



# PHP基礎オンライン講座 1回目

2026/01/06

Kazuma SEKIGUCHI

# 自己紹介



関口和真

株式会社コムセントCTO

Webシステム開発、スマートフォンアプリ制作、  
サーバー構築、運用など

スマートフォンを使ったアプリの制作

サーバーサイドシステムの作成

フロントエンド部分の作成

# 目標

- PHPで簡単なWebアプリが作成できるようになる
- PHPの基本的な文法を理解する
- データベースの使い方を習得する
- セッションなどのWebならではの扱いについて知る

# 今回のアジェンダ

- PHPとは何か
- PHPの動作環境
- PHPの記述方法
- 変数、文字列連結
- if文、ループ文
- 関数の利用
- ユーザー定義関数の作成

# PHPとは

- 動的なWebページを作成することに特化したスクリプト言語
  - 動的なWebページ=ユーザーの操作や送り込んでくる情報に基づいてページの内容を変化させるようなWebページ
- スクリプト言語でもあり、PHPというスクリプト言語を実行するための実行環境でもある
  - スクリプト言語なので、コンパイルしてもらう必要がある
  - PHPで書かれたプログラムを実行するためにはPHPが必要



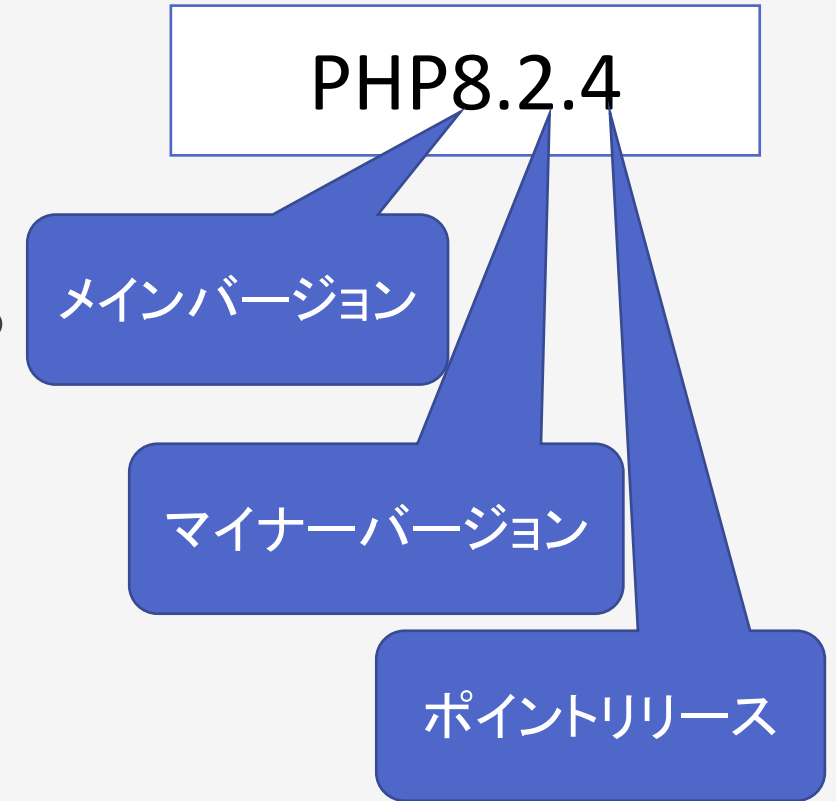
言語

実行環境



# PHPとは

- 元々はWebサイトの作成に特化しているが、Webサイトの機能強化に伴い、PHP自体の機能も強化されている
- バージョンごとに機能や関数が少しずつ異なる
  - 使用するPHPのバージョンに注意する必要がある
- マイナーバージョン以上で何らかの変更が行われる
  - ポイントリリースはバグフィックスかセキュリティフィックス



