

Formation Gus05 Atelier 3

Auteur : Augustin Delale

Code : GUS05A003FR

Date : 29/05/2011

Avant-propos: Ce document s'inscrit dans une suite d'ateliers pratiques conçus pour faire découvrir aux développeurs Java le framework gus05. Si vous ne l'avez pas fait, je vous encourage à lire le document GUS05B001FR qui constitue une introduction au framework et à ce tutoriel.

Objectifs : Pour débiter la programmation gus05, nous allons développer une première entité graphique très simple : l'affichage d'un panneau de couleur verte.

Etapes :

1. Définition d'une entité graphique
2. Conventions de nommage et pseudo développeur
3. Ajouter un package et une classe au répertoire entity
4. Coder et compiler la classe

Etape 1 : Définition d'une entité graphique

Avant de développer une entité graphique, il est nécessaire de bien comprendre la définition générale d'une entité d'une part, et la définition d'une entité graphique d'autre part. Une entité se définit au sens strict selon les trois règles suivantes :

1. Il s'agit d'une ou plusieurs classes Java regroupées dans un unique package.
2. L'une des classes doit implémenter l'interface *Entity* définie par le framework gus05
3. Le nom du package est défini comme tel : <package-name> = gus05.entity.<entity-name>

Une entité graphique se définit quant à elle en ajoutant la règle ci-dessous :

1. La classe principale qui implémente l'interface *Entity* doit aussi implémenter l'interface *Graphic* définie de même par le framework gus05

Comme vous avez sans doute pu le constater, le framework gus05 comporte une grande majorité d'Interfaces (18 interfaces et seulement 2 classes) qui permettent notamment de définir les entités ainsi que de les caractériser. Dans le cas précis de l'entité graphique seulement deux interfaces nous intéressent dont nous allons détailler le code source :



Projet gus05

<http://gus05.forumactif.com>

```
package gus05.framework.core;  
  
public interface Entity {  
    public String getName();  
    public String getCreationDate();  
}
```

Interface de définition des entités comportant 2 méthodes :

public String getName() : fournit le nom de l'entité <entity-name>

public String getCreationDate() : fournit la date de création de l'entité (format aaaa.mm.jj)

```
package gus05.framework.features;  
  
import javax.swing.JComponent;  
  
public interface Graphic {  
    public JComponent gui();  
}
```

Interface de caractérisation graphique comportant 1 méthode :

public JComponent gui() : fournit l'objet Swing choisi pour représenter graphiquement l'entité.

L'exemple de code source d'entité présenté à l'étape 4 vous fixera très certainement les idées sur la manière dont ces différentes méthodes peuvent être implémentées par la classe de l'entité. Mais avant cela, il nous reste à préciser certains points concernant le nommage des packages et des entités.

Etape 2 : Conventions de nommage et pseudo développeur

Vous avez dû remarquer qu'il existe des conventions de nommage générales pour les packages du projet gus05 suivant la catégorie de code auxquels ils appartiennent.

Catégorie de code	Le package commence par
Framework gus05	gus05.framework.
Gestionnaires	gus05.manager.
Entités	gus05.entity.
Ressources	gus05.resource.

Tous vos packages d'entité doivent donc impérativement débiter par gus05.entity. Par ailleurs, pour reprendre la définition de l'entité, le nom du package est lié au nom de l'entité de la manière suivante : <package-name> = gus05.entity.<entity-name>



Projet gus05

<http://gus05.forumactif.com>

Dans la pratique, vous n'aurez donc qu'à vous soucier de la manière de nommer vos entités, ce qui nous amène aux conventions de nommage de celles-ci.

Une entité se nomme en respectant les mêmes conventions qu'un package (succession de mots en minuscule séparés par des points) et afin d'éviter que deux développeurs ne créent des entités en les nommant de la même manière, le premier mot choisi devra impérativement être spécifique à chaque développeur. Ce sera le "pseudo développeur".

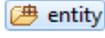
Mon pseudo développeur est "gus" et concernant les exemples de l'atelier, nous allons choisir le pseudo "charly". Quant à vous, dans la mesure où vous souhaitez contribuer au projet gus05, vous allez devoir vous choisir votre pseudo et vous assurer que celui-ci n'a pas déjà été choisi par un autre développeur. Vous pouvez pour cela me contacter et me soumettre le pseudo que vous souhaitez adopter. Il est à noter que plusieurs mots-clés tels que "gus05", "entity", "manager", "package", "default" ou "kassia" ne peuvent être utilisés.

La suite du nom est laissée à votre soin et vous êtes donc libre de nommer vos entités comme vous le désirez, en gardant à l'esprit que votre manière de les nommer reflétera l'organisation générale de votre travail et qu'elle mérite donc que vous y réfléchissiez sérieusement.

<entity-name> = <pseudo>.<ce que vous voulez>

Etape 3: Ajouter un package et une classe au répertoire entity

Il est temps de commencer à développer notre première classe d'entité ! Notre première entité consistant en l'affichage d'un panneau vert, nous allons la nommer *charly.test.panel.green* correspondant au nom de package: *gus05.entity.charly.test.panel.green*

Faites un clic droit sur l'icône du répertoire de source "entity" () et choisissez dans le menu contextuel : *New / Package*, enfin saisissez le nom du package et cliquez sur le bouton *Finish*.

Un package vide () est créé dans votre répertoire "entity". Nous allons y ajouter une classe Java que nous appellerons *GreenPanel*. De manière généralement, les noms des classes d'entités ne sont soumis à aucune contrainte. Faites un clic droit sur l'icône du package et choisissez dans le menu contextuel : *New / Class*, enfin saisissez le nom de la classe et cliquez sur le bouton *Finish*.

Une classe *GreenPanel* est créée dans le package et s'ouvre directement dans la partie de droite d'Eclipse. Cette classe est encore vide mais nous allons la compléter à l'étape suivante.

Remarques : Si vous vous êtes trompé dans vos noms de package ou de classe, vous pouvez les renommer grâce au menu contextuel *Refactor* d'Eclipse. Prenez garde à ne pas modifier les noms de classe ou de package directement dans le code source de la classe et de manipulez pas (déplacement, renommage) les fichiers java du répertoire *Workspace* d'autres moyens que le menu



contextuel Refactor d'Eclipse sous peine d'obtenir des erreurs sous Eclipse qui seront difficilement corrigables.

Etape 4 : Coder et compiler la classe

A présent, saisissez le code de votre classe comme ci-dessous :

```
package gus05.entity.charly.test.panel.green;

import java.awt.Color;
import javax.swing.JComponent;
import javax.swing.JPanel;
import gus05.framework.core.Entity;
import gus05.framework.features.Graphic;

public class GreenPanel implements Entity, Graphic {

    public String getName(){return "charly.test.panel.green";}
    public String getCreationDate(){return "2009.08.30";}

    private JPanel panel;

    public GreenPanel()
    {
        panel = new JPanel();
        panel.setBackground(Color.GREEN);
    }

    public JComponent gui()
    {return panel;}
}
```

Une fois que vous avez saisi le code source, sauvegardez les modifications dans l'éditeur en appuyant sur CTRL-S, puis compilez en appuyant sur CTRL-B.

Je ne m'étendrai pas sur le code lui-même qui est assez simple et d'un intérêt limité en lui-même (mais il faut bien commencer simplement). La compréhension du paramétrage qui sera détaillé dans l'atelier suivant est en revanche primordiale.

