{PHPhrame}

Framework Usage Documenatation

Oni Adesola Ayomide & Mama Meshac Eberenna

Introduction

Welcome to PHPhrame, a lightweight PHP framework designed to simplify web development. This usage documentation will guide you through setup, features, and best practices for using the framework effectively.

Database Setup

Configuring Your Database

Configure your database connection in the `app/core/config.php` file:

```
define('DBNAME', 'phphrame_db');
define('DBHOST', 'localhost');
define('DBUSER', 'root');
define('DBPASS', '');
define('DBDRIVER', '');
```

Setting the ROOT

Ensure the `ROOT` is set to the public folder:

```
define('ROOT', 'http://localhost/phphrame/public');
```

Passing Data from Controller

Controller Code

Controllers pass data to views using the `$data` array:

```
// Controller
public function index()
{
    $data['title'] = "Home";
    $this->view('home', $data);
}
```

Using Data in Views

Use the passed data in your views:

```
<title><?=$title ?? 'Unknown'?></title>
```

Global Variables

Defining Globals

Global variables are defined in the configuration file:

```
define('APP_NAME', "PHPhrame");
define('APP_DESC', "Best Framework on the planet!!!");
define('DEBUG', true);
```

Using Global Variables

Use them in your views:

```
<title><?=$title ?? 'Unknown'?> | <?=APP_NAME?></title>
```

Linking CSS & JavaScript

Example

Assets should be linked using the `ROOT` constant:

```
<link rel="stylesheet" href="<?=ROOT?>/assets/css/styles.css">
<script src="<?=ROOT?>/assets/js/bootstrap.bundle.min.js"></script>
```

Handling Images

Displaying Images

Display images directly or use the `Image` class for advanced handling:

```
<img src="<?=ROOT?>/assets/images/sample.jpg" alt="Sample image">
```

Using the `Image` Class

```
<?php $image = new \Model\Image; ?>
<img src="<?=get_image($image->getThumbnail('assets/images/image3.jpg',
width, height))?>">
```

Routing & Forms

Route links using the `ROOT` constant:

```
<a href="<?=ROOT?>">Home</a>
<a href="<?=ROOT?>/photos">Photos</a>
```

Session Management

Manage user sessions using the `Session` class:

```
<?php $ses = new \Core\Session; ?>
<?php if($ses->is_logged_in()): ?>
    <li>Hello, <?=$ses->user('username')?></li>
<?php else: ?>
    <li><a href="<?=ROOT?>/login">Login</a></li>
<?php endif; ?>
```

Database Migration

Example

Create tables and insert data programmatically:

```
$this->addColumn('id int(11) NOT NULL AUTO_INCREMENT');
$this->addColumn('username varchar(30) NULL');
$this->addPrimaryKey('id');
$this->createTable('users');
```

Models

Allowed Columns

```
protected $allowedColumns = [
    'username',
    'email',
    'role',
    'password',
    'date',
    'date_updated',
    'date_created',
];
```

Validation Rules

- `required`
- `alpha`
- `email`
- `numeric`
- `unique`
- `symbol`
- `longer_than_8_chars`
- `alpha_numeric_symbol`

Authentication

Example Login Controller

```
namespace Controller;

defined('ROOTPATH') OR exit('Access Denied!');

class Login
{
    use MainController;

    public function index()
    {
        $data['title'] = "Login";

        $data['user'] = new \Model\User;
        $req = new \Core\Request;
        if($req->posted())
        {
            $data['user']->login($_POST);
        }

        $this->view('login', $data);
    }
}
```

File Management

Example Upload Controller

```
namespace Controller;

class Upload
{
    use MainController;

    public function index()
    {
        $data['title'] = "Upload Photo";

        $req = new \Core\Request;
        $photo = new \Model\Photo;

        if($req->posted())
        {
            $post_data = $req->post();
            $files = $req->files();
            // Additional code here...
        }
```

```
        $this->view('upload', $data);
    }
}
```

User Messages

Sending Messages to Users

Use the `message()` function to pass messages to the user:

```
public function signup($data)
{
    if($this->validate($data))
    {
        $this->insert($data);
        message("Your account was created! Please login to continue.");
        redirect('login');
    }
}
```

To display the message in the view:

```
<?php if(message()): ?>
    <div><?=message('', true)?></div>
<?php endif; ?>
```

File Uploads

Managing File Uploads

Uploads can be managed within a single controller and page. Example:

Controller Code:

```
namespace Controller;

defined('ROOTPATH') OR exit('Access Denied!');
use \Model\Photo;

/**
 * Upload class
 */
class Upload
{
        use MainController;

    public function index()
        {
                $data['title'] = "Upload photo";
                $data['mode'] = 'new';
```

```
                //index backend code

                $data['photo'] = $photo;
                $this->view('upload', $data);
        }

        public function edit($id = null)
        {
                $data['title'] = "Edit photo";
                $data['mode'] = 'edit';

                //edit backend code

                $data['photo'] = $photo;

                $this->view('upload', $data);
        }

        public function delete($id = null)
        {
                $data['title'] = "Delete photo";
                $data['mode'] = 'delete';

                //delete backend code

                $data['photo'] = $photo;

                $this->view('upload', $data);
        }
}
```