# PHPとは

- HTMLに限らずさまざまなデータを出力することが可能
  - JSON、画像データ、テキストデータ、PDF、Wordドキュメントなど
- 結果を表示するのは通常ブラウザー
  - ターミナルなどのCLI画面で実行、出力させることも可能
  - ブラウザーで表示できないものはダウンロード扱いになる

# 現状のPHP

- いつでもPHPはオワコンと言われている言語
  - 言われ続けて10年以上経つはずだが、未だに健在
  - プログラミング言語なんてそんなもの
- サーバーサイドだけではなく、デスクトップアプリを作成できるようになりつつある
  - が、誰も望んではいないように思う
- サーバー側で動作する言語としては、Ruby、Go、JavaScriptなどが他の有力候補
  - レンタルサーバーで動作させるならPHPが一番楽
    - 他の言語はサポートされていないことが多い



# PHPの勉強方法

- 公式のチュートリアルはイマイチなので、他のを参照した方が良い
  - JavaDriveのPHP入門  
<https://www.javadrive.jp/php/>
- 標準関数はPHP公式マニュアルを参照する
- 書籍も比較的多く出ているので、それらを使うのも手
  - 自分が読みやすいものを1冊選んで勉強
- PHP自体は比較的初心者向きのプログラミング言語
  - 習得が容易
  - キレイに書こうとすると結構大変なところも多い
  - 初心者であれば、Pythonもオススメ

# PHPの勉強方法

PHPの基  
本的な文  
法

関数を利用した記  
述

フォーム  
との連動  
(データ  
受け取  
り)

データ  
ベースを  
利用した  
プログラ  
ム

オブジェ  
クト指向  
を利用し  
た記述

ライブラ  
リーを利用したプ  
ログラム  
記述

フレーム  
ワークを利用した  
プログラ  
ム

# PHPの特徴

- 弱い動的型付け言語
  - 現在はメソッドや関数の引数として型を指定することが可能
- 比較的容易に記述することが可能
- スクリプト言語なため、確認→修正が容易
- 標準で多数の関数が存在するため、関数を作ること無く機能を実行可能
- 多くのWebサーバーに組み込まれているため動作させやすい
- レンタルサーバーの多くで実行可能
- 開発事例が多いため、参考となる資料が豊富

