International Conference "Risk in Contemporary Economy"
ISSN online 2344-5386 ISSN print 2067-0532
XIV[th] Edition, 2013, Galati, Romania,
"Dunarea de Jos" University of Galati – Faculty of Economics and Business Administration

ZEND FRAMEWORK

Lupaşc Adrian
Adrian.Lupasc@ugal.ro
Dumitru Eugenia Georgiana
Niculescu Ana-Maria
Faculty of Economics and Business Administration
Dunarea de Jos University of Galati

In this paper we present Zend Architecture, which is an open source technology for developing web applications and services, based on object-oriented components, and the Model-View-Controller architectural pattern, also known as MVC, which is the fundament of this architecture. The MVC presentation emphasises its main characteristics, such as facilitating the components reuse by dividing the application into distinct interconnected modules, tasks distribution in the process of developing an application, the MVC life cycle and also the essential features of the components in which it separates the application: model, view, controller. The controller coordinates the models and views and it's responsible with manipulating the user events through the corresponding actions. The model contains application rules, respectively the scripts that implement the database manipulation. The third component, the view represents the controllers interface with the user or the way it displays the response to the event triggered by the user. Another aspect treated in this paper consists in highlighting the Zend architecture advantages and disadvantages. Among the framework advantages, we can enumerate good code organization, due to its delimitation into three sections, presentation, logic and data access, and dividing the code into components, which facilitates the code reuse and testing. Other advantages are the open-source license and the support for multiple database systems. The main disadvantages are represented by its size and complexity, that makes it hard to understand for a beginner programmer, the resources it needs etc. The last section of the paper presents a comparison between Zend and other PHP architectures, like Symphony, CakePHP and CodeIgniter, which includes their essential features and points out their similarities and differences, based on the unique functions that set them apart from others. The main thing that distinguishes ZF from the others it's represented by the various components that help developers build efficient applications and create application models founded on the database engine they require. In the papers conclusion we argue the superiority of Zend architecture, by presenting solid motives that justify this affirmation, such as the implementation of the last features of HTML5 and PHP5.3, open source license, vast documentation, server support, large classes library and advanced security level.

Keywords: Zend Framework, MVC, Symfony, CodeIgniter, CakePHP, "glue" framework, PHP5, community, object oriented programming, Model, View, Controller, dynamic.

1. Introduction

Zend Framework represents an open source technology for developing web applications and services, based on components, which implements the MVC paradigm (Model-View-Controller), using the object oriented principle.

Developed by Zend Technologies, Zend architecture was launched in march 2006, under the New Berkeley Software Distribution License, an open source initiative, which states that all the code contributors, have to sign the Contributors License Agreement, based on the Apache Software Foundation's CLA, in order to prevent intellectual property issues.

The last stable version, Zend Framework 2.2.5, developed in PHP5 programming language and launched on 31.10.2013, uses 100% object oriented code and also the last PHP 5.3 features (namespace, dynamic binding etc.).

Simple and extensible, Zend represents a well documented software architecture tested by the members of Zend Community, who are permanently searching ways of improving it.

2. The structure of Zend components

Zend it is composed of object oriented components, from the standard library, which although have minimal interdependencies and can be used separately, they form a powerful web architecture when combined.

Components – organized by categories

1. Core – Zend_Loader, Zend_Exception, Zend_Version
2. MVC – Zend_Controller, Zend_View, Zend_Layout
3. Database Access – Zend_Db
4. Validation and filtering –Zend_Form, Zend_Validate, Zend_Filter
5. Authentication – Zend_Auth, Zend_Session, Zend_Acl
6. Web services- Zend_Http_Client, Zend_Rest_Route, Zend_Feed, Zend_Service
7. Utilities – Zend_Cache, Zend_Config, Zend_Registry.

3. MVC (MODEL–VIEW–CONTROLLER)

The main principle which is the fundament of Zend Framework, represents a high performance architectural pattern, MVC, which separates the application into 3 main components: the models, the views and the controllers.

The MVC pattern, introduced in 1979 by Trygve Reenskang, it's a very popular architectural design in nowadays web industry. Software engineers have adopted this technique because it represents an efficient method for developing and maintaining web applications.

