**Step 7: Implementing the Post Model**

* It represents the data and the business logic related to blog posts in your application. It's responsible for fetching, inserting, updating, and deleting post data from the database.

**Basic Structure**:

* Create the **Post** model class in the **app/models** directory
* This class will contain methods that correspond to the various operations you can perform on a blog post.

**app/models/Post.php**

<?php

namespace app\models;

// Include the bootstrap file for global setup

require\_once BASE\_DIR . '/bootstrap.php';

// Define class for managing posts

class Post {

    private $db; // Database connection object

    // Initialize connection in constructor

    public function \_\_construct() {

        $this->db = new \Database; // Create a Database object for connection

    }

   /\*\*

     \* Create a new post with title and content.

     \*

     \* @param string $title The title of the post.

     \* @param string $content The content of the post.

     \*/

    public function createPost($title, $content) {

        // Prepare an SQL statement to insert a new post into the 'posts' table

        $stmt = $this->db->conn->prepare('INSERT INTO posts (title, content) VALUES (?, ?)');

        // Execute the prepared statement with the provided title and content

        $stmt->execute([$title, $content]);

    }

    /\*\*

     \* Retrieve a single post by its ID.

     \*

     \* @param int $id The ID of the post to retrieve.

     \* @return object The fetched post as an object.

     \*/

    public function getPost($id) {

        // Prepare an SQL statement to select a post from the 'posts' table by its ID

        $stmt = $this->db->conn->prepare('SELECT posts.\* FROM posts WHERE posts.id = :id');

        // Execute the prepared statement with the provided ID

        $stmt->execute(['id' => $id]);

        // Return the fetched post as an object

        return $stmt->fetch(\PDO::FETCH\_OBJ);

    }

    /\*\*

     \* Retrieve all posts, ordered by ID in descending order.

     \*

     \* @return array An array of post objects.

     \*/

    public function getPosts() {

        // Prepare an SQL statement to select all posts from the 'posts' table, ordered by ID

        $stmt = $this->db->conn->prepare('SELECT posts.\* FROM posts ORDER BY posts.id DESC');

        // Execute the SQL statement

        $stmt->execute();

        // Return all fetched posts as an array of objects

        return $stmt->fetchAll(\PDO::FETCH\_OBJ);

    }

    // Update an existing post with new title and content

    public function updatePost($id, $title, $content) {

        $sql = 'UPDATE posts SET title = :title, content = :content WHERE id = :id';

        $stmt = $this->db->conn->prepare($sql);

        try {

            $stmt->execute(['title' => $title, 'content' => $content, 'id' => $id]);

        } catch (PDOException $e) {

            // Log the error

            error\_log("Database error: " . $e->getMessage() . " in " . $e->getFile() . ":" . $e->getLine());

            // Optionally, display a user-friendly error message

            echo "<p>An error occurred while updating the post. Please try again later.</p>";

        }

    }

    public function deletePost($id) {

        $stmt = $this->db->conn->prepare('DELETE FROM posts WHERE id = ?');

        $stmt->execute([$id]);

    }

}

**Difference Between $db and $conn**

In the context of your **Post** class, **$db** and **$conn** refer to different aspects of handling a database connection:

1. **$db**:
   * In your **Post** class, **$db** is a private property that holds an instance of the **Database** class.
   * When you create a new **Post** object, the constructor (**\_\_construct()**) instantiates the **Database** class and assigns the resulting object to **$db**.
   * This property (**$db**) is used within the **Post** class to access the functionality provided by the **Database** class, such as establishing a connection to the database and executing SQL queries.
2. **$conn**
   * **$conn** typically refers to the actual database connection object within the **Database** class.
   * In a common implementation of a **Database** class, **$conn** would be a property that holds the PDO (PHP Data Objects) connection or another type of database connection resource.
   * This **$conn** property is usually used internally within the **Database** class to perform database operations like querying, fetching data, etc.

We will then create an ***AdminController.php*** which will contain methods for the crud functionality for our app.

***app/controllers/AdminController.php***

<?php

namespace app\controllers;

require\_once BASE\_DIR . 'app/models/Post.php';

class AdminController {

    private $model;

    public function \_\_construct() {

        $this->model = new \app\models\Post;

    }

    // Read: Display a list of posts

    public function index() {

        // Fetch all posts without pagination

        $posts = $this->model->getPosts();

        // Load the view for displaying posts

        require BASE\_DIR . '/app/views/admin/posts/index.php';

    }

    public function create() {

        // Load the view for creating a new post

        require BASE\_DIR . '/app/views/admin/posts/create.php';

    }

    // Read: Display a single post

    public function show($id) {

        // Fetch the post with the given ID

        $post = $this->model->getPost($id);

        if ($post === null) {

            // Post not found, display 404 page

            require BASE\_DIR . '/app/views/404.php';

            return;

        }

        // Load the view for displaying the single post

        require BASE\_DIR . '/app/views/admin/posts/show.php';

    }

    // Create: Store a new post

      public function store($postData) {

        // Retrieve form data

        $title = $postData['title'];

        $content = $postData['content'];

        $category\_id = $postData['category\_id'];

        try {

            $this->model->createPost($title, $content);

            // Redirect to the posts page after successful creation

            header('Location: /admin/posts');

            $\_SESSION['message'] = 'Post created successfully!.';

            exit;

        } catch (Exception $e) {

            // If there's a database error, display it

            echo 'Error creating post: ' . $e->getMessage();

            exit;

        }

    }

    public function edit($id) {

        $post = $this->model->getPost($id);

        // Pass post and categories to the view

        require BASE\_DIR . '/app/views/admin/posts/edit.php';

