         telefone(blank: false, matches: /\d{2}\d{4}-\d{4}$/)
15         endereco(blank: false)
16     }
17 }
```

Banco de Dados

```
+-----+
| Tables_in_locadora_development |
+-----+
| estudio |
| filme  |
+-----+
2 rows in set (0.00 sec)
```



```
+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+
| id    | bigint(20) | NO | PRI | NULL | auto_increment |
| version | bigint(20) | NO | | NULL | |
| data_lancamento | varchar(255) | NO | | NULL | |
| numero_de_oscars | int(11) | NO | | NULL | |
| sinopse | text | NO | | NULL | |
| titulo | varchar(255) | NO | | NULL | |
| estudio_id | bigint(20) | NO | | NULL | |
+-----+
7 rows in set (0.00 sec)
```

```
+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+
| id    | bigint(20) | NO | PRI | NULL | auto_increment |
| version | bigint(20) | NO | | NULL | |
| email | varchar(255) | NO | | NULL | |
| endereco | varchar(255) | NO | | NULL | |
| nome | varchar(255) | NO | | NULL | |
| telefone | varchar(255) | NO | | NULL | |
+-----+
5 rows in set (0.01 sec)
```

e o belongsTo ?

- Ele gera uma relação de dependência;
- Relação **pertence a ...**
- Por exemplo, se fizer sentido que, caso um estúdio seja excluído, automaticamente seus filmes também sejam, devemos utilizar uma relação **belongsTo**.

Relacionamento Filme - Estudio

```
1 package locadora
2
3 class Filme {
4     String titulo
5     String sinopse
6     int numeroDeOscars
7     Date dataLancamento
8
9     static belongsTo = [estudio : Estudio]
10
11     static constraints = {
12         titulo(blank: false, unique: true)
13         sinopse(blank: false, widget: 'textarea')
14         numeroDeOscars(min: 0)
15         dataLancamento(max: new Date())
16     }
17
18     static mapping = {
19         sinopse sqlType: 'longtext'
20     }
21 }
```

Criando novas instâncias

- `def instancia = new ClasseDominio()`
ou
- `def instancia = new ClasseDominio(dicionário)`

Listas

- Listas
 - `def lista = []`
 - `def lista2 = ['banana', 'abacate', 'abacaxi']`
 - `lista << 'Valor'`
 - `lista << 'Outro valor'`
 - `print lista[0]`
 - `lista[1] = 'Muda o valor'`

Dicionários

- `def dicionario = [:]`
- `dicionario['idade'] = 30`
- `dicionario['altura'] = 1.60`
- `dicionario['nome'] = 'Fulano'`
- `def dicionario2 = [idade: 30, altura: 1.60]`

Gravando no Banco de Dados

- `instancia.save()`
- `instancia.save(opções)`
- Opções:
 - `flush`
 - `failOnError`
- Exemplo:
 - `instancia.save(flush: true, failOnError: true)`

Recuperando uma Instância do Banco de Dados

- `def instancia = ClasseDeDominio.get(id)`

Alterando uma Instância

- `instancia.atributo1 = valor1`
- `instancia.atributo2 = valor2`
- ...
- `instancia.atributoN = valorN`
- `instancia.save()`

Alterando uma Instância

- `instancia.properties = dicionário`
- `instancia.save()`

Excluindo uma Instância

- `instancia.delete()`
ou
- `instancia.delete(opções)`

Opções:

- `failOnError`
- `flush`

Navegando nos Relacionamentos

- `def matrix = Filme.get(1)`
- `def estudio = matrix.estudio`
- `def filmes = estudio.filmes`
- `def avatar = new Filme(...)`
- `estudio.addToFilmes(avatar)`

Consultas

- `ClasseDeDominio.list(opções)`
- Opções:
 - `max`
 - `offset`
 - `sort: "campo"`
 - `order: "asc|desc"`

Dynamic Finders

- `ClasseDeDominio.findByAtributo(valor)`
- `ClasseDeDominio.findAllByAtributo(valor)`
- Exemplo:
 - `Filme.findByTitulo('Matrix')`
 - `Filme.findAllByNumeroDeOscarsGreaterThanOrEquals(1)`

Dynamic Finders

- Podem combinar até 2 atributos
- `ClasseDeDominio.findByAtributo1AndAtributo2(valor1, valor2)`
- `ClasseDeDominio.findByAtributo1AndAtributo2(valor1, valor2)`

Operadores de Comparação

- LessThan
- LessThanEquals
- GreaterThan
- GreaterThanEquals
- Between
- Like
- Ilike (i.e. ignorecase like)
- IsNotNull
- IsNull
- Not
- NotEqual
- And
- Or
- InList

Exemplos

```
def b = Book.findByTitle("The Shining")
b = Book.findByTitleAndAuthor("The Sum of All Fears", "Tom Clancy")
b = Book.findByReleaseDateBetween(firstDate, new Date())
b = Book.findByReleaseDateGreaterThanOrEquals(firstDate)
b = Book.findByReleaseDateLessThanOrEquals(firstDate)
b = Book.findByTitleLike("%Hobbit%")
b = Book.findByTitleIlike("%Hobbit%") // ignores case
b = Book.findByTitleNotEqual("Harry Potter")
b = Book.findByReleaseDateIsNull()
b = Book.findByReleaseDateIsNotNull()
b = Book.findPaperbackByAuthor("Douglas Adams")
b = Book.findNotPaperbackByAuthor("Douglas Adams")
b = Book.findByAuthorInList(["Douglas Adams", "Hunter S. Thompson"])
```

Criteria

```
ClasseDeDominio.withCriteria {  
    operador('atributo', valor)  
}
```


Exemplo 1

```
def consumidorInstanceList = Consumidor.withCriteria {  
  if (params.estadoEmail && params.estadoEmail != 'TODOS') {  
    eq('estadoEmail', EstadoEmail.valueOf(params.estadoEmail))  
  }  
  maxResults max  
  firstResult offset  
  if (params.sort) {  
    order(params.sort, params.order)  
  } else {  
    order('nome', 'asc')  
  }  
}
```

Exemplo 2

```
def results = Account.withCriteria {  
    like("holderFirstName", "Fred%")  
    and {  
        between("balance", 500, 1000)  
        eq("branch", "London")  
    }  
    maxResults(10)  
    order("holderLastName", "desc")  
}
```

Operadores Criteria

- between
- eq
- lt
- le
- gt
- ge
- like
- ilike
- inList
- isNull
- isNotNull
- maxResults
- firstResult
- order