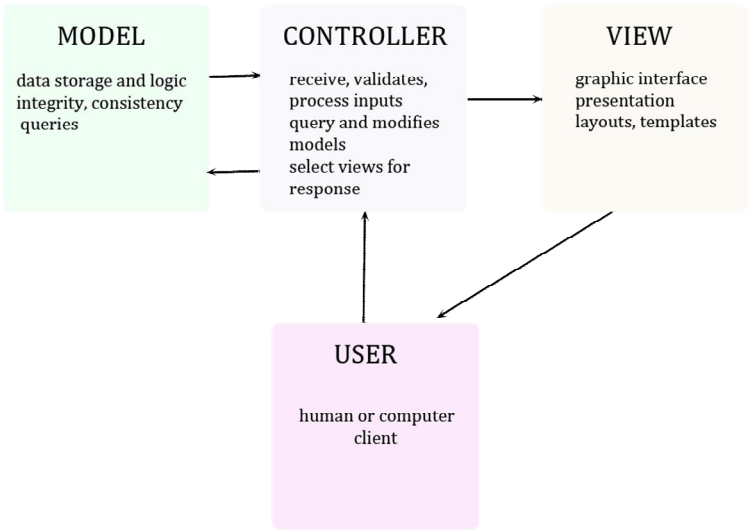


Figure 4 MVC Pattern
Sursa: *A. Padilla, Beginning Zend Framework, USA, 2009*

MVC CHARACTERISTICS

1. Facilitates tasks distribution in developing an application, which reduces the estimated time. For example, for those applications that have an user interface and an administration module, the interface developer will be in charge of the views, while the backend developer will create the controllers and models, which contain the business logic.
2. Components reusability, by separating the application into distinct interconnected modules, that can be integrated into other applications.

The MVC architectural design separates an application into 3 modules: controller, model and view.

CONTROLLER

The controller contains the code for manipulating all the user events and coordinates all the models and views, returning a response to the user.

In Zend architecture, every user event it is represented through a request, which it is forwarded to an action, that knows how to process it and every controller can have an infinite number of actions.

For example, the controller that allows an user to visualize all the system accounts or a certain account or to add a new account, will contain all the corresponding actions for these events, that will handle them.

MODEL

The next step the controller takes, after accessing one of it's actions is to process the information stored in a model.

The model contains domain specific instructions. In opposition to the controller, which directs the functionality of certain sections of the application, the model contains the business logic, which consists in application rules, materialized through the scripts that realize the connection to the database and process the create, read, update and delete commands, respectively, methods for extracting different information, such as data saved in a database, the content of certain xml files, etc.

For example, for logging into a system, we can implement a module which contains the user authentication logic.

The model main role is to separate the "business" logic, also known as the domain logic, from the rest of the application.

VIEW

The view represents the user interface of the controller, the way it displays the answer to events triggered by the user. For web applications, the view contains front-end code such as HTML code, CSS, Javascript, forms flash, images etc and also PHP code for rendering the logic such as the decisional instructions or the include() function for loading the menus, headers or external php files.

The user utilizes the views for filling and submitting forms, reading the content of a web page, clicking a button or interacting with a flash application. In principle, a view allows integrating different designs inside an application, without changing it's functionality.

