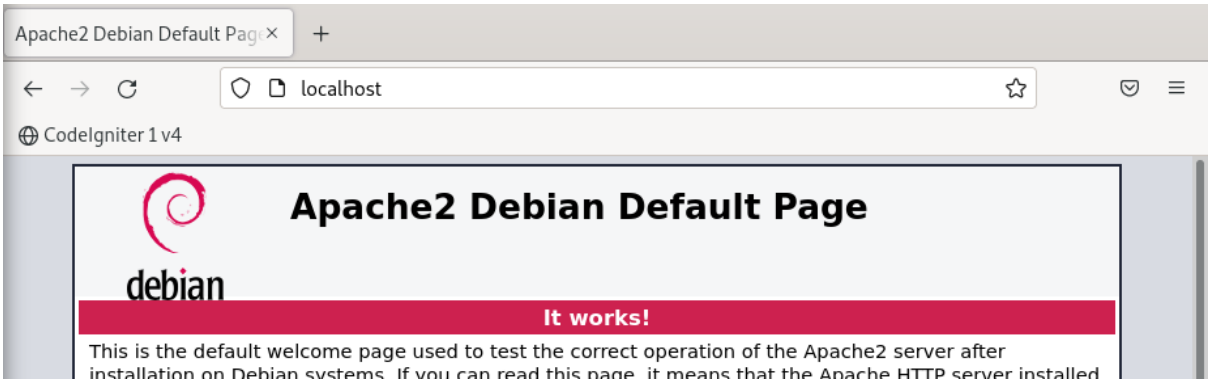
Prérequis :
Installation de apache 2:



Vérifier la version de php:

```
test@debian:~$ php -version
PHP 7.3.31-1~deb10u1 (cli) (built: Oct 24 2021 15:18:08) ( NTS )
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.3.31, Copyright (c) 1998-2018 Zend Technologies
    with Zend OPcache v7.3.31-1~deb10u1, Copyright (c) 1999-2018, by Zend Techno
logies
```

Mettre à jour la version de java :
Faire une sudo apt update
puis faire un sudo apt upgrade pour mettre à jour

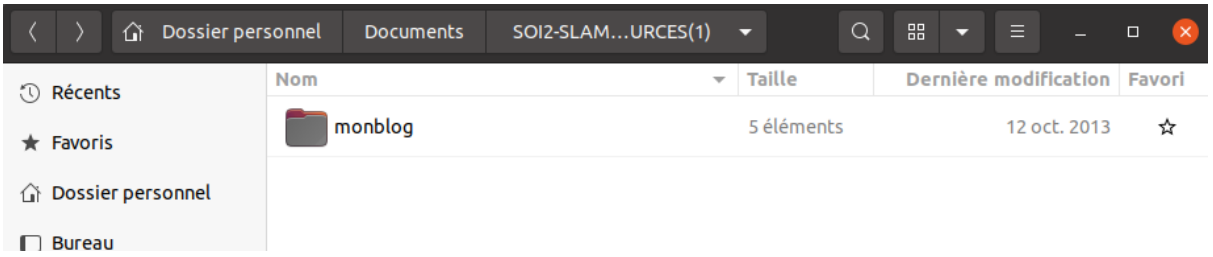
```
test@debian:~$ java -version
openjdk version "11.0.16" 2022-07-19
OpenJDK Runtime Environment (build 11.0.16+8-post-Debian-1deb10u1)
OpenJDK 64-Bit Server VM (build 11.0.16+8-post-Debian-1deb10u1, mixed mode, shar
ing)
```

Pour l'étape 0 je n'ai pas besoin de la faire car j'ai déjà netbeans sur ma VM, j'utilise une codeigniter.

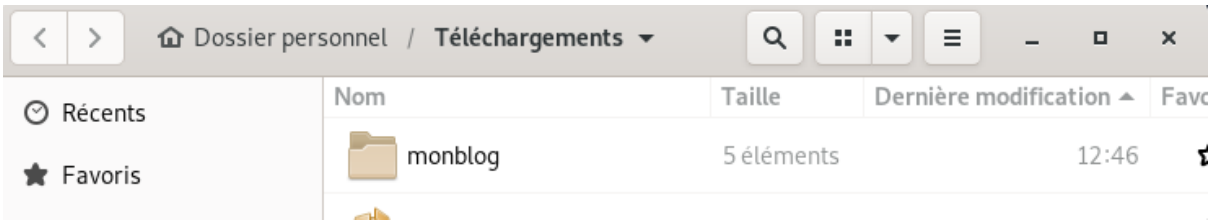
INSTALLATION :

1) **Ouvrir le projet « monblog » dans NetBeans**

Je dois récupérer le fichier MonBlog-sans-mrv.tar.gz, le décompresser pour le copier dans ma vm MVC.
Préalablement j'ai donc fait la commande $ sudo usermod -a -G vboxsf test pour pouvoir déplacer le fichier dans ma VM.
Voici donc mon dossier renommé :



Je transfère donc mon dossier dans ma VM dans les téléchargement :



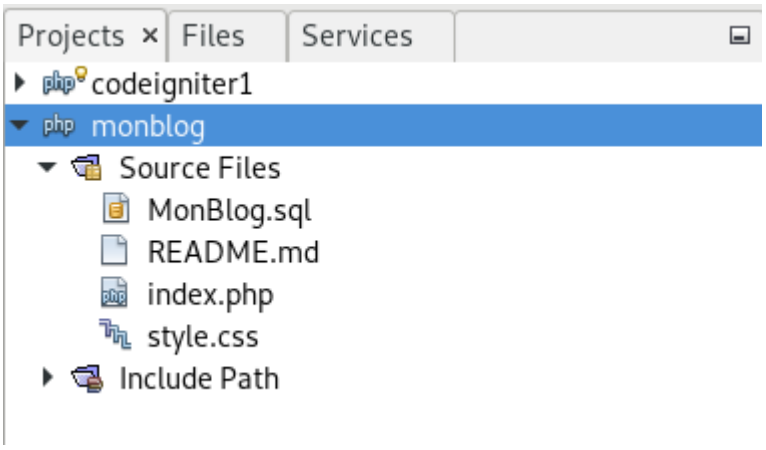
Voici le dossier monblog copier dans /var/www/html :

```
test@debian:~$ sudo cp -r ~/Téléchargements/monblog /var/www/html
test@debian:~$ cd /var/www/html/
test@debian:/var/www/html$ ls
codeigniter1  codeigniter1BACK  index.html  monblog  phpinfo.php
```

Puis je dois affecter les droits :

```
test@debian:/var/www/html$ sudo chmod -R 777 monblog/
test@debian:/var/www/html$ ls
codeigniter1  codeigniter1BACK  index.html  monblog  phpinfo.php
test@debian:/var/www/html$
```

Puis ouvrir ce dossier dans netbeans :



Je n'ai bien évidement pas le fichier .htaccess donc je dois le créer dans le dossier /var/www/html/monblog :

```
test@debian:/var/www/html/monblog$ sudo nano .htaccess
```

Ce fichier permet de ne pas avoir à écrire /index.php lorsque l'on fait une requête URL web.

2) Créer la base "monblog" dans mysql

Dans la description de ma VM je peux voir que les identifiants de connexions à mysql sont :

MySQL 8.0 installé
psw : admin / adminsql1234ADC+

Je me connecte donc avec ses identifiant au serveur mysql :

```
test@debian:~$ sudo mysql -u admin -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.31 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
mysql>
```

Une fois connecté au serveur je vérifie que ma databases monblog existe:
Celle-ci est bien présente :

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| codeigniter        |
| information_schema |
| monblog            |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
6 rows in set (0,11 sec)
```

Puis je me connect et vérifie que les tables sont présente et les bonnes :

```
mysql> connect monblog;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Connection id:    12
Current database: monblog

mysql> show tables;
+------------------+
| Tables_in_monblog |
+------------------+
| T_BILLET         |
| T_COMMENTAIRE    |
+------------------+
2 rows in set (0,01 sec)
```

Puis je vérifie que les valeurs dans les tables soient correctes :
Celle-ci sont bien correctes :

```
mysql> select* from T_BILLET
    -> ;
+--------+---------------------+---------------+-----------------------------------------------------------+
| BIL_ID | BIL_DATE            | BIL_TITRE     | BIL_CONTENU                                               |
+--------+---------------------+---------------+-----------------------------------------------------------+
|      1 | 2020-06-14 19:07:21 | Premier billet | Bonjour monde ! Ceci est le premier billet sur mon blog. |
|      2 | 2020-06-14 19:07:21 | Au travail     | Il faut enrichir ce blog dès maintenant.                  |
+--------+---------------------+---------------+-----------------------------------------------------------+
2 rows in set (0,01 sec)

