SEFP ASSIGNMENT-5
SYMFONY-PHP FRAMEWORK

INTRODUCTION:-

Symfony is a PHP web application framework for MVC applications. Symfony is free software. Symfony is a complete framework designed to optimize the development of web applications by way of several key features. For starters, it separates a web application's business rules, server logic, and presentation views. It contains numerous tools and classes aimed at shortening the development time of a complex web application.

Additionally, it automates common tasks so that the developer can focus entirely on the specifics of an application. The end result of these advantages means there is no need to reinvent the wheel every time a new web application is built!

Symfony is written entirely in PHP 5. It has been thoroughly tested in various real-world projects, and is actually in use for high-demand e-business websites. It is compatible with most of the available databases engines, including MySQL, PostgreSQL, Oracle, and Microsoft SQL Server. It runs on *nix and Windows platforms.

Features:-

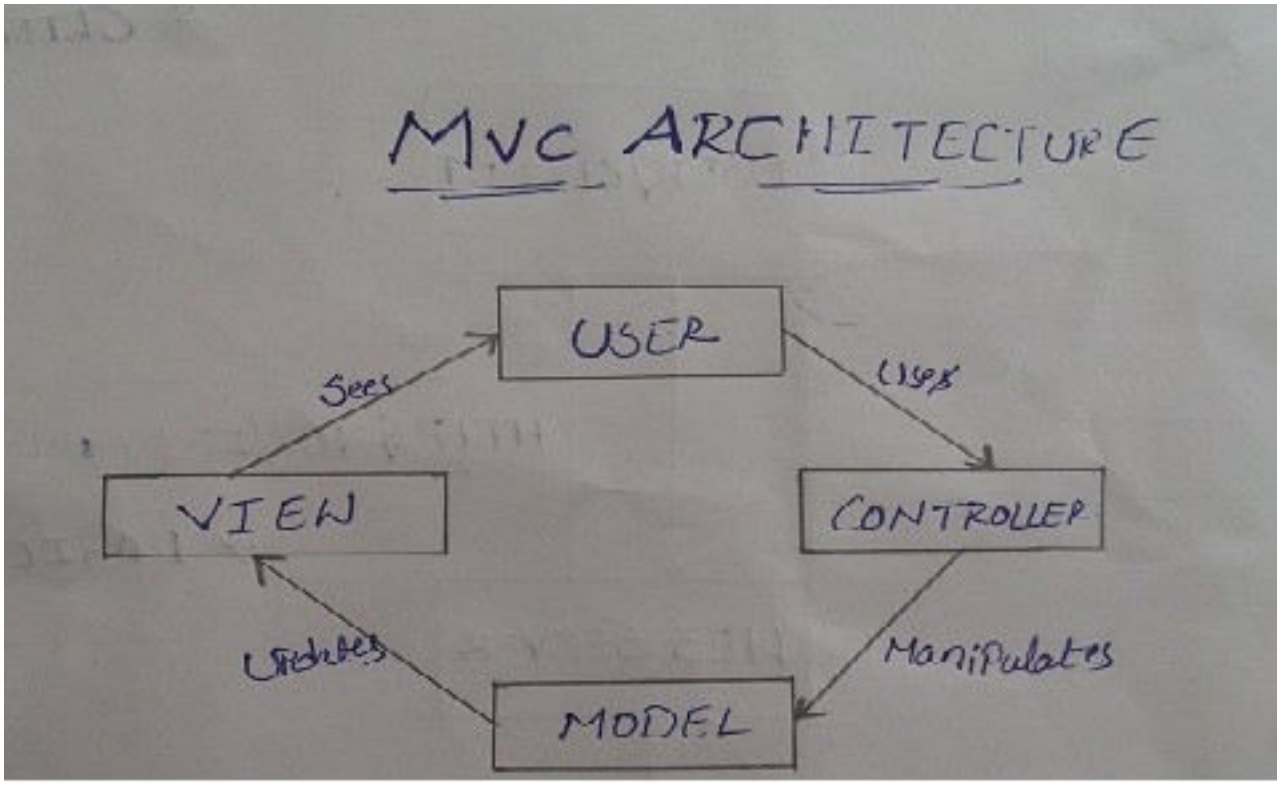
- Easy to install and configure on most platforms (and guaranteed to work on standard *nix and Windows platforms)
- Database engine-independent
- Simple to use, in most cases, but still flexible enough to adapt to complex cases
- Based on the premise of convention over configuration--the developer needs to configure only the unconventional
- Compliant with most web best practices and design patterns
- Enterprise-ready--adaptable to existing information technology (IT) policies and architectures, and stable enough for long-term projects
- Very readable code, with phpDocumentor comments, for easy maintenance
- Easy to extend, allowing for integration with other vendor libraries

