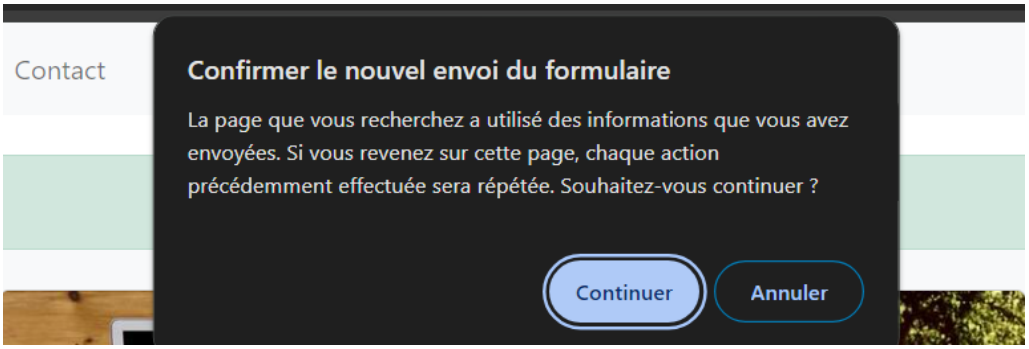
PHP Blog Project

4. Failles XSS et injections SQL

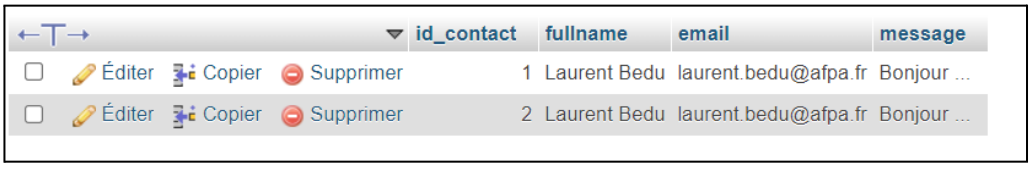
Le code que nous avons écrit dans le TP précédent comporte plusieurs problèmes

1.Doublons dans la DB au refresh de la page

Après avoir envoyé votre message sur le formulaire de contact, une ligne est créée en DB.
Si vous faites un refresh de la page vous obtenez le popup suivant



Si vous cliquez sur continuer, une deuxième ligne identique à la première (à l'exception de l'id) sera créée en DB



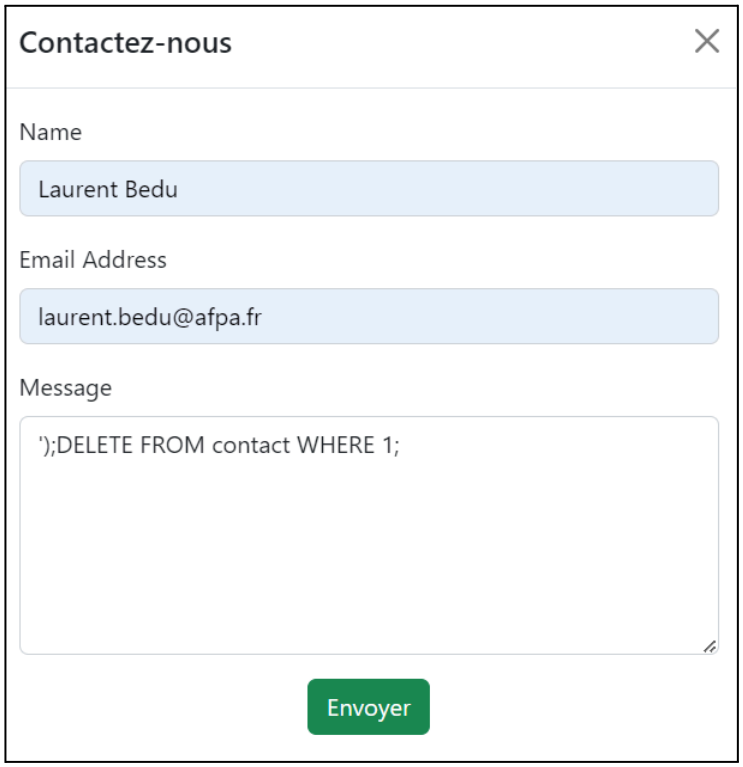
	id_contact	fullname	email	message
Éditer Copier Supprimer	1	Laurent Bedu	laurent.bedu@afpa.fr	Bonjour ...
Éditer Copier Supprimer	2	Laurent Bedu	laurent.bedu@afpa.fr	Bonjour ...

