W3 - Rattrapages

W-RAT-010

My_CRUD_generator

Rattrapages rush 3

1.0

{EPITECH.}

My_CRUD_generator

repository name: Rattrapages_Rush_03
repository rights: ramassage-tek
language: PHP

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

MY_CRUD_GENERATOR

+ INTRODUCTION

Le but du projet est de réviser une notion basique du métier de développeur web : le CRUD en conception MVC.
My CRUD generator simule le comportement de **php bin/console generate:doctrine:crud** de Symfony.

+ NOTIONS

- PHP
- POO
- CRUD
- MVC

+ PROCéDURE

Au fur et à mesure de ce sujet, vous serez guidé dans les étapes permetant de simuler le comportement de la commande CLI **generate:doctrine:crud** de Symfony.

- Etape 1 : La génération de l'entité et de son model
- Etape 2 : La génération des views de lecture de l'entité
- Etape 3 & 4 : La génération des views d'écriture de l'entité

ETAPE 01 - L'ENTITé

Créez un script PHP CLI nommé **crud_generator**. Ce script devra demander à l'utilisateur le nom de l'entité à créer puis proposer à l'utilisateur de créer des champs dans cette entité. Par exemple :

Terminal

```
~/W-RAT-010> $> crud_generator -g entity my_CRUD_generator
=> Quel est le nom de l'entité que vous voulez créer ?
product
=> Quel est le nom du nouveau champ à ajouter à l'entité product ? (Tapez "done"
pour arrêter d'ajouter des champs et générer l'entité)
name
=> Quel est le type du champ name ? (string || integer) (string par défaut)
=> Quel est le nom du nouveau champ à ajouter à l'entité product ? (Tapez "done"
pour arrêter d'ajouter des champs et générer l'entité)
done
=> Entité product créée dans my_CRUD_generator avec les champs : name (string).
      => Table products créée dans la base de données my_CRUD_generator.
      => Model Product créé dans Product.php
$>
```

Au terme de cette étape vous devez avoir :

- Généré une table products dans votre base de données avec les champs
  - **id** => int primary ai
  - **name** => text
- Généré un fichier **product/Product.php** contenant une classe Product avec :
  - Les attributs :
    - **private id**
    - **private name**
  - Les méthodes :
    - Construct($name)** qui prends en paramètre les valeur des champs de l'entité
    - **public save()** qui sauve l'entité en base de donnée
    - **public static index()** qui renvoie une liste de tous les products en base de données
    - **public static show($id)** qui prends en paramètre un id de product et renvoie le product correspondant en base de données

- **public load($id)** qui prends en paramètre un id et initialise l'entité aux valeurs d'un product correspondant en base de données
- **public delete()** qui delete le product en base de données

Attention ! Pour ce sujet, vous avez tout intérêt à créer un modèle de … votre Model afin de vous servir de guide lors de la génération ce fichier. Ce qui est valable aussi pour la suite…

Pour générer un fichier, pensez aux fonctions de manipulation des fichiers de PHP : http://php.net/manual/fr/ref.filesystem.php.

Pour exécuter un script PHP facilement, pensez à son en-tête : **#!/usr/bin/php.**

Checkpoint : Attention, il est inutile de poursuivre si vous n'êtes pas arrivé correctement à ce stade. En cas de doute, vérifiez à nouveau votre travail.

Etape 02 - Voir les enregistrements

Continuez votre script **crud_generator** en lui ajoutant la possibilité de générer les vues **show** (affichage d'un enregistrement dont l'**id** est fourni) et **list** (affichage de tous les enregistrements) de votre entité. Par exemple :

Terminal

```
~/W-RAT-010> $> crud_generator -g crud product
=> Les fichiers suivants ont été créés :
      - product/list.view.php
      - product/show.view.php
=> Ces fichiers sont accessibles aux URL suivantes :
      - localhost/product/list
      - localhost/product/show?id=xx
$>
```

Etape 03 – Créer un enregistrement

Continuez votre script **crud_generator** en lui ajoutant la possibilité de générer la vue new (ajout d'un enregistrement en base de données). Par exemple :

Terminal

```
~/W-RAT-010> $> crud_generator -g crud product
=> Les fichiers suivants ont été créés :
      - product/list.view.php
      - product/show.view.php
      - product/new.php
=> Ces fichiers sont accessibles aux URL suivantes :
      - localhost/product/list
      - localhost/product/show?id=xx
      - localhost/product/new
$>
```

Etape 04 – Modifier des enregistrements

Continuez votre script **crud_generator** en lui ajoutant la possibilité de générer la vue edit (modification d'un enregistrement en base de données dont l'**id** est fourni). Par exemple :

Terminal

```
~/W-RAT-010> $> crud_generator -g crud product
=> Les fichiers suivants ont été créés :
      - product/list.view.php
      - product/show.view.php
      - product/new.php
      - product/edit.php
=> Ces fichiers sont accessibles aux URL suivantes :
      - localhost/product/list
      - localhost/product/show?id=xx
      - localhost/product/new
      - localhost/product/edit?id=xx
$>
```

Votre script devra également mettre à jour la vue list pour ajouter les possibilités de :

- Voir un enregistrement
- Modifier un enregistrement
- Supprimer un enregistrement

Votre script devra également mettre à jour la vue show pour ajouter les possibilités de :

- Modifier un enregistrement
- Supprimer un enregistrement