



40. メール送信

1. メール送信の仕組み

はじめに

Webアプリやシステム開発でよく使われる「メール送信」。ユーザー登録通知や問い合わせフォームなど、多くの場面で必要になります。この記事では、初心者でも理解しやすいように、メール送信の基本と簡単なコード例、さらにSPF/DKIM/DMARCといった認証技術についても紹介します。

メール送信フロー

メールは送信者から受信者に届くまで、いくつかのサーバーを経由します。

1. 差出人がメールを作成して送信
2. メール送信サーバー (SMTP) がメールを受け取り、宛先のサーバーへ転送
3. メール受信サーバー (POP3/IMAP) がメールを保管し、受信者がアクセスできるようにする
4. 宛先（受信者）がメールクライアントで受信

参考) <https://cmc-japan.co.jp/blog/gmail%E3%81%AEsmtp/>

メールの基本構成

メール送信するには、必須項目があります。

項目	説明	備考
From	差出人	送信者のメールアドレス
To	宛先	受信者のメールアドレス
Subject	件名	メールのタイトル
Body	本文	メールの内容（テキスト/HTML）

メール送信の選択肢

Webアプリでメール送信するには、自己サーバにメールサーバ構築、または `Gmail` をはじめとする外部メール送信サービスを利用します

SMTP

SMTP (Simple Mail Transfer Protocol) はメールを送信するためのプロトコルです。

例：SMTPサーバ x SSL

```
smtp.gmail.com:465
```

メールAPI

メールAPIは `SendGrid` / `Mailgun` / `AWS SES` などのトランザクションメールサービスの HTTP API を利用する方法です。高い到達率・ログ・統計・不正メールブロックなどの機能があり、本運用で推奨されます。

開発擬似サーバ

自己サーバでメールサーバ構築が難しい場合、 MailHog / Papercut / smtp4dev といった擬似サーバも利用できます。ローカルでメール送信を確認でき、開発時に誤送信を防止できます。

SPF/DKIM/DMARC

Webアプリかメール送信する場合、スパムメールや改竄などのセキュリティに注目する必要があります。

SPF

SPF (Sender Policy Framework) はドメインのDNSにこのサーバーから送信するのは正当と記録する設定です。

- 受信側は送信元IPとDNS情報を照合して正当性を判断
- 偽装送信を防ぐ基本の仕組み

DKIM

DKIM (DomainKeys Identified Mail) は、メールに電子署名を付け、メール内容が改ざんされていないか確認する仕組みです。

- 署名は送信サーバーが秘密鍵で付け
- 受信サーバーは公開鍵で検証

DMARC

DMARC (Domain-based Message Authentication, Reporting and Conformance) は、 SPF や DKIM の結果を元に「受信側がどう扱うか」をルール指定します。

- DNSにポリシーを設定し、認証失敗時に拒否や隔離を指示
- レポート機能で不正送信の把握が可能

2. アプリパスワードの生成

Googleアカウント

Googleアカウント にアクセスします。



The screenshot shows the main menu of a Google Account page. The menu items are:

- ホーム (Home)
- 個人情報 (Personal Information)
- データとプライバシー (Data & Privacy)
- セキュリティ (Security) ようこ
- 情報共有と連絡先 (Sharing & Contacts) Google サービスを便利にご利用いた
- お支払いと定期購入 (Payments & Subscriptions)

2段階認証プロセスをオン

セキュリティにアクセスし、「2段階認証プロセス」をオンにします。



The screenshot shows the 'Googleへのログイン' (Login to Google) section of the security settings. It includes:

- パスワード (Password) 前回の変更: 2021/02/27
- 2段階認証プロセス (2-Step Verification) オフ (Off) ON
- アプリ パスワード (App Passwords) 2 個のパスワード

