|  |
| --- |
| **KAPsCore DSP** |
| Noframework Framework |
|  |
| KAPsCore and this document are provided under GNU GPL version 3 licenses. Check ‘License’ section for more details. |
|  |
| **Kapil Sharma (www.KapilSharma.info)** |
| **2/2/2013** |
|  |

Table of Contents

[License 5](#_Toc348186399)

[Introduction 7](#_Toc348186400)

[MVC: Basic structure and configuration 9](#_Toc348186401)

[Assumption: 9](#_Toc348186402)

[Introduction to MVC 9](#_Toc348186403)

[Controller: 9](#_Toc348186404)

[Model: 9](#_Toc348186405)

[View: 9](#_Toc348186406)

[What all this mean: 9](#_Toc348186407)

[Structure 10](#_Toc348186408)

# License

This file is part of KAPsCore DSP.

KAPsCore is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

KAPsCore is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the General Public License for more details.

You should have received a copy of the GNU General Public License along with KAPsCore. If not, see <http://www.gnu.org/licenses/>.

# Introduction

No boring and big introduction, just what we are trying to achieve with KAPsTOP.

End goal is to provide a good base for core PHP projects but here the target is to define each and every step of base work so that it can be modified wherever needed. Obviously the target audience here is students and software engineers/trainees with little professional experience (called ‘trainees’ or ‘Software trainees’ or ‘readers’ in rest of document). **Experienced IT professionals can easily skip this guide as code is properly commented and easy to understand.**

Again, this guide majorly covers how we are creating, arranging our first core PHP.

**Chapter 1**

# MVC: Basic structure and configuration

## Assumption:

For rest of the tutorial, I’m assuming you have some knowledge PHP. You had written some basic PHP applications, even if it is for learning PHP language basics. This importantly suggest you have PHP development environment ready; it may be WAMP on windows or separate installation on Linux, Mac etc.

I’m also assuming you have some knowledge of OOPs principles in other language like C++/Java etc & worked on some example OOPs programs in PHP.

If any of these assumptions are not correct, I’d like to suggest learning PHP and its OOPs concepts before you start with this tutorial as this tutorial do not cover language basics. We are just assuming you know language basics along with OOPs basics.

## Introduction to MVC

MVC is one of the most common structures to arrange any software application. MVC stands for Model, View, and Controller.

### Controller:

Controller, as name suggest, control everything. When a request comes to the server, it first goes to the controller. Controller decides if request is legal (correct structure, authentication, authorization etc). If the request is legal, it is controller’s responsibility to determine which model contains the requested data and initiate model object to fetch that data and finally pass control to view to display that data.

### Model:

Model stands to data. Most application has data. Mostly that data comes from database but it may also come from XML files, text files, 3rd party servers etc. Models will contain all the logic to fetch that data. Outside Model, it doesn’t matter where the data is coming from, what is the logic to fetch that data, or how we are saving it (single table/source or multiple.

### View:

View is responsible to determine how the requested data will be presented to end user. View is called by the controller which passes object of model containing requested data. View just pick up the data from model object and arrange it in presentable manner. View might sent its response directly to end user or to controller, as commanded by controller.

### What all this mean:

Readers, if not familiar with MVC till now, might be asking what this entire mean? How to apply this knowledge practically?

We will shortly see how to apply all this but for now, just remember basic concept. **Controller** controls the code flow. When a request comes, it checks authenticity of request, data fetching through model and display data through view. Thus it controls whole business logic. **Model** contains data fetching logic. It does not handle any request by itself but called by controller but once controller calls a model, it is model’s responsibility to decide how to fetch requested data from data source (database). So the difference is *controller decides what data is to be fetched but model decides how to fetch that data (data requested by controller).* Once model completes data fetching operation, controller has a object(s) of model class containing requested data. Then it decides which the correct view is to present that data and calls that view by passing model object (data) to view. **View** then decides how to present that data to the end user. Thus views also contain **Presentation logic.**

## Project setup

Enough theory lets apply all this. Create an empty folder to represent website on your apache. I’m using WAMP on windows 7 so I’m referring settings accordingly but you need to change settings to match your apache settings. My WAMP server is installed on location ‘D:\wamp’ so web path on my apache is ‘D:\wamp\www’. Again I’m naming this project as ‘kapscoredspmaking’. So important paths on my system are:

Web URL: http://localhost/kapscoredspmaking

Web path: D:\wamp\www\kapscoredspmaking

If you are using different operation system or different path, you need to make required changes in those two paths in following tutorial.

## Folder structure

Most important part is to design is to arrange files and folder properly. It is important to plan ahead how to arrange our files and folders. I’m proposing a folder structure, for now, just follow that. As you go ahead in this tutorial, you will know the reasons of such folder structure and more importantly, during this whole tutorial, we will make few changes in this folder structure, to address our increasing needs.

To start, just make index.php file in ‘kapscoredspmaking’ folder and add following code

<?PHP

echo 'Hi from kapscoredsp';

Start apache, open browser and enter URL <http://localhost/kapscoredspmaking>

It should display ‘Hi from kapscoredsp’. **If it says ‘Parse error: syntax error, unexpected T\_STRING, expecting ',' or ';' in D:\wamp\www\kapscoredspmaking\index.php on line *2’,* check single quote. This has to be taken care with rest of the examples in this document as word processor handle quotes differently than source code editor.**

Well this example is just to confirm we have a perfect structure. Now first important file in any web application is it configuration file. Let’s create configuration file.

## Application Configuration files

Configuration files are most important for any system. Why it is important? Well while development of any web application, there could be one or more developers involved. These developers work on their local system. Once development finishes, they combine the code and put on server. Now all these dev system and server may have different operating system, file paths etc. We put all these system specific information in configuration files so that when code move from one system to other, we just need to change configuration file.

Create a folder ‘kapscoredspmaking/lib/vendor/KAPsCore/config’. This means create folder ‘lib’ in ‘kapscoredspmaking’, than ‘vendor’ in ‘lib’ and so on. Now create a file ‘basic.config.php’ in that folder and add following code.



**An important note** before we discuss about this code. Check ‘Code example name’, which is dspmaking1.1 in this example. You can download the code from folder codeexample/project/files/name.php. This code can be downloaded as file ‘codeexample/dspmaking/files/eg1.1-basic.config.php’. Else you can download KAPsCore and can find codes in example/codeexample folder.

Now let’s discuss what this code is doing. We first defined an array to hold configuration.

Create a file ‘app.php’ and save it in the folder ‘kapscoredspmaking/hr/config’. Why we selected that structure? Reason will follow but quick reason, we will be making a ‘Human resources’ application in this tutorial and along with that, we will make KAPsCore. Our tutorial app name is ‘HR’. So folder ‘hr’ represents application folder. In that folder, ‘config’ folder represents the folder containing configuration files. File name ‘app.php’ represent general application configuration.