Afin d'éviter cela, nous ajoutons un script JS sous l'alert bootstrap affichant le message.

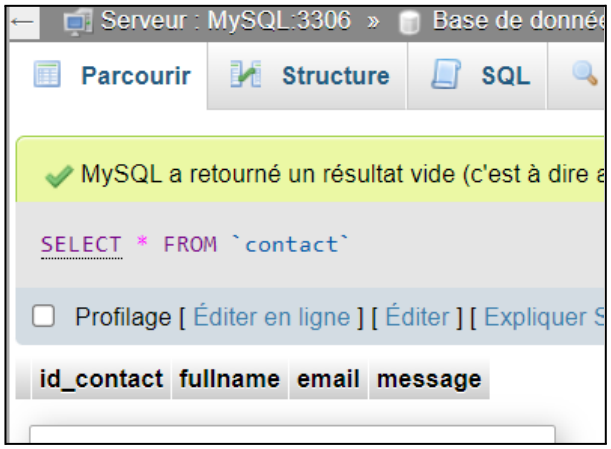
```php
if($result){ ?>
    <div class="alert alert-success alert-dismissible fade show"
        role="alert" style="margin-top:80px; margin-bottom:-
        Votre message est envoyé.
        <button type="button" class="btn-close" data-bs-dismiss=
    </div>
    <script>
        if (history.replaceState) {
            history.replaceState(null, null, location.href);
        }
    </script>
<?php }
```

2. Injections SQL

Si quelqu'un de mal intentionné remplit le formulaire comme ci-dessous en ajoutant du code SQL, il peut y avoir des dégâts irréversibles sur la DB.



La table contact est complètement vidée !

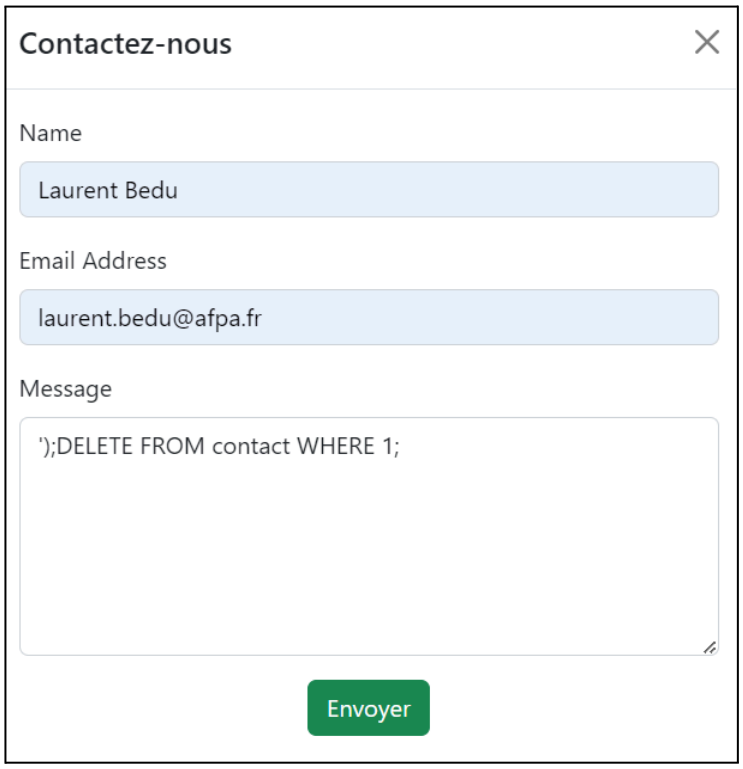


Pour remédier à cela, nous allons utiliser des requêtes préparées en modifiant légèrement le code pour l'insertion en DB dans contact.php

```
48          );
49          $sql = "INSERT INTO contact (fullname, email, message) VALUES (?,?,?);";
50          $stmt = $db->prepare($sql);
51          $result = $stmt->execute([$fullname, $email, $message]);
52          $result = $result && $stmt->rowCount() == 1;
53          if($result){ ?>
```

