06.6주차:Validation설정및Tip

GORM Validation & Tip

Table of Contents

Validation

grails에서는 도메인 모델을 정의할 때 기본적인 Validation과 사용자가 직접 custom Validation을 정의할 수 있다.
grails Validation

attributes

scaffolding기능을 사용하여 화면을 동적으로 생성될 때 입력할 수 있는 값(콤보박스)가 attributes 태그로 설정한 범위 내만 표시되도록 정의할 수 있다.

```
1 birthDate attributes: [years: 2000..2011]
```

bindable

도메인 프러퍼티의 data binding 여부를 설정한다.

```
1 salary bindable: false
2 firstName size: 5..15, bindable: true
3 department bindable: true
```

grails bindable

blank

도메인 프러퍼티에 공백이 입력가능한지 여부를 설정한다.

```
login blank: false
```

creditCard

도메인 프러퍼티의 신용카드 번호 입력 여부를 설정한다.

```
cardNumber creditCard:true
```

email

도메인 프러퍼티의 이메일 입력 여부를 설정한다.

```
homeEmail email: true
```

inList

도메인 프러퍼티의 입력값이 컬렉션 안에 있는 값들만 입력 가능하도록 설정한다.

```
name inList: ["Joe", "Fred", "Bob"]
```

matches

도메인 프러퍼티의 입력값이 정규식에 맞는 값이 입력 가능하도록 설정한다.

```
login matches: "[a-zA-Z]+"
```

max

도메인 프러퍼티에 입력가능한 최대값을 설정한다.

```
age max: new Date()
price max: 999F
```

maxSize

도메인 프러퍼티에 입력가능한 최대 문자 Size를 설정한다.

```
children maxSize: 25
```

min

도메인 프러퍼티에 입력가능한 최소값을 설정한다.

```
age min: new Date()
price min: 0F
```

minSize

도메인 프러퍼티에 입력가능한 최소 문자 Size를 설정한다.

```
children minSize: 25
```

notEqual

도메인 프러퍼티에 일치하지 않아야 하는 데이터를 설정한다.

```
username notEqual: "Bob"
```

nullable

도메인 프러퍼티에 NULL 설정 가능 여부를 설정한다.

```
age nullable: true
```

기본적으로 grails의 domail의 nullable은 true로 설정되어 있다. 해당 설정을 바꾸기 위해서는
grails-app/conf/Config.groovy의 grails.databinding.convertEmptyStringsToNul 설정값을 false로 변경하면 된다.

```
// grails-app/conf/Config.groovy// the default value for this property is true
grails.databinding.convertEmptyStringsToNull = false
```

range

도메인 프러퍼티의 입력값 범위를 설정한다.

```
age range: 18..65
```

scale

Relationships의 갯수를 설정한다.
1:N의 관계에서 1이 소유할 수 있는 N의 갯수를 정의한다.

```
salary scale: 2
```

unique

도메인 프러퍼티가 데이터베이스 레벨에서 유니크 여부를 설정한다.

```
username unique: true
```

한개의 컬럼이 아닌 여러 컬럼이 합쳐져서 unique여야 할 경우는 아래와 같이 설정한다.

```
username(unique: ['group', 'department'])
```

size

도메인 프러퍼티의 입력되야 하는 최소,최대 사이즈를 정의한다.

```
children size: 5..15
```

url

도메인 프러퍼티의 입력값이 URL 형식에 맞게 입력되도록 정의한다.

```
homePage url: true
```

validator(Custom Validation)

