SEFP Asssignment-5
Zend Framework

What is Zend Framework:

Zend Framework 2 is an open source framework for developing web applications and services using php 5.3+. Zend Framework 2 uses 100% object-oriented code and utilises most of the new features of PHP 5.3, namely namespaces, late static binding, lambda functions and closures. The component structure of Zend Framework 2 is unique; each component is designed with few dependencies on other components. ZF2 follows the SOLID object oriented design principle. This loosely coupled architecture allows developers to use whichever components they want. We call this a “use-at-will” design.While they can be used separately, Zend Framework 2 components in the standard library form a powerful and extensible web application framework when combined. Also, it offers a robust, high performance MVC implementation, a database abstraction that is simple to use, and a forms component that implements HTML5 form rendering, validation, and filtering so that developers can consolidate all of these operations using one easy-to-use, object oriented interface. Other components, such as Zend\Authentication and Zend\Permissions\Acl, provide user authentication and authorization against all common credential stores.

FEATURES:

1.Zend Framework has a very extensive Validation component.
2.Object Oriented Goodness.
3.ZF, by default, is a glue framework. Its decoupled nature makes it easy to use as "glue" to your already existing application.
4.Integration with other libraries.
5.Contributor guide
6.recycling
7.Web and CLI programming.
8.No model implementation.

Architecture:

Zend framework follows MVC/VC architecture.

MVC Architecture:

MVC is much more than just a three-letter acronym (TLA) that you can whip out anytime you want to sound smart; it has become something of a standard in the design of modern web applications. And for good reason. Most web application code falls under one of the following three categories: presentation, business logic, and data access.

- ***Model*** - This is the part of your application that defines its basic functionality behind a set of abstractions. Data access routines and some business logic can be defined in the model.
- ***View*** - Views define exactly what is presented to the user. Usually controllers pass data to each view to render in some format. Views will often collect data from the user, as well.

This is where you're likely to find HTML markup in your MVC applications.

- ***Controller*** - Controllers bind the whole pattern together. They manipulate models, decide which view to display based on the user's request and other factors, pass along the data that each view will need, or hand off control to another controller entirely. Most MVC experts recommend for keeping comtrollers as simple a possible.

ARCHITECTURE:

No Model Implementation:

This is actually one of the reasons **most** developers don't use Zend Framework - it has no Model implementation on its own. For those who don't know what a Model is, it's the **M** in **MVC**, which stands for "Model-View-Controller", a programming architecture that's used by most PHP Frameworks.

Does that mean that ZEND framework is only a “VC” framework??

Yes, and no.

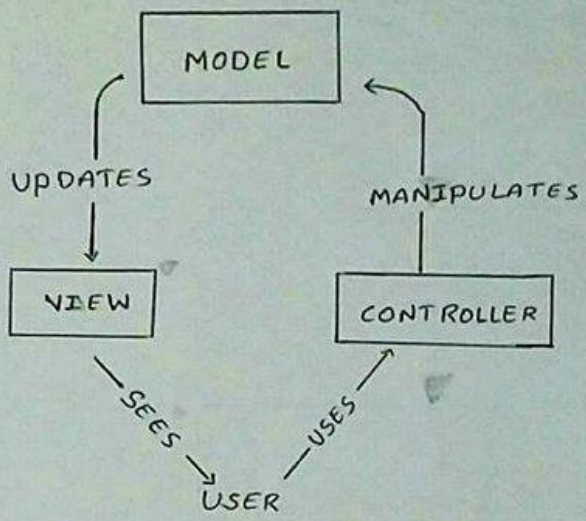
Yes, it's a VC framework because it doesn't have its own Model implementation. This makes it hard for some people to use ZF, especially if they're coming from a framework which does have a Model implementation (like CakePHP, Symfony, or even Ruby on Rails).

On the other hand, no, it's an MVC framework as well, since apart from providing the generic ways to access the database (using Zend_Db), it actually still relies on some sort of Model implementation. What it does differently is that it leaves this kind of implementation up to the developer ñ which some say should be the case since models are actually where the business logic of the application resides, and therefore, they're not something which can be developed as a generic component. Zend Framework Philosophy states that model implementations are unique to the projectóit's impossible to create an abstract implementation of it since they don't really know what you need. They believe that models should be implemented by the developers themselves.

