Mise en situation (60 min)

Les pizzérias

réalisation **Novembre 2016**

par **Guillaume Orain** *et* **Brice Bruneau**

Table des matières

Introduction

Bonjour,

Cette mise en situation reprend dans les grandes lignes une partie des compétences qui seront utilisées dans le cadre du projet proposé.

Ce « test » à vocation d'évaluer votre niveau sans attente de résultats particuliers.

Voici les critères qui nous servirons à évaluer votre niveau :

- La clarté et qualité du code
- La bonne compréhension de l'architecture MVC
- Savoir expliquer ses choix techniques
- Savoir admettre qu'on ne sait pas

Durant toute la durée de la mise en situation (60 min) nous sommes à votre disposition pour répondre aux questions.

Voici les technologies qui sont utilisées dans la mise en situation :

- PHP 5.6 (documentation)
- Symfony (documentation)
- Doctrine (documentation)
- Twig (documentation)
- GIT (cheat sheet)

Sur votre poste vous avez :

- Un IDE (PhpStrom) et un éditeur de texte (Sublime Text)
- Un environnement de développement : Wamp (MySQL, PhpMyAdmin, Apache)
- Un navigateur (Chrome ou Firefox)
- Un client GIT (GitBash)

Votre environnement de travail est prêt à fonctionner

L'application de mise en situation traite d'un site web qui référence des pizzérias. On y trouve notamment des pizzérias, des pizzaiolos, des pizzas, …

Chaque pizzéria propose différentes pizzas sur sa carte.

Chaque pizza est composée des différents ingrédients.

Étape 1 : Mapping des entités

Voici le diagramme de classe de l'application :