    }

    // Update: Update an existing post

    public function update($postData, $id) {

// Retrieve form data

        $title = $postData['title'];

        $content = $postData['content'];

        try {

            // Call the updatePost method with the image filename

            $this->model->updatePost($id, $title, $content);

            // Redirect to the posts page after successful update

            $\_SESSION['message'] = 'Post updated successfully!.';

              header('Location: /admin/posts');

              exit;

            exit;

        } catch (Exception $e) {

            // If there's a database error, display it

            echo 'Error updating post: ' . $e->getMessage();

            exit;

        }

    }

    // Delete: Remove a post

    public function delete($id) {

        $this->model->deletePost($id);

        // Redirect or display success message

        header('Location: /admin/posts');

    }

}

**Create the View Files**

We will now create the views files for our application based on what we have in the ***AdminController.php.***

***'/app/views/admin/posts/index.php';***

<div>

  <!-- Display message -->

    <?php if (isset($\_SESSION['message'])): ?>

        <div id="message" class="<?php echo strpos($\_SESSION['message'], 'error') !== false ? 'message-error' : 'message-success' ?> text-center">

            <?php

                echo $\_SESSION['message'];

                unset($\_SESSION['message']);

            ?>

        </div>

    <?php endif; ?>

      <div style="text-align: right;">

        <!-- Create Post Button -->

        <a href="/admin/posts/create">Create New Post</a>

<a href="/admin/categories/create">Create Categories</a>

    </div>

    <?php foreach ($posts as $post): ?>

        <div style="margin-bottom: 20px; margin-top: 20px;">

            <div>

                <h2><?php echo htmlspecialchars($post->title, ENT\_QUOTES); ?></h2>

                <p><?php echo htmlspecialchars($post->content, ENT\_QUOTES); ?></p>

                <a href="/admin/posts/show/<?php echo $post->id; ?>">View</a>

                <a href="/admin/posts/edit/<?php echo $post->id; ?>">Edit</a>

                <!-- Delete Post -->

                <form action="/admin/posts/delete/<?php echo $post->id; ?>" method="POST" style="display: inline;">

                    <input type="hidden" name="\_method" value="DELETE">

                    <button type="submit">Delete</button>

                </form>

            </div>

        </div>

    <?php endforeach; ?>

</div>

<style>

.message-success {

color: green;

/\* other styling \*/

}

.message-error {

    color: red;

    background-color: #ffd6d6;

    padding: 10px;

    border: 1px solid red;

    margin-bottom: 20px;

}

</style>

***'/app/views/admin/posts/create.php';***

<!-- admin/posts/create.php -->

<div>

    <!-- View all Post Button -->

    <div style="text-align: right; margin-bottom: 20px;">

        <a href="/admin/posts/">View Posts</a>

    </div>

    <h1>Create Post</h1>

    <form method="POST" action="/admin/posts/store">

        <div>

            <label for="title">Title:</label>

            <input type="text" id="title" name="title">

        </div>

        <div>

            <label for="content">Content:</label>

            <textarea id="content" name="content"></textarea>

        </div>

        <button type="submit">Create Post</button>

    </form>

</div>

***'/app/views/admin/posts/show.php';***

<div>

    <!-- View all Post Button -->

    <div style="text-align: right; margin-bottom: 20px;">

        <a href="/admin/posts/">View Posts</a>

    </div>

    <h2><?php echo htmlspecialchars($post->title, ENT\_QUOTES); ?></h2>

    <p><?php echo htmlspecialchars($post->content, ENT\_QUOTES); ?></p>

</div>

***'/app/views/admin/posts/edit.php';***

<div>

    <?php if ($post === null): ?>

        <div>Post not found</div>

        <?php return; ?>

        <?php endif; ?>

    <form method="POST" action="/admin/posts/update/<?php echo $post->id ?>" style="margin-bottom: 20px;">

        <input type="hidden" name="id" value="<?php echo $post->id ?>">

        <div style="margin-bottom: 20px;">

            <label for="title">Title</label>

            <input type="text" id="title" name="title" value="<?php echo htmlspecialchars($post->title, ENT\_QUOTES); ?>">

        </div>

        <div style="margin-bottom: 20px;">

            <label for="content">Content</label>

            <textarea id="content" name="content"><?php echo htmlspecialchars($post->content, ENT\_QUOTES); ?></textarea>

        </div>

        <button type="submit">Update Post</button>

    </form>

</div>

**UpdateThe Routes**

$routes = [

    'GET' => [

  // Frontend Routes

        '' => ['controller' => '\app\controllers\PostController', 'method' => 'index'],

        'posts/show/([0-9]+)' => ['controller' => 'app\controllers\PostController', 'method' => 'show'],

        // Admin Posts Routes

        'admin/posts' => ['controller' => '\app\controllers\AdminController', 'method' => 'index'],

        'admin/posts/create' => ['controller' => '\app\controllers\AdminController', 'method' => 'create'],

        'admin/posts/show/([0-9]+)' => ['controller' => '\app\controllers\AdminController', 'method' => 'show'],

        'admin/posts/edit/([0-9]+)' => ['controller' => '\app\controllers\AdminController', 'method' => 'edit'],

'POST' => [

 // Admin Post Routes

        'admin/posts/store' => ['controller' => '\app\controllers\AdminController', 'method' => 'store'],

        'admin/posts/update/([0-9]+)' => ['controller' => '\app\controllers\AdminController', 'method' => 'update'],

        'admin/posts/delete/([0-9]+)' => ['controller' => '\app\controllers\AdminController', 'method' => 'delete'],

    ]

];