PHP-based URL shortener

Overview

This PHP-based URL shortener application allows users to input a URL, generate a shortened version, and track the number of times the short link is accessed. The shortened URL redirects users to the original link and updates the click count in the database.

- **Shortening Process**: Users enter a long URL, which is stored in the `urls` table.
- **Redirection**: When accessing the short URL, `u/index.php` looks up the corresponding long URL, increments the click count, and redirects the user.
- **Dynamic Update**: A jQuery function refreshes the page periodically, updating the displayed click count for each URL.

Code Explanation:

first file (config.php):

- this file serves as the central configuration for database connection management.
- config.php contains the necessary parameters and code to establish a connection to the MySQL database, which allows the rest of your application to interact with the database (e.g., to store URLs, update click counts, etc.).

```
<?php

try {
    // Database connection parameters
    $host = "localhost"; // Database host, typically "localhost'
    $dbname = "short-urls"; // The name of the database you're 
    $user = "root"; // The username for the MySQL database.
    $pass = ""; // The password for the MySQL database. Here it

    // Creating a new PDO instance to connect to the MySQL datal
    $conn = new PDO("mysql:host=$host;dbname=$dbname", $user, $
    // Setting the error mode for the PDO instance
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTI
} catch (PDOException $e) {
    // If there is an error, catch the exception and display the
    echo "Connection failed: " . $e->getMessage();
}
```

Main File (`index.php`):

1.**Connecting to the Database**:

```
<?php require "config.php"; ?>
```

→This line pulls in the database connection setup, so the `$conn` object is available for running queries.

2.
Fetching All URLs from the Database:

```
$select = $conn->query('SELECT * FROM urls');
$select->execute();
$rows = $select->fetchAll(PDO::FETCH_OBJ);
```

- A SQL query selects all records from the `urls` table, and `$select->execute()` executes the query.
- `fetchAll(PDO::FETCH_OBJ)` fetches the results as an array of objects, where each object represents a row in the table, with properties corresponding to the table columns (`id`, `url`, `clicks`).

3.**Handling Form Submission**:

```
if (isset($_POST['submit'])) {
if ($_POST['url'] == '') {
echo "Input empty, please enter the URL.";
} else {
```

```
$url = $_POST['url'];
$insert = $conn->prepare("INSERT INTO urls (url) VALUES (:url)"
$insert->execute(['url' => $url]);
}
}
```

- When the form is submitted, it checks if `$_POST['submit']` is set.
- If the URL field is empty, it prompts the user to enter a URL. Otherwise, it inserts the provided URL into the `urls` table.
- `prepare("INSERT INTO urls (url) VALUES (:url)")` prevents SQL injection by using a placeholder (`:url`). The `execute()` function binds `$url` to this placeholder and executes the insertion.

4.**HTML Structure and Displaying URLs**:

```
<form class="card p-2 margin" method="POST" action="index.php">
<div class="input-group">
<input type="text" name="url" class="form-control" placeholder="
<div class="input-group-append">
<button type="submit" name="submit" class="btn btn-success">Sho
</div>
</div>
</form>
```

This form allows users to enter a URL to be shortened. The form submits data to the current page (
`index.php`), and the `name="url"` attribute ensures it matches `$_POST['url']` in PHP.

5.
Displaying URLs and Short Links in a Table:

```
<table class="table mt-4">
<thead>
<tr>
```

```
<th scope="col">Long_url</th>
<th scope="col">Short_url</th>
<th scope="col">Clicks</th>
</tr>
</thead>
<tbody>
<?php foreach ($rows as $row) : ?>
<tr>
<th scope="row"><?php echo $row->url; ?></th>
<td><a href="http://localhost/url_shortner/u/index.php?id=<?php
http://localhost/url_shortner/u/index.php?id=<?php echo $row->i
<td><?php echo $row->clicks; ?></td>
</tr>
<?php endforeach; ?>
</tbody>
</table>
```

- This table displays each URL from the `urls` table, showing the original (long) URL, the short link, and the number of clicks.
- The `href` attribute in the `<a>` tag generates a short URL with the `id` of each entry, which is sent to `u/index.php` for redirection and click tracking.

6.**JavaScript for Page Refresh**:

```
<script>
$(document).ready(function() {
$("#refresh").click(function() {
setInterval(function() {
$("body").load('index.php')
}, 5000);
});
});
</script>
```

- This jQuery script sets an interval to reload the page every 5 seconds. When the `#refresh` div is clicked, it begins loading `index.php` into the `<body>`, allowing the clicks count to be updated in real time.

Redirection and Click Tracking (`u/index.php`)

1.
Connecting to the Database:

```
<?php require "../config.php"; ?>
```

2.**URL Redirection and Click Tracking**:

```
if (isset($_GET['id'])) {
$id = $_GET['id'];
$select = $conn->query("SELECT * FROM urls WHERE id='$id'");
$select->execute();
$data = $select->fetch(PDO::FETCH_OBJ);
```

- The `if (isset($_GET['id']))` checks if an `id` parameter was passed in the URL.
- The query `"SELECT * FROM urls WHERE id='$id'"` fetches the URL data for the given `id`.
- `$data = $select->fetch(PDO::FETCH_OBJ);` retrieves the result as an object, allowing you to access `$data->url`.

3.**Updating Click Count**:

```
$clicks = $data->clicks + 1;
$update = $conn->prepare("UPDATE urls SET clicks = :clicks WH
ERE id = '$id'");
$update->execute([':clicks' => $clicks]);
```

- `$clicks = $data->clicks + 1;` increments the current click count by 1.
- `UPDATE urls SET clicks = :clicks WHERE id = '$id'` updates the click count for this specific URL in the database.

4.**Redirecting to the Original URL**:

```
header("location: " . $data->url);
```

Finally, `header("location: " . $data->url);` redirects the user to the original long URL, using the URL stored in `$data->url`.