mysql> select* from T_COMMENTAIRE;
+--------+---------------------+------------+----------------------------------------+--------+
| COM_ID | COM_DATE            | COM_AUTEUR | COM_CONTENU                            | BIL_ID |
+--------+---------------------+------------+----------------------------------------+--------+
|      1 | 2020-06-14 19:07:22 | A. Nonyme  | Bravo pour ce début                    |      1 |
|      2 | 2020-06-14 19:07:22 | Moi        | Merci ! Je vais continuer sur ma lancée |      1 |
|      3 | 2020-06-15 19:39:47 | Valy       | Super mon enrichissement.              |      2 |
+--------+---------------------+------------+----------------------------------------+--------+
3 rows in set (0,01 sec)
```

Après cela je dois créer le User monblog :
Pour cela je vérifie qu'il est pas créer mais celui-ci est déjà :

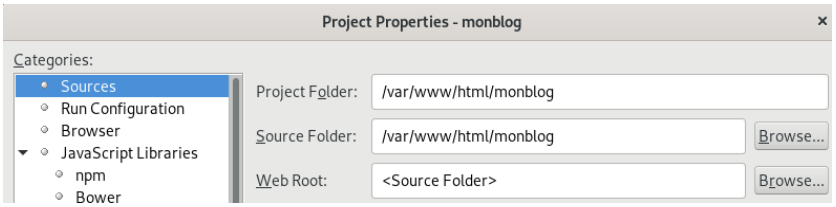
```
mysql> select user,host from mysql.user;
+------------------+-----------+
| user             | host      |
+------------------+-----------+
| admin            | localhost |
| monblog          | localhost |
| mysql.infoschema | localhost |
| mysql.session    | localhost |
| mysql.sys        | localhost |
| root             | localhost |
+------------------+-----------+
6 rows in set (0,00 sec)
```

Je dois donc seulement vérifier qu'il est bien tous les privilèges :

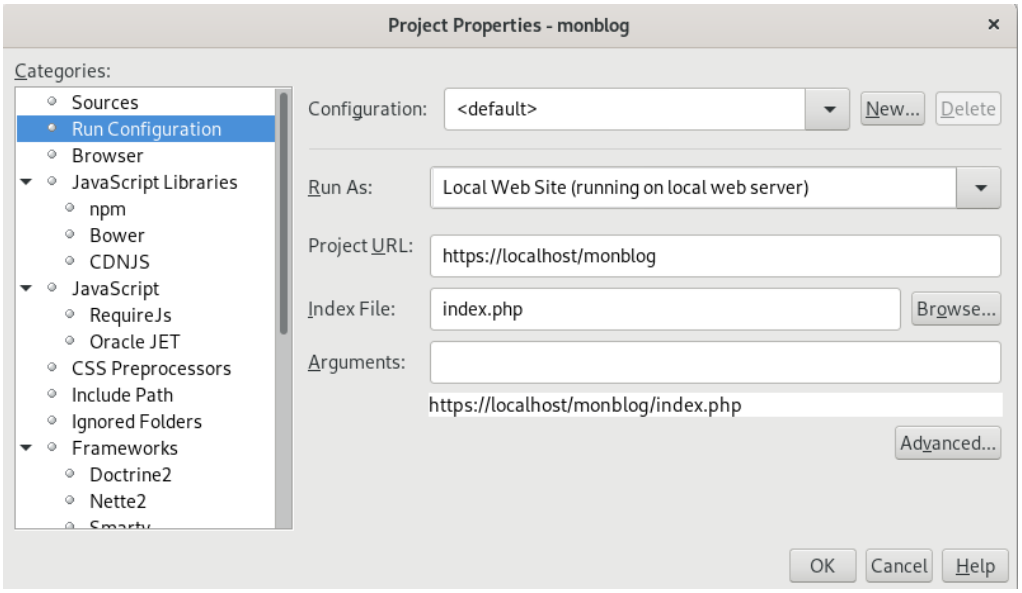
```
mysql> show grants for 'monblog'@'localhost';
+-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------
| Grants for monblog@localhost



+-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------
| GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, SHUTDOWN, PROCESS, FILE, REFEREN
E, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTIN
onblog`@`localhost`

| GRANT APPLICATION_PASSWORD_ADMIN,AUDIT_ABORT_EXEMPT,AUDIT_ADMIN,AUTHENTICATION_POLICY_ADMIN,
ION_KEY_ADMIN,FIREWALL_EXEMPT,FLUSH_OPTIMIZER_COSTS,FLUSH_STATUS,FLUSH_TABLES,FLUSH_USER_RESOU
SS_USER_ADMIN,PERSIST_RO_VARIABLES_ADMIN,REPLICATION_APPLIER,REPLICATION_SLAVE_ADMIN,RESOURCE_
TION_ADMIN,SESSION_VARIABLES_ADMIN,SET_USER_ID,SHOW_ROUTINE,SYSTEM_USER,SYSTEM_VARIABLES_ADMIN
+-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------
2 rows in set (0,00 sec)
```

3) tester le projet initial

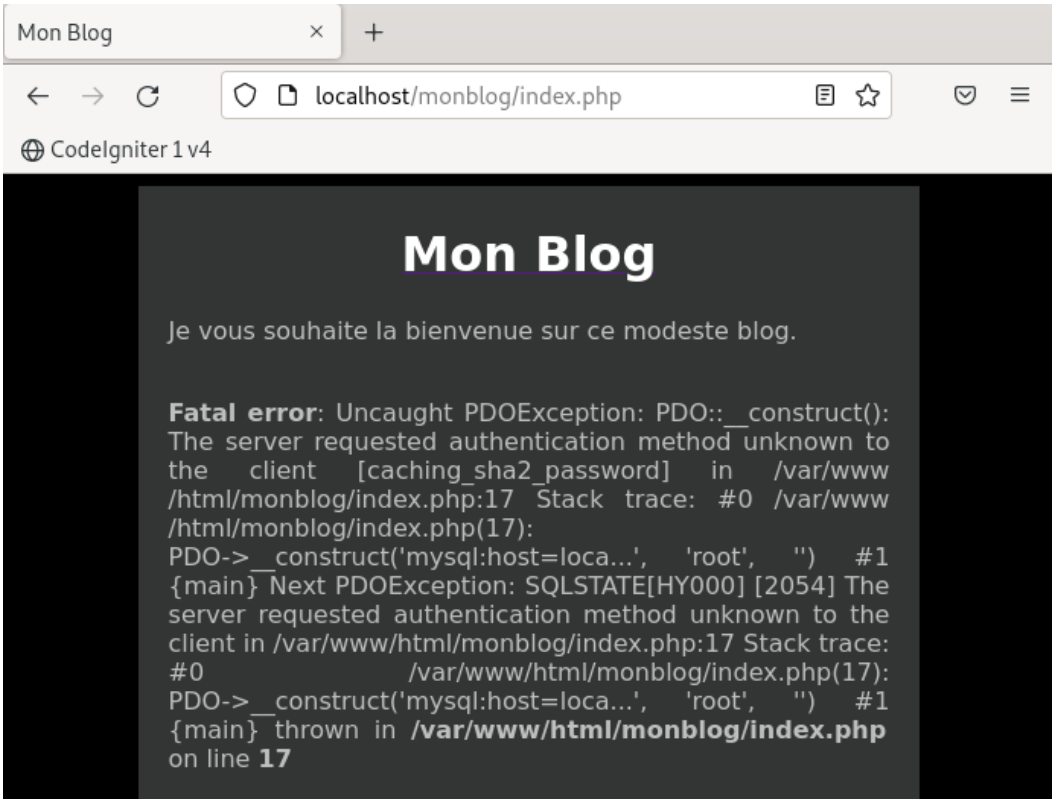
Sources :



Run Configuration :



Lorsque que je lance j'obtiens donc une erreur :



Je dois donc afficher les messages d'erreur avec le try catch :

```php
<?php
try {
$bdd = new PDO('mysql:host=localhost;dbname=monblog;charset=utf8',
        'root', '');
$billets = $bdd->query('select BIL_ID as id, BIL_DATE as date,'
        . ' BIL_TITRE as titre, BIL_CONTENU as contenu from T_BILLET'
        . ' order by BIL_ID desc');
}
catch (Exception $e) {
    echo '<html><body>Erreur !' . $e->getMessage() . '</body></html>';
}
foreach ($billets as $billet):
    ?>
```

Quand on relance on obtient donc cette erreur :

Nous allons résoudre cette erreur dans la prochaine étape.

ÉTAPE 1 : PRISE EN MAIN :

Demandez l'affichage des erreurs php

Entrer dans le fichier php.ini :

```
test@debian:~$ sudo nano /etc/php/7.3/apache2/php.ini
[sudo] Mot de passe de test :
```

Puis mettre en On le paramètre de display_errors :

```
display_errors = On
;   Default Value: On
;   Development Value: On
;   Production Value: On

error_reporting = E_ALL | E_STRICT
;   Default Value: E_ALL & ~E_NOTICE & ~E_STRICT & ~E_DEPRECATED
;   Development Value: E_ALL
;   Production Value: E_ALL & ~E_DEPRECATED & ~E_STRICT
```

Puis restart le serveur apache 2 :

```
test@debian:~$ sudo service apache2 restart
```

Affichage de l'application actuelle

Dans cette partie nous devons simplement modifier la ligne PDO qui permet de se connecter à la base de données afin de mettre les bon identifiants de connexions :
USER : monblog
PASSWORD : Monblog1234+

