GFS Example for Grails 1.1

Table of Contents

Introduction

In order to explain how this plugin works we are going to write some code for a dummy example app which contains three type of relations (one-to-many, many-to-one and one-to-one) between four domain classes.

Resources

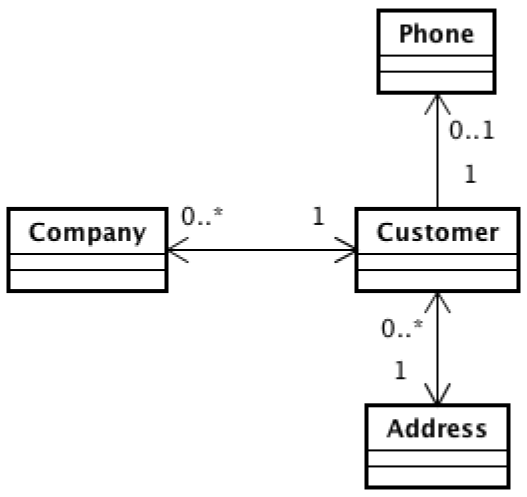
- GFSv0.1.1 plugin
- GFSv0.1.1 example application
- GFS screencast example

Prerequisites

- Grails-1.1
- Adobe Flex 3.0.0 or major

FLEX_HOME and GRAIL_HOME must be defined as enviroment variable!!

Domain Model



Scaffolding

Grails Project Creation

```
user@cubikalabs:~$ grails create-app gsf-test
```

Plugin installation

```
user@cubikalabs:~$ cd gfs-test
user@cubikalabs:~/gfs-test$ grails install-plugin flex-scaffold
user@cubikalabs:~/gfs-test$ grails compile #After installing Grails doesn't compile
it.
```

Generating domain class model

```
user@cubikalabs:~/gfs-test$ grails create-domain-class customer
user@cubikalabs:~/gfs-test$ grails create-domain-class company
user@cubikalabs:~/gfs-test$ grails create-domain-class phone
user@cubikalabs:~/gfs-test$ grails create-domain-class address
```

Let's edit generated domain classes

Company

```
importorg.cubika.labs.scaffolding.annotation.FlexScaffoldProperty
//@FlexScaffoldProperty(labelField="name") is
//The label field is displayed in edit-view
//of the relation external
@FlexScaffoldProperty(labelField="name")
class Company
{
  String name
  String address
  //One-to-Many
  static hasMany = [customers:Customer]

  static mapping =
  {
    customers lazy:false, cascade:"none"
  }

  static constraints =
  {
    name(blank:false)
    address(blank:false)
    customers(display:false)//Not view customer in Company's edit-view
  }

}
```

Customer

```
class Customer
{
  String firstName
  String lastName
  String email
  Date dateOfBirth
  Phone phone
  List addresses
  Company company
  String maritalStatus
  Integer age
  Boolean enabled

  static hasMany = [addresses:Address]
  static belongsTo = Company

  static mapping =
  {
    addresses lazy:false, cascade:"all-delete-orphan"
    company lazy:false, cascade:"none"
  }

  static constraints =
  {
    firstName(minSize:2, blank:false)
    lastName(maxSize:20)
    dateOfBirth()
    age(range:18..99)
    email(email:true, blank:false)
    //if not declared widget, the default component is a ComboBox

maritalStatus(inList:["Single","Married","Divorce","Widower"],widget:"autocomplete")
    addresses()
    //if inPlace:false, a ComboBox is created in the "edit-view"
    //of the class containing it, and it's filled with the information that makes a
    // reference of it.
    //Besides, it allows to create a new record from the edit-view of the referenced
class
    //through a button ("add")
    //by default inPlace is true
    company(inPlace:false, nullable:true)
    //The componente will be hidden when the form is setted CREATE_VIEW mode
    //and sets the defaultValue, in this case the value is true
    //If you wish, use the metaConstraint editView:false to hide component in
EDIT_VIEW mode
    enabled(createView:false,defaultValue:'true')
  }
}
```

Address

```
class Address
{
  String street
  Integer number
  String zip
  String observation
  Customer customer

  static constraints =
  {
    street(blank:false)
    //if not declared widget, the default
    //component is a NumericStepper
    number(widget:"textinput")
    zip(blank:false)
    observation(widget:"textarea")
    //Not view customer in Address' edit-view
    customer(display:false)
  }
}
```

Phone

```
class Phone
{
  String number
  String type

  static belongsTo = Customer
  static constraints =
  {
    //if not declared widget, the default
    //component is a NumericStepper
    number(widget:"textinput")
    type(inList:["Home","Movil"])
  }
}
```

CRUD Company and Customer Generation

```
user@cubikalabs:~/gfs-test$ grails generate-all-flex company
user@cubikalabs:~/gfs-test$ grails generate-all-flex customer
```

Flex compilation

```
user@cubikalabs:~/gfs-test$ grails flex-tasks
```

It's time to start up our app-server and navigate our application

```
user@cubikalabs:~/gfs-text$ grails run-app
```

open browser and go to http://localhost:8080/gfs-test

Success tips (Important information)

Relations

- many-to-one supports only inPlace:false (this declaration is not required because it's setted as a default)
- one-to-many both cases are supported inPlace:false/inPlace:true.
- one-to-one supports only inPlace:true (this declaration is not required because it's setted as a default).
- If relations are declared as inPlace:true, e.g: Customer <-> Address o Customer -> Phone, the included class (Address, Phone) must define the constraint diplay:false for property that is including "custormer(display:false)". At this moment, this restriction is not valid in generation code time and ends by abort process
- Relations must ever be lazy:false if not, BlazeDS throws a LazyInitialization exception (in future versions we are going to support this feature with DPHibernate or similar).

Constraints

- Front-End supported constraints
  - blank, email, size, minSize, maxSize, min, max, range, url, inList
  - For each Front-End constraint, a Flex validator is generated. This avoids user to persist the entity without the need of Back-End validation.

- All other constraints (Grails constraints) follows Grails validation way, doing validation on Back-End side which have the responsibility of getting feedback about errors to Front-End. This kind of errors are supported by i18n.

Import into Eclipse

If you want to know how to import a project into FlexBuilder see: How-To import project into eclipse