# PHPの特徴

- HTMLファイル内にPHPプログラムを埋め込むことができる
  - 拡張子は変える必要がある
  - PHPだけのファイルもちろん作成可能

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="UTF-8">
    <title>HTML</title>
  </head>
  <body>
    <h1><?php echo $username; ?></h1>
```

\$usernameという変数の  
値を表示する

<?phpで始まり、?>で終わる  
その間にPHPプログラムを  
書くと、PHPがプログラムとして  
実行してくれる

# PHPを作成、動作させる

- PHPを記述する
  - 普通のエディタなどで記述は可能
  - Visual Studio Codeなどでも記述可能
    - 機能拡張を入れると更に便利
  - PHPStormなどの有料IDEも存在
- PHPの動作はOSには影響を受けない
  - PHP自体はさまざまなOS向けに提供されている
  - macOSでもWindowsでもLinuxでも記述、動作可能



# PHPを作成、動作させる

- PHPを実行する環境は必要
  - 今回はMAMPを利用してPHPを実行
  - レンタルサーバーなどでも通常PHPは動作するので、あまり環境を選ばずに実行できる
    - 多少レンタルサーバーごとに設定やPHPのバージョンの違いなどにより動作に違いが出ることはある
- PHPの動作自体はphp.iniファイルで設定を変更できる
  - MAMPの場合少し変更した方が良い



# MAMPの設定

- Windows版
  - MAMP→Preferenceを選択し、Stop serversにチェックを入れておく
  - PHPタブでVersionを選択、8.3.1を選択する
  - Portsタブで80&3306のボタンを押しておく
- mac版
  - PHP versionで8.3.28を選択
  - PreferencesでPortsタブを開き、80&3306のボタンを押しておく
  - ServerタブでUse MySQL serverで5.7.44を選択しておく



# php.iniの設定

- Windowsの場合

C:¥MAMP内の「conf」→「php8.3.1」→php.iniをエディタ等で開く

- Macの場合

「Finder」→「アプリケーション」→「MAMP」→  
「bin」→「php」→「php8.3.28」→「conf」→php.ini  
をエディタ等で開く

# php.iniの設定 (Windowsの場合)

374行目付近

`display_errors = off`



`display_errors = on`

エラーが出たときにエラー内容を画面に表示するかどうかを設定する

704行目付近

`;date.timezone =`



`date.timezone = "Asia/Tokyo"`

時間を出すときに利用される。**先頭のセミicolonも外すこと！**  
日本であれば、GMT+9なので、タイムゾーンを変えないと変な時間が使われる

## php.iniの設定（Windowsの場合）

- 変更をしたらファイル名をphp.iniとして上書き保存する
- ファイルを保存したらMAMPのパネルから「Stop Servers」を押してサーバーを停止
- 「Start Servers」を押して再度サーバーを起動
  - 問題なく起動すればOK
  - 起動しない場合、設定をどこか間違えている可能性があるため、再度確認する（セミコロンを外す、残しておくとか、スペルなど）

# php.iniの設定(macの場合)

705行目付近

`post_max_size = 8M`

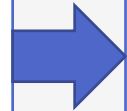


`post_max_size = 128M`

データ量として受け入れる最大サイズ。ファイルサイズではなく、パケットの総合計サイズになる

859行目付近

`upload_max_filesize = 32M`

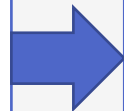


`upload_max_filesize = 64M`

アップロードが可能なファイルサイズの最大サイズを指定。post\_max\_sizeよりも下の値を指定する

949行目付近

`;date.timezone = "Europe/Berlin"`



`date.timezone = "Asia/Tokyo"`

時間を出すときに利用される。**先頭のセミコロンも外すこと！**  
日本であれば、GMT+9なので、タイムゾーンを変えないと変な時間が使われる

## php.iniの設定（macの場合）

- 変更をしたら上書きして保存する
- ファイルを保存したらMAMPのパネルから「Stop Servers」を押してサーバーを停止
- 「Start Servers」を押して再度サーバーを起動
  - 問題なく起動すればOK
  - 起動しない場合、設定をどこか間違えている可能性があるため、再度確認する（セミコロンを外す、残しておくとか、スペルなど）

# PHP

- 変数（配列を含む）
- 関数
- 制御
- クラス

の4つから構成される

# 書式

- <?phpから始まり?>で終わる
  - HTMLで言う<body>タグと同じ
- 空の行などは一切意味がない
  - 見やすいように適宜改行を行って良い
    - スクリプトによっては、ファイルサイズを小さくするため、改行、スペースを削除する場合もある
- 1行で一つの動きを書く
- 「;」で行の終わりを示す

```
<?php  
    print('PHPで文字を表示します');  
?>
```



# 変数

- 変数名には「\$」をつける

```
$a = 5;
```

- この中のaが変数。aには5が入る
- 変数は自由に値を格納できる箱みたいなもの
- 変数名は自由に設定できる
  - 但し、予め別の役割が決められているものは使用不可（予約語）
  - 変数名の最初は英字で始まること。数字は文頭には使えない
  - 「\_（アンダーバー）」から始まる変数は特殊な用途に利用される

## 変数（2）

- 変数は宣言せずに使用可能
  - 初期化できる（現在は初期化をすることが推奨）

```
$variables = "";
```

- 型は一切なし
  - 関数の引数やクラスなどでは指定することもできる
  - \$txt = "";とすれば文字列型
  - \$txt = 0;とすれば数値

## 変数 (3)

- 文字はシングルクォーテーションで括る