grails에서 정의한 Validation이 아닌 사용자가 Validation의 정의가 필요할 경우 Custom Validation을 아래와 같이 정의할 수 있다.

```
Adds custom validation to a field.

// Closure with two arguments, the second being the object itself
password1 validator: { val, obj ->
    obj.password2 == val
}

// Closure with three arguments, the third being the errors object
password1 validator: { val, obj, errors ->
    if (!(obj.password2 == val)) errors.rejectValue('password1', 'noMatch')
}

// Examples that pass arguments to the error message in message.properties

// Example 1: Using the "implicit" argument 0 (property name)

class Person {

    String name

    static constraints = {
        name validator: {
      if (!it) return ['entryMissing']
       }
}

// This maps to a message
// person.name.entryMissing=Please enter a name in the field {0}

// Example 2: Using the "implicit" arguments 0 (property name) and 2 (property value)

class Person {

    Integer yearOfBirth

    static constraints = {
        yearOfBirth validator: {
      if (yearOfBirth>2013) return ['yearTooBig']
       }
}

// This maps to a message
// person.name.entryMissing=The value {2} entered in the field {0} is not valid because it lies in the future.
// You will note when using this kind of message on Integers that years are displayed as 1,990 instead of 1990.
// This is addressed in the next example.

// Example 3: Much more complex
```

```
// Example 3: much more complex

class Astronaut {

    Integer yearOfBirth
    Integer yearOfFirstSpaceTravel

        yearOfFirstSpaceTravel validator: { val, obj ->
                if (val < obj.yearOfBirth) ['datePriorTo', val.toString(), obj.yearOfBirth]
                else if (val < (obj.yearOfBirth+18)) ['maybeABitTooYoung', val-obj.yearOfBirth]
        }
}

// Respective messages
// Note that argument 3 is the property value converted toString to avoid the unwanted formatting as described before.
astronaut.yearOfFirstSpaceTravel.datePriorTo=The value {3} entered for the year of the first space travel is prior to the year of birth ({4})
astronaut.yearOfFirstSpaceTravel.maybeABitTooYoung={3} years seems a bit young for travelling to space, dude!
```

widget

gsp에서 도메인 프러퍼티를 출력할 때 사용하는 html 컴포넌트를 정의할 수 있다.

```
description widget: 'textarea'
```

grails & hibernate

grails에서 gorm을 사용하지 않고 hibernate를 Annotation을 정의하여 사용할 수 있다.

추후 작성 예정

Eager and Lazy Fetching

Eager Fetching : 관계되어 있는 인스턴스를 필요할 때마다 각각 개별로 1개씩 질의를 수행하여 가져온다.
Lazy Fetching : 관계되어 있는 인스턴스를 한번에 모두 조회한다.
추후 보충 작성 예정

Events and Auto Timestamping

GORM의 경우네 database의 레벨의 트리거와 같이 CRUD 이벤트가 발생할 때 특정 클로저를 등록할 수 있다.
등록할 수 있는 클로저는 아래와 같다.

The beforeInsert event

```
class Person {
   Date dateCreated
   def beforeInsert = {
       dateCreated = new Date()
   }
}
```

The beforeUpdate event

```
class Person {
   Date dateCreated
   Date lastUpdated
   def beforeInsert = {
       dateCreated = new Date()
   }
   def beforeUpdate = {
       lastUpdated = new Date()
   }
}
```

The beforeDelete event

```
class Person {
   String name
   Date dateCreated
   Date lastUpdated
   def beforeDelete = {
      new ActivityTrace(eventName:"Person Deleted",data:name).save()
   }
}
```

The onLoad event

```
class Person {
   String name
   Date dateCreated
   Date lastUpdated
   def onLoad = {
      name = "I'm loaded"
   }
}
```

Automatic timestamping

gorm의 경우 'lastUpdated' 'dateCreated'을 도메인 모델에 정의하는 것만으로도 자동으로 해동 프러퍼티에 값을 입력 수정해준다. 해당 기능을 사용하지 않으려면 아래와 같이 정의하면 된다.

```
class Person {
   Date dateCreated
   Date lastUpdated
   static mapping = {
      autoTimestamp false
   }
}
```

실제 수행 장면 추가

링크 목록

- grails Validation - http://grails.org/doc/2.0.2/ref/Constraints/Usage.html
- grails bindable - http://grails.org/doc/2.0.2/ref/Constraints/bindable.html

---