```
$bdd = new PDO('mysql:host=localhost;dbname=monblog;charset=utf8','monblog'
    ,'Monblog1234+');
```

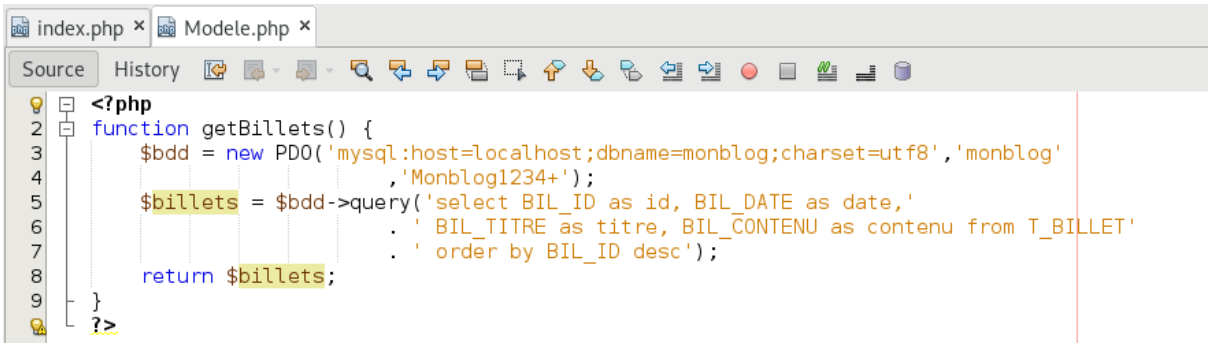
Désormais la page index.php affiche le bon résultat :

Je vais maintenant apprendre à structurer correctement le code et différencier les différentes parties entre présentation traitement et données. C'est donc le modèle MVC.

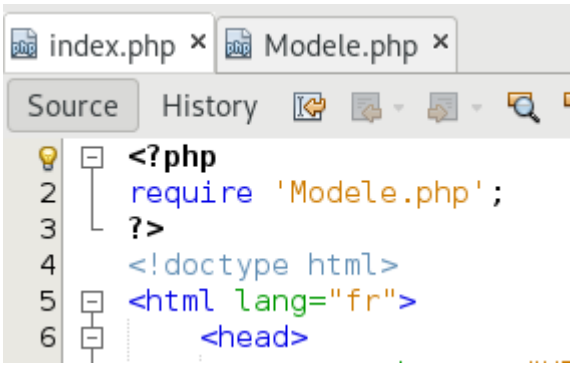
ÉTAPE 2 : MISE EN PLACE DU MODÈLE MVC

1) Isolation du code d'accès aux données

Dans un premier temps je créer une nouvelle page Modele.php dans laquelle j'ajoute une fonction getBillets() qui reprend simplement le PDO du fichier index.php :



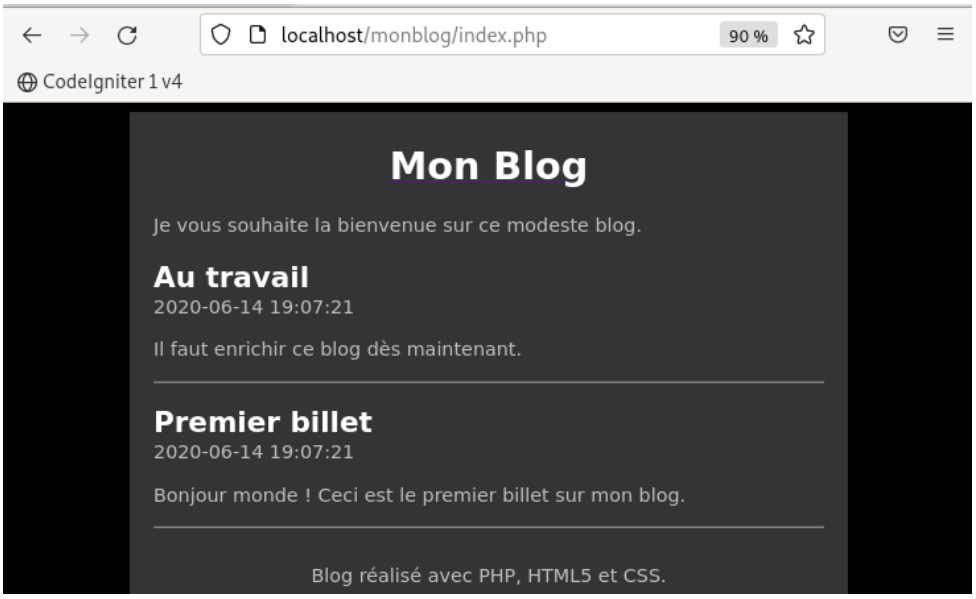
Ensuite je dois donc appeler le fichier Modele.php dans mon fichier index.php grâce à ma fonction créée avec require :



Puis supprimer totalement le try de index.php pour le remplacer par la fonction getBillet() :

```php
<?php
try {
   //Accés aux données
    $billets = getBillets();
} catch (Exception $ex) {

}
catch (Exception $e) {
    echo '<html><body>Erreur !' . $e->getMessage() . '</body></html>';
}
```

Je test en rappelant ma page index.php :



2) Isolation du code d'affichage

Dans cette isolation nous allons seulement garder le code d'affichage donc tout le code html avec les différentes balises, en gardant le foreach :

index.php | Modele.php | vueAccueil.php

Source | History

```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="UTF-8" />
        <link rel="stylesheet" href="style.css" />
        <title>Mon Blog</title>
    </head>
    <body>
        <div id="global">
            <header>
                <a href="index.php"><h1 id="titreBlog">Mon Blog</h1></a>
                <p>Je vous souhaite la bienvenue sur ce modeste blog.</p>
            </header>
            <div id="contenu">
                <?php
                foreach ($billets as $billet):
                ?>
                <article>
                        <header>
                            <h1 class="titreBillet"><?= $billet['titre'] ?></h1>
                            <time><?= $billet['date'] ?></time>
                        </header>
                        <p><?= $billet['contenu'] ?></p>
                    </article>
                    <hr/>
                <?php endforeach; ?>
            </div> <!-- #contenu -->
            <footer id="piedBlog">
                Blog réalisé avec PHP, HTML5 et CSS.
            </footer>
        </div> <!-- #global -->
    </body>
</html>
```

3) Isolation des traitement

Dans cette partie nous allons garder les reste de index.php en ajoutant les require des page Modele.php et vueAccueil.php ;

index.php | Modele.php | vueAccueil.php

Source | History

```php
<?php
require 'Modele.php';
try {
    //Accés aux données
    $billets = getBillets();
} catch (Exception $ex) {
    echo '<html><body>Erreur !' . $e->getMessage() . '</body></html>';
}
require 'vueAccueil.php';
?>
```

Résultats : MVC

Modele.php permet de se connecter à notre base de données en PDO, **vuAccueil.php** permet d'afficher la page html et notre **index.php** permet de faire la liaison entre les deux pages précédentes.

ÉTAPE 3 : MISE EN PLACE DU MODÈLE MVC

Mise en place d'un gabarit

Nous créons la Gabarit.php qui contiendra tous les élément du $contenu ainsi que du $titre, on peut dire que cette page va permettre de créer des variables pour alléger notre page vuAccueil.php en récupérant tous le code html :

index.php | Modele.php | vueAccueil.php | Gabarit.php

Source | History

```php
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="UTF-8" />
        <link rel="stylesheet" href="style.css" />
        <title><?= $titre ?></title> <!-- Élément spécifique -->
    </head>
    <body>
        <div id="global">
            <header>
                <a href="index.php"><h1 id="titreBlog">Mon Blog</h1></a>
                <p>Je vous souhaite la bienvenue sur ce modeste blog.</p>
            </header>
            <div id="contenu">
                <?= $contenu ?> <!-- Élément spécifique -->
            </div>
            <footer id="piedBlog">
                Blog réalisé avec PHP, HTML5 et CSS.
            </footer>
        </div> <!-- #global -->
    </body>
</html>
```

Puis nous allons créer la nouvelle page vuAcceuil.php :
Nous pouvons donc retrouver notre $titre et notre $contenu sans tout le code html.

index.php | Modele.php | vueAccueil.php | Gabarit.php

Source | History

```php
<?php $titre = 'Mon Blog';
ob_start();
foreach ($billets as $billet): ?>
    <article>
        <header>
            <h1 class="titreBillet"><?= $billet['titre'] ?></h1>
            <time><?= $billet['date'] ?></time>
        </header>
        <p><?= $billet['contenu'] ?></p>
    </article>
    <hr />
<?php endforeach;
$contenu = ob_get_clean();
require 'Gabarit.php'; ?>
```

Factorisation de la connexion à la base

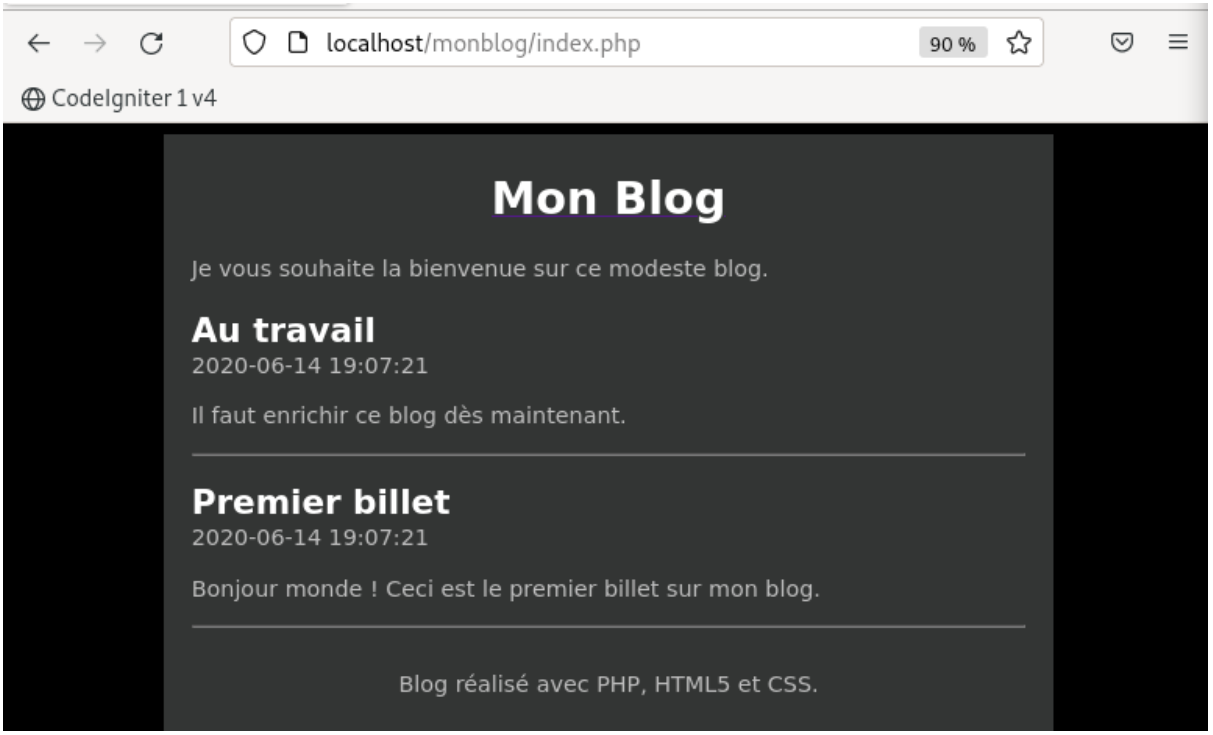
Dans cette partie je vais créer une nouvelle fonction dans Modele.php qui permet d'éviter la réécriture du code de la connexion à la bdd. On pourra aussi utiliser la variable $bdd qui se trouve dans la fonction getbdd() sans utiliser la variable $billets qui est dans la fonction getbillets().
Voici le nouveaux code source de Modele.php:

index.php | Modele.php | vueAccueil.php | Gabarit.php

Source | History

