

Создание контактной формы на PHP с отправкой данных на электронную почту через PHPMailer

Акентьев Данила Денисович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данной работе рассматривается разработка контактной формы на PHP с использованием библиотеки PHPMailer для отправки сообщений на электронную почту через SMTP-сервер. Описание включает в себя создание формы для сбора данных от пользователей, обработку этих данных с применением фильтрации и валидации, а также настройку отправки сообщений с использованием безопасного протокола SMTP через сервер Gmail. Использование PHPMailer позволяет реализовать отправку сообщений в формате HTML и корректно обрабатывать кодировку, что обеспечивает удобство и безопасность работы с электронной почтой. В процессе разработки были учтены важные аспекты безопасности, такие как защита от инъекций и использование пароля приложения для подключения к SMTP-серверу. Работа демонстрирует использование современных подходов для создания безопасных и функциональных контактных форм на веб-сайтах.

Ключевые слова: PHP, контактная форма, отправка данных, безопасность, электронная почта, SMTP, защита от атак, криптография, валидация данных, PHPMailer.

Creating a contact form in PHP with sending data via email via PHPMailer

Akentev Danila Denisovich

Sholom-Aleichem Priamursky State University

Student

Abstract

This paper discusses the development of a contact form in PHP using the PHPMailer library to send messages to email via an SMTP server. The description includes creating a form to collect data from users, processing this data using filtering and validation, and configuring the sending of messages using the secure SMTP protocol via the Gmail server. Using PHPMailer allows you to send messages in HTML format and correctly process the encoding, which ensures the convenience and security of working with email. During the development process, important security aspects were taken into account, such as injection protection and the use of the application password to connect to the SMTP server. The work demonstrates the use of modern approaches to create secure and functional contact forms on websites.

Keywords: PHP, contact form, sending data, security, email, SMTP, attack protection, cryptography, data validation, PHPMailer.

1 Введение

1.1 Актуальность

Создание контактных форм для сбора данных пользователей является важной частью разработки современных веб-приложений. Одним из наиболее распространенных способов обработки таких форм является отправка данных на электронную почту, что требует надежного и безопасного решения. В данной работе актуальность заключается в использовании библиотеки PHPMailer для отправки сообщений через SMTP-серверы, такие как Gmail. Эта библиотека предоставляет широкие возможности для безопасной отправки email-сообщений с поддержкой HTML-формата и кодировок, а также предоставляет механизмы защиты при подключении к SMTP-серверу. В условиях повышенных требований безопасности, включая необходимость использования паролей приложений и шифрования данных, использование PHPMailer становится важным инструментом для реализации контактных форм, которые обеспечивают высокий уровень защиты и функциональности.

1.2 Обзор исследований

Обзор исследований по использованию PHPMailer и SMTP в различных областях показал разнообразие подходов и возможностей применения этих технологий. Так, в работе И.М. Суданы, Н.Кудуса и С.Э. Прасети, авторы исследуют использование PHPMailer для автоматической отправки уведомлений в системах электронного обучения. В этом контексте библиотека PHPMailer использовалась для отправки уведомлений о выполнении различных действий, таких как результаты тестов и прогресс в обучении, как для учеников, так и для преподавателей. Это исследование демонстрирует, как интеграция PHPMailer может улучшить взаимодействие между участниками образовательного процесса, обеспечивая оперативное информирование через электронную почту [1].

В работе А.А. Колосова анализируются методы защиты электронной почты с помощью асимметричной криптографии. Автор исследует возможности применения этих технологий для создания защищенных систем обмена сообщениями, что особенно важно при работе с конфиденциальной информацией. В данном исследовании особое внимание уделено использованию сертификатов открытых ключей для обеспечения безопасности почтовых систем, что подтверждает необходимость усиленной защиты при использовании электронных средств коммуникации, в том числе через протоколы типа SMTP [2].

Далее, в статье А.А. Шайдурова подробно расписан процесс разработки e-mail рассылок для образовательного учреждения. В этом контексте автор использует PHPMailer для обеспечения отправки массовых рассылок и уведомлений, что позволяет оперативно информировать студентов и преподавателей о новостях и событиях в университете. Работа также

акцентирует внимание на оптимизации процесса рассылки и улучшении надежности, что является важным аспектом при разработке подобных сервисов [3].

