

## TP PHP 5 - MVC, Moteur de gabarits (Twig)

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

**Exercice 1.** Suivez les instructions du **README** pour installer le paquet **Twig** et créer la base de données dénommée `l3_dw_tp_php_mvc_blog`. Cette base contient les tables `T_BILLET` et `T_COMMENTAIRE`. Notez la définition d'une clé étrangère 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 MariaDB (ou MySQL selon votre environnement)).

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



FIGURE 2 – Suppression de billet réussie

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

**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.** L'objectif est ici d'utiliser le moteur de gabarits **Twig** et de transformer vos gabarits HTML/-PHP au format HTML/Twig. Pour accéder à l'API de Twig, il suffit d'importer l'autochargeur dans le fichier `Vue.php` :

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

Modifiez le constructeur de `Vue` en changeant l'extension `.php` des fichiers vues à générer par l'extension `.html`.

Modifiez la méthode `Vue::generer($donnees)` pour charger et évaluer le gabarit avec Twig (voir [documentation](#)) :

```
public function generer($donnees)
{
    $loader = new \Twig\Loader\FilesystemLoader('Vue');
    $twig = new \Twig\Environment($loader); /* no template caching : , array('cache' => '.'); */
    $template = $twig->load($this->fichier);
    echo $template->render($donnees);
}
```

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](#).

**Exercice 6.** Créez des copies avec l'extension `html` des autres gabarits (`vueErreur.php`, `vueAccueil.php` et `vueBillet.php`). Utilisez l'héritage de gabarits pour transformer ces fichiers au format Twig.