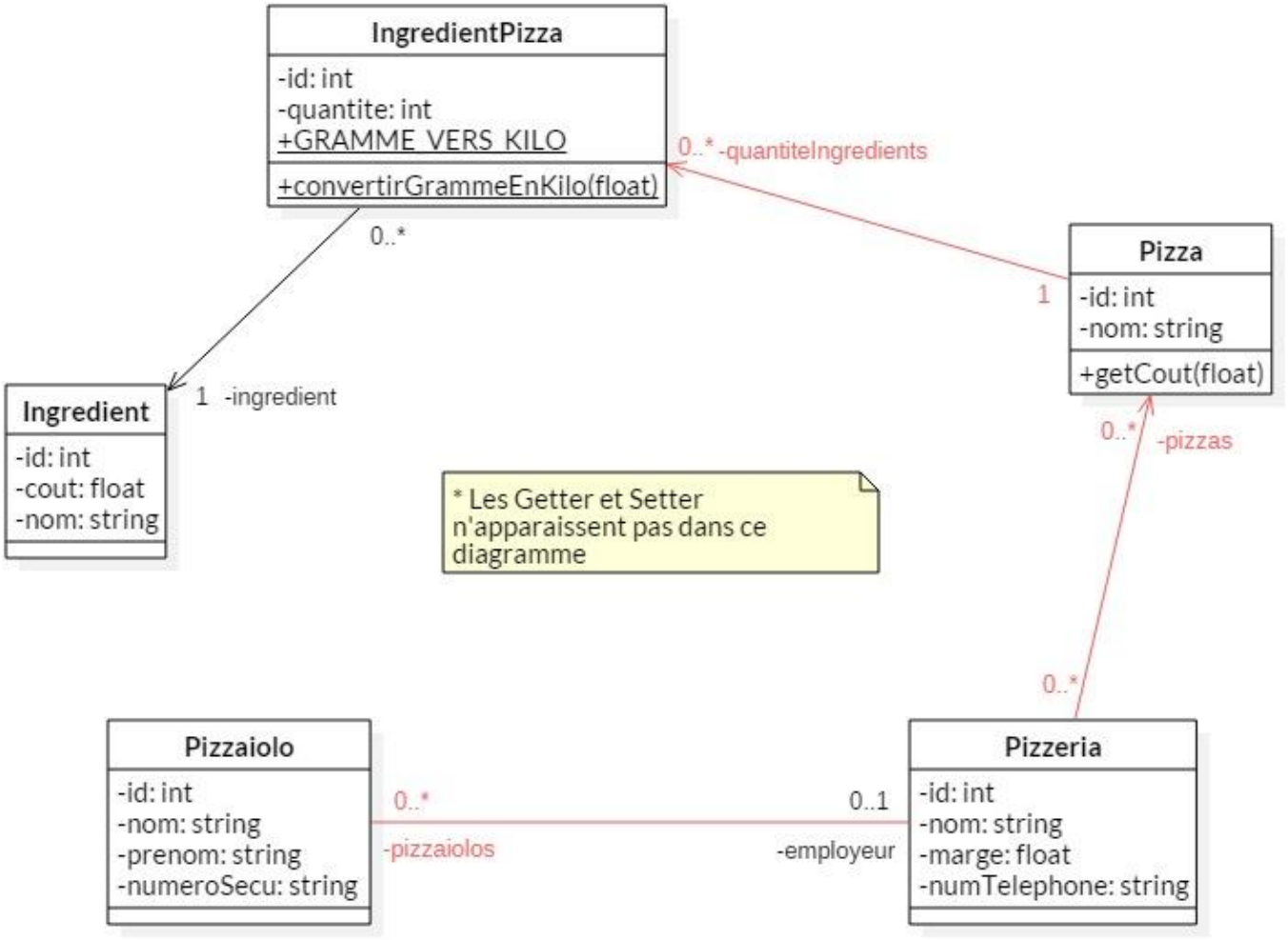

Figure 1: Diagramme de classe de l'application des pizzérias

Les relations **en rouges** ne sont pas présentes dans les différentes entités. **Commencez par réaliser les relations manquantes.**

Petite précision, dans Symfony, Doctrine est une dépendance, ce qui signifie que contrairement à la documentation officielle, il faut utiliser le namespace complet de Doctrine.

```
use Doctrine\ORM\Mapping as ORM;
```

Soit **@ORM**NomDeLaMethodeDoctrine

Dans une application Symfony, les entités se trouve généralement dans **/src/NomDeMonBundle/Entity/**

Il est impératif de respecter le nommage des différents attributs, nom de classes, …

Étape 2 : Le détail d'une pizza

Une fois le mapping réalisé, il est temps de générer la base de données.

Pour ce faire, ouvrez un terminal (dans >Tools>Run Command… sur PhpStorm) à la racine du projet.

Exécutez les commandes suivantes :

```
php bin/console doctrine:database:create
php bin/console doctrine:schema:update --force
```

Si tout s'est bien passé, vous pouvez exécuter les fixtures. Les fixtures sont tout simplement des jeux d'essais qui peuvent être facilement lancés depuis l'exécution d'un script.

Pour ce faire, exécutez alors la commande suivante puis suivez les instructions :

```
php bin/console doctrine:fixtures:load
```

Vous devriez voir apparaître ceci dans le terminal :

```
> C:\wamp64\bin\php\php5.6.16\php.exe bin/console doctrine:fixtures:load
Careful, database will be purged. Do you want to continue y/N ?y
  > purging database
  > loading [0] AppBundle\DataFixtures\ORM\IngredientData
  > loading [1] AppBundle\DataFixtures\ORM\PizzaData
  > loading [2] AppBundle\DataFixtures\ORM\PizzeriaData
  > loading [3] AppBundle\DataFixtures\ORM\PizzaioloData

Process finished with exit code 0 at 11:53:32.
Execution time: 3 207 ms.
```

Dans le cas contraire, vous avez surement raté quelque chose dans l'étape 1.

Une fois les fixtures exécutées, vous pouvez désormais vous rendre sur l'application à l'adresse suivante

```
http://localhost/app_dev.php/
```

L'objectif de cette étape est de créer une vue qui affiche le détail d'une pizza, en effet sur la page **http://localhost/app_dev.php/pizzas**, vous visualiser une liste de pizzas.