View File:

```
<?php $this->view('includes/header', $data); ?>

<div class="mx-auto col-md-4 bg-light shadow m-4 p-4 border">
    <h3><?=$title?></h3>

    <?php if($mode == 'new' || (($mode == 'edit' || $mode == 'delete') &&
$row)):?>

        <?php if($mode == 'delete'):?>
            <div class="alert alert-danger text-center">Are you sure you want
to delete this image?!</div>
        <?php endif?>

        <form method="post" enctype="multipart/form-data">
            <input class="form-control my-3" value="<?=old_value('title',
$row->title ?? '')?>" name="title" placeholder="Image title">
            <div><small class="text-danger"><?=$photo-
>getError('title')?></small></div>

            <label class="d-block">
                <?php if($mode != 'delete'):?>
                    <div class="input-group mb-3">
```

```
                        <input onchange="display_image(this.files[0])"
name="image" type="file" class="form-control" id="inputGroupFile02">
                        <label class="input-group-text"
for="inputGroupFile02">Select Image</label>
                    </div>
                    <div><small class="text-danger"><?=$photo-
>getError('image')?></small></div>
                <?php endif?>
                <div>
                    <img src="<?=get_image($row->image ?? '')?>" class="js-
image-preview img-thumbnail mx-auto d-block" style="cursor: pointer;">
                </div>
            </label>
            <?php if($mode == 'delete'):?>
                <button class="btn btn-danger my-3">Delete</button>
            <?php else:?>
                <button class="btn btn-primary my-3">Save</button>
            <?php endif?>
        </form>
    <?php else:?>
        <div class="p-2 text-center">Image not found!</div>
    <?php endif?>
</div>

<script type="text/javascript">
    function display_image(file) {
        let allowed = ['image/jpeg','image/png','image/webp'];
        if(!allowed.includes(file.type)) {
            alert("The only files supported are: " +
allowed.toString().replaceAll("image/", ""));
            return;
        }
        document.querySelector(".js-image-preview").src =
URL.createObjectURL(file);
    }
</script>
<?php $this->view('includes/footer', $data); ?>
```

Fetching Data

Fetching Data from the Database

For instance, fetching multiple images from the database:

```
// Photos Controller
namespace Controller;

defined('ROOTPATH') OR exit('Access Denied!');
use \Model\Photo;
use \Model\Image;
use \Core\Pager;

/**
 * Photos class
```

```
 */
class Photos
{
        use MainController;

        public function index()
        {
                $photo = new Photo;

                $limit = 24;
                $pager = new Pager($limit);
                $offset = $pager->offset;

                $photo->limit = $limit;
                $photo->offset = $offset;

                $data['rows'] = $photo->findAll();
                $data['image'] = new Image;
                $data['pager'] = $pager;

                $this->view('photos', $data);
        }
}
```

Viewing Individual Records

Viewing an Individual Photo

Controller:

```
namespace Controller;

defined('ROOTPATH') OR exit('Access Denied!');
use \Model\Image;

/**
 * Photo class
 */
class Photo
{
    use MainController;

    public function index($id = null)
    {
        $photo = new \Model\Photo;

        $query = "select photos.*, users.username from photos join users on
users.id = photos.user_id where photos.id = :id limit 1";
        $data['row'] = $photo->query($query, ['id'=>$id]);
        if($data['row'])
            $data['row'] = $data['row'][0];

        $data['image'] = new Image;
        $data['ses'] = new \Core\Session;
```

```
        $this->view('photo', $data);
    }
}
```

View File:

```
<?php $this->view('includes/header', $data); ?>

<div class="p-4 text-center"><h3>Single Photo View</h3></div>
<div class="row p-4 justify-content-center">

    <?php if(!empty($row)):?>
        <div class="col-sm-12 text-center bg-light">
                <div class="card-header"><h4><?=esc($row->title)?></h4></div>
                <div class="card-header"><a href="<?=ROOT?>/profile/<?=$row-
>user_id?>"><i>By: <?=esc($row->username)?></i></a></div>
                <img src="<?=get_image($row->image)?>" class="img-thumbnail"
style="object-fit: cover;width:250px;height:250px;">
            <br>

            <?php if($ses->is_logged_in() && $ses->user('id') == $row-
>user_id):?>
                <a href="<?=ROOT?>/upload/edit/<?=$row->id?>">Edit Image</a>
                |
                <a href="<?=ROOT?>/upload/delete/<?=$row->id?>">Delete
Image</a>
            <?php endif?>
        </div>
    <?php else:?>
        <div class="p-2 text-center">Image not found!</div>
    <?php endif?>
</div>

<?php $this->view('includes/footer', $data); ?>
```

