Arnaud Parot 2019-2021

Fiche de production

Système de gestion d’affichage de produit

Table des matières

1) Partie Base de Données/SQL

1.1) Définition du MCD et des tables/attributs

La première partie va consister à créer dans une base de données, toutes les tables nécessaires à l'affichage d'un produit et à sa définition comme son nom, son prix, sa catégorie, etc...

Dans le cadre de notre projet, nous avons décidé de créer 4 tables liées entre elles (Product, Pictures, Category, Status) contenant les informations des produits, elles disposent chacune respectivement des informations suivantes :

Products :

- Id
- Nom
- Description
- Prix HT
- Prix TTC
- Quantité en stock
- Catégorie
- Statut

Pictures :

- Id
- Id produit
- Chemin de l'image

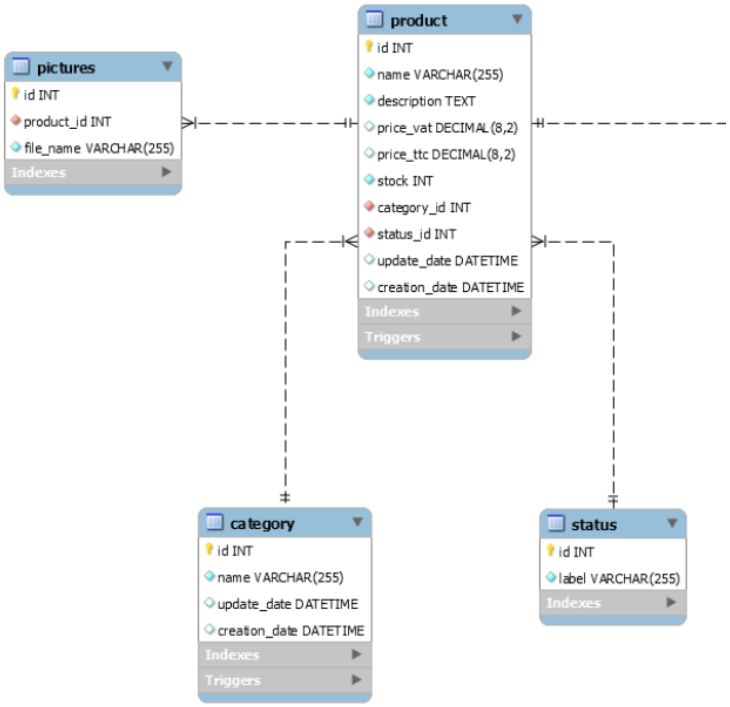
Category :

- Id
- Nom

Status :

- Id
- Libelle

Certaines tables disposent également d'un attribut « Date de modification » et « Date de création » mis à jour automatiquement lors de modification dans l'une de ses tables par le biais de triggers pour pouvoir ainsi garder une trace de quand date l'attribut et quand une de ses modifications a été effectuée. La réalisation de cette partie de la base de données correspond au MCD ci-après.



1.2) Script de création des tables

Afin de permettre la création des tables au mieux possible, nous avons créé un fichier de script appelé « Script.sql » contenant l'intégralité du script de création de la base de données (Tables, Association, Triggers, etc…) ainsi qu'un script permettant de mettre des données de base dans la BDD, ces données par défaut vont permettre le bon affichage du site lors d'une présentation, car lorsqu'il n'y a aucune donnée, rien ne s'affiche sur certaines pages web rendant la présentation de celle-ci moins tape a l'œil et agréable.

Ce fichier permet également le déploiement facile et rapide de la base de données étant donné que nous avons juste besoin d'importer le script de création via PHPMyAdmin ou tout autre moyen d'import de script a la base de données.

