Tema 7: Crear una API propia

Introducción

Crear una API sencilla en PHP puede ser un buen ejercicio para comprender los conceptos básicos. Vamos a desarrollar una API simple que maneje operaciones CRUD (Crear, Leer, Actualizar, Eliminar) para una lista de usuarios. Esta API será basada en RESTful y utilizará el método GET para obtener usuarios, POST para agregar un nuevo usuario, PUT para actualizar un usuario y DELETE para eliminar un usuario.

Paso 1: Estructura del Proyecto

Crea un directorio para tu proyecto y dentro de él, crea un archivo index.php. Este será el punto de entrada para todas las solicitudes a tu API.

```
<?php
// index.php echo "¡Bienvenido a la API!";
?>
```

Paso 2: Configuración básica

Vamos a configurar algunos encabezados básicos para permitir el acceso desde cualquier origen (CORS) y para asegurarnos de que PHP maneje los datos como JSON.

```
<?php
 // index.php
// Configurar encabezados para permitir el acceso desde cualquier origen
 header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
// Otras configuraciones…
 ?>
```

Paso 3: Manejo de Solicitudes

Ahora, vamos a manejar las solicitudes GET, POST, PUT y DELETE en nuestro index.php. Dependiendo del método de solicitud, realizaremos diferentes acciones.

```
<?php
// index.php
header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");

// Manejar solicitudes GET
if ($_SERVER['REQUEST_METHOD'] === 'GET') {
        // Lógica para obtener usuarios y devolverlos como JSON
        echo json_encode(["message" => "GET request handled"]);
}

// Manejar solicitudes POST
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        // Lógica para agregar un nuevo usuario
        echo json_encode(["message" => "POST request handled"]);
}
```

```
// Manejar solicitudes PUT
if ($_SERVER['REQUEST_METHOD'] === 'PUT') {
        // Lógica para actualizar un usuario existente
        echo json_encode(["message" => "PUT request handled"]);
}

// Manejar solicitudes DELETE
if ($_SERVER['REQUEST_METHOD'] === 'DELETE') {
        // Lógica para eliminar un usuario existente
        echo json_encode(["message" => "DELETE request handled"]);
}
?>
```

Paso 4: Ejecutar la API

Puedes usar un servidor web local para ejecutar tu API. Puedes probar tu API usando herramientas como Postman o cURL.

Paso 5: Expandir la Lógica

Ahora que tienes la estructura básica, puedes expandir la lógica dentro de cada bloque de solicitud para realizar operaciones más significativas, como interactuar con una base de datos para almacenar y recuperar datos de usuarios.

Crear una API con base de datos

Crear una API en PHP que interactúe con una base de datos puede ser más complejo, pero también más poderoso. En este ejemplo, utilizaré MySQL como base de datos y PDO (PHP Data Objects) para la conexión y ejecución de consultas. Asegúrate de tener una base de datos MySQL en funcionamiento antes de comenzar.

Paso 1: Configuración de la Base de Datos

Crea una base de datos llamada usuarios y una tabla llamada usuarios con columnas id y nombre.

```
CREATE DATABASE usuarios;
USE usuarios;
CREATE TABLE usuarios (
        id INT AUTO_INCREMENT PRIMARY KEY,
        nombre VARCHAR(255) NOT NULL
);
```

Paso 2: Archivo de Configuración

Crea un archivo config.php para almacenar la configuración de la base de datos.

```
<?php
// config.php
$servername = "localhost";
$username = "tu_usuario";
$password = "tu_contraseña";
$dbname = "usuarios";
try {
        $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password); $
        conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
        echo "Error de conexión: " . $e->getMessage();
}
?>
```