MVC LIFE CYCLE

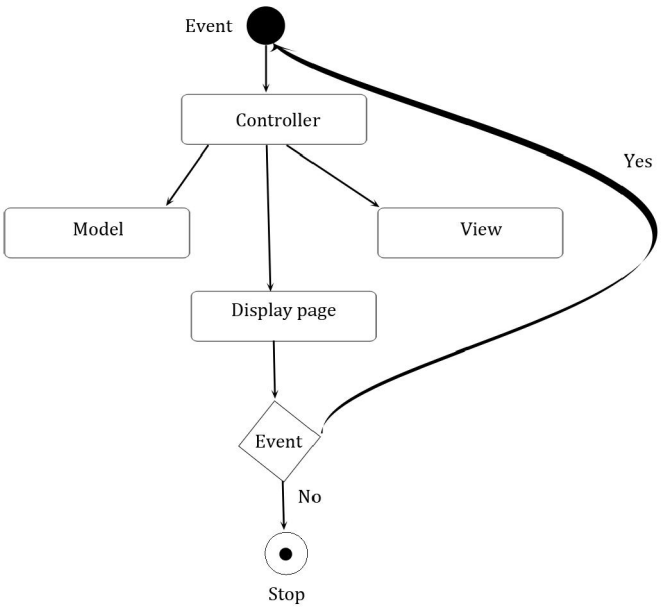


Figure 5 MVC Life cycle

EXPLAINING THE MVC LIFE CYCLE

The first step in MVC it's represented by the event triggered by the user (eg. accessing a web page).Events always appear by requesting a certain url by the user or another application, such as a web service.

The second step consists in translating that user event into a request to the controller. The controller than decides if it has to load a model in order to format, process or transform any information that the user has submitted or if the user wants only to visualize a page. The model returns to the controller the processed data.

Finally, the controller specifies the design that has to be displayed to the user, by extracting a view.

After displaying the view, the request leaves the MVC session and it waits for another request. If the user requests another page, the entire process is resumed until the user stops or leaves the application.

The controller contains different actions that allow the user to trigger certain events inside a module. It contains two default actions, init() and indexAction(). By default, these 2 methods are automatically created when we use Zend_Tool. The init() method executes before calling any other action. For example, if the user requests the index action, the init method will be initially processed, followed by the index action, which is the second default action.

ROUTING IN ZEND FRAMEWORK

All the actions that can be requested by the user by introducing their name in the browser, are public methods. The actions that are internally used by the controller, are private or protected, according to the security level.

Submitting a form, accessing a link, typing a url into the address bar of a browser are typical things that a user does while navigating the web and represent actions which are equivalent to url request which allow the connection with the action-controller pair.

One of the common problems in creating new actions for a controller it is represented by their name length, which makes it more difficult to access, when typing their name in a browser. The main components of a url are the domain, followed by the controller name and the action.

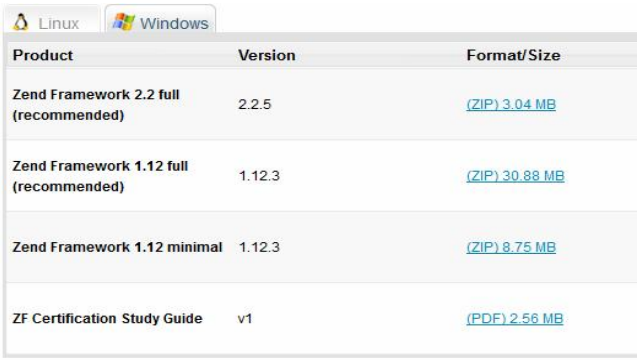
4. ZEND framework advantages

- Zend Framework represents **an open source architecture**, with minimal restrictions over the source code redistribution, which makes it **accessible** to all the web developers.
- Facilitates the organization of the source code into three distinct structures: presentation, logic and data, which leads to a **fair tasks distribution** in the process of developing a software application.
- Allows **code reusability**, by dividing it into components with little or no dependencies, which offers the possibility of including them into other applications.
- Offers **high quality code**, due to its strict standards and to the permanent testing and reviewing by the large community of users, according to the Contributor Guide, which states that code MUST be tested and covered by a unit test using PHPUni and must adhere to strict Coding Standards.
- Helps developers in producing **scalable and secure web applications** and services, due to its components that incorporate essential functions for a site and to the ones that have a high degree of customization, providing tools and measures to prevent and protect data.
- Zend Framework is a **fully object-oriented framework**, which makes most of all the ZF's components **extensible**, according to the inheritance concept. It allows developers to customize components in order to create functionality that is unique to their projects and also to use them in other projects as well.
- ZF is a **"glue" framework**, which unlike the full-stack frameworks that impose strict conventions on a project, is more flexible, allowing the use of components as individual libraries instead of the whole framework and also the integration of other libraries.
- ZF uses **test-driven development**, a programming technique that supposes writing the tests for a function before writing the code, which ensures the coverage of all the use case scenarios.
- Supports multiple **database systems**, including: MySQL, Oracle, IBMDB, MSSQL Server, PostgreSQL, SQLLITE etc.