Les scripts de création des tables nécessaires à l'affichage des produits sont les suivants, ils correspondent respectivement au tables Category, Status, Product et Pictures.

```sql
-- -----------------------------------------------------
-- Table `dendo_jitensha`.`category`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dendo_jitensha`.`category` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NOT NULL,
  `update_date` DATETIME NULL,
  `creation_date` DATETIME NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC))
ENGINE = InnoDB;
```

```sql
-- -----------------------------------------------------
-- Table `dendo_jitensha`.`status`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dendo_jitensha`.`status` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `label` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC))
ENGINE = InnoDB;
```

```sql
-- -----------------------------------------------------
-- Table `dendo_jitensha`.`product`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dendo_jitensha`.`product` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NOT NULL,
  `description` TEXT NOT NULL,
  `price_vat` DECIMAL(8,2) NULL,
  `price_ttc` DECIMAL(8,2) NULL,
  `stock` INT NOT NULL,
  `category_id` INT NOT NULL,
  `status_id` INT NOT NULL,
  `update_date` DATETIME NULL,
  `creation_date` DATETIME NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_product_category1_idx` (`category_id` ASC),
  INDEX `fk_product_status1_idx` (`status_id` ASC),
  CONSTRAINT `fk_product_category1`
    FOREIGN KEY (`category_id`)
    REFERENCES `dendo_jitensha`.`category` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_product_status1`
    FOREIGN KEY (`status_id`)
    REFERENCES `dendo_jitensha`.`status` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```sql
-- -----------------------------------------------------
-- Table `dendo_jitensha`.`pictures`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dendo_jitensha`.`pictures` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `product_id` INT NOT NULL,
  `file_name` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC),
  INDEX `fk_pictures_product1_idx` (`product_id` ASC),
  CONSTRAINT `fk_pictures_product1`
    FOREIGN KEY (`product_id`)
    REFERENCES `dendo_jitensha`.`product` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

1.3) Création de la base de données

Dans le cadre de notre projet nous avons utilisé PHPMyAdmin pour importer la BDD depuis le script de créations que nous avons réalisé auparavant, pour l'importer, nous nous rendons sur la page principale de PHPMyAdmin et utilisons l'onglet « Importer » prévu à cet effet.

Il nous suffit ensuite de sélectionner notre fichier « Script.sql » et de cliquer sur le bouton exécuter pour que la création se fasse automatiquement si aucune erreur de syntaxe n'est présente, a noté que s'il y a des erreurs de syntaxe, le script s'exécutera jusqu'à l'erreur mais ne continuera pas après.

Nous avons maintenant créé notre base de données nécessaire à la sauvegarde et l'affichage des produits disponible sur le site Dendo Jitensha.

	Table	Action	Lignes	Type	Interclassement	Taille	Perte
☐	category	Parcourir Structure Rechercher Insérer Vider Supprimer	3	InnoDB	utf8_general_ci	32,0 kio	-
☐	orders	Parcourir Structure Rechercher Insérer Vider Supprimer	3	InnoDB	utf8_general_ci	48,0 kio	-
☐	order_line	Parcourir Structure Rechercher Insérer Vider Supprimer	8	InnoDB	utf8_general_ci	64,0 kio	-
☐	payment	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8_general_ci	48,0 kio	-
☐	pictures	Parcourir Structure Rechercher Insérer Vider Supprimer	11	InnoDB	utf8_general_ci	48,0 kio	-
☐	product	Parcourir Structure Rechercher Insérer Vider Supprimer	11	InnoDB	utf8_general_ci	48,0 kio	-
☐	role	Parcourir Structure Rechercher Insérer Vider Supprimer	2	InnoDB	utf8_general_ci	32,0 kio	-
☐	status	Parcourir Structure Rechercher Insérer Vider Supprimer	2	InnoDB	utf8_general_ci	32,0 kio	-
☐	user	Parcourir Structure Rechercher Insérer Vider Supprimer	3	InnoDB	utf8_general_ci	48,0 kio	-
☐	user_address	Parcourir Structure Rechercher Insérer Vider Supprimer	3	InnoDB	utf8_general_ci	48,0 kio	-
	10 tables	Somme	46	InnoDB	utf8_general_ci	448,0 kio	0 o

Extrait de la table Product :

☐	Éditer Copier Supprimer	1	Vélo ville 1	Par défaut pour test	79.99	99.99	5	1	2	2021-03-09 11:05:12	2021-03-05 14:44:11
☐	Éditer Copier Supprimer	2	Vélo ville 2	Par défaut pour test	91.99	114.99	7	1	2	2021-03-09 17:33:17	2021-03-05 14:44:12
☐	Éditer Copier Supprimer	3	Vélo ville 3	par défaut pour test	59.99	69.99	1	1	2	2021-03-09 17:23:57	2021-03-05 14:44:12
☐	Éditer Copier Supprimer	4	VTT 1	Par défaut pour test	39.99	49.99	1	2	2	2021-03-09 17:32:24	2021-03-05 14:44:12
☐	Éditer Copier Supprimer	5	VTT 2	Par défaut pour test	63.99	79.99	5	2	2	2021-03-09 10:34:09	2021-03-05 14:44:12
☐	Éditer Copier Supprimer	6	VTT 3	Par défaut pour test	43.99	54.99	4	2	2	2021-03-05 14:44:12	2021-03-05 14:44:12
☐	Éditer Copier Supprimer	7	Vélo course 1	Par défaut pour test	119.99	149.99	0	3	2	2021-03-05 14:52:03	2021-03-05 14:44:12
☐	Éditer Copier Supprimer	8	Vélo course 2	Par défaut pour test	159.99	199.99	2	3	2	2021-03-09 17:24:25	2021-03-05 14:44:12
☐	Éditer Copier Supprimer	9	Vélo course 3	Par défaut pour test	179.99	224.99	2	3	2	2021-03-09 10:34:09	2021-03-05 14:44:12

2) Partie PHP/Backend

Lors de la réalisation de la partie principale de site web, nous avons décidé de suivre une architecture MVC pour notre code PHP, cette architecture a pour avantage de simplifier un peu le codage mais surtout de bien organiser notre code, nous avons mis en place un système performant permettant d'ajouter facilement de nouveau attribut si besoin en BDD ou d'autre tables sans altérer le fonctionnement de notre code et en rendant l'intégration de ces nouveaux attributs et nouvelles tables très facile.

2.1) Création des model

Les modèles suivent tous un modèle par défaut en héritant de celui-ci, il contient notamment une fonction permettant d'attribuer automatiquement les attributs d'une table au model correspondant via ses getters et setters appelé automatiquement, cette fonction est appelée fonction d'hydratation.

Le constructeur des model hérite donc aussi de celui du model principal et appel donc automatiquement la fonction quand on leurs passent un tableau avec les attributs, il dispose également d'un attribut id toujours présent dans chaque autre model.

