Grails Model Domain Classes

Alexis Plantin

Sommaire

- Création des Domain Classes
- Persister la donnée
- Contraintes & Validation
  - Contraintes sur un exemple
  - Contraintes disponibles
  - L'interface Errors de Spring
  - Custom Validators
- Propriétés Transient
- Mapping avec la Base de données
- Relations
  - One-to-one
  - One-to-many
  - Many-to-many
- Classes & Héritage
- Objects embarqués

Création des Domain Classes

Création des Domain Classes

```
grails create-domain-class wrh.Product
```

- 2 fichiers créés:
  - La Domain Class dans `grails-app/domain`
  - Le fichier de tests unitaires dans `src/test`

Persister la donnée

Persister la donnée

```
package wrh

class Product {

    String name
    Double weight

    static constraints = {
    }
}
```

Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	NULL	auto_increment
version	bigint(20)	NO		NULL	
name	varchar(255)	NO		NULL	
weight	double	NO		NULL	

Contraintes & Validation

Constraintes & Validation

- Contraintes d'un Product (exemple)

```
static constraints = {
    name(blank: false, maxSize: 30)
    weight(min: 0d, scale: 2)
}
```

Contraintes & Validation

- Contraintes disponibles

Name	Example	Description
blank	login(blank:false)	Set to false if a string value cannot be blank
creditCard	cardNumber(creditCard:true)	Set to true if the value must be a credit-card number
email	homeEmail(email:true)	Set to true if the value must be an e-mail address
inList	login(inList:['Joe', 'Fred'])	Value must be contained within the given list
length	login(length:5..15)	Uses a range to restrict the length of a string or array
min	duration(min:1)	Sets the minimum value
minLength	password(minLength:6)	Sets the minimum length of a string or array property
minSize	children(minSize:5)	Sets the minimum size of a collection or number property
matches	login(matches:'[a-zA-Z]+')	Matches the supplied regular expression
max	age(max:99)	Sets the maximum value
maxLength	login(maxLength:5)	Sets the maximum length of a string or array property
maxSize	children(maxSize:25)	Sets the maximum size of a collection or number property
notEqual	login(notEqual:'Bob')	Must not equal the specified value
nullable	age(nullable:false)	Set to false if the property value cannot be null
range	age(range:16..59)	Set to a Groovy range of valid values
scale	salary(scale:2)	Set to the desired scale for floating-point numbers
size	children(size:5..15)	Uses a range to restrict the size of a collection or number
unique	login(unique:true)	Set to true if the property must be unique
url	homePage(url:true)	Set to true if a string value is a URL address

Contraintes & Validation

- L' interface Errors de Spring

```
package org.springframework.validation;
interface Errors {
List getAllErrors();
int getErrorCount();
FieldError getFieldError(String fieldName);
int getFieldErrorCount();
List getFieldErrors(String fieldName);
Object getObjectName();
boolean hasErrors();
boolean hasFieldErrors(String fieldName);
// ...x remaining methods
}
```

Contraintes & Validation

- Peut être utilisé pour retrouver les erreurs sur nos objets

```
if(!product.save()) {
    product.errors.allErrors.each {
      println it.defaultMessage
    }
}
```

Contraintes & Validation

- Custom Validator

```
weight(min: 0d, scale: 2, validator: { val, obj ->
    if(obj.name == 'Weight must be 20' && val != 20) {
        return false
    }
})
```

- Une closure avec 2 paramètres:
  - val: la valeur de l'attribut à valider
  - obj: l'objet à valider
- Peut également retourner un message code

Propriétés Transient

Propriétés Transient

- Propriétés non stockées dans le base de données
- getter + setter => Un attribut persisté

```
class Product {

    String name
    Double weight
    String code

String getUpperCaseName() {
    name.toUpperCase()
}

static transients = ['code', 'upperCaseName']
```

Mapping avec la Base de données

Mapping avec la Base de données

- Utile avec les bases de données existantes
- Permet de changer le nom des tables et des colonnes

```
class Product {

    String name
    Double weight

    static mapping = {
        id column: 'product_id'
        name column: 'product_name'
        weight column: 'product_weight'
        version false
        table 'product_table'
    }
}
```

Relations

Relations: One-to-one

- Relation uni-directionnelle
  - Déclarer simplement un attribut avec le type d'une domain class
  - Les modifications sur l'objet référencé sont automatiquement sauvées lorsque l'objet référent est sauvé

```
class Product {
    Description description
}
```

Relations: One-to-one

- Product est l'élément important
- `myDescription.product = myProduct`
  ne fera pas automatiquement
  `myProduct.description = myDescription`

```
class Product {
    Description description
}

Class Description {
    static belongsTo = [product: Product]
}
```

Relations: One-to-one

- Pas de référence au owner

```
class Product {
    Description description
}

Class Description {
    static belongsTo = [Product]
}
```

Relations: One-to-many

```
class Location {

    String name
    String country
    String address

    static hasMany = [warehouses: Warehouse]
}

class Warehouse {

    String name

    static belongsTo = [location: Location]
}
```

Relations: Many-to-many

- A besoin d'un belongsTo d'un côté

```
class Warehouse {

    String name

    static hasMany = [staffs: Staff]
}

class Staff {

    String name
    Date since

    static hasMany = [warehouses: Warehouse]
    static belongsTo = Warehouse
}
```

Relations: Many-to-many

- En utilisant une table intermédiaire

```
class Warehouse {
    String name
}

class Product {
    String name
    Double weight
}

class Entry {
Integer quantity
Warehouse warehouse
Product product
    static constraints = {
    }
}
```

Classes & Héritage

Classes & Héritage

- Etendez simplement vos classes

```
class Person {
    String firstName
    String lastName
}

class Employee extends Person {
    String employeeNumber
    Double salary
}

class Customer extends Person {
    String customerNumber
    Integer fidelity
}
```

Classes & Héritage

- Un attribut class pour distinguer les sous-classes
- Tous les attributs de toutes les sous-classes dans la même table

Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	NULL	auto_increment
version	bigint(20)	NO		NULL	
first_name	varchar(255)	NO		NULL	
last_name	varchar(255)	NO		NULL	
class	varchar(255)	NO		NULL	
customer_number	varchar(255)	YES		NULL	
fidelity	int(11)	YES		NULL	
employee_number	varchar(255)	YES		NULL	
salary	double	YES		NULL	

Classes & Héritage

```
class Person {

    String firstName
    String lastName

    static mapping = {
        tablePerHierarchy false
    }
}
```

Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	NULL	auto_increment
version	bigint(20)	NO		NULL	
first_name	varchar(255)	NO		NULL	
last_name	varchar(255)	NO		NULL	

Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	NULL	
employee_number	varchar(255)	NO		NULL	
salary	double	NO		NULL	

Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	NULL	
customer_number	varchar(255)	NO		NULL	
fidelity	int(11)	NO		NULL	

Objects embarqués

Objects embarqués

```
class Product {

    String name
    Double weight
    Description description

    static embedded = ['description']
}
```

Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	NULL	auto_increment
version	bigint(20)	NO		NULL	
description_name	varchar(255)	YES		NULL	
name	varchar(30)	NO		NULL	
weight	double	NO		NULL	

Questions?