```
$a = 'text';
```

- ダブルクォーテーションで括った場合は、中の変数が展開される

```
$text = 'php';  
print("this is a {$text} program.");
```

\$textの部分が変数として  
処理されてthis is a php  
programとして表示される

- 文字列で改行を表すときは¥n (バックスラッシュ + n) を記述する
  - ダブルクォーテーションで括っておく

```
print("this is a ¥n php program");
```

this is a  
php program  
と改行されて表示される

# コメント

- スクリプトの実行時には無視される
- プログラムの動きなどを記述しておき、後から見直ししやすいようにする

//1行だけコメント

/\*この挟まれた間がコメント\*/

# 文字列連結

- 変数同士をくっつける
- 「.(ドット)」を使う

```
$txt_first = "PHP";  
$txt_second = "Program";  
$moji = $txt_first . $txt_second;
```

- \$mojiにはPHPProgramが入る
- 「.」を使う際には「" ." 」とする

# 条件（条件分岐）

- 条件に応じて、処理を分けるケースは多い
  - クリックされてもこれ以上右に動かさない→左に動かす
  - いわゆる判断をさせる場合に利用
  - 条件式を記述して判断させる
  - ifのみ必須
    - 他は任意
  - elseifは複数回利用可能

```
if(条件式1){  
    条件式1に当てはまるならここが動作  
}  
elseif(条件式2){  
    条件式2に当てはまるならここが動作  
}  
else{  
    条件式に当てはまらない場合に動作  
}
```

# 条件式

- 0以外の数、true , 何らかの文字列は全て条件式で当てはまる
  - 0,falseだと条件式に当てはまらない
- 比較して判断することも
  - `if($a > 3)`→aが3よりも大きければtrueになる
  - `if($a == 3)`→aが3のときだけtrueになる
  - `if(($a >= 3) && ($a < 100))`→aが3以上で、100未満のときだけtrueになる
  - `if(($a >= 100) || ($a < 10))`→aが100以上または10未満のときだけtrueになる



# 条件式の成立

- 成立する場合

- `if(true)`
- `if(1以上の数値)`
- `if( '何らかの文字' )`
- 条件式が成立しない場合

- 不成立の場合

- `if(false)`
- `if(null)`
- `if(undefined)`
- `if(0)`
- 条件式が成立しない場合

# ループ（繰り返し処理）

- 同じ処理を繰り返す際に利用
  - いくつか方法がある
  - プログラムは楽するために利用する＝同じ処理はできるだけループを使って処理する
  - 条件が成立している限りループする
    - 初期値に0を入れて1回回るごとに1を足していくと100回回った段階で条件を満たさなくなる

```
for($i =0; $i< 100; $i++){  
    print($i);  
    //変数iに格納されている値を表示。0～  
    99まで表示される  
}
```

```
for(初期値; 条件; カウントアップ){  
    繰り返す内容  
}
```

# ループ

- 条件が成立している限りループする
  - while文
  - 気をつけないと永久にループする
  - データベースからデータを取り出すときに良く利用する

```
$a = 0;  
while($a < 100){//( )内の条件がtrueな限りループ  
    $a++;//変数aに1を加算  
    print($a);//変数aを表示(1~100まで表示)  
}
```

```
while(条件式){  
    繰り返す内容  
}
```

# 関数

- 多数の関数が存在
- 引数を与えることで任意の動作が可能
- PHPの場合、関数が大量にあるため、上手く組み合わせる

```
<?php
print("今の時刻を10回表示します");
for($i = 0; $i < 10; $i++){
    print(time());
}
?>
```

# ユーザ定義関数

- 何らかの処理をまとめたもの
- 同じような処理を繰り返す際に利用する
  - イチイチ同じ処理を何度も書くのは面倒。間違えも増える
- ()内に引数を指定可能
  - 引数はカンマで区切ることで複数指定可能
  - 引数に値を与えて関数を呼び出せば、その引数を利用して処理してくれる
- ユーザ定義関数から更にほかのユーザ定義関数を呼び出すことも可能
- 記述場所は自由
  - 通常まとめて書いておく

# ユーザ定義関数