← 2段階認証プロセス

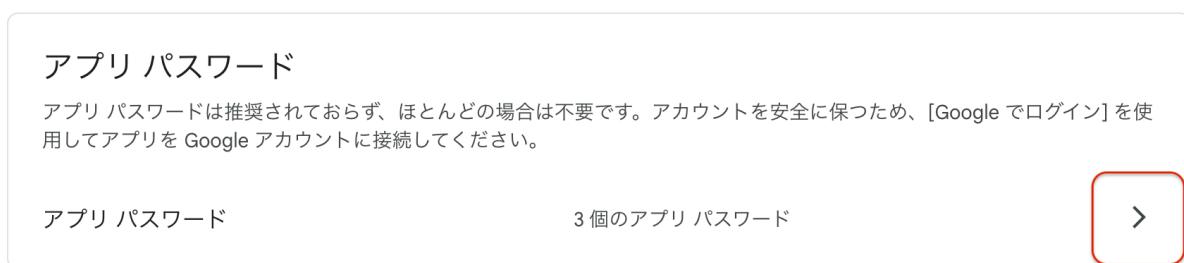
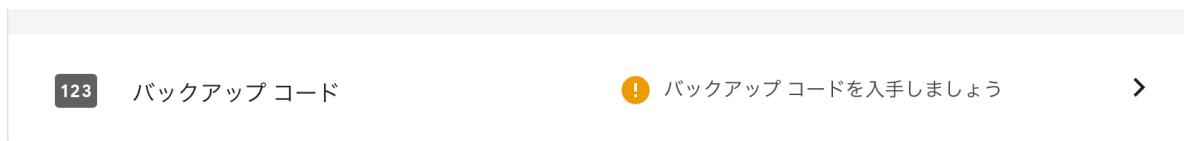
2段階認証プロセスは 2020/10/27より有効になっています