В статье М. Е. Кочитова приводится простой и эффективный способ использования библиотеки RHPMailer для отправки сообщений через SMTP-серверы. Автор приводит примеры настройки и интеграции этой библиотеки в веб-приложения, акцентируя внимание на её удобстве и функциональности. В работе детально описан процесс отправки сообщений с использованием RHPMailer и показаны основные преимущества, такие как поддержка HTML-формата, настройка SMTP и удобство в использовании [4].

Наконец, в статье И. Тетерина рассматривается использование электронной почты в качестве инструмента автоматизации процессов на веб-сайтах. Автор акцентирует внимание на применении e-mail рассылок для уведомлений о различных событиях на сайте, таких как обновления контента или важные административные задачи. В этом контексте RHPMailer используется для автоматизации отправки сообщений и уведомлений пользователям, что помогает оптимизировать рабочие процессы и улучшить взаимодействие с клиентами [5].

1.3 Цель исследования

Целью данной работы является разработка безопасной контактной формы на языке программирования PHP с отправкой данных на электронную почту, а также анализ и реализация методов защиты данных от внешних угроз. Исследование направлено на повышение уровня безопасности веб-приложений при обработке пользовательских данных и отправке информации через электронную почту.

2 Материалы и методы

Для разработки контактной формы использован язык программирования PHP, а также стандартные инструменты для обработки HTTP-запросов, такие как `$ _POST`. Для обеспечения безопасности использованы методы валидации и фильтрации пользовательских данных с применением встроенных функций PHP, таких как `filter_input()` и `htmlspecialchars()`. Для отправки сообщений использован механизм SMTP с использованием библиотеки RHPMailer, что позволяет повысить безопасность при отправке почты через защищенное соединение. Методы защиты от атак, таких как CSRF и XSS, реализованы с использованием токенов безопасности и правильной обработки ввода данных. Анализ реализованных методов безопасности выполнен на основе современных стандартов защиты веб-приложений.

3 Результаты и обсуждения

Структура данного проекта выглядит следующим образом (рис.1).

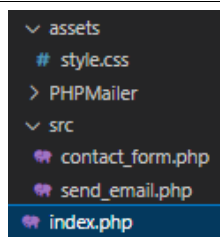


Рисунок 1. Структура проекта

```
<?php
// index.php
include 'src/contact_form.php';
?>

<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Контактная форма</title>
  <link rel="stylesheet" href="assets/style.css">
</head>
<body>
  <div class="form-container">
    <?php echo showContactForm(); ?>
  </div>
</body>
</html>
```

Рисунок 2. Index.php

Этот код представляет собой веб-страницу, на которой отображается контактная форма (рис.2).

В начале файла подключается внешний PHP-скрипт `contact_form.php`, который содержит функцию `showContactForm()`. Эта функция генерирует HTML-разметку формы, которая включает поля для ввода имени, email и сообщения, а также кнопку отправки.

HTML-разметка страницы начинается с объявления типа документа (HTML5) и языка страницы — русский. Внутри тега `<head>` задаются метатеги, такие как кодировка UTF-8 и мета-тег для адаптивности на мобильных устройствах, а также ссылка на внешний файл стилей (`style.css`), который будет отвечать за оформление страницы.

Внутри тега `<body>` создается контейнер с классом `form-container`, который используется для размещения формы на странице. Функция `showContactForm()` вызывается внутри этого контейнера для отображения сгенерированного HTML-кода формы.

Таким образом, код организует отображение контактной формы на веб-странице, а также подключает необходимые стили для её оформления.

```
/* style.css */
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  background-color: #f4f4f4;
}

.form-container {
  max-width: 500px;
  margin: 50px auto;
  padding: 20px;
  background-color: #fff;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

form label {
  font-weight: bold;
}

input[type="text"],
input[type="email"],
textarea {
  width: 100%;
  padding: 10px;
  margin: 5px 0;
  border: 1px solid #ccc;
  border-radius: 4px;
}

textarea {
  height: 150px;
}

input[type="submit"] {
  background-color: #4caf50;
  color: white;
  border: none;
  padding: 10px 20px;
  cursor: pointer;
  border-radius: 4px;
}

input[type="submit"]:hover {
  background-color: #45a049;
}
```