```php
public function __construct(array $donnees){
    $this->hydrate($donnees);
}

private function hydrate(array $donnees)
{
    foreach ($donnees as $key => $values) {
        $methode = 'set_'.$key;
        if (method_exists($this, $methode)) {
            $this->$methode($values);
        }
    }
}
```

Le model qui contiens les informations des produits hérite donc du model principal et dispose de ses propres attributs avec getters et setters associé à chacun des attributs.

```php
class ProductModels extends Models
{
    private $_price_vat;
    private $_price_ttc;
    private $_name;
    private $_description;
    private $_stock;
    private $_category_id;
    private $_status_id;
```

2.2) Création de contrôleur pour lier à la base de données

Tout nos contrôleurs sont également basés sur un contrôleur en class abstraite qui contient quelques attributs de base mais aussi les fonctions de base permettant d'effectuer les actions principales sur la BDD, à savoir un INSERT, un UPDATE, un DELETE et un SELECT. Ce contrôleur dispose aussi d'attribut commun à chaque autre contrôleur :

- Db : Contiens un objet PDO permettant la connexion à la BDD
- Table_name : Nom de la table a laquelle est lié le manager
- Class : Le model associé à la table
- ManagerClass : Le nom du manager actuel
- Column : Le nom de chaque attribut présent dans la table

```php
abstract class Manager
{
    protected $_db;
    protected $table_name;
    protected $class;
    protected $managerClass;
    protected $column;

    const PRIMARY_KEY = 'id';
```

```php
class ManagerProduct extends Manager
{
    protected $table_name = 'product';
    protected $class = ProductModels::class;
    protected $managerClass = ManagerProduct::class;
    protected $column =[
        'id',
       'price_vat',
       'name',
       'price_ttc',
       'description',
       'stock',
       'category_id',
        'status_id'
    ];
```

Les fonctions SELECT, INSERT, UPDATE et DELETE sont construite tous de la même façon, elles utilisent les attributs définit dans le manager pour créer automatiquement la requête correspondante à exécuter dans la base de données en remplissant également tout les paramètres de la requête automatiquement.