doc :
https://www.php.net/manual/en/pdo.prepare.php
https://www.php.net/manual/en/pdostatement.execute.php

Nous essayons à nouveau d'injecter du code SQL malveillant



Cela ne fonctionne plus !

Il nous reste à modifier les requêtes exécutées dans les pages index.php et article.php pour protéger toute notre application des injections SQL

```php
    $sql = "SELECT * FROM article ORDER BY ? DESC LIMIT 12;";
    $stmt = $db->prepare($sql);
    $stmt->execute(['published_at']);
    $articles = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

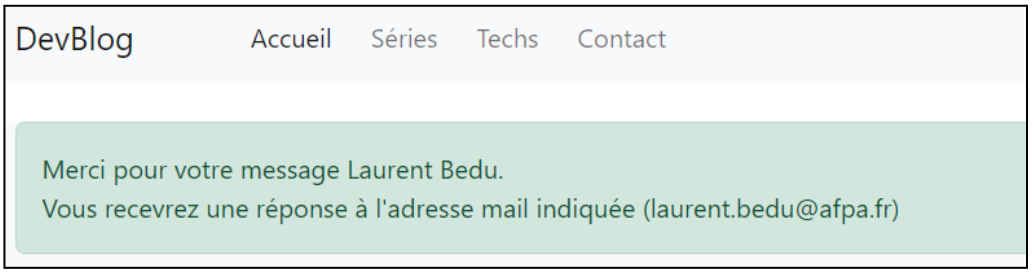
```php
    $sql = "SELECT * FROM article WHERE id_article = ?;";
    $stmt = $db->prepare($sql);
    $stmt->execute([$_GET['id']]);
    $article = $stmt->fetch(PDO::FETCH_ASSOC);
```

3. Failles XSS

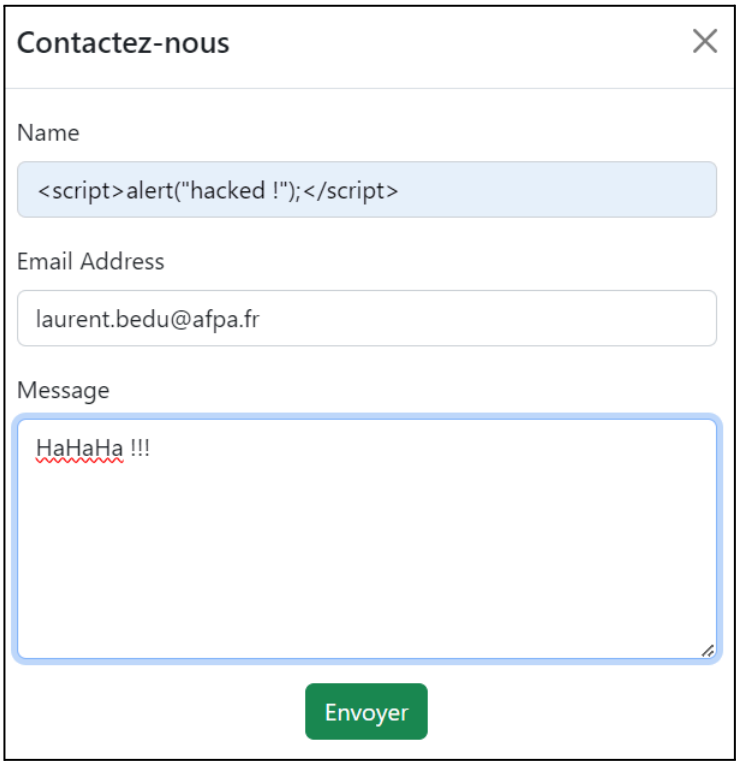
Si notre alert bootstrap confirmant l'envoi du message comporte par exemple le nom saisi dans le formulaire de contact comme ci dessous