5. Disadvantages

At first sight**, Zend Framework** doesn't seem to be very friendly, mostly because:

- **it is big** – it is composed of a large collection of classes, useful, but very hard to understand. It may be even impossible to be learned by a novice in web development;
- **it takes quite enough resources** (for example, the last version, that is 2.2.5 in ZIP format has 3.04MB). This disadvantage gives more credit to the affirmation "*software is more expensive than hardware is*";


Linux Windows

Product	Version	Format/Size
Zend Framework 2.2 full (recommended)	2.2.5	(ZIP) 3.04 MB
Zend Framework 1.12 full (recommended)	1.12.3	(ZIP) 30.88 MB
Zend Framework 1.12 minimal	1.12.3	(ZIP) 8.75 MB
ZF Certification Study Guide	v1	(PDF) 2.56 MB

- **it requires previous knowledges about design patterns** - it is knowen the fact that Singleton Pattern was the foundation of the first version of this PHP Framework. Singleton is implemented by the Front Controller, which is used inside the Model-View-Controller applications;
- **it also requires previous Object-Oriented-Programming knowledges** because all its components are entirely object-oriented and to be more précised, all its components are entirely PHP5 object-oriented;
- **it has a substantial documentation**, which is not such a bad thing because it is always better to have a choice than to have no information to use or to help whenever it is needed. The large number of pages from manuals, that are available on official websites, can be a sure source of intimidation and they can even make somebody turn around and give away the attempt to learn this PHP and MVC framework;
- **the update is not automatic** – the website administrator or a web developer has to be carefully when making the update whenever is needed. In Windows for example, the update and even the install need's to be made in command prompt (console);
- **the update requires a lot of attention** because sometimes certain aspects of the framework's behavior from the previous version can suffer major changes or even completely disappear in the current version. For example: Zend Framework 2 was completely rewritten and its Model-View-Controller layer suffered several changes, but despite these things, a projected built in the first version of Zend Framework must be able, and it does, to migrate to the latest one, which is currently 2.2.5;
- the web developers are creative people and they are often solving the same problem with different choices because they like variety; sometimes the web developers chose to ignore the framework's suggestions and to follow their instinct taking decisions on their own and writing the code that they want. In some cases**, the framework considers these decisions strange and rejects them**; In conclusion, a framework isn't really suitable for people who want to write their own solutions for things that Zend already has;
- some developer consider Zend Framework to be more a View-Controller rather than a Model-View-Controller and this is due to the fact that Zend doesn't come with any model implemented letting the developers to be the first ones to implement one.
- **it isn't suitable for any kind of application** – it is rather appropriate for large websites because it has a large number of classes, which are included and used very often, for everything.

6. ZEND framework and other frameworks

Before making any comparison of Zend Framework with other frameworks it is necessary to motivate the choice of a framework, whatever that may be.

Why a framework?

Of course a framework is not suitable for a website with only 4 or 5 pages, but it's an excellent choice for a web application which are composed of a collection of sessions, cookies, even forms and which are based on a consistent database. Speaking of forms…we all know that a form should be validated, secured and a framework defines the PHP code, generates the form as HTML code, validates and secures it also.

The most popular frameworks PHP, besides Zend Framework are:

- Symfony
- CakePHP
- CodeIgniter

Here are a few things about each of these frameworks:

Zend Framework

- was launched by Zend Technologies, the company which is behind the great PHP;

- has important partners like: Google, IBM, Adobe and Microsoft;
- was "born" in 2005;
- licence: BSD (Berkeley Software Distribution);
- it is a "glue" framework because it allows usage of its components either separately or together;
- it is well tested;
- has a large class library;
- its community is very big and very supportive;
- it also provides server support.

Symfony Framework

- was launched in 2005;
- is under MIT (Massachusetts Institute of Technology) licence;
- it must not be confused with Symfony CMS;
- uses the ORM (Object-Relational-Mapping) doctrine;
- also has a large library, but until the second version, there were some performance issues;
- has good documentation;
- it is supported by a pretty large community.

