FACULTÉ DES SCIENCES
Unité de formation et de recherche
université angers

Département Informatique
L3 Informatique - Développement Web
David Lesaint

TP PHP : MVC, Moteur de templates (Twig)

Téléchargez et décompressez dans votre répertoire de travail l'archive donnée en annexe sur l'espace Moodle. Le répertoire obtenu `blog_mvc_objet` est une implémentation de type MVC du site "blog" présenté en cours.

Exercice 1 - Créez une base de données sous phpmyadmin. Y importez le script `BD/MonBlog.sql` qui construira les tables `T_BILLET` et `T_COMMENTAIRE`. Notez la définition d'une clé etrangère dans la table `T_COMMENTAIRE`. Créez via phpmyadmin des enregistrements supplémentaires pour la table `T_BILLET`.

Exercice 2 - Modifiez le corps de la méthode `Modele::getBdd()` avec vos paramètres (DSN, compte utilisateur et mot de passe MySQL).

Exercice 3 - Chargez le site dans votre navigateur. Testez les différents hyperliens et créez des commentaires. Retracez dans le code source l'exécution des différentes actions entreprises.

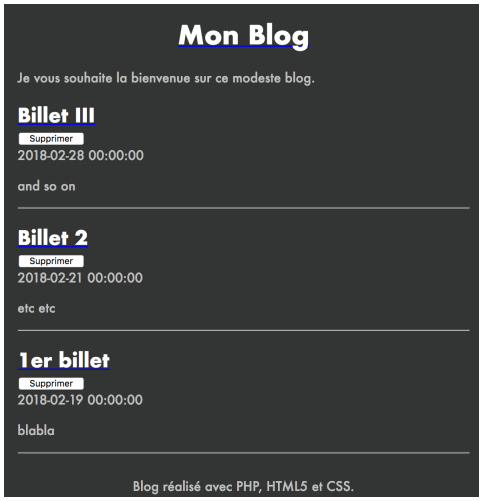

FIGURE 1 – Page d'accueil avec boutons de suppression de billets

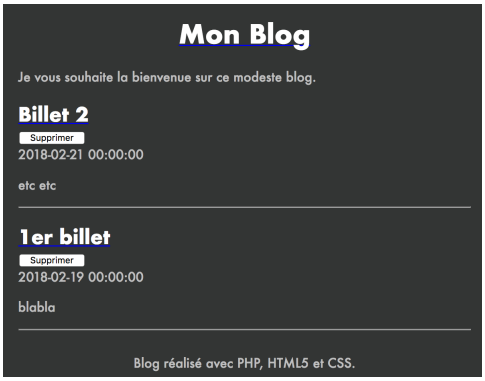

FIGURE 2 – Suppression de billet réussie

Exercice 4 - Enrichir le site pour permettre la suppression de billets :

- La page d'accueil doit afficher un bouton de suppression sous chaque article (Figure 1).
- Cliquer sur un bouton doit supprimer le billet correspondant dans la base de données ainsi que les commentaires associés puis réafficher la page d'accueil actualisée (Figure 2).

Rajouter une action `supprimer` à l'application :

- La vue à modifier doit pouvoir communiquer l'action de suppression et l'identifiant du billet à supprimer (utilisez un champ caché pour ce dernier).
- Le routeur doit intégrer cette nouvelle action.
- Le contrôleur concerné doit alors supprimer les commentaires avant de supprimer le billet pour ne pas violer les contraintes de clés étrangères.
- Les méthodes de suppression doivent être implémentées dans les modèles concernés.

Exercice 5 - Téléchargez `composer` dans votre répertoire `blog_mvc_objet` (ou exécuter les commandes d'installation décrites sur la page dans un nouveau shell). Installez ensuite le moteur de templates `Twig` en exécutant la commande suivante dans `blog_mvc_objet` :

```
./composer.phar require twig/twig:^2.0
```

Un répertoire `vendor` contenant Twig est créé à la source de votre répertoire `blog_mvc_objet` ainsi que deux fichiers `composer.json` et `composer.lock`.

Exercice 6 - Créez une copie de `gabarit.php` nommée `gabarit.html`.
Adaptez ce fichier en y substituant les scripts PHP par des instructions Twig : référez-vous à la syntaxe de Twig décrite dans la documentation.
Autochargez Twig dans le fichier `Vue.php` :

```
require_once './vendor/autoload.php';
```

Modifiez ensuite la méthode `Vue::generer($donnees)` pour charger le gabarit avec Twig (voir documentation) :

```
public function generer($donnees) {
...
$loader = new Twig_Loader_Filesystem('Vue');
$twig = new Twig_Environment($loader);
$template = $twig->load(...);
echo $template->render(...);
}
```

Exercice 7 - Utilisez l'héritage de templates pour transformez les vues restantes au format Twig (`erreur.php`, `Vue/Accueil/index.php` et `Vue/Billet/index.php`) et adaptez le constructeur et la méthode `generer()` de la classe `Vue` en conséquence.