RxLogix Corporation

Grails - Domain

RxLogix

Agenda

- ***Relational Objects and ORM Introduction***
- ***Grails Domain***
- ***Domain class structure***
- ***Domain Persistence Flow***
- ***Domain Relationship***
- ***Questions***

Relational Objects and ORM Introduction

Relational Objects are the objects which map with the database tables and operation on that table is done (indirectly) through relations objects.

The mechanism through which an object is mapped with the database tables is called Object Relational Mapping (ORM)

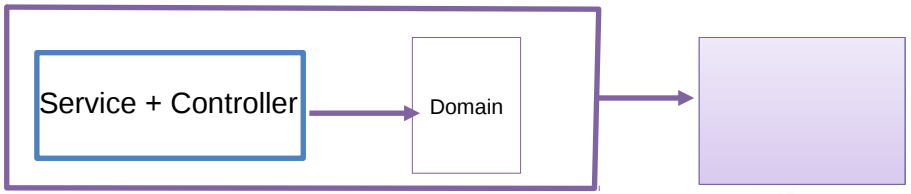
Some Benefits:-

- Reduction of code and application structure is maintained.
- No need to write the complex sql code.

Grails Domain

Domain is the building block of the Grails application. It is the gateway by which our application interacts with the database.

Grails uses Hibernate to Map domain classes with database



Domain Class as Grails Artefact

Creating a domain class:

grails create-domain-class <class name>

Class will be created in project -> grails-app -> domain

Domain Class Structure

- Domain contains the properties, mapping and constraints etc.
- Constraints are the data validation rule that we impose on the properties of domain. (Ref: Constraints)
- Mapping represents how we map a table with the domain class. (Ref: Domain mapping)

```
class Employee {
        String firstName
        String lastName
        String email
        String password
        String city // getter & setters are generated automatically
        int age


        static constraints = {
                city nullable : true
                firstName blank: false, nullable: false
                email(unique: true)
        }


        static mapping = {
                table ‘EMP’
                lastName column ‘LNAME’
        }
```

Persistent & Transient Fields

Persistent fields:

- All the fields in a domain class are persisted to the database.
- Each field in the class will map to a column in the database.
- By default each persitence field is required (cannot have null value)

Transient fields:

- Transient properties are never written to the database
- Can apply constraints on them.
- Don't have a corresponding column in the database.

Eg:

```
class Employee {
    String firstName
    String lastName
    static transients = ['name']
    String getName() {
        return “${firstName} ${lastName}”
    }
}
```

TimeStamps

- dateCreated and lastUpdated are special date fields in grails.
- Both the fields updated automatically when the new object is created
- When object is updated only lastUpdated field gets updated

Domain Persistence Flow

- Validating a domain object:

  Employee employee = new Employee(firstName: ‘Sample’, lastName: ‘’)
  employee.validate() //returns true or false

- Saving/Updating a domain object:

  Employee employee = new Employee(
  name: ‘Test_Name’
  city: ‘Test_city’
  age: 22
)
  employee.save() //returns instance if saved otherwise null

  Valid arguments - flush,failOnError,validate

DataSource mapping

In grails DataSource.groovy plays vital role in determining db connections etc.

Datasource's name can be explicitly given to the domain class like:-

```
class Employee {
        String firstName
        String lastName


        static mapping = {
                datasource "db2"
                table 'EMP'
                firstName column: 'FNAME'
        }
}
```

So in the above example the datasource_db2 will be referred for the db connection for that domain.

Domain relationships

One-to-One

```
class Car {
    Engine engine
}

class Engine {
    static belongsTo = [car:Car]
}
```

Domain relationships

One-to-Many (No Owner)

```
class Project {
    String name
    static hasMany = [tasks: Task]
}

class Task {
    String name
    Employee assignedTo
    Employee assignedBy
    Date dateCreated
    Date lastUpdated
}
```

Domain relationships

One-to-Many (Owner)

```
class Company {
    String name
    static hasMany = [employees: Employee]
}

class Employee {
    String firstName
    String lastName
    static belongsTo = [company: Company]

    }
```

Domain relationships

Many-to-Many

```
class Employee {
    String firstName
    String lastName
    static hasMany = [projects: Project]
}

class Project {
    String name
    static belongsTo = [Employee]
    static hasMany = [employees: Employee]
    }
```

Exercise

Problem : - Create a domain class Department which will have following properties:-

departmentName
departmentId
location

Department can have many Employees.

Now create another domain class Employee with following properties:-

empName
empNumber
location

Employee can have many departments.

Create a domain system. The system can be assigned to only one employee.

Additional Task :
Identify how mapping tables are created for the above relationship in the database.

Questions??

Thank you!!