```php
<?php
function getBillets() {
    $bdd = getBdd();
    $billets = $bdd->query('select BIL_ID as id, BIL_DATE as date,'
                    . ' BIL_TITRE as titre, BIL_CONTENU as contenu from T_BILLET'
                    . ' order by BIL_ID desc');
    return $billets;
}
//Effectuer la connexion à la BDD
//Instancie et renvoie le PDO de la fonction
function getBdd(){
    $bdd = new PDO('mysql:host=localhost;dbname=monblog;charset=utf8','monblog'
                    ,'Monblog1234+');
    return $bdd;
}
?>
```

Voici le test quand je refresh la page du site :



Gestion des erreurs

Nous allons modifier notre gestion d'erreur afin de la rendre plus propre et de mieux restituer la chaîne de cause à effet.
Modification du try/catch de la page index.php, je déplace seulement le require 'vueAccueil.php'; dans le try :

index.php | Modele.php | vueAccueil.php | Gabarit.php

Source | History

```php
<?php
require 'Modele.php';
try {
    //Accés aux données
    $billets = getBillets();
    require 'vueAccueil.php';
} catch (Exception $ex) {
    echo '<html><body>Erreur !' . $e->getMessage() . '</body></html>';
}
?>
```

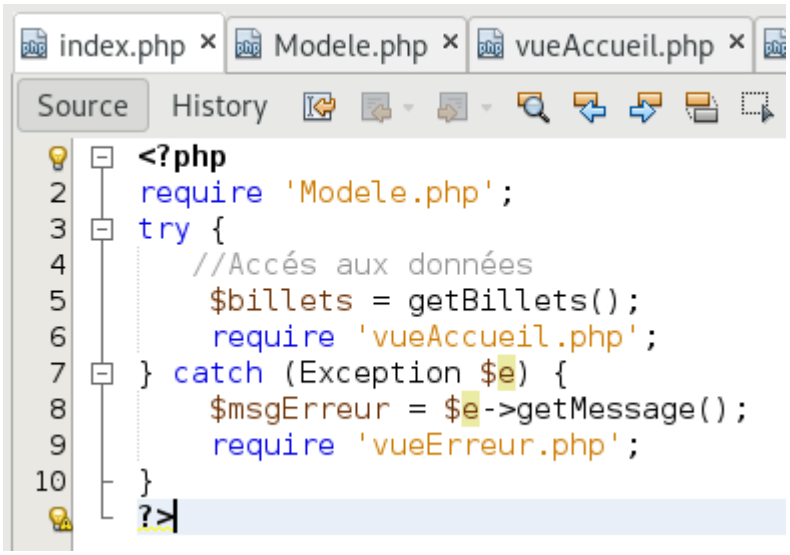
Création du fichier vueErreur.php :
Il permet de gérer les erreurs sans avoir à les gérer dans le catch du fichier index.php :

index.php | Modele.php | vueAccueil.php | Gabarit.php | vueErreur.php

Source | History

```php
<?php $titre = 'Mon Blog'; ?>
<?php ob_start() ?>
    <p>Une erreur est survenue : <?= $msgErreur ?></p>
<?php $contenu = ob_get_clean(); ?>
<?php require 'Gabarit.php'; ?>
```

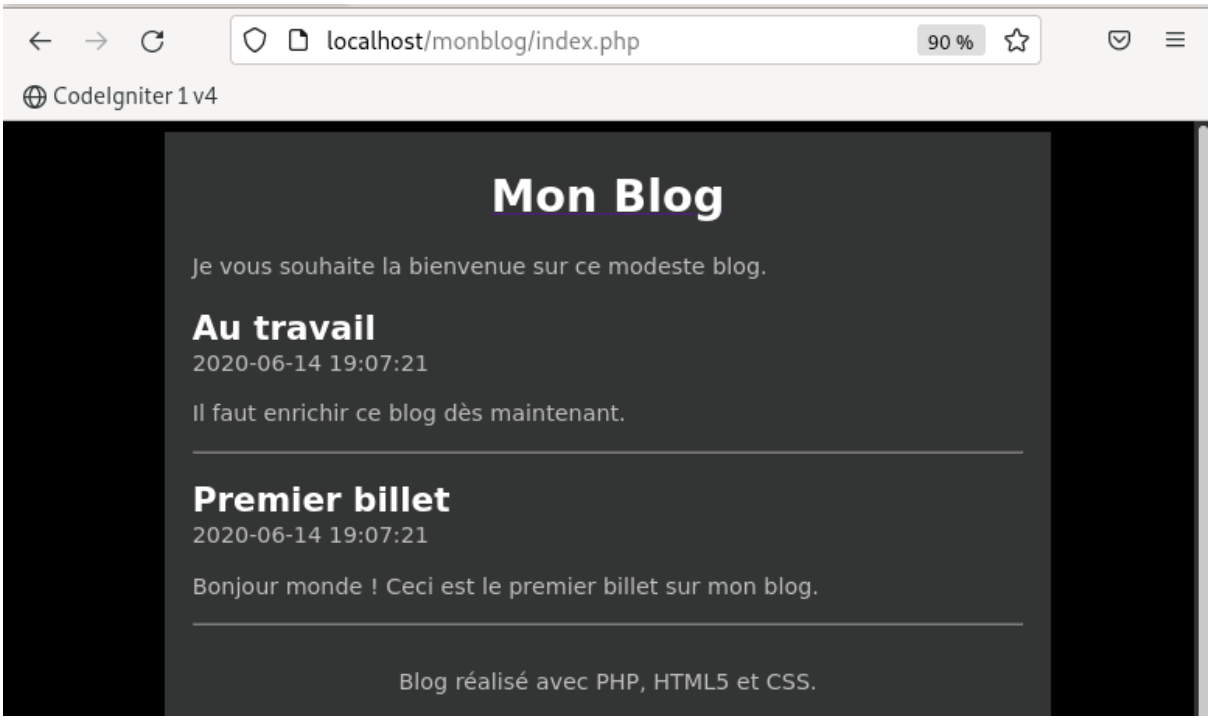
Il nous reste encore à modifier la partie catch du try/catch de index.php et il y aura une meilleur gestion d'erreur avec le fichier vueErreur.php :



Nous avons pu aussi supprimer le code html car c'est vueErreur.php en appelant le Gabarit.php qui le réalise tout cela est donc plus propre.
Je vais faire un test avec une erreur en changeant le mdp de connexion dans le PDO par 'xxx' :

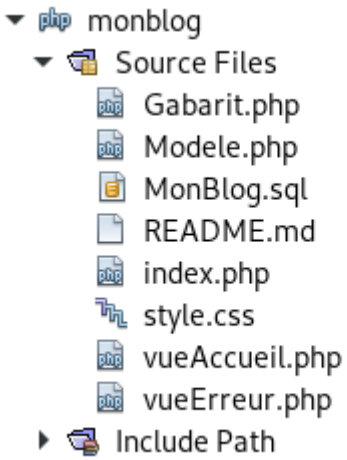
Voici l'erreur générée :

Puis je corrige mon erreur volontaire :



Bilan provisoire

Nous respectons désormais le modèle MVC :
Voici l'architecture :



Ajout d'une nouvelle fonctionnalité de restitution

Nous respecterons désormais tout le temps le modèle MVC.

Création d'une nouvelle fonctionnalité du modèle (M)

Dans cette création du nouveau modèle nous allons donc modifier le fichier Modele.php en modifiant la fonction getBillets et en ajoutant la fonction getCommentaires qui permet de récupérer un commentaires associées à un billet dans notre table de notre bdd T_COMMENTAIRE.

Voici Modele.php :

Modele.php × Billet.php × Gabarit.php × vueErreur.php ×

Source History

```php
<?php
//Sur tous les billets
function getBillets() {
    $bdd = getBdd();
    $billets = $bdd->query('select BIL_ID as id, BIL_DATE as date,'
            . ' BIL_TITRE as titre, BIL_CONTENU as contenu from T_BILLET'
            . ' order by BIL_ID desc');
    return $billets;
}
//Liste des informations sur un billet
function getBillet($idBillet) {
    $bdd = getBdd();
    $billet = $bdd->prepare('select BIL_ID as id, BIL_DATE as date,'
            . ' BIL_TITRE as titre, BIL_CONTENU as contenu from T_BILLET'
            . ' where BIL_ID=?');
    $billet->execute(array($idBillet));
    if ($billet->rowCount() == 1)
        return $billet->fetch(); // Accès à la première ligne de résultat
    else
        throw new Exception("Aucun billet ne correspond à l'identifiant '$idBillet'");
}

