[PHP] WebPush

Step-by-step implementation of Web Push Notifications using plain PHP and the **Minishlink/web-push** library:

---

0. Install Composer in Windows

How to Install Composer in Windows, click this link

1. Install the Minishlink Web Push Library

Install the library using Composer:

```
composer require minishlink/web-push
```

---

2. Generate VAPID Keys

Use the library to generate the VAPID keys for authentication:

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

use Minishlink\WebPush\VAPID;

// Generate VAPID keys
$vapidKeys = VAPID::createVapidKeys();

echo 'Public Key: ' . $vapidKeys['publicKey'] . PHP_EOL;
echo 'Private Key: ' . $vapidKeys['privateKey'] . PHP_EOL;
```

Save the `Public Key` and `Private Key` for use in the frontend and backend.

---

3. Create the Subscription Storage

Store user subscriptions in a database.

Example Table:

```
CREATE TABLE push_subscriptions (
    id INT AUTO_INCREMENT PRIMARY KEY,
    endpoint TEXT NOT NULL,
    public_key TEXT NOT NULL,
    auth_key TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

---

4. Create an Endpoint to Save Subscriptions

Create a PHP script (`subscribe.php`) to save subscription data sent by the browser.

`subscribe.php`:

```
<?php
require 'vendor/autoload.php';

// Database connection
$dsn = 'mysql:host=localhost;dbname=your_database;charset=utf8';
$username = 'your_username';
$password = 'your_password';

try {
    $pdo = new PDO($dsn, $username, $password);
} catch (PDOException $e) {
    die('Database connection failed: ' . $e->getMessage());
}

// Retrieve subscription data from the request
$data = json_decode(file_get_contents('php://input'), true);
```

```
$endpoint = $data['endpoint'];
$publicKey = $data['keys']['p256dh'];
$authKey = $data['keys']['auth'];

// Store subscription in the database
$stmt = $pdo->prepare("INSERT INTO push_subscriptions (endpoint, public_key, auth_key) VALUES (?, ?, ?)");
$stmt->execute([$endpoint, $publicKey, $authKey]);

echo json_encode(['success' => true]);
```

---

5. Create a Script to Send Notifications

Create a PHP script (`send_notification.php`) to send notifications to all subscribers.

`send_notification.php`:

```
<?php
require 'vendor/autoload.php';

use Minishlink\WebPush\WebPush;
use Minishlink\WebPush\Subscription;

// Database connection
$dsn = 'mysql:host=localhost;dbname=your_database;charset=utf8';
$username = 'your_username';
$password = 'your_password';

try {
    $pdo = new PDO($dsn, $username, $password);
} catch (PDOException $e) {
    die('Database connection failed: ' . $e->getMessage());
}

// Retrieve all subscriptions
$stmt = $pdo->query("SELECT * FROM push_subscriptions");
```

```
$subscriptions = $stmt->fetchAll(PDO::FETCH_ASSOC);

// VAPID keys
$vapid = [
    'subject' => 'mailto:example@example.com',
    'publicKey' => 'YOUR_PUBLIC_KEY',
    'privateKey' => 'YOUR_PRIVATE_KEY',
];

// WebPush instance
$webPush = new WebPush(['VAPID' => $vapid]);

// Notification payload
$payload = json_encode([
    'title' => 'Reminder',
    'body' => 'You have a new task to complete!',
]);

// Send notifications
foreach ($subscriptions as $subscription) {
    $webPush->sendNotification(
        Subscription::create([
            'endpoint' => $subscription['endpoint'],
            'publicKey' => $subscription['public_key'],
            'authToken' => $subscription['auth_key'],
        ]),
        $payload
    );
}

// Flush notifications
foreach ($webPush->flush() as $report) {
    if (!$report->isSuccess()) {
        echo "Push failed: {$report->getReason()} for endpoint: {$report->getEndpoint()}\n";
    }
}
```

6. Frontend Implementation

Register a Service Worker

Create a `sw.js` file in the public directory to handle incoming push notifications.

```
self.addEventListener('push', event => {
    const data = event.data.json();
    self.registration.showNotification(data.title, {
        body: data.body,
        icon: '/icon.png', // Optional
    });
});
```

Subscribe to Push Notifications

Add JavaScript to your frontend to subscribe users to push notifications.

```
<script>
    const publicKey = 'YOUR_PUBLIC_KEY'; // Replace with your VAPID public key

    // Convert VAPID public key to Uint8Array
    function urlBase64ToUint8Array(base64String) {
        const padding = '='.repeat((4 - base64String.length % 4) % 4);
        const base64 = (base64String + padding).replace(/-/g, '+').replace(/_/g, '/');
        const rawData = window.atob(base64);
        return Uint8Array.from([...rawData].map((char) => char.charCodeAt(0)));
    }

    // Register the service worker
    navigator.serviceWorker.register('/sw.js').then(registration => {
        // Subscribe the user to push
        return registration.pushManager.subscribe({
            userVisibleOnly: true,
            applicationServerKey: urlBase64ToUint8Array(publicKey),
        });
    }).then(subscription => {
        // Send subscription data to the server
        return fetch('/subscribe.php', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
```

```
            },
            body: JSON.stringify(subscription),
        });
    }).then(() => {
        console.log('Subscribed to push notifications!');
    }).catch(err => {
        console.error('Failed to subscribe:', err);
    });
</script>
```

7. Test Your Implementation

1. Serve your application over HTTPS (required for the Push API).
2. Open the page in a browser and subscribe to push notifications.
3. Trigger `send_notification.php` to send a push notification.

Revision #2
Created 2 December 2024 06:08:06 by Martin
Updated 2 December 2024 06:22:30 by Martin