Voilà ce que l'on souhaite. Lorsqu'un utilisateur clique sur le nom d'une pizza, il devrait être rediriger vers une page contenant les informations sur la pizza (son prix de fabrication et la liste des ingrédients)

L'action du contrôleur qui gère cette étape **/src/PizzeriaBundle/Controller/PizzaController::detailAction()** existe déjà, il ne vous reste plus qu'à l'implémenter

Le fichier html **/src/PizzeriaBundle/Ressources/views/Pizza/detail.html.twig** contenant la vue de détail d'une pizza existe également. Il contient la définition des différents blocs de la vue de base de l'application. En effet **Twig** est un générateur de template permettant l'héritage.

Gardez à l'esprit qu'une pizza peut posséder plusieurs ingrédients.
Chaque ingrédient contenu dans une pizza possède une certaine quantité (exprimé en gramme)
Chaque ingrédient possède un prix au kilo.

Voilà un exemple de ce qu'on attend de vous à l'étape 2 :

Figure 2 : Exemple de la vue de détail de la pizza Flamenckuche

Félicitation, vous avez terminé l'étape 2.

Étape 3 : Créer une requête avec Doctrine

L'objectif de cette étape sera d'implémenter la méthode du *répository* des pizzérias pour trouver les informations nécessaires à l'obtention de la carte d'une pizzéria en particulier.

La méthode à implémenter existe déjà, elle se trouve dans /src/PizzeriaBundle/Repository/PizzeriaRepository::findCartePizzeria()

Vous pouvez supprimer le contenu actuel de la méthode

Les informations à afficher dans la carte d'une pizzéria sont :

- Le nom de la pizzeria
- La marge de la pizzeria
- Les différentes pizzas qui composent la carte
- Le prix (incluant la marge de la pizzeria) de vente de chaque pizza

Étape 4 : Afficher la carte d'une pizzeria

Un fois la requête récupérant les informations nécessaires à l'affichage de la carte de la pizza effectuée. Nous voulons désormais qu'un utilisateur cliquant sur le lien d'une pizzeria dans la page **http://localhost/app_dev.php/pizzerias** soit redirigé vers une vue qui représente la carte de la pizzeria.

L'action du contrôleur qui gère cette étape **/src/PizzeriaBundle/Controller/PizzeriaController::detailAction()** existe déjà, il ne vous reste plus qu'à l'implémenter

Le fichier html **/src/PizzeriaBundle/Ressources/views/Pizzeria/carte.html.twig** contenant la vue de la carte d'une pizzéria existe également.

Voilà un exemple de ce qu'on attend de vous à l'étape 4 :

Figure 3 : Exemple de la carte de la pizzéria "La Belle Rouge"

Étape 5 : Sauvegarde de votre travail

La mise en situation touche à sa fin, il est temps de sauvegarder votre travail avec GIT. Vous allez tout simplement créer une branche au format suivant :

test-prénom-nom-année-mois
Exemple pour Guillaume Orain (date du jour le 05 Novembre 2016), le nom de la branche serais **test-guillaume-orain-2016-11**

Puis envoyer cette branche sur le dépôt distant.

Pour ce faire, ouvrez un terminal Git BASH à la racine du projet.

Exécutez ensuite les commandes suivantes :

```
git branch nom-de-votre-branche     # permet de créer une branche
git checkout nom-de-votre-branche   # se déplace sur la branche créée
git status   # affiche les différents changements depuis le derniers commit
git add -A   # ajout tous les changements dans le commit
git commit -m ''Message de votre commit''   # permet de sauvegarder les changements dans le dépôt local
git push origin nom-de-votre-branche   # envoi de vos modifications vers le dépôt distant
```

Les identifiants de connexion pour envoyer vos modifications sur le dépôt distant sont :
Utilisateur : **dave-test**
Mot de passe : **Pizza2016**

Félicitation, vous avez terminé la mise en situation.