//Liste des commentaires à un billet
function getCommentaires($idBillet) {
    $bdd = getBdd();
    $commentaires = $bdd->prepare('select COM_ID as id, COM_DATE as date,'
            . ' COM_AUTEUR as auteur, COM_CONTENU as contenu from T_COMMENTAIRE'
            . ' where BIL_ID=?');
    $commentaires->execute(array($idBillet));
    return $commentaires;
}
//Effectuer la connexion à la BDD
//Instancie et renvoie le PDO de la fonction
function getBdd() {
    $bdd = new PDO('mysql:host=localhost;dbname=monblog;charset=utf8', 'monblog'
            , 'Monblog1234+');
    return $bdd;
}
?>
```

Création d'une nouvelle fonctionnalité de la vue (V)

Ensuite pour afficher les nouvelle information du billet ainsi que des commentaires je vais créer une nouvelle page vueBillet.php :

index.php | Modele.php | vueAccueil.php | Gabarit.php | vueErreur.php | vueBillet.php

Source | History

```php
<?php $titre = "Mon Blog - " . $billet['titre']; ?>
<?php ob_start(); ?>
<article>
    <header>
        <h1 class="titreBillet"><?= $billet['titre'] ?></h1>
        <time><?= $billet['date'] ?></time>
    </header>
    <p><?= $billet['contenu'] ?></p>
</article>
<hr />
<header>
    <h1 id="titreReponses">Réponses à <?= $billet['titre'] ?></h1>
</header>
<?php foreach ($commentaires as $commentaire): ?>
    <p><?= $commentaire['auteur'] ?> dit :</p>
    <p><?= $commentaire['contenu'] ?></p>
<?php endforeach; ?>
<?php $contenu = ob_get_clean(); ?>
<?php require 'Gabarit.php'; ?>
```

Création d'un nouveau contrôleur (C)

Maintenant nous allons afficher les informations de vueBillet.php en créant un nouveau modele Billet.php :

Billet.php | Gabarit.php | Modele.php | index.php | vueAccueil.php...

Source | History

```php
<?php
require 'Modele.php';
try {
    if (isset($_GET['id'])) {
// intval renvoie la valeur numérique du paramètre ou 0 en cas d'échec
        $id = intval($_GET['id']);
        if ($id != 0) {
            $billet = getBillet($id);
            $commentaires = getCommentaires($id);
            require 'vueBillet.php';
        } else
            throw new Exception("Identifiant de billet incorrect");
    } else
        throw new Exception("Aucun identifiant de billet");
} catch (Exception $e) {
    $msgErreur = $e->getMessage();
    require 'vueErreur.php';
}
?>
```

Modification du contexte

Pour cette dernière partie nous devons modifier la page principale du contexte vueAcceuil.php afin de permettre d'afficher les bons billets de Billet.php:
Je dois donc modifier le header :

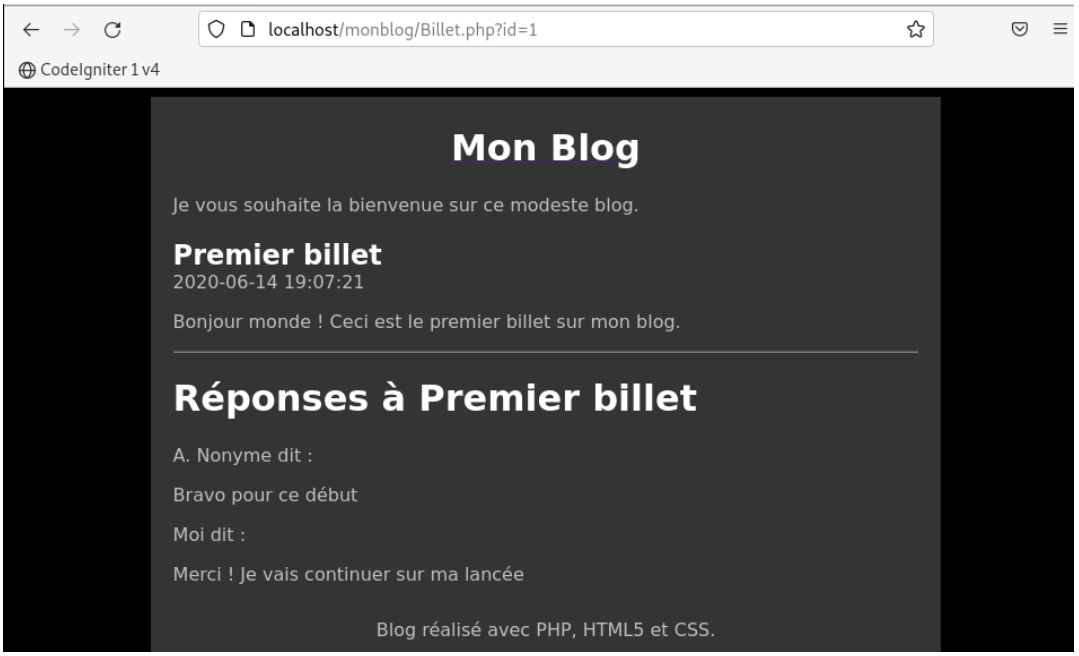
```
7   <header>
8       <a href="<?= "Billet.php?id=" . $billet['id'] ?>">
9           <h1 class="titreBillet"><?= $billet['titre'] ?></h1>
10      </a>
11      <time><?= $billet['date'] ?></time>
12  </header>
```

Puis on ajoute la classe #titreReponses a notre style.css :

```
45  #titresReponses {
46      font-size: 100%;
47  }
```

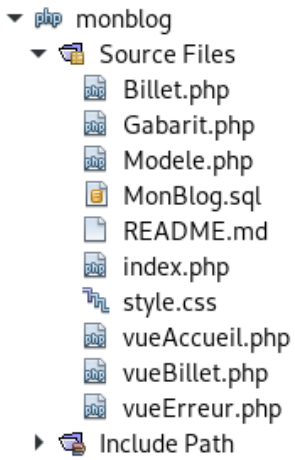
Test final :

J'obtiens bien la bonne page.



Bilan de l'étape

Voici mon archi chitecture finale :



ÉTAPE 4 : Rationalisation de l'architecture MVC :

Dans cette étape nous allons régler notre problème de visibilité de nom des fichiers appelé comme par exemple Billet.php qui peut révéler à un attaquant la structure de notre projet. Pour débuter nous allons utiliser le contrôleur frontal.

Le contrôleur frontal

Premièrement je vais recréer le fichier index.php pour passer les méthode en GET afin d'appeler les fonctions du fichier Controleur.php que je vais créer par la suite :

index.php

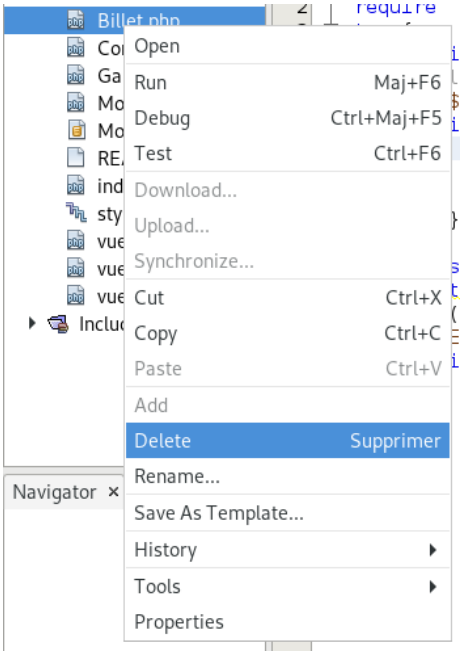
```php
<?php
require('Controleur.php');
try {
    if (isset($_GET['action'])) {
        if ($_GET['action'] == 'billet') {
            if (isset($_GET['id'])) {
                $idBillet = intval($_GET['id']);
                if ($idBillet != 0)
                    billet($idBillet);
                else
                    throw new Exception("Identifiant de billet non valide");
            } else
                throw new Exception("Identifiant de billet non défini");
        } else
            throw new Exception("Action non valide");
    } else {
        accueil(); // action par défaut
    }
} catch (Exception $e) {
    erreur($e->getMessage());
}
?>
```

Deuxièmement création du fichier Controleur.php qui contient les traitements de données de

index.php | Controleur.php

```php
<?php
require 'Modele.php';

// Affiche la liste de tous les billets du blog
function accueil() {
    $billets = getBillets();
    require 'vueAccueil.php';
}

// Affiche les détails sur un billet
function billet($idBillet) {
    $billet = getBillet($idBillet);
    $commentaires = getCommentaires($idBillet);
    require 'vueBillet.php';
}

// Affiche une erreur
function erreur($msgErreur) {
    require 'vueErreur.php';
}
```

Troisièmement je supprime le fichier Billet.php qui devient donc inutile car nous avons un nouveau fichier contrôleur.



Quatrièmement je modifie le code de vueAccueil.php pour mettre le bon lien de redirection : Je remplace donc l'ancien lien par le nouveau :