Рисунок 3. Style.css

В данном файле прописаны внешние стили страницы (рис.3).

```
<?php
// contact_form.php

// Функция для отображения формы
function showContactForm() {
    $output = '<h2>Контактная форма</h2>';
    $output .= '<form action="src/send_email.php" method="POST" enctype="multipart/form-data">
        <label for="name">Имя:</label><br>
        <input type="text" id="name" name="name" required><br><br>

        <label for="email">Email:</label><br>
        <input type="email" id="email" name="email" required><br><br>

        <label for="message">Сообщение:</label><br>
        <textarea id="message" name="message" required></textarea><br><br>

        <input type="submit" value="Отправить">
    </form>';
    return $output;
}
?>
```

Рисунок 4. contact_form.php

Этот код представляет собой PHP-скрипт, который содержит функцию `showContactForm()` (рис.4). Функция генерирует HTML-код для отображения контактной формы. В форме присутствуют поля для ввода имени, email и сообщения, а также кнопка для отправки данных.

При вызове функции создаётся строка, содержащая HTML-разметку. Сначала выводится заголовок формы, затем идет само тело формы с полями для ввода. Каждое поле снабжено соответствующим `label` для улучшения доступности и идентификации каждого поля. Атрибут `required` в полях ввода гарантирует, что пользователь не сможет отправить форму, не заполнив эти поля.

Форма использует метод `POST` для отправки данных на серверный скрипт `send_email.php`, расположенный в папке `src`. Также задан атрибут `enctype="multipart/form-data"`, что указывает на возможность загрузки файлов через форму. В конце формы находится кнопка отправки с текстом "Отправить" (рис.5).

После того как форма будет обработана на сервере, данные будут отправлены на сервер для дальнейшей обработки (отправки email).

Рисунок 5. Внешний вид контактной формы

```

<?php
require '../PHPMailer/src/Exception.php';
require '../PHPMailer/src/SMTP.php';
require '../PHPMailer/src/PHPMailer.php';

use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    // Сбор данных из формы
    $name = filter_input(INPUT_POST, 'name', FILTER_SANITIZE_FULL_SPECIAL_CHARS);
    $email = filter_input(INPUT_POST, 'email', FILTER_SANITIZE_EMAIL);
    $message = filter_input(INPUT_POST, 'message', FILTER_SANITIZE_FULL_SPECIAL_CHARS);

    // Проверка на валидность email
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        die('Неверный email формат');
    }

    // Настройка PHPMailer
    $mail = new PHPMailer(true);

    try {
        // Настройки для отправки через SMTP
        $mail->isSMTP(); // Используем SMTP
        $mail->Host = 'smtp.gmail.com'; // SMTP сервер Google
        $mail->SMTPAuth = true; // Включаем аутентификацию
        $mail->Username = '*****@gmail.com'; // Ваш email
        $mail->Password = '*****'; // Пароль приложения
        $mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS; // Включаем TLS
        $mail->Port = 587; // Порт для TLS

        // Получатель и отправитель
        $mail->setFrom('*****@gmail.com', 'OpenServer'); // Указываем ваш email и имя
        $mail->addAddress('*****@gmail.com', 'Recipient Name'); // Почта получателя

        // Контент письма
        $mail->isHTML(true); // Устанавливаем формат HTML
        $mail->Subject = 'Новое сообщение из контактной формы';
        $mail->Body = "<html><body><h2><p><strong>Имя:</strong> $name</p><p><strong>Email:</strong> $email</p><p><strong>Сообщение:</strong><br>$message</p></body></html>";
        $mail->Subject = "=?UTF-8?<?> . base64_encode('Новое сообщение из контактной формы') . '?='"; // Кодировка Subject в UTF-8

        // Отправка письма
        $mail->send();
        echo 'Сообщение отправлено!';
    } catch (Exception $e) {
        echo "Сообщение не было отправлено. Ошибка: {$mail->ErrorInfo}";
    }
}
?>

```

Рисунок 6. Send_email.php

Данный PHP-скрипт (рис.6) обрабатывает данные контактной формы и отправляет их на указанный адрес электронной почты с помощью библиотеки PHPMailer, используя SMTP-сервер Gmail.

В начале скрипта подключаются необходимые файлы библиотеки PHPMailer, чтобы иметь доступ к классам PHPMailer, Exception и SMTP. Затем начинается основной блок обработки данных формы.