```php
public function insert($objet){
        $q = $this->_db->prepare('INSERT INTO
'.$this->table_name.'('.implode(',
',array_slice($this->column, 1)).') VALUES ('.implode(',
', array_map(function ($values){return ':'.$values;},
array_slice($this->column, 1))).')');
        foreach (array_slice($this->column, 1) as $row){
            $methode = 'get_'.$row;
            $q->bindValue(':'.$row, $objet->$methode());
        }
        $q->execute();
    }
```

2.3) Récupérer les articles en BDD

Pour ensuit récupérer les articles nous définissons une fonction spécifique permettant de récupérer les produits spécifiques à une catégorie avec une limite ou non, nous appelons cette fonction « getProductByCategorie() », elle prendra en paramètres l'id d'une catégorie aussi que la quantité de produits que l'on veut récupérer dans cette catégorie, la quantité sera définit comme paramètre facultatif.

```php
public function getProductByCategorie($categorieId, $amount = 0)
    {
        $tableau = [];

        if ($amount == 0){
            $q = $this->_db->prepare('SELECT * FROM
'.$this->table_name. ' WHERE category_id = '.$categorieId.' AND
status_id = 2');
        }else{
            $q = $this->_db->prepare('SELECT * FROM
'.$this->table_name. ' WHERE category_id = '.$categorieId.' AND
status_id = 2 LIMIT '.$amount);
        }
        $q->execute();
        $result = $q->fetchAll(PDO::FETCH_ASSOC);
        $q->closeCursor();

        foreach ($result as $value){
            $tableau[] = new $this->class($value);
        }

        return $tableau;
    }
```

Concrètement nous effectuons juste une requête SELECT sur la BDD puis nous renvoyons le résultat sous forme d'un tableau associatif exploitable pour la partie affichage sur les différentes pages du site.

3) Partie HTML/CSS (Affichage)

Pour l'affichage nous avons décidé d'utiliser le Framework Bootstrap, celui-ci nous facilitant grandement le positionnement du contenu de notre site tout en nous économisant du temps de développement en plus de fournir un affichage mobile et responsif très satisfaisant s'adaptant parfaitement a tout type d'écran. On utilise également un système de colonnes ainsi que des « cartes » pour afficher les produits récupérés en BDD.

```
<div class="container">
            <div class="text-center">
                <a class="text-decoration-none" href="rayon.php?categoryId=<?= $catname->get_id(); ?>">
                    <h2><?= $catname->get_name(); ?></h2>
                </a>
            </div>
            <div class="row">
                <?php
                $managerProduct = new ManagerProduct();
                $managerImage = new ManagerPicture();
                $products = $managerProduct->getProductByCategorie($catname->get_id(), 3);
                foreach ($products as $product):
                    ?>
                    <div class="col-lg-4 col-md-6">
                        <div class="card mt-5 border border-warning mx-auto" style="width: 18rem;">
                            <img class="card-img-top" src="<?php $image =
$managerImage->getProductPicture($product->get_id()); echo $image[0]->get_file_name();?>" alt="Card image cap">
                            <div class="card-body">
                                <h5 class="card-title"><?= $product->get_name(); ?></h5>
                                <p class="card-text"><?= $product->get_price_vat(); ?>€</p>
                                <a href="produit.php?productId=<?= $product->get_id(); ?>"
                                   class="btn btn-primary-o">Acheter</a>
                            </div>
                        </div>
                    </div>
                <?php endforeach; ?>
            </div>
        </div>
```

Le tout va être contenu dans un conteneur suivi d'un nom de catégorie récupéré au préalable, ensuit nous allons faire une colonne « Row » où sera contenu toutes les cartes représentant les produits.

Pour récupérer les produits nous créons un manager pour récupérer les produits ainsi que leurs images associées puis nous récupérons une liste de 3 produits dans une catégories définit, nous utilisons ensuit le système de colonnes de Bootstrap ainsi que le système de carte modifié pour l'adapté a la charte graphique de notre site.

Les informations que nous affichons en plus de l'image sont le nom du produit et son prix. Pour récupérer l'image nous passons par le manager de la table Pictures en récupérant le ou l'image associée à l'id de notre produit, dans le cas de plusieurs images nous n'en affichons qu'une seul en récupérant seulement le chemin de celle-ci.

4) Rendu final

Le rendu est à peu près le même sur tout le site, il ce présente de la façon suivante :