Search Functionality

Usage of search functionality:
To enable a search feature in the application, implement the following:

```
<?php
namespace Controller;

defined('ROOTPATH') OR exit('Access Denied!');
use \Model\Photo;
use \Model\Image;
use \Core\Pager;

/**
 * Search class
 */
class Search
```

```
{
    use MainController;

    public function index()
    {
        $photo = new Photo;

        $limit = 24;
        $pager = new Pager($limit);
        $offset = $pager->offset;

        $photo->limit = $limit;
        $photo->offset = $offset;

        $find = $_GET['find'] ?? '';
        if (!empty($find)) {
            $find = "%$find%";
            $query = "SELECT * FROM photos WHERE title LIKE :find LIMIT
$limit OFFSET $offset";
            $data['rows'] = $photo->query($query, ['find' => $find]);
        }
        $data['image'] = new Image;
        $data['pager'] = $pager;

        $this->view('search', $data);
    }
}
```

Search Page View File:

```
<?php $this->view('includes/header', $data); ?>

<div class="p-4"><h3>Search</h3></div>
<div class="px-4">Searching for: <?=!empty($_GET['find']) ? $_GET['find'] :
'empty string'?></div>
<div class="row p-4 justify-content-center">

    <?php if (!empty($rows)): ?>
        <?php foreach ($rows as $row): ?>
            <?php $this->view('includes/photo-card', ['row' => $row, 'image'
=> $image]) ?>
        <?php endforeach ?>
    <?php else: ?>
        <div class="p-2 text-center">No images found!</div>
    <?php endif ?>

</div>
<div>
    <?php $pager->display() ?>
</div>
<?php $this->view('includes/footer', $data); ?>
```

User Profile

How the user profile can be viewed:

```
<?php
namespace Controller;

defined('ROOTPATH') OR exit('Access Denied!');

/**
 * Profile class
 */
class Profile
{
    use MainController;

    public function index($id = null)
    {
        $user = new \Model\User;
        $data['ses'] = $ses = new \Core\Session;

        $id = $id ?? $ses->user('id');
        $data['row'] = $user->first(['id' => $id]);

        $this->view('profile', $data);
    }
}
```

Profile Page View File:

```
<?php $this->view('includes/header', $data); ?>

<div class="p-4 text-center"><h3>User Profile</h3></div>
<div class="row p-4 justify-content-center">

    <?php if (!empty($row)): ?>
        <div class="col-md-6 text-center bg-light">

            <table class="table table-striped table-hover text-start">
                <tr><th>#</th><td><?=$row->id?></td></tr>
                <tr><th>Username</th><td><?=esc($row->username)?></td></tr>
                <tr><th>Email</th><td><?=esc($row->email)?></td></tr>
                <tr><th>Role</th><td><?=esc($row->role)?></td></tr>
                <tr><th>Date Created</th><td><?=get_date($row-
>date_created)?></td></tr>
                <tr><th>Date Updated</th><td><?=get_date($row-
>date_updated)?></td></tr>
            </table>

            <br>

            <?php if ($ses->is_logged_in() && $ses->user('id') == $row->id):
?>
                <a href="<?=ROOT?>/profile/edit/<?=$row->id?>">
                    Edit Profile
                </a>
                |
                <a href="<?=ROOT?>/profile/delete/<?=$row->id?>">
```

```
                    Delete Profile
                </a>
            <?php endif ?>
        </div>
    <?php else: ?>
        <div class="p-2 text-center">Profile not found!</div>
    <?php endif ?>

</div>
<?php $this->view('includes/footer', $data); ?>
```

Conclusion

In conclusion, **PHPhrame** is a versatile, lightweight framework designed to streamline your PHP development process, offering flexibility and efficiency in building modern web applications. This documentation has walked you through its key features, from database setup to routing, session management, and advanced functionalities like file management, search, and user authentication.

The framework provides powerful tools, such as:

- **Effortless Routing**: Simplify navigation with ROOT-based routing.
- **Dynamic Data Management**: Seamless data passing between controllers and views.
- **Robust Authentication and Session Handling**: Manage users securely and efficiently.
- **File Upload and Image Handling**: Add and display media with minimal effort.
- **Global Configuration**: Easily manage app-wide settings through a single file.
- **Database Migrations and Models**: Reduce redundancy and ensure data integrity.
- **Search and Profile Features**: Provide end-users with enhanced interactivity and functionality.

By following this guide, you can unlock the full potential of PHPhrame to create scalable and maintainable web applications. Whether you're working on a simple project or a complex enterprise-level application, PHPhrame offers the necessary tools to keep your workflow productive.

For any additional queries, support is readily available via our documentation or by contacting our team at **phphramesupport@phphrame.com.ng**. We are committed to providing a smooth development experience.

Thank you for choosing PHPhrame as your framework of choice. We look forward to seeing the innovative applications you build with it!

Happy Coding!
The PHPhrame Development Team