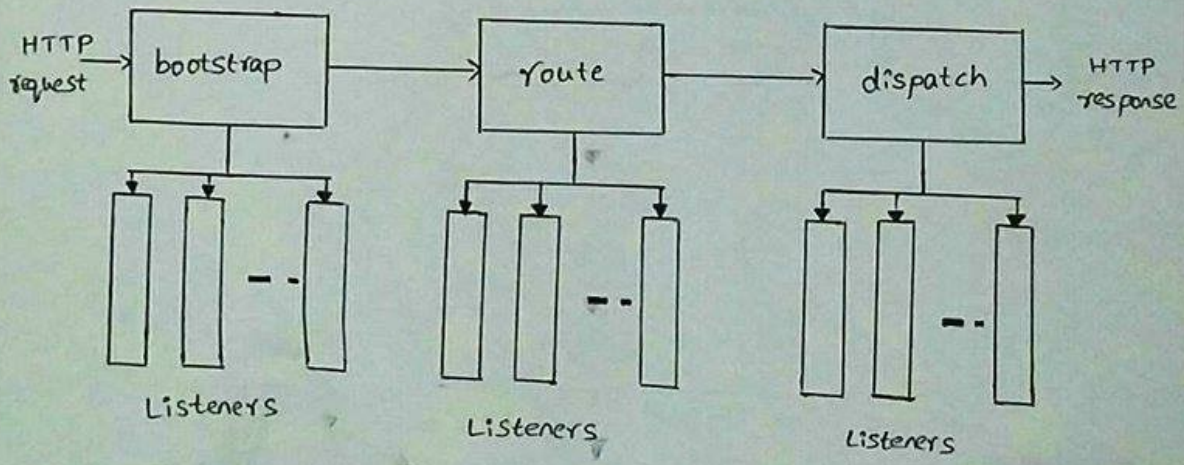
Not having a Model implementation means that the developer is free to use whatever means he has to implement it, or even just integrate existing implementations. Being free of predefined restraints, the developer is then allowed to create more complex implementations, rather than just simple representations of tables, which is how usual Model implementations are created. Models contain your business logic. They should not be restrained by your database tables; rather, they should dictate the connections of these tables to one another. This lets you put most of your programming code in your Models, therefore satisfying the "Thin Controllers, Fat Models" paradigm of MVC.

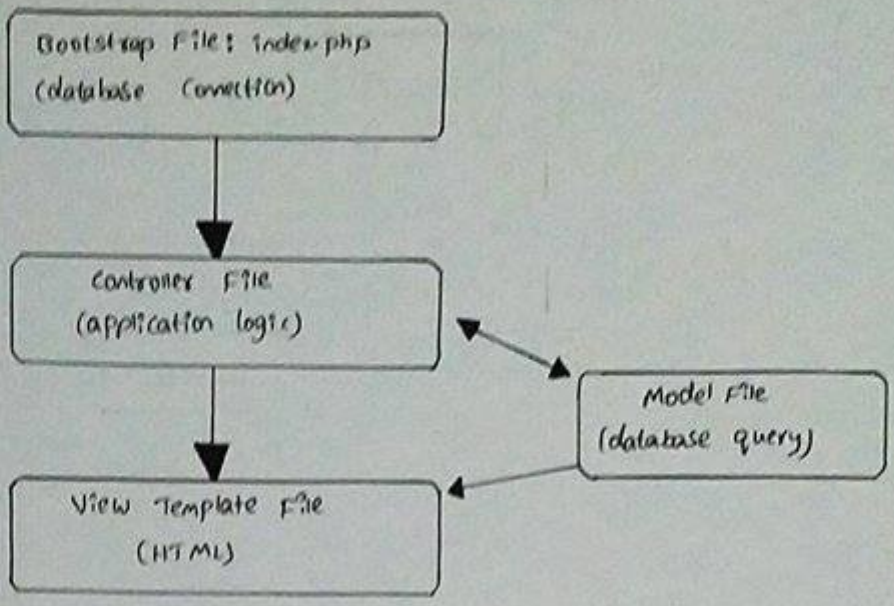
they implement an ORM approach to the models, wherein you would create three filesóthe actual Model, which is an abstract representation of your object; a Mapper, which maps data from the database to your Model; and a Database Table object, which is used by the mapper to get the data.