オフにする

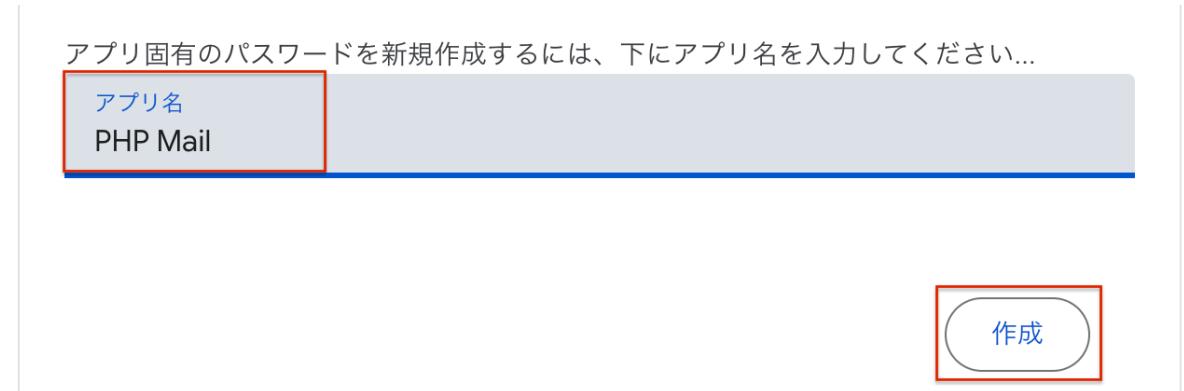
アプリパスワード

パスワード作成

「アプリパスワード」のボタンをクリックします。

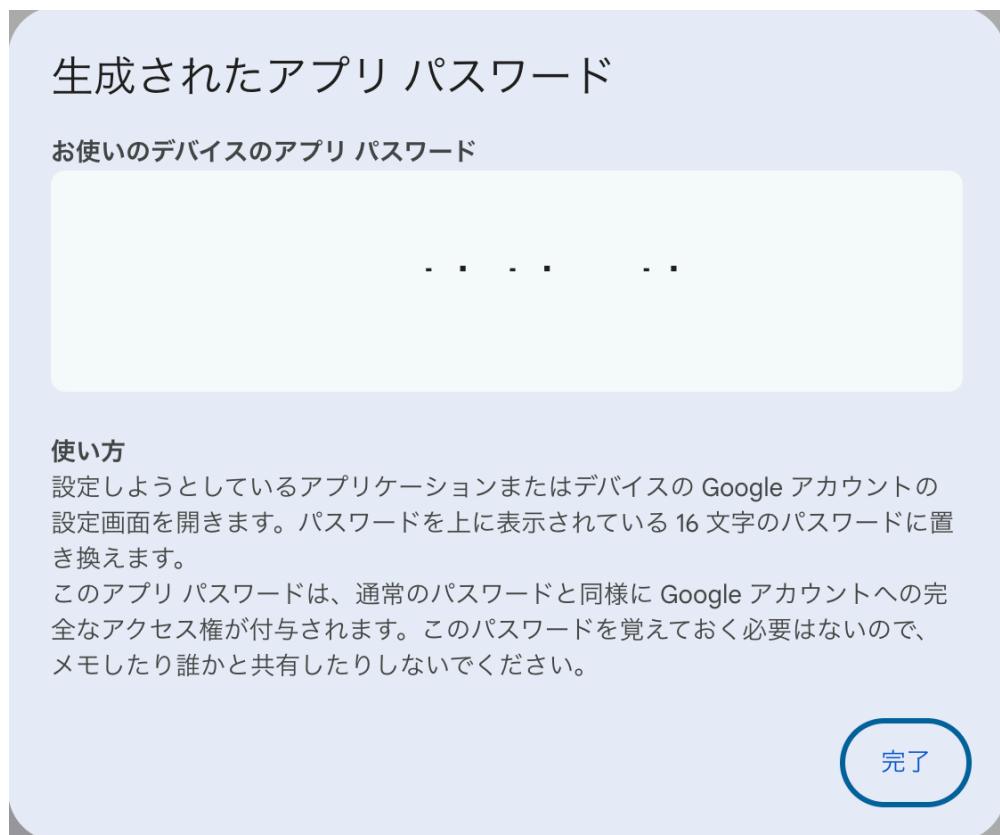


任意の「アプリ名」を入力して作成します。



パスワード確認

パスワードが発行されました。画面を閉じるとパスワードが見れなくなるのでメモしておきます。



3. 問い合わせフォーム1

事前準備

- PHP 8.x 以上推奨
- Composer が使えること
- SMTP 情報（例：Gmail のアプリパスワード, SendGrid / Mailgun / AWS SES の SMTP）
- ローカル確認用に php -S localhost:8000 -t public などで簡易サーバを起動

- Gmail を使う場合は 2段階認証 + アプリパスワードが必要です（通常パスワード不可）

ディレクトリ構成

```
└ contact
  └ templates
    └ contact_mail.html
  └ index.php
  └ Contact.php
  └ result.php
  └ send.php
  └ .env
```

フォーム画面

問い合わせフォーム画面を作成します。

お問い合わせフォーム

お名前

東京 太郎

メールアドレス

tokyo@test.com

問い合わせ内容

問い合わせテスト

送信する

index.php

```
<?php
session_start();

// send.php のセッションデータを読み込み
$name = isset($_SESSION['name']) ? $_SESSION['name'] : '';
$email = isset($_SESSION['email']) ? $_SESSION['email'] : '';
$body = isset($_SESSION['body']) ? $_SESSION['body'] : '';
$error = isset($_SESSION['error']) ? $_SESSION['error'] : '';
?>
<!DOCTYPE html>
<html lang="ja">

<head>
    <meta charset="UTF-8">
    <title>お問い合わせフォーム</title>
    <script src="https://cdn.tailwindcss.com"></script>
</head>

<body class="bg-gray-100 p-8 text-gray-800">
    <!-- ローディングオーバーレイ -->
    <div id="loadingOverlay" class="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center z-50 hidd
```

```
<div class="flex flex-col items-center">
    <!-- スピナー -->
    <svg class="animate-spin h-12 w-12 text-white mb-4" xmlns="http://www.w3.org/2000/svg" fill="none">
        <circle class="opacity-25" cx="12" cy="12" r="10" stroke="currentColor" stroke-width="4"></circle>
        <path class="opacity-75" fill="currentColor" d="M4 12a8 8 0 018-8v8z"></path>
    </svg>
    <p class="text-white text-lg font-semibold">送信中...</p>
</div>
</div>

<!-- Main -->
<div class="max-w-xl mx-auto bg-white p-6 rounded shadow">
    <h1 class="text-2xl text-center font-bold mb-4">お問い合わせフォーム</h1>

    <?php if ($error): ?>
        <div class="bg-red-100 border border-red-400 text-red-700 px-4 py-3 rounded mb-4">
            <?= $error ?>
        </div>
    <?php endif ?>
    <form action="send.php" method="POST" class="space-y-4" onsubmit="handleSubmit(event)">
        <div>
            <label class="block font-semibold mb-1">お名前</label>
            <input type="text" name="name" value="<?= $name ?>" required class="w-full border px-4 py-2 rounded mb-1">
        </div>
        <div>
            <label class="block font-semibold mb-1">メールアドレス</label>
            <input type="email" name="email" value="<?= $email ?>" required class="w-full border px-4 py-2 rounded mb-1">
        </div>
        <div>
            <label class="block font-semibold mb-1">問い合わせ内容</label>
            <textarea name="body" rows="6" required class="w-full border px-4 py-2 rounded"><?= $body ?></textarea>
        </div>
        <div class="flex justify-center">
            <button type="submit"
                    class="bg-blue-600 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded flex items-center justify-content-center">
                <span>送信する</span>
            </button>
        </div>
    </form>
</div>

<script>
    function handleSubmit(event) {
        document.getElementById("loadingOverlay").classList.remove("hidden");
    }
</script>
</body>

</html>
```