Paso 3: Crear API CRUD

Crea un archivo api.php que maneje las solicitudes CRUD.

```
<?php
// api.php
include 'config.php';
header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
// Obtener todos los usuarios
if ($_SERVER['REQUEST_METHOD'] === 'GET') {
        $stmt = $conn->prepare("SELECT * FROM usuarios"); $
        stmt->execute();
        $result = $stmt->fetchAll(PDO::FETCH_ASSOC);
        echo json_encode($result);
}

// Agregar un nuevo usuario
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        $data = json_decode(file_get_contents("php://input"), true);
        if (isset($data['nombre'])) {
                $nombre = $data['nombre'];
                $stmt = $conn->prepare("INSERT INTO usuarios (nombre) VALUES (:nombre)");
                $stmt->bindParam(':nombre', $nombre);
                $stmt->execute();
                echo json_encode(['message' => 'Usuario agregado']);
        } else {
                echo json_encode(['error' => 'Datos incompletos']);
        }
}

// Actualizar un usuario
if ($_SERVER['REQUEST_METHOD'] === 'PUT') {
        $data = json_decode(file_get_contents("php://input"), true);
        if (isset($data['id']) && isset($data['nombre'])) {
                $id = $data['id'];
                $nombre = $data['nombre'];
                $stmt = $conn->prepare("UPDATE usuarios SET nombre = :nombre WHERE id
= :id");
                $stmt->bindParam(':nombre', $nombre);
                $stmt->bindParam(':id', $id);
                $stmt->execute();
                echo json_encode(['message' => 'Usuario actualizado']);
        }
        else {
                echo json_encode(['error' => 'Datos incompletos’]);
        }
}

// Eliminar un usuario
if ($_SERVER['REQUEST_METHOD'] === 'DELETE') {
        $data = json_decode(file_get_contents("php://input"), true);
        if (isset($data['id'])) {
                $id = $data['id'];
                $stmt = $conn->prepare("DELETE FROM usuarios WHERE id = :id");
                $stmt->bindParam(':id', $id); $
                stmt->execute();
                echo json_encode(['message' => 'Usuario eliminado']); }
         else {
                echo json_encode(['error' => 'Datos incompletos']);
```

```
        }
} ?>
```

Paso 4: Probar la API

Puedes probar la API usando las mismas herramientas que mencioné anteriormente. Aquí hay algunos ejemplos:

- GET: http://localhost:8000/api.php para obtener la lista de usuarios.
- POST: Enviar una solicitud POST a http://localhost:8000/api.php con datos JSON en el cuerpo para agregar un nuevo usuario.
-
- PUT: Enviar una solicitud PUT a http://localhost:8000/api.php con datos JSON en el cuerpo para actualizar un usuario existente.
- DELETE: Enviar una solicitud DELETE a http://localhost:8000/api.php con datos JSON en el cuerpo para eliminar un usuario existente.

Este ejemplo utiliza PDO para interactuar con la base de datos. Asegúrate de configurar el archivo config.php con las credenciales correctas de tu base de datos. Recuerda que este es un ejemplo educativo y, en un entorno de producción, deberías implementar medidas de seguridad adicionales.

Cómo usar PUT, POST y DELETE para enviar datos a la API

Enviar datos usando POST
Este caso es también muy típico, querer enviar datos a un servidor para dar de alta recursos, un post, una tarea o cualquier otra cosa, mira qué sencillo es:

Tenemos los datos que tenemos que mandar en un array $todo.

```
$curl = curl_init();

curl_setopt($curl, CURLOPT_URL, $url);

curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);

curl_setopt($curl, CURLOPT_POST, true);

curl_setopt($curl, CURLOPT_POSTFIELDS, json_encode($todo));

$response = curl_exec($curl);

curl_close($curl);

var_dump($response);
```

Actualizar datos con cURL utilizando PUT
Una vez hemos creado un recurso, posiblemente necesitaremos actualizarlo, para ello podemos hacer lo siguiente:

```
$curl = curl_init();

curl_setopt($curl, CURLOPT_URL, $url . '/' . $todo[‘id']);

curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);

curl_setopt($curl, CURLOPT_CUSTOMREQUEST, 'PUT');
```

```
curl_setopt($curl, CURLOPT_POSTFIELDS, json_encode($todo));

curl_setopt($curl, CURLOPT_HTTPHEADER, [
    'Content-Type: application/json'
]);

$response = curl_exec($curl);

curl_close($curl);

var_dump($response);
```

Eliminar datos con cURL utilizando DELETE
Una vez hemos creado, listado y actualizado, es el momento de eliminar, para ello, siguiendo el ejemplo con PUT, podemos hacer lo mismo, pero en este caso una petición DELETE.

```
$curl = curl_init();

curl_setopt($curl, CURLOPT_URL, $url . '/' . $todo['id']);

curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);

curl_setopt($curl, CURLOPT_CUSTOMREQUEST, 'PUT');

curl_setopt($curl, CURLOPT_POSTFIELDS, json_encode($todo));

curl_setopt($curl, CURLOPT_HTTPHEADER, [
    'Content-Type: application/json'
]);

$response = curl_exec($curl);

curl_close($curl);

var_dump($response);
```

Basic Auth con cURL en PHP
En el contexto de una transacción HTTP, la autenticación de acceso básica es un método diseñado para permitir a un navegador web, u otro programa cliente, proveer credenciales en la forma de usuario y contraseña cuando se le solicita una página al servidor.

Realizar una petición utilizando Basic Auth con cURL en PHP es realmente sencillo.

```
$curl = curl_init();

curl_setopt($curl, CURLOPT_URL, $url);

curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);

curl_setopt($curl, CURLOPT_USERPWD, $username . ':' . $password);

$response = curl_exec($curl);

curl_close($curl);

var_dump($response);
```