MVCは、プログラムを3つの要素 [ ] ・ [ ] ・ [ ] に分割し、それぞれの役割を以下のように決めてWEBアプリ用のテンプレートを作成します。

Mのパーツでは、アプリケーション固有の処理を記述するようにします。その為、他から呼び出す [ ] の定義を行う事になります。

Vのパーツでは、ユーザにとって重要な [ ] を定義します。殆どの記述をクライアント側で処理される [ ] ・ [ ] ・ [ ] で記述し、これらの特性より、職種としての [ ] の介入を容易にする目的もあります。

Cのパーツでは、アプリケーション全体を見渡せるような基本的な部分を記述します。Cの意味通りに、ブラウザより [ ] された内容を正しく処理する為の条件を設定したり、処理の呼び出しを制御します。

MVC の役割を持つ PHP のファイルを3つ作成し、それぞれ [ ] .php・ [ ] .php・ [ ] .php とします。ブラウザから呼び出されるのは C の役割を持ったファイルになり、他の二つのファイルは PHP の [ ] で、読み込むことになります。

V の部分で動的に PHP で作成する必要がある場合は、**PHP の変数埋め込み**方法を使用して変数内に必要なクライアント側の記述を構築します。その変数を **$dynamic** とすると、その記述は [ ] となります。

PHP には、 [ ] と言う『すべてのスコープで使用できる組み込みの変数』が存在します。この中でも特に WEBアプリケーションにおいて、**FORM からの送信**で作成される変数を [ ] ・ [ ] と書きます。この二つの変数名は、FORM 要素の [ ] 属性の値として指定可能な文字列より命名されたものです。

その次に重要な変数は、 [ ] です。この中には常にその時の PHP を取り巻く環境変数が設定されているので、WEB アプリケーションではこのうちのいくつかを利用する事になります。

さらに、WEBアプリケーションに**ログイン**を実装する為に必ず使用する事となる [ ] という変数があります。これは、**一定時間ユーザがページを移動してもサーバ側で値を維持可能**な変数です。

これらの変数は、連想配列と呼ばれるキーに対して値を持つ配列として定義されているので、**PHP をデバッグ**する上で重要な値を常に保持しています。そこで、 [ ] という関数でページの最後に出力したり、WEBアプリの都合上画面上に出す事が容易で無い場合、この関数の第二引数を true にして出力を文字列に変換し、 [ ] 関数を使ってテキストファイルとして出力する事も考えましょう。

WEBアプリケーションは、基本的には**文字列を扱う操作が重要**になります。その為、文字列の中に変数を埋め込んだり、HTML の中に PHP の変数を埋め込んだりする事が多々あります。文字列の中に変数を埋め込む場合は [ ] を使用してその間に変数を記述します。また、埋め込まずに他の文字列と変数を連結する方法もあります。その場合 PHP では連結記号(結合演算子)として [ ] が使用されます。

しかし、埋め込む事を目的とした文字列は通常ダブルクォート内で作成される為、ダブルクォート文字そのものを表現するには [ ] する必要があります。その為可読性が悪くなるので [ ] と言う方法を使って、変数に直接記述したそのままの文字列をセットする事によって記述も楽になり、視認性も良くなります。

PHP で WEBアプリケーションを作成する上で、ブラウザに対して HTTP [ ] を送る [ ] 関数という**重要な関数**が存在します。簡単に言うと、この処理は **PHP がブラウザに対して行う指示**と考えるといいと思います。そして、プログラムの最初で送る事になる **Content-Type** が重要になります。これは、ブラウザに送るデータの種類を説明するもので、一般的に [ ] タイプと呼ばれる分類方法が定義されています。これは、MDN でも説明されていますが、『**文書、ファイル、またはバイト列の性質や形式を示す標準**』です。