完了画面

完了画面を作成します。

result.php

```
<!DOCTYPE html>
<html lang="ja">

<head>
  <meta charset="UTF-8">
  <title>お問い合わせ完了</title>
  <script src="https://cdn.tailwindcss.com"></script>
</head>

<body class="bg-gray-100 p-8 text-gray-800">

  <!-- Main -->
  <div class="max-w-xl mx-auto bg-white p-6 rounded shadow text-center">
    <!-- チェックアイコン -->
    <div class="mx-auto mb-4 w-16 h-16 flex items-center justify-center rounded-full bg-green-100">
      <svg class="w-10 h-10 text-green-600" fill="none" stroke="currentColor" stroke-width="2"
           viewBox="0 0 24 24">
        <path stroke-linecap="round" stroke-linejoin="round"
              d="M5 13l4 4L19 7" />
      </svg>
    </div>

    <h1 class="text-2xl font-bold mb-2">お問い合わせを受け付けました</h1>
    <p class="text-gray-600 mb-6">
      担当者が内容を確認し、必要に応じてご連絡いたします。<br>
      しばらくお待ちください。
    </p>

    <a href="./" class="inline-block bg-blue-600 hover:bg-blue-700 text-white font-semibold px-6 py-2 rounded">
      フォームに戻る
    </a>
  </div>

</body>
</html>
```

リダイレクト

メール送信実行ファイルから、完了画面にリダイレクトできるか確認します。

```
<?php  
session_start();  
  
header("Location: result.php");  
exit;
```



4. 問い合わせフォーム2

phpmailer ライブライリ導入

Composer で `phpmailer` と `phpdotenv` をインストールします。

```
composer require phpmailer/phpmailer vlucas/phpdotenv
```

.env 設定

プロジェクト直下に `.env` を作成し、メール設定をします。

```
MAIL_HOST=smtp.gmail.com  
MAIL_PORT=587
```

```
MAIL_ENCRYPTION=tls
MAIL_USERNAME=Gmailメールアカウント
MAIL_PASSWORD="Gmailキー"
MAIL_FROM_ADDRESS=Gmailメールアドレス
MAIL_FROM_NAME="差出人名前"
```

クラス作成

Contact クラスを作成して、ライブラリを読み込みます。

Contact.php

```
<?php
require '../vendor/autoload.php';

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

class Contact
{
    private $mailer;
    private $template = __DIR__ . "/templates/contact_mail.html";
    private $from_address = "";
    private $from_name = "";
    private $subject = '[お問い合わせ]ご確認のメール';

}

?>
```

メール基本設定

コンストラクタで `.env` を読み込み、`setupMailer()` でメール送信の基本設定をします。

Contact.php

```
class Contact
{
    ...

    public function __construct()
    {
```

```

// .env 読み込み
$dotenv = Dotenv::createImmutable(__DIR__);
$dotenv->load();

// クラスメンバに代入
$this->from_address = $_ENV['MAIL_FROM_ADDRESS'];
$this->from_name = $_ENV['MAIL_FROM_NAME'];
$this->host = $_ENV['MAIL_HOST'];
$this->username = $_ENV['MAIL_USERNAME'];
$this->password = $_ENV['MAIL_PASSWORD'];
$this->encryption = $_ENV['MAIL_ENCRYPTION'];
$this->port = (int) $_ENV['MAIL_PORT'];

$this->setupMailer();
}

private function setupMailer(): void
{
    $this->mailer = new PHPMailer(true);
    $this->mailer->isSMTP();
    $this->mailer->SMTPOAuth = true;
    $this->mailer->Host = $this->host;
    $this->mailer->Username = $this->username;
    $this->mailer->Password = $this->password;
    $this->mailer->SMTPSecure = $this->encryption;
    $this->mailer->Port = $this->port;
    $this->mailer->CharSet = 'UTF-8';
    $this->mailer->Encoding = 'base64';
}
}