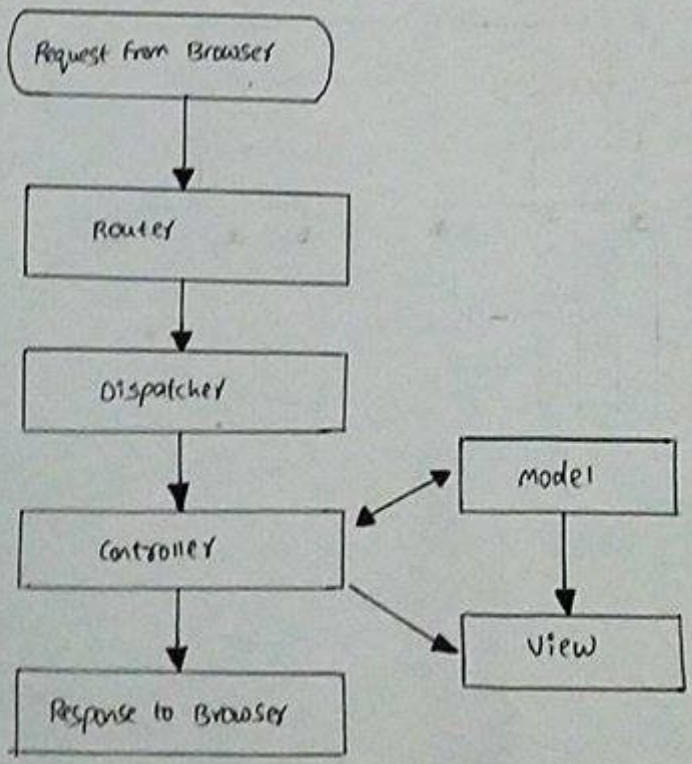
MVC - Model, View, Controller



Everything is an event




Bootstrap File: index.php
(database connection)
Controller File
(application logic)
Model File
(database query)
View Template File
(HTML)


Request from Browser
Router
Dispatcher
Controller
Model
View
Response to Browser

Benefits:

- Zend Framework focuses on quality of the code. It follows industry best practices. It uses proven object oriented design patterns. Almost all the components can be easily extended.The components of Zend Framework are unit tested using PHPUnit. All components have at least 80% code coverage. There is also a continuous integration server in place. The framework also provides a testing component.

- Zend Framework is a fully object-oriented framework, and as such, it utilizes a lot of object-oriented concepts like inheritance and interfaces. This makes ZF's components extendable to some point. Being able to customize ZF this way allows you to create functionality that is unique to your project.

- Tooling and rapid application development: The initial application setup can be quite difficult and tedious and RAD offers tooling support and a command line client. This helps PHP web application by easily generating project architecture, MVC and other features.

- Zend framework uses the best practices of standard database programs and it allows PHP developers to build application models based on the database engines they require.

- Almost all major frameworks follow the concept of making your URLs look attractive and easy to access.

- Framework help the developers to write code faster and thus speed up the coding phase and yet manage quality, readability and versatility.

Disadvantages:

- Not well suited for rapid development ala Rails or Django .
- No ORM.
- When developers ignore the suggested "Zend" way of doing things it can get *very* messy
- It's not as loosely coupled as advertised. Just try making a Zend Framework MVC project without using Zend Loader.
- Slower than some other frameworks (but fast enough for 90% of websites, DB is almost always the bottleneck anyway).
- I've found the Zend Forms to be clunky and slow to implement.
- Last I used it, the Zend_Feed_Writer class didn't seem to work right.

Conclusion:

PHP developers tend to dislike ZF2 because it's "heavy-weight" and has a steep learning curve. The reason why most PHP developers dislike these "bulky" frameworks with steep (yes, it's true!) learning curves. Why spend time writing intelligible code when I can throw something together in a couple of minutes? ZF2 has the opposite philosophy: let's build maintainable, testable, secure web apps. It may take a longer, but it's more pragmatic.