SYMFONY ARCHITECTURE:-

The MVC model

Today will be the first dive in the world of the MVC Architecture. What does this mean? Simply that the code used to generate one page is located in various files according to its nature.
If the code concerns data manipulation independent from a page, it should be located in the **Model**(most of the time in askeet/lib/model/). If it concerns the final presentation, it should be located in the **View**; in symfony, the view layer relies on templates (for instance inaskeet/apps/frontend/modules/question/templates/) and configuration files. Eventually, the code dedicated to tie all this together and to translate the site logic into good old PHP is located in the**Controller**, and in symfony the controller for a specific page is called an

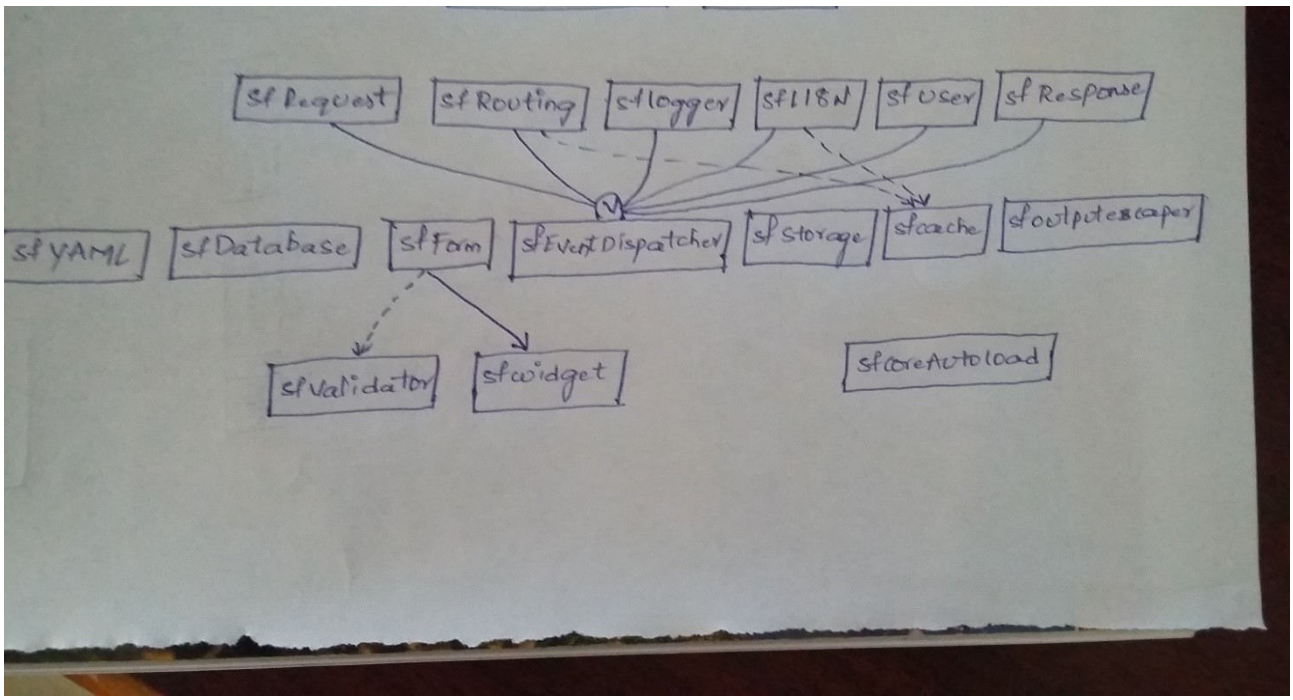
action (look for actions inaskeet/apps/frontend/modules/question/actions/). You can read more about this model in the MVC implementation in symfony of the symfony book. While our applications view will only change slightly today, we will manipulate a lot of different files. Don't panic though, since the organization of files and the separation of the code in various layers will soon become evident and very useful.