```php
if($result){ ?>
    <div class="alert alert-success alert-dismissible fade show"
        role="alert" style="margin-top:80px; margin-bottom:-40px;">
        Merci pour votre message <?= $fullname; ?>.<br/>
        Vous recevrez une réponse à l'adresse mail indiquée (<?= $email ?>)
        <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
    </div>
    <script>
        if (history.replaceState) {
            history.replaceState(null, null, location.href);
        }
    </script>
<?php }
```

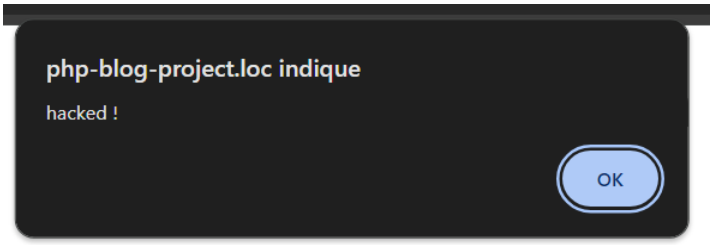
Résultat suite à l'envoi du formulaire



Il y a alors un risque d'injection de code (Faille XSS) comme dans l'exemple ci dessous



Résultat suite à l'envoi du formulaire



Afin de prévenir ces injections de code, nous allons nettoyer (sanitize) les données saisies dans le formulaire et stockées dans la variable prédéfinie $_POST à l'aide de filter_var

```php
<?php
    if (isset($_POST['send'])) {
        $fullname = filter_var($_POST['fullname'], FILTER_SANITIZE_FULL_SPECIAL_CHARS);
        $email = filter_var($_POST['email'], FILTER_SANITIZE_EMAIL);
        $message = filter_var($_POST['message'], FILTER_SANITIZE_FULL_SPECIAL_CHARS);
        include_once "./configs/db.config.php";
```

De cette manière, le script saisi dans le formulaire ne s'exécute plus.
Voyez en DB la différence entre les 2 lignes insérées avant (4) et après (5) l'utilisation de filter_var

	id_contact ▲ 1	fullname	email	message
Éditer Copier Supprimer	3	Laurent Bedu	laurent.bedu@afpa.fr	');DELETE FROM contact WHERE 1;
Éditer Copier Supprimer	4	<script>alert("hacked !");</script>	laurent.bedu@afpa.fr	HaHaHa !!!
Éditer Copier Supprimer	5	<script>alert("hacked !");</scr...	laurent.bedu@afpa.fr	HaHaHa !!!

doc :
https://www.php.net/manual/en/function.filter-var.php
https://www.php.net/manual/en/filter.filters.php
https://www.php.net/manual/en/filter.filters.sanitize.php

Tout ce qui est saisi ou modifiable par l'utilisateur doit être vérifié, validé, nettoyé.
Nous pouvons appliquer la méthode filter_var à l'id passé en paramètre GET dans l'url pour la page article.php (puisqu'il est modifiable)

```php
    $sql = "SELECT * FROM article WHERE id_article = ?;";
    $stmt = $db->prepare($sql);
    $id = filter_var($_GET['id'], FILTER_SANITIZE_NUMBER_INT);
    $stmt->execute([$id]);
    $article = $stmt->fetch(PDO::FETCH_ASSOC);
    if($article == false){
        $redirect_url = 'index.php';
        header('Location: '.$redirect_url);
        die();
    }
?>
```

Si l'id saisi dans l'URL n'est pas correct (s'il n'est pas un nombre entier ou n'existe pas en DB), le fetch de la ligne 38 renvoie false, nous serons donc redirigé vers la page d'accueil.

git :
https://github.com/DWWM-23526/PHP_BLOG_PROJECT/tree/Step04