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**

```
json_decode( $value, $assoc, $depth, $options );
```

戻り値：デコード成功時はPHPで扱える値
失敗時はnull

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**

```
json_encode( $value, $options, $depth );
```

戻り値：エンコード成功時はJSON文字列
失敗時はfalse

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
```

④

③

②

①

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。

② 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。

③ 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**

```
Client::request( string $method , string $url, array $options )
```

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

④ 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。