```
<a href="<?= "index.php?action=billet&id=" . $billet['id'] ?>">
    <h1 class="titreBillet"><?= $billet['titre'] ?></h1>
</a>
```

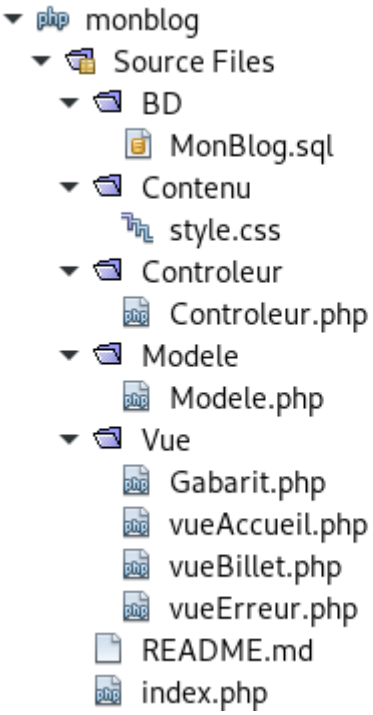
Puis je vérifie que l'url à bien été changé :

On a donc plus le fichier Billet.php qui est appelé pour gérer les données, on ne sait donc plus quel est le fichier appelé donc cela permet de sécuriser un peut plus notre application.

L'organisation des fichiers source

Je vais créer différent répertoire pour correctement organiser l'application
Avec 5 répertoires :

BD, Contenu, Contrôleur, Modèle, Vue



Puis je dois modifier les différent chemin de chaque require :
Par exemple dans le fichier Controleur.php il faut modifier le require de chaque fonction:

```
function accueil() {
    $billets = getBillets();
    require 'Vue/vueAccueil.php';
}

function billet($idBillet) {
    $billet = getBillet($idBillet);
    $commentaires = getCommentaires($idBillet);
    require 'Vue/vueBillet.php';
}

// Affiche une erreur
function erreur($msgErreur) {
    require 'Vue/vueErreur.php';
}
```

Par exemple pour index.php :

```
<?php
require('Controleur/Controleur.php');
try {
```

Je fais ca avec tous les fichiers qui doivent être modifier et je rappelle la page index.php :

Tout cela fonctionne correctement.

ÉTAPE 5 : L'architecture MVC orientée OBJET :

Passage au mode objet

Dans cette étape je vais passer notre Modèle MVC en objet.

Passez le Modèle en objet

Premièrement le modèle.php va être passé en objet. Nous allons créer une classe mère Modèle qui sera hérité par Billet et Commentaire ses deux classes filles.

Ainsi je vais modifier le fichier Modele.php ou il y a la classe mère Modele :

index.php × Modele.php ×

Source History

```php
<?php
abstract class Modele {

// Objet PDO d'accès à la BD
    private $bdd;

// Exécute une requête SQL éventuellement paramétrée
    protected function executerRequete($sql, $params = null) {
        if ($params == null) {
            $resultat = $this->getBdd()->query($sql); // exécution directe
        } else {
            $resultat = $this->getBdd()->prepare($sql); // requête préparée
            $resultat->execute($params);
        }
        return $resultat;
    }

// Renvoie un objet de connexion à la BD en initialisant la connexion au besoin
    private function getBdd() {
        if ($this->bdd == null) {
// Création de la connexion
            $this->bdd = new PDO('mysql:host=localhost;dbname=monblog;charset=utf8',
                'monblog', 'Monblog1234+', array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION));
        }
        return $this->bdd;
    }
}
?>
```

Deuxièmement je créer un fichier Billet.php qui contiendra la première classe fille Billet de la classe Modele dans laquelle il y a la méthode getBillets() et getBillet() :

index.php × Modele.php × Billet.php ×

Source History

```php
<?php
require_once 'Modele/Modele.php';

class Billet extends Modele {

// Renvoie la liste des billets du blog
    public function getBillets() {
        $sql = 'select BIL_ID as id, BIL_DATE as date,'
                . ' BIL_TITRE as titre, BIL_CONTENU as contenu from T_BILLET'
                . ' order by BIL_ID desc';
        $billets = $this->executerRequete($sql);
        return $billets;
    }

// Renvoie les informations sur un billet
    public function getBillet($idBillet) {
        $sql = 'select BIL_ID as id, BIL_DATE as date,'
                . ' BIL_TITRE as titre, BIL_CONTENU as contenu from T_BILLET'
                . ' where BIL_ID=?';
        $billet = $this->executerRequete($sql, array($idBillet));
        if ($billet->rowCount() == 1)
            return $billet->fetch(); // Accès à la première ligne de résultat
        else
            throw new Exception("Aucun billet ne correspond à l'identifiant '$idBillet'");
    }

}
?>
```

Troisièmement je créé donc la seconde classe fille Commentaire dans un nouveau fichier Commentaire.php dans laquelle il y a la méthode getCommentaires() :

Modele.php × Billet.php × Commentaire.php ×

Source History

```php
<?php
require_once 'Modele/Modele.php';

class Commentaire extends Modele {

// Renvoie la liste des commentaires associés à un billet
    public function getCommentaires($idBillet) {
        $sql = 'select COM_ID as id, COM_DATE as date,'
                . ' COM_AUTEUR as auteur, COM_CONTENU as contenu from T_COMMENTAIRE'
                . ' where BIL_ID=?';
        $commentaires = $this->executerRequete($sql, array($idBillet));
        return $commentaires;
    }
}
?>
```

Désormais notre architecture Modèle est en Objet ce qui facilite donc l'enrichissement de notre application si nous voulons créer une nouvelle classe elle sera fille de la classe Modele comme Billet et Commentaire.

Passez les vues en objet

Comme dans la partie précédente je vais créer un classe mère appeler Vue qui sera hérité par vueAccueil et vueBillet :

Tout d'abord, création du fichier Vue.php dans lequelle il y aura la classe Vue ainsi que le constructeur et les méthodes generer, genererFichier :

Gabarit.php × Vue.php ×

Source History

```php
<?php
class Vue {
// Nom du fichier associé à la vue
    private $fichier;
// Titre de la vue (défini dans le fichier vue)
    private $titre;

    public function __construct($action) {
// Détermination du nom du fichier vue à partir de l'action
        $this->fichier = "Vue/vue" . $action . ".php";
    }
// Génère et affiche la vue
    public function generer($donnees) {
// Génération de la partie spécifique de la vue
        $contenu = $this->genererFichier($this->fichier, $donnees);
// Génération du gabarit commun utilisant la partie spécifique
        $vue = $this->genererFichier('Vue/Gabarit.php',
                array('titre' => $this->titre, 'contenu' => $contenu));
// Renvoi de la vue au navigateur
        echo $vue;
    }
// Génère un fichier vue et renvoie le résultat produit
    private function genererFichier($fichier, $donnees) {
        if (file_exists($fichier)) {
// Rend les éléments du tableau $donnees accessibles dans la vue
            extract($donnees);
// Démarrage de la temporisation de sortie
            ob_start();
// Inclut le fichier vue
// Son résultat est placé dans le tampon de sortie
            require $fichier;
// Arrêt de la temporisation et renvoi du tampon de sortie
            return ob_get_clean();
        } else {
            throw new Exception("Fichier '$fichier' introuvable");
        }
    }
}
?>
```

Ensuite, modification du fichier vueAccueil.php pour qu'il soit dans un environnement objet :

Gabarit.php | Vue.php | vueAccueil.php | vueBillet.php

Source | History

```php
<?php $this->titre = "Mon Blog"; ?>
<?php foreach ($billets as $billet):
    ?>
    <article>
        <header>
            <a href="<?= "index.php?action=billet&id=" . $billet['id'] ?>">
                <h1 class="titreBillet"><?= $billet['titre'] ?></h1>
            </a>
            <time><?= $billet['date'] ?></time>
        </header>
        <p><?= $billet['contenu'] ?></p>
    </article>
    <hr />
<?php endforeach;
?>
```

Désormais pour afficher un vue j'appelle donc la méthode generer() .

Passez les contrôleurs en objet

Modification du fichier Controleur.php pour ma part je l'avais déjà fait mes je montre quand même les modifications :

Controleur.php

Source | History

```php
<?php
require 'Modele/Modele.php';

// Afficher la liste de tous les billets
function accueil() {
    $billets = getBillets();
    require 'Vue/vueAccueil.php';
}

// Afficher un billet
function billet($idBillet) {
    $billet = getBillet($idBillet);
    $commentaires = getCommentaires($idBillet);
    require 'Vue/vueBillet.php';
}

// Afficher une erreur
function erreur($msgErreur) {
    require 'Vue/vueErreur.php';
}
?>
```

Ainsi que index.php que j'avais aussi déjà modifier :

Controleur.php × index.php ×

Source History

```php
<?php
require('Controleur/Controleur.php');
try {
    if (isset($_GET['action'])) {
        if ($_GET['action'] == 'billet') {
            if (isset($_GET['id'])) {
                $idBillet = intval($_GET['id']);
                if ($idBillet != 0)
                    billet($idBillet);
                else
                    throw new Exception("Identifiant de billet non valide");
            } else
                throw new Exception("Identifiant de billet non défini");
        } else
            throw new Exception("Action non valide");
    } else {
        accueil(); // action par défaut
    }
} catch (Exception $e) {
    erreur($e->getMessage());
}
?>
```

Puis je créer controleurAccueil.php :

Controleur.php × index.php × ControleurAccueil.php ×

Source History

```php
<?php
require_once 'Modele/Billet.php';
require_once 'Vue/Vue.php';

class controleurAccueil {

    private $billet;

    public function _contruct() {
        $this->billet = new Billet();
    }

    //Afficher ma miste des billet du blog
    public function accueil() {
        $billets = $this->billet->getBillets();
        $vue = new Vue("Accueil");
        $vue->generer(array("billets" => $billets));
    }
}
?>
```

Création de controleurBillet.php :

Controleur.php | index.php | ControleurAccueil.php | ControleurBillet.php

Source | History

```php
<?php

require_once 'Modele/Billet.php';
require_once 'Modele/Commentaire.php';
require_once 'Vue/Vue.php';

class ControleurBillet {

    private $billet;
    private $commentaire;

    public function _contruct() {
        $this->billet = new Billet();
        $this->commentaire = new Commentaire();
    }