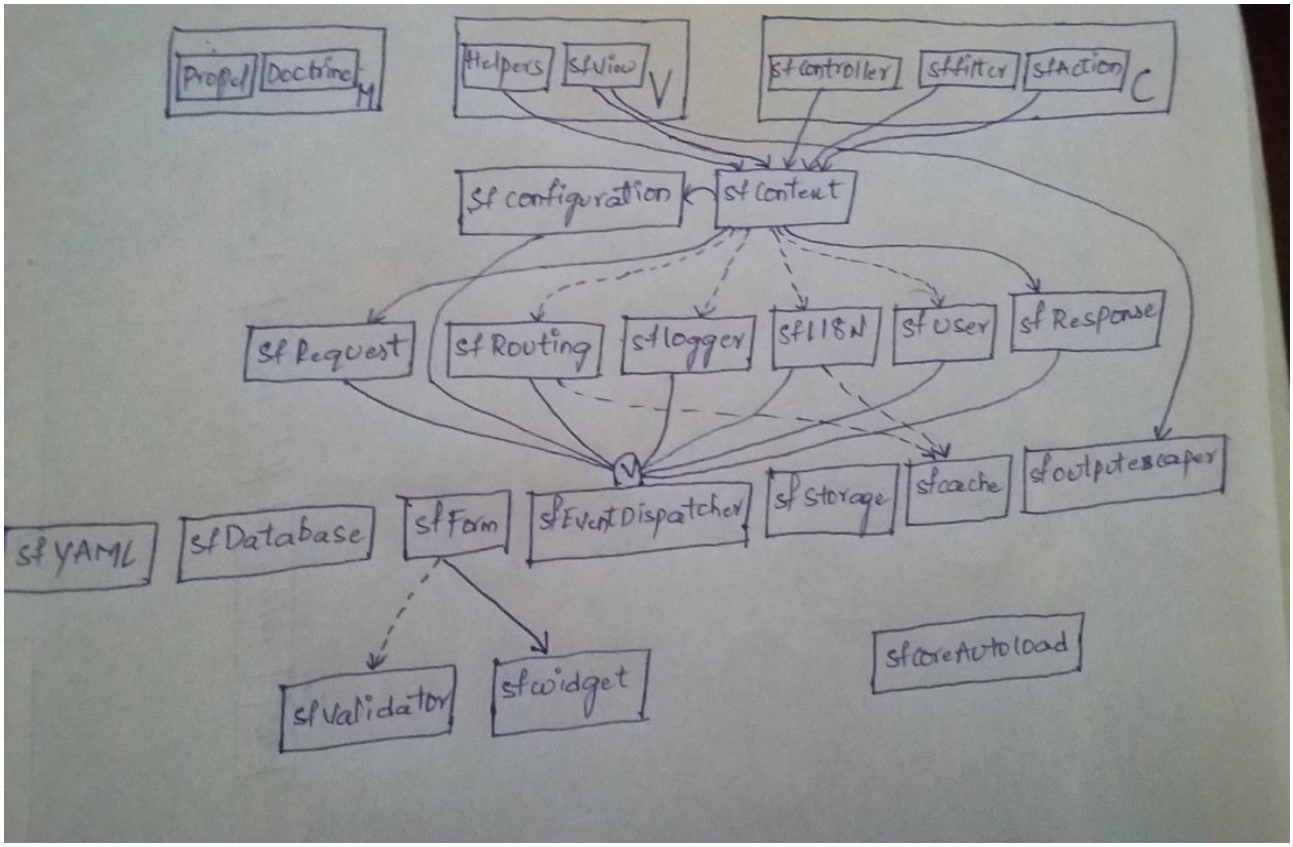


The symfony platform:-

symfony is based on a set of cohesive but decoupled classes, the **symfony platform**.Each class in the symfony platform is useable without the whole MVC architecture. The symfony platform classes have no dependency, and the only prerequisite to use them is the registration of the symfony autoloader:

- ➔ It follows the MVC pattern
- ➔ M-Model it’s known as the Backend of the programming.it controls the databases for storing and retrieving the data BY using languages MYSQL or MYSQLITE.
- ➔ V-View it’s known as the front end of the programming.it displays the webpages by using the languages HTML,CSS and JAVASCRIPT.
- ➔ C-Controller,Here in this layer we will control the both front end and back end by PHP script.it is also called as a logic layer.The main logic is managed in this layer.


sfRequest
sfRouting
sfLogger
sfI18N
sfUser
sfResponse
sfYAML
sfDatabase
sfForm
sfEventDispatcher
sfStorage
sfCache
sfOutputEscaper
sfValidator
sfWidget
sfCoreAutoload


Propel
Doctrine
M
Helpers
sfView
V
sfController
sfFilter
sfAction
C
sfConfiguration
sfContext
sfRequest
sfRouting
sfLogger
sfI18N
sfUser
sfResponse
sfYAML
sfDatabase
sfForm
sfEventDispatcher
sfStorage
sfCache
sfOutputEscaper
sfValidator
sfWidget
sfCoreAutoload

- ➔ The **symfony framework** itself is powered by the **symfony platform**:

The **sfConfiguration** class provides a way to configure and to customize your applications. The sfContext class acts as a registry that holds references to all core objects. And thanks to thefactories.yml configuration file, you can customize all the registry classes very easily, just by editing a YAML file.

- ➔ The **symfony MVC framework** is provided by a set of additional classes on top of the **symfony framework.**

- ➔ The *Model layer* is provided by third-party libraries, Propel or Doctrine. Even if symfony 1.1 is bundled with the Propel plugin, it's very easy to switch to Doctrine by installing thesfDoctrinePlugin. Both ORMs provide the same level of integration with symfony.

- ➔ The*View layer*is provided by the sfView class, a bunch of helpers, and templates written by the developer.

- ➔ The*Controller layer*is based on a filter chain and actions defined by the developer.

- ➔ As of version 1.1, symfony is one of the most decoupled framework available in PHP, even more than the Zend Framework. For example, the sfForm framework is useable without any of the MVC classes whereas Zend_Form is somewhat tied to the controller and the view layers.

CONCLUSION:-

I want to design the framework by using two layers ;-)