```

メールテンプレート

テンプレートファイル (HTML)

メールテンプレートファイル `templates/contact_mail.html` を作成します。

`templates/contact_mail.html`

```

<h2>お問い合わせ内容</h2>
<p>
    お問い合わせありがとうございます。<br>
    以下の内容でお問い合わせを受け付けました。
</p>
<p><strong>名前:</strong> {{name}}</p>

```

```
<p><strong>メールアドレス:</strong> {{email}}</p>
<p><strong>本文:</strong><br>{{body}}</p>
```

テンプレート読み込み

`loadTemplate()` でメールテンプレートを読み込み、HTMLメールにします。

Contact.php

```
private function loadTemplate($values)
{
    $template = file_get_contents($this->template);
    foreach ($values as $key => $value) {
        $template = str_replace("{{{$key}}}", $value, $template);
    }
    return $template;
}
```

メール送信処理

`send()` でメール送信処理をします。メール本文は `loadTemplate()` でHTMLメールに変換します。

Contact.php

```
public function send($name, $email, $body)
{
    try {
        // HTMLメール作成
        $html = $this->loadTemplate([
            "name"  => $name,
            "email" => $email,
            "body"  => nl2br($body),
        ]);

        $this->mailer->setFrom($this->from_address, $this->from_name);
        $this->mailer->addAddress($email, $name);
        $this->mailer->addReplyTo($this->from_address, $this->from_name);
        $this->mailer->isHTML(true);
        $this->mailer->Subject = $this->subject;
        $this->mailer->Body = $html;
    }
}
```

Copy

```
        return $this->mailer->send();
    } catch (Exception $e) {
        return $this->mailer->ErrorInfo;
    }
}
```

メール送信実行

POSTデータを取得し、`Contact` クラスでメール送信実行をします。

```
<?php
session_start();
require './Contact.php';

// POSTデータ取得
$name = $_POST['name'] ?? '';
$email = $_POST['email'] ?? '';
$body = $_POST['body'] ?? '';

// セッションに保存
$_SESSION['name'] = $name;
$_SESSION['email'] = $email;
$_SESSION['body'] = $body;

// メール送信
$contact = new Contact();
$result = $contact->send($name, $email, $body);

header("Location: result.php");
exit;
```

4. 運用のコツ

本番で必要な設定（到達率UP）

- SPF：どのサーバがあなたのドメインを送信してよいか（DNS TXT）
- DKIM：メールに署名を付与（送信サービスが鍵発行）
- DMARC：SPF/DKIMのポリシーとレポート
- 逆引き（PTR）：自前SMTPサーバ運用時は必須級

送信サービス（SendGrid / Mailgun / AWS SES 等）を使うと、ウィザードに従って DNSを設定するだけで通過率が上がります。

失敗しないために

- 本番ドメインでの「認証（SPF/DKIM/DMARC）」は最優先**
- バウンス/苦情のハンドリング：送信停止や抑止リストを持つ
- テンプレはテキスト版も同梱**（HTMLのみだとスパム判定を受けやすい）
- ログとメトリクス：配信成功/失敗、開封/クリック（プライバシー配慮）

よくあるエラー対処

- `535 Authentication failed`：ユーザー/パス誤り、2段階認証やアプリパスワード要確認
- `ECONNREFUSED / timeout`：ポート・ファイアウォール・TLS設定を確認
- 迷惑メール行き：SPF/DKIM/DMARC 未設定、差出人ドメインとSMTPの不一致

<< Composer

無名関数とアロー関数 >>

当サイトの教材をはじめとするコンテンツ（テキスト、画像等）の無断転載・無断使用を固く禁じます。これらのコンテンツについて権利者の許可なく複製、転用等する事は法律で禁止されています。尚、当ウェブサイトの内容をWeb、雑誌、書籍等へ転載、掲載する場合は「ロジコヤ」までご連絡ください。

© 2021-2025 logicoya.com