    //Afficher les détails d'un billet
    public function billet($idBillet) {
        $billet = $this->billet->getBillet($idBillet);
        $commentaires = $this->commentaire->getCommentaires($idBillet);
        $vue = new Vue("Billet");
        $vue->generer(array('billet' => $billet, 'commentaires' => $commentaires));
    }
}
?>
```

Création du Routeur

Dans cette partie je vais créer le fichier Routeur.php qui va permettre d'analyser la requête qui entre pour ensuite savoir l'action par rapport à cette requête.

Controleur.php | Routeur.php

Source | History

```php
<?php
require_once 'Controleur/ControleurAccueil.php';
require_once 'Controleur/ControleurBillet.php';
require_once 'Vue/Vue.php';

class Routeur {

    private $ctrlAccueil;
    private $ctrlBillet;

    public function __construct() {
        $this->ctrlAccueil = new ControleurAccueil();
        $this->ctrlBillet = new ControleurBillet();
    }

// Traite une requête entrante
    public function routerRequete() {
        try {
            if (isset($_GET['action'])) {
                if ($_GET['action'] == 'billet') {
                    if (isset($_GET['id'])) {
                        $idBillet = intval($_GET['id']);
                        if ($idBillet != 0) {
                            $this->ctrlBillet->billet($idBillet);
                        } else
                            throw new Exception("Identifiant de billet non valide");
                    } else
                        throw new Exception("Identifiant de billet non défini");
                } else
                    throw new Exception("Action non valide");
            } else { // aucune action définie : affichage de l'accueil
                $this->ctrlAccueil->accueil();
            }
        } catch (Exception $e) {
            $this->erreur($e->getMessage());
        }
    }
```

```php
// Affiche une erreur
    private function erreur($msgErreur) {
        $vue = new Vue("Erreur");
        $vue->generer(array('msgErreur' => $msgErreur));
    }

}
?>
```

Puis ensuite modifier index.php pour qu'il utilise le class Routeur :

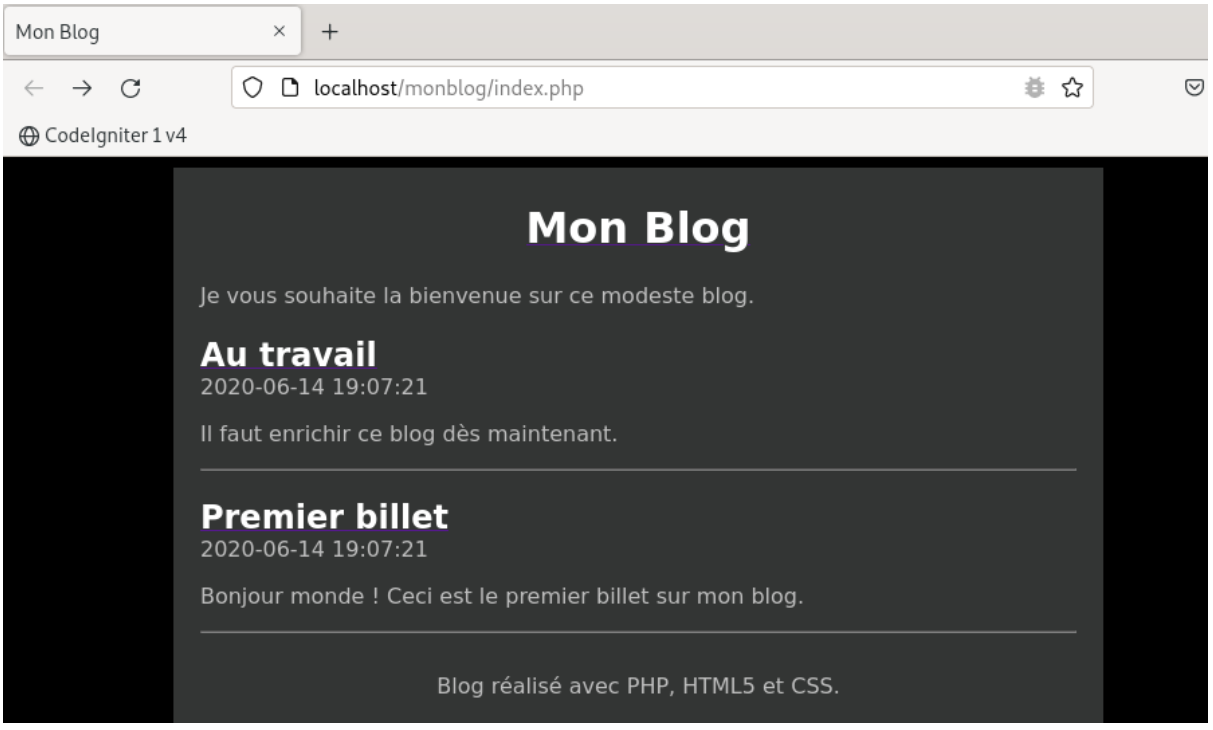
index.php ×

Source History

```php
<?php
require 'Controleur/Routeur.php';

$routeur = new Routeur();
$routeur->routerRequete();
?>
```

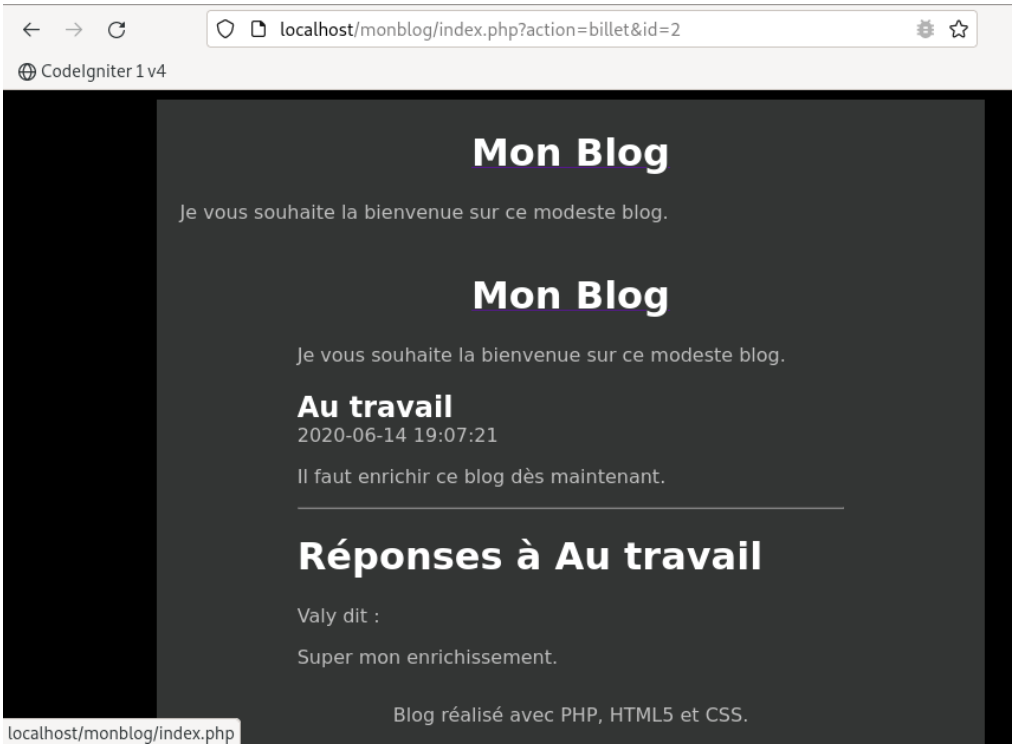
J'ai donc modifier le fichier index.php au maximum.

Pour finir il faut run la page pour voir si cela fonctionne.



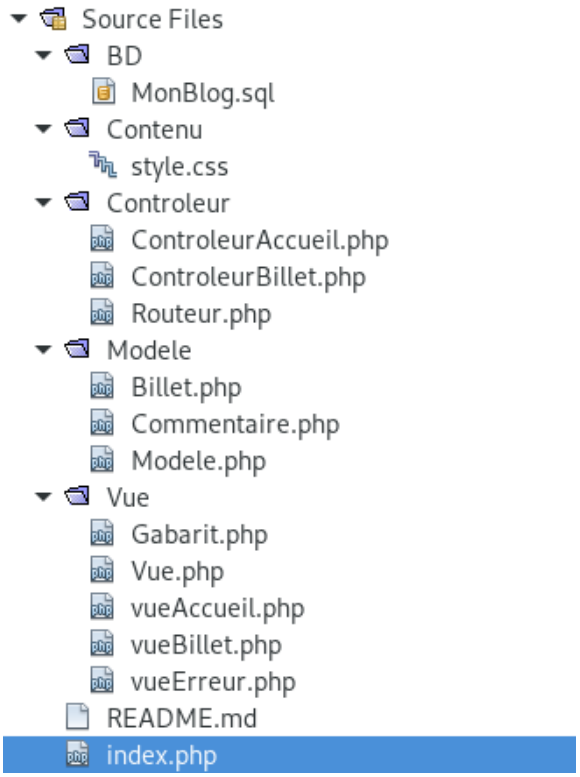
Cela fonctionne correctement, on a le même affichage mais pas la même méthode.

Lorsque l'on clique sur Au travail donc dans l'id 2 de notre bdd:



Bilan de cette étape

Nous avons donc augmenté la robustesse de notre projet en ajoutant toutes ses nouvelles méthodes.
Voici l'architecture de mon Projet :



ÉTAPE 6 :NOUVEAU BESOIN : AJOUTS :

Besoin : ajout d'un commentaire

Dans cette étape je vais créer la possibilité de pouvoir commenter dans la base de données en remplissant des champs dans un formulaire.

Application

Dans un premier temps je vais modifier le contenu de Commentaire en ajoutant la possibilité d'introduire des commentaires dans la base de données avec une méthode s'appelant ajouterCommentaire, je rappelle que cette classe est une fille de la classe Modèle.

Modification de Commentaire.php