Если форма была отправлена методом POST, скрипт получает данные из полей формы (имя, email и сообщение), используя функцию filter_input(),

которая очищает данные от нежелательных символов для предотвращения атак через инъекции. Валидация email осуществляется с помощью `filter_var()`, и если email не соответствует правильному формату, скрипт завершает выполнение и выводит сообщение об ошибке.

Далее создается объект `PHPMailer`, и настраивается отправка письма через SMTP-сервер Gmail. В настройках указываются:

- Хост для SMTP (`smtp.gmail.com`),
- Использование аутентификации с логином и паролем приложения (пароль генерируется в аккаунте Gmail для безопасного доступа),
- Включение шифрования TLS и установка порта 587.

После этого указывается отправитель (ваш email) и получатель письма, а также формируется тело письма в формате HTML, которое содержит данные, отправленные через форму (имя, email и сообщение).

В строке с темой письма используется кодировка UTF-8 для правильного отображения русских символов в теме письма.

Если письмо успешно отправлено, скрипт выводит сообщение о успешной отправке. Если возникает ошибка, например, при подключении к SMTP-серверу или отправке письма, выводится описание ошибки (рис.7).

Таким образом, скрипт надежно обрабатывает данные формы и отправляет их по почте с использованием безопасного подключения через SMTP.

Так же для корректной работы данного скрипта потребуется установить `PHPMailer` вручную с GitHub [6].



Рисунок 7. Сообщение на почте

4 Выводы

В ходе выполнения работы была разработана контактная форма с возможностью отправки сообщений через электронную почту с использованием библиотеки `PHPMailer` и SMTP-сервера Gmail. В ходе реализации были решены несколько ключевых задач:

Разделение логики и представления: Был использован подход с разделением PHP-скриптов, что позволило организовать структуру проекта. Логика для отображения формы была вынесена в отдельный файл, а HTML-код страницы был размещён в `index.php`. Это улучшило читаемость кода и упрощает его поддержку.

Обработка и валидация данных: Важным этапом было использование функций фильтрации и валидации данных формы с помощью `filter_input()` и `filter_var()`. Это гарантирует, что данные, отправленные пользователем, будут

очищены от нежелательных символов и соответствовать корректному формату (например, email).

Использование PHPMailer: Для отправки email-сообщений была использована библиотека PHPMailer, что позволило обрабатывать отправку сообщений с использованием SMTP, обеспечивая высокую степень безопасности при передаче данных. Библиотека также предоставляет возможности для работы с HTML-сообщениями и кодировками.

SMTP-аутентификация через Gmail: Реализована отправка сообщений через SMTP-сервер Gmail, что позволяет использовать защищённый канал для отправки email-сообщений. Был настроен процесс аутентификации через пароль приложения для повышения безопасности, что соответствует рекомендациям Google по использованию сервисов сторонними приложениями.

Обработка ошибок: В случае возникновения ошибок при отправке сообщений предусмотрена обработка исключений, которая информирует пользователя о причине неудачной отправки сообщения. Это повышает надёжность работы приложения и улучшает взаимодействие с пользователем.

Реализация интерфейса: Контактная форма была стилизована и адаптирована для различных устройств, что обеспечивает удобный пользовательский интерфейс и улучшает взаимодействие с веб-страницей.

Библиографический список

1. Sudana I.M., Qudus N., Prasetyo S.E. Implementation of PHPMailer with SMTP protocol in the development of web-based e-learning prototype // Journal of Physics: Conference Series. 2019. Т. 1321. №. 3.
2. Колосов А.А. Защищенная система обмена электронной почтой на основе сертификатов открытых ключей // Безопасность информационных технологий. 2009. № 1. С. 112-113
3. Шайдуров А.А. Разработка информационной системы "сервис e-mail рассылок" для ПГУ им. Шолом-Алейхема // Постулат. 2019. № 6.
4. Кочитов М.Е. Отправка писем на электронную почту с помощью PHPMailer // Постулат. 2018. №. 7.
5. Тетерин И. Автоматизация веб-проектов через электронную почту // Системный администратор. 2004. № 4 (17). С. 76-79.
6. URL: <https://github.com/PHPMailer/PHPMailer>