CodeIgniter Framework

- was launched in 2006 by EllisLab;
- under special licence of OSL (Open Source Licence);
- boasts performance and flexibility;
- good documentation;
- large community;
- has many libraries, from which unfortunately is missing a set of classes that would compose the library responsible with the authorizing.

CakePHP Framework

- as old as Zend Framework and Symfony;
- has another thing in common with Symfony Framework, which is: the MIT licence;
- has a large library;
- has many development tools;
- has good documentation;
- has a large community;
- its performance isn't as good as any of the other three framework listed above.

As is comes out from these four short overviews, these frameworks can be grouped into two categories of different complexity or by the preferences of web developers:

- *Zend Framework* and *Symfony Framework* – are both large systems, with many configurations and not very easy to understand;
- *CodeIgniter* and *CakePHP* – pretty approachable, even for beginners.

Because we already determined that Zend Framework and Symfony Framework are about on the same level of complexity and preference among web developers regarding the frameworks, we can discuss a little about the two technologies in parallel.

SIMILARITIES

- used for large web projects;
- a good choice in terms of developing complex websites;
- high performance;
  Example: the number of request per second from Apache Benchamrk are:
  - Zend Framework: 36
  - Symfony Framework: 39
- large communities;
- have thick documentations;
  Example: Zend Framework's 2.2.1 version's manual has almost 1500 pages.
- challenging for those who wish to learn one of these frameworks;
- PHP frameworks with MVC design;
- have large libraries;
- born in the same year, 2005;
- currently, each reached second version;
- open-source;
- have encouraging writing source code.

DIFFERENCES

- unlike his direct opponent (Symfony), Zend Framework needs -less memory, less resources. The last version of Symfony in ZIP format has a size of 9.3 MB, and the last version of Zend Framework also in ZIP format, has a size of 3.04 MB;
- the best thing about each of the two frameworks are, of course, different.
  So, Symfony Framework is counting on plugins developed by users, while Zend Framework is counting on its internal structure, on its architecture which allows the usage by will of its available resources.
  Precisely because of this, some projects developed in Symfony Framework were improved using some pieces of Zend Framework, to cover Symfony's gaps;
- Zend Framework requires the existence of previous knowledge of object-opriented programming (because everything in Zend Framewor is object) and some design patterns (for example: Singleton).
  Symfony Framework uses the ORM doctrine.
  Most of the time, web developers know not only PHP, HMTL, JavaScript and CSS, they also know other programming languages and programming paradigms, so it is more likely for them to know object oriented programming rather than ORM doctrine;
- although it has a great documentation, Symfony has enough gaps, so most of the times, the developers have to use directly the source code of some libraries and classes to get to understand them;
- some people consider an advantage for Zend Framework, the fact that it is the "baby" of Zend Technologies, which is also behind the great PHP.

Analyzing these aspects, we can certainly say that Zend is a better choice as framework than Symfony for developing a great, complex, dynamic web project.

7. Conclusions

In our opinion, if we had to chose from any PHP framework, we would definitely chose Zend Framework because:

- is working with HTML5;
- is open-source;
- is a "glue" framework;
- has a great community;
- has a helpful documentation;
- has server support;
- has a lots of useful classes and libraries.

The greatest thing about Zend Framework is that it is among the few PHP frameworks that uses dependency injection over composition as a design pattern (from Java).

In terms of security, no other PHP framework can beat up Zend Framework.

References

[1] A. Padilla, Beginning Zend Framework, USA, 2009, pp: 35-63
[2] http://en.wikipedia.org/wiki/Zend_Framework, date: 08.12.2013
[3] http://www.slideshare.net/shahar/zend-framework-components-for-nonframework-use, date: 08.12.2013
[4] http://net.tutsplus.com/tutorials/php/10-compelling-reasons-to-use-zend-framework/, date: 08.12.2013
[5] http://en.wikipedia.org/wiki/Symfony
[6] http://en.wikipedia.org/wiki/CodeIgniter
[7] http://en.wikipedia.org/wiki/Cakephp
[8] http://framework.zend.com/
[9] http://www.youtube.com/watch?v=A1dmxvN6vRs, date: 03.12.2013