```php
    // Ajoute un commentaire dans la base
    public function ajouterCommentaire($auteur, $contenu, $idBillet) {
        $sql = 'insert into T_COMMENTAIRE(COM_DATE, COM_AUTEUR, COM_CONTENU, BIL_ID)'
                . ' values(?, ?, ?, ?)';
        $date = date(DATE_W3C); // Récupère la date courante
        $this->executerRequete($sql, array($date, $auteur, $contenu, $idBillet));
    }

}
```

On doit ensuite ajouter le formulaire en html permettant la saisi d'un commentaire dans VueBillet.php aprés la ligne <?php endforeach; ?>:

Modification de vueBillet.php

```php
<?php endforeach; ?>
<form method="post" action="index.php?action=commenter">
    <input id="auteur" name="auteur" type="text" placeholder="Votre pseudo"
           required /><br />
    <textarea id="txtCommentaire" name="contenu" rows="4"
              placeholder="Votre commentaire" required></textarea><br />
    <input type="hidden" name="id" value="<?= $billet['id'] ?>" />
    <input type="submit" value="Commenter" />
</form>
<?php $contenu = ob_get_clean(); ?>
<?php require 'Gabarit.php'; ?>
```

Par ailleurs on va ajouter le paramètre #txtCommentaire dans notre css afin d'augmenter la largeur 'width' de saisie commentaire.

Modification de style.css

```css
#txtCommentaire {
    width: 50%;
}
```

Il faut aussi ajouter un contrôle permettant d'ajouter un commentaire à un billet, de sauvegarder et d'actualiser l'affichage du billet dans ControleurBillet.

Modification de ConttroleurBillet.php:

```
25      // Ajoute un commentaire à un billet
26      public function commenter($auteur, $contenu, $idBillet) {
27  // Sauvegarde du commentaire
28          $this->commentaire->ajouterCommentaire($auteur, $contenu, $idBillet);
29  // Actualisation de l'affichage du billet
30          $this->billet($idBillet);
31      }
32
33  }
```

Pour finir les modifications il me manque d'ajouter dans Routeur.php la possibilité de router vers ajoutcommentaire. On ajoute donc la fonction getParametre afin de rechercher un paramètre dans un tableau.

Modification de Routeur.php ajout fonction getParametre:

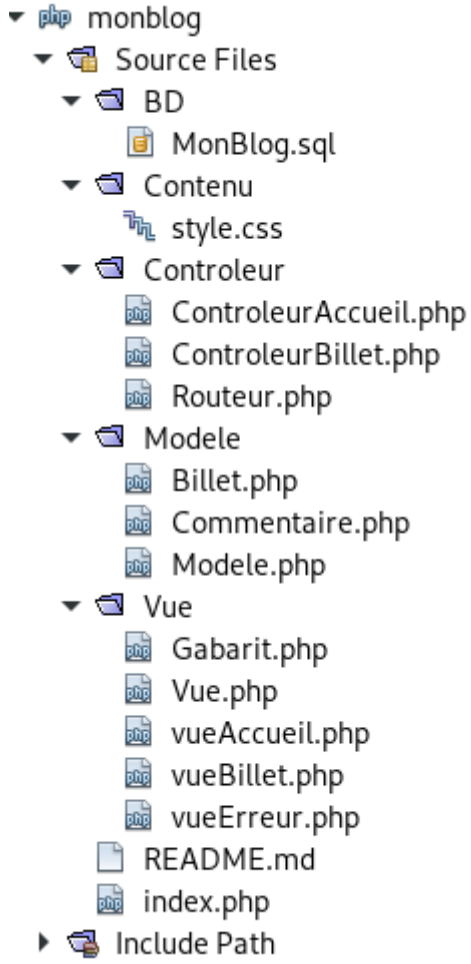
```
46      // Recherche un paramètre dans un tableau
47      private function getParametre($tableau, $nom) {
48          if (isset($tableau[$nom])) {
49              return $tableau[$nom];
50          } else
              throw new Exception("Paramètre '$nom' absent");
52      }
53
```

Ajout de la méthode de routage modification de routerRequete pour appeler à la fonction getParametre.

Modification de Routeur.php modification de routerRequete:

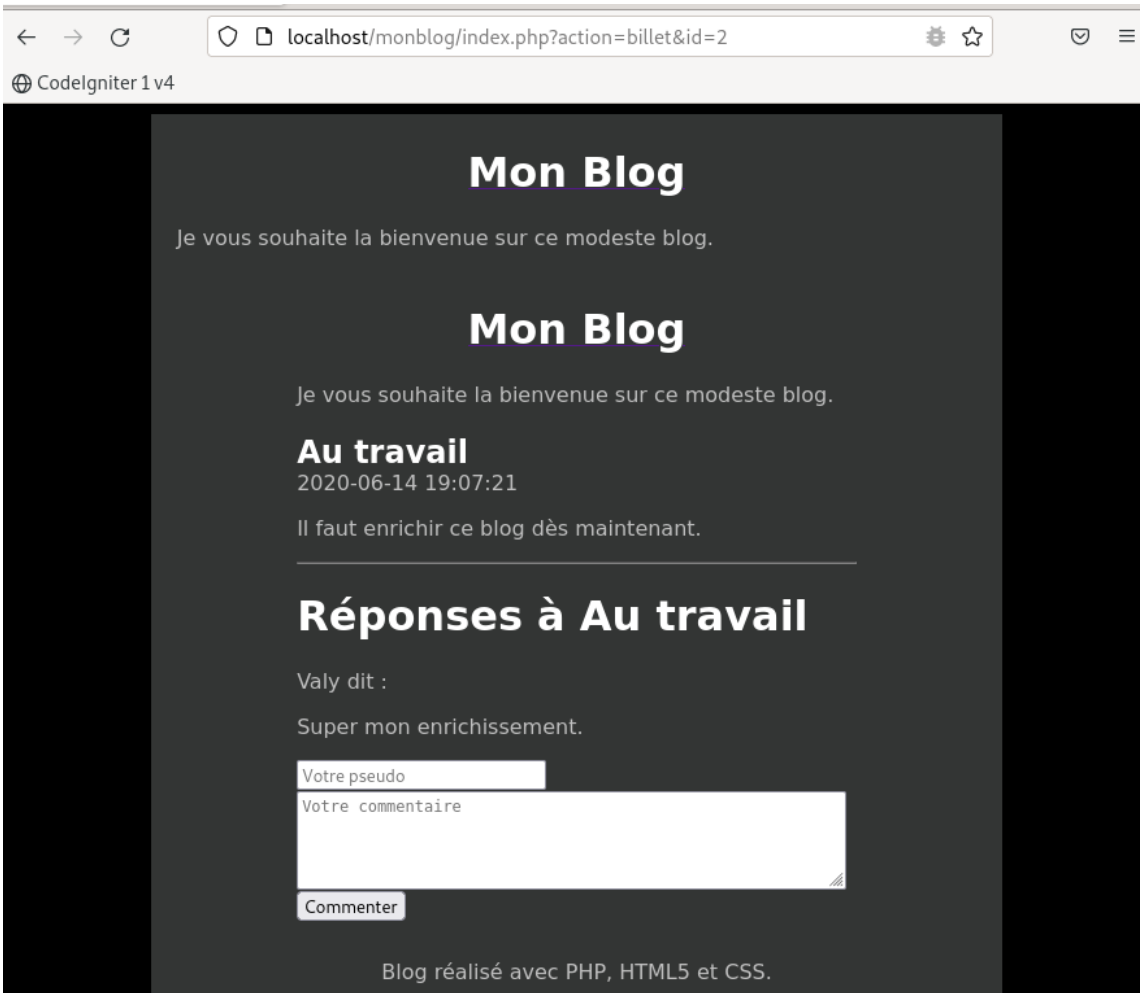
```
17  // Traite une requête entrante
    public function routerRequete() {
19      try {
            if (isset($_GET['action'])) {
                if ($_GET['action'] == 'billet') {
                    $idBillet = intval($this->getParametre($_GET, 'id'));
                    if ($idBillet != 0) {
24                      $this->ctrlBillet->billet($idBillet);
25                  } else
                        throw new Exception("Identifiant de billet non valide");
                } else if ($_GET['action'] == 'commenter') {
                    $auteur = $this->getParametre($_POST, 'auteur');
                    $contenu = $this->getParametre($_POST, 'contenu');
                    $idBillet = $this->getParametre($_POST, 'id');
31                  $this->ctrlBillet->commenter($auteur, $contenu, $idBillet);
32              } else
                    throw new Exception("Action non valide");
34          } else { // aucune action définie : affichage de l'accueil
35              $this->ctrlAccueil->accueil();
                }
37      } catch (Exception $e) {
38          $this->erreur($e->getMessage());
39      }
40  }
```

On a donc pour architecture finale:

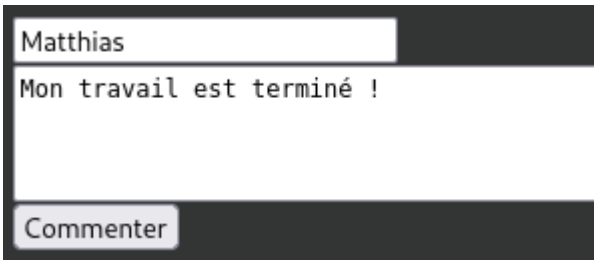


Test de la modification

Premièrement voici la page par défaut obtenu:



Saisie d'un commentaire:



Retourne: