第１２回　JSONと外部API

１２－１．JSON

１２－１－１． JSONとは

JSON（JavaScript Object Notation）はデータ交換フォーマットになります。

JSON は 、JavaScript のサブセットに似ており、多くのプログラミング言語がJSON をサポートしていますが、ウェブサイトやブラウザの拡張機能を含むJavaScript ベースのアプリケーションでの利用が特に便利です。

　・JSON は数値、真偽値、文字列、null、配列で構成されるオブジェクトを表すことができます。

　・JSON はCSV 形式とは異なり、XML のような階層データを格納することができます。

１２－１－１－２．JavaScript とJSON の違い

　　JSON は、数値、文字列、論理値、null、配列、オブジェクトをシリアライズする構文です。これはJavaScript の構文

に基づいていますが、区別されるもので、JavaScript ならば JSON であるというわけではありません。

　・オブジェクトと配列

　　プロパティ名は二重引用符で括った文字列にしなければなりません。末尾のカンマを置いてはいけません。

　・数値

　　先頭にゼロを置くことは禁止されています。NaN とInfinity には対応していません。

１２－２．JSON関数(DECODE)

JSON 形式の文字列をデコード（複号）します。

json\_decode関数

テキスト

低い精度で自動的に生成された説明

|  |  |
| --- | --- |
| 引数 | 説明 |
| $json | デコード対象のJSONデータ |
| $assoc | trueの場合、JSON文字列中のオブジェクト型がPHPの連想配列に変換される |
| $depth | 許容する階層の深さ |
| $options | デコード時の挙動を指定するオプション定数 |

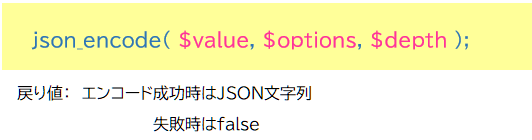
■json\_decode関数の第４引数 $options に指定できる定数

|  |  |
| --- | --- |
| オプション | 意味 |
| JSON\_ＢＩＧＩＮＴ\_AS\_STRING | 大きな整数値を文字列データで表現する |
| JSON\_OBJECT\_AS\_ARRAY | JSONのオブジェクト型をPHPの連想配列に変換する  ※JSON\_OBJECT\_AS\_ARRAYを使うことは、第２引数をtrueにすることと同じ意味です。 |

１２－３．JSON関数(ENCODE)

値をJSON 形式にして返します。

json\_encode関数



|  |  |
| --- | --- |
| 引数 | 説明 |
| $value | エンコード対象の値（スカラー、配列、連想配列、オブジェクトなど） |
| $options | エンコード時の挙動を措定するオプション変数（下記、別表を参照） |
| $depth | 引数$valueが配列/連想配列だった場合に許容する階層の深さ |

■json\_encode関数の第２引数 $options に指定できる定数

|  |  |
| --- | --- |
| オプション | 意味 |
| JSON\_HEX\_TAG | 「＜」と「＞」をエスケープする |
| JSON\_HEX\_AMP | 「＆」記号をエスケープする |
| JSON\_HEX\_APOS | 「’」(アポストロフィー)をエスケープする |
| JSON\_HEX\_QUOT | 「”」（ダブルクォート）をエスケープする |
| JSON\_FORCE\_OBJECT | 連想配列ではない値（おもに配列を想定）を強制的にオブジェクト型で出力する |
| JSON\_NUMERIC\_CHECK | 数値が文字列型で表されていた場合に、数値として出力する |
| JSON\_BIGINT\_AS\_STRING | 大きな整数値を文字列型として出力する |
| JSON\_PRETTY\_PRINT | 改行とインデントを使って出力結果を見易くする |
| JSON\_UNESCAPED\_SLASHES | 「＼」（バックスラッシュ）をエスケープしない |
| JSON\_UNESCAPED\_UNICODE | Unicode文字列を１６進数エスケープしない |
| JSON\_PARTIAL\_OUTPUT\_ON\_ERROR | エンコードできない値を代替値に置き換える |
| JSON\_PRESERVE\_ZERO\_FRACTION | float型の値を常にfloat値としてエンコードする |

１２－２．外部API

今回は外部APIを利用します。ネットワーク経由で外部の情報を取得するために、Guzzleというライブラリを使用します。

今回は、Guzzuleライブラリを含んだ「lib」ファルダを配布していますので、「PHP1」フォルダ内に「lib」フォルダを配置してください。

下記に、郵便番号から住所を検索するAPIを例に使用方法を見ていきましょう。

■サンプルコード　※外部API（郵便番号から住所を返すAPI）を利用する。

　※各コードの説明は後述。

require './lib/vendor/autoload.php';

$cli = new GuzzleHttp\Client([

'base\_uri' => 'https://zipcloud.ibsnet.co.jp', //外部API

]);

$res = $cli->request('get', '/api/search', [

'query' => [

'zipcode' => $zipcode // 検索したい郵便番号を指定

],

'verify' => false //開発用環境なので、証明書の検証をオフにする**（本来はオンで使用する）**

]);

$response = json\_decode($res->getBody(), true); //JSONデータを連想配列に変換

1. autload.phpをインポートする

autoloadはライブラリを自動ロードするためのコードです。Guzzleに限らず、Composerでインストールしたライブラリを利用する際には、あらかじめautoload.phpをインポートしておきます。

※ComposerについてはLaravelで説明します。今は仕組みだけ知っておいてください。

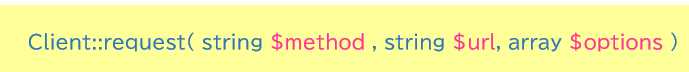
1. GuzzuleHttp\Clientオブジェクトを生成する

コンストラクターには「 **オプション=>値** 」の形式で、動作オプションを指定できます。ここでは、base\_urlで基底のパスだけを指定しておきましょう。

1. リクエストを送信する

実際にリクエストを送信するのは、requestメソッドで行います。

requestメソッド



|  |  |
| --- | --- |
| $method | HTTPメソッド |
| $url | アクセス先のURI |
| $options | 動作オプション |

1. レスポンスを処理する

今回はgetBodyメソッドでレスポンスの本体を取得しています。取得した情報をJSON　DECODE関数で連想配列に変換しています。

以上で、データは取得できたので、課題では取得データを用いて画面に表示してみましょう。

また、郵便番号を固定にしているので、課題では入力値に応じて検索ができるようにします。