```
function 関数名(引数1,引数2・・・){  
    処理の内容  
    return 処理の結果  
}
```

基本的な記述方法

```
function Calc($a,$b){  
    $c = $a + $b;  
    return $c;  
}
```

作成(Calcという関数を作成)

```
$g = Calc(4,6); //gには10が格納される  
function Calc($a,$b){  
    $c = $a + $b;  
    return $c;  
}
```

Calcを利用

```
function caution(){  
    print('注意！！');  
}
```

returnの無いものも可能

# PHPの記述

- 単一のファイルに記述して保存
- 拡張子は**php**とする
  - アクセスされるときもhttp://example.jp/example.phpとしてアクセスされる
- サーバ上で動作するため、サーバにFTPなどでアップして実際にアクセスして動作を確認する
  - ローカルで開発環境を作成して確認しておく
- 文字コードは**UTF-8**で作成する
  - Shift-JISでは作らないこと



# サーバ上のPHP環境情報

<div> <div>PHP Version 5.1.6</div>  </div>	
<b>System</b>	Linux serverlx01 2.6.18-92.1.10.el5 #1 SMP Tue Aug 5 07:41:53 EDT 2008 i686
<b>Build Date</b>	Jul 16 2008 19:54:37
<b>Configure Command</b>	./configure '--build=i686-redhat-linux-gnu' '--host=i686-redhat-linux-gnu' '--target=i686-redhat-linux-gnu' '--program-prefix=' '--prefix=/usr' '--exec-prefix=/usr' '--libdir=lib' '--with-config-file-path=/etc' '--with-config-file-scan-dir=/etc/php.d' '--disable-debug' '--with-pic' '--disable-rpath' '--without-pcre' '--with-bz2' '--with-layout=GNU' '--enable-exif' '--enable-fpm' '--enable-magic-quotes' '--enable-sockets' '--enable-sysvsem' '--enable-sysvshm' '--enable-sysvmsg' '--enable-without-sqlite' '--with-libxml-dir=/usr' '--with-xml' '--with-system-ztdat' '--with-apxs2=/usr/sbin/apxs' '--without-mysql' '--without-gd' '--without-odbc' '--di
<b>Server API</b>	Apache 2.0 Handler
<b>Virtual Directory Support</b>	disabled
<b>Configuration File (php.ini) Path</b>	/etc/php.ini
<b>Scan this dir for additional .ini files</b>	/etc/php.d
<b>additional .ini files parsed</b>	/etc/php.d/dbase.ini, /etc/php.d/dom.ini, /etc/php.d/eaccelerator.ini, /etc/php.d/gd.ini, /etc/php.d/ldap.ini, /etc/php.d/mbstring.ini, /etc/php.d/mcrypt.ini, /etc/
<b>PHP API</b>	20041225
<b>PHP Extension</b>	20050922
<b>Zend Extension</b>	220051025
<b>Debug Build</b>	no
<b>Thread Safety</b>	disabled
<b>Zend Memory Manager</b>	enabled
<b>IPv6 Support</b>	enabled
<b>Registered PHP Streams</b>	php, file, http, ftp, compress, bzp2, compress, zlib, https, ftps
<b>Registered Stream Socket Transports</b>	tcp, udp, unix, udg, ssl, sslv3, sslv2, tls
<b>Registered Stream Filters</b>	string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, convert.iconv.*, bzp2.*, zlib.*

- サーバで動いているPHP  
の情報を入手

```
<?php
    phpinfo();
?>
```

- 適当な名前で保存して実行する

## 参考書など

- 特に本講座で使う訳ではありません
- 大重美幸, 『詳細! PHP 8 + MySQL入門ノート  
XAMPP + MAMP 対応』, ソーテック社, 2021
- 山田祥寛, 『独習PHP 第4版』, 翔泳社, 2021

ありがとうございました。  
また次回。