



Zend Framework functionality and web application methodology

Mihai-Marius, Vlăsceanu

Bachelor's thesis of the Degree Programme in Business Information Technology
Bachelor of Business Administration

TORNIO 2012

ABSTRACT

KEMI-TORNIO UNIVERSITY OF APPLIED SCIENCES

Degree programme:	Business Information Technology
Writer:	Mihai-Marius, Vlăsceanu
Thesis title:	Zend Framework functionality and web application methodology
Pages (of which appendices):	87 (7)
Date:	December 4, 2012
Thesis instructor:	Johanna, Vuokila
<p>This research work is a methodology analysis of web frameworks with focus on the Zend Framework software. The objective of this thesis is to emphasize the business context that is suitable for using a web framework, to determine the rate of web developers that use a framework solution in their daily work and also to analyze the most important aspects of the Zend Framework software.</p> <p>With the increasing evolution of the web applications and the complex requirements made by clients of organizations regarding the final product it is important for web developers especially to stay current with the technology trends. With the respect to evolving technology it is preferable to concentrate on the aspects that the technology is capable of and its compliance with best practices and service quality. For that reason, this research work follows the Service-Oriented Architecture framework to assist in analyzing web application frameworks, the Zend Framework software in this case.</p> <p>Exploratory research is the core research methodology which is used to accomplish the objectives of this research work and answer the research questions. Additionally, the survey research methodology is used to answer some of the research questions. The data found within this research was collected from scientific sources such as printed publications and electronic books. While this research is largely theoretical, it also includes practical instructions regarding software use.</p> <p>Theoretical evidence is provided by this research work regarding the benefits and also downsides of using web application frameworks in web development projects. The Zend Framework provides a wide range of rather loosely coupled components which web developers can use either entirely or individually within their projects. Moreover, the web developers can also create custom components in Zend Framework. The rate of web developers that use a web application framework, not necessarily the Zend Framework, is relatively consistent among those with some work experience in web development. However, prospective web developers are also interested in experiencing the capabilities of web application frameworks in training sessions. Further research could emphasize ways to overcome the downsides of using web application frameworks in software development projects.</p>	
Keywords: web development, framework, Zend Framework, framework usage	

CONTENTS

ABSTRACT

FIGURES

TABLES

PICTURES

1 INTRODUCTION.....	8
1.1 Motivation and background.....	8
1.2 Objectives.....	9
1.3 Thesis structure.....	11
2 RESEARCH TOPIC, QUESTIONS AND METHODOLOGY.....	12
2.1 Research scope.....	12
2.2 Research questions.....	13
2.3 Research methodology.....	14
2.4 Theoretical framework.....	15
3 THE APPLICATION FRAMEWORK CONCEPT.....	18
3.1 Defining the framework concept.....	18
3.2 Frameworks in Software Projects.....	19
4 THE ZEND FRAMEWORK APPLICATION.....	22
4.1 The Zend Framework overview.....	23
4.2 The Zend Framework setup.....	25
4.3 MVC implementation.....	25
4.3.1 Introduction to MVC.....	27
4.3.2 MVC life cycle model in web applications.....	28
5 THE ZEND FRAMEWORK ARCHITECTURE.....	29
5.1 Building Projects in the Zend Framework.....	52
5.2 Security.....	53
5.3 Advanced components.....	58
6 FRAMEWORK USAGE SURVEY.....	64
6.1 Survey motivation.....	64
6.2 Survey methodology.....	65
6.3 Survey questions.....	66
6.4 Results.....	67

7 CONCLUSIONS.....	74
REFERENCES.....	77
APPENDICES.....	81

FIGURES

Figure 1. PHP Frameworks features comparison (PHPFrameworks.com 2012).....	10
Figure 2. SOA Governance Relationships (The Open Group 2009, 9).....	15
Figure 3. Key Leverage Points for Policies (SOA Governance) (Afshar 2007, 5).....	16
Figure 4. MVC life cycle model (Padilla 2009, 56).....	28
Figure 5. The Zend Framework MVC controller (Allen & Lo 2007, 11).....	29
Figure 6. Confirmation E-Mail in Development stage.....	31
Figure 7. Confirmation E-Mail in Production stage.....	32
Figure 8. Application environment variable (Gilmore 2009, 131).....	32
Figure 9. Bootstrap config (Gilmore 2009, 131).....	32
Figure 10. <i>config.ini</i> contents sample.....	33
Figure 11. Database parameters example.....	36
Figure 12. Database connection link in <i>bootstrap.php</i> (Gilmore 2009, 139).....	36
Figure 13. Create parameters and assign values to <i>Zend_View</i>	47
Figure 14. Manual rendering of a view script.....	48

TABLES

Table 1. Answers to question #1.....	44
Table 2. Age distribution of respondents.....	68
Table 3. Gender distribution of respondents.....	69
Table 4. Working status distribution among respondents.....	69
Table 5. Awareness of web application frameworks.....	70
Table 6. Experience distribution of web developers responding to survey.....	71
Table 7. Answers distribution regarding trainings.....	72
Table 8. <i>Zend_Tool</i> commands (Padilla 2009, 28).....	74

PICTURES

Picture 1. Zend Server versions comparison (Zend.com 2012).....	24
Picture 2. <i>fetchAll()</i> output example (Evans 2008, 78).....	38
Picture 3. <i>fetchAssoc()</i> output example (Evans 2008, 79).....	39
Picture 4. <i>fetchPairs()</i> output example (Evans 2008, 80).....	39
Picture 5. Turning on the database profiler.....	40
Picture 6. Database profiler view script (Evans 2008, 82).....	41
Picture 7. Database profiling report (Lyman 2009, 206).....	42
Picture 8. Render the profiler (Lyman 2009, 205).....	42
Picture 9. Enabling <i>Zend_Log</i> (Pope 2009, 113).....	43
Picture 10. Three-step views with <i>Zend_View</i> , <i>Zend_Layout</i> and <i>skins</i> (Lyman 2009, 19).....	51
Picture 11. <i>skin.xml</i> contents sample (Lyman 2009, 31).....	51
Picture 12. Document Root server variable (Lyman 2009, 7).....	52
Picture 13. Sample <i>Zend_Acl</i> Usage (Lyman 2009, 165).....	56
Picture 14. Sample <i>Zend_Acl</i> permissions usage (Pope 2009, 253).....	57
Picture 15. Front-end Cache Types (Padilla 2009, 360).....	59
Picture 16. Cache Record Types (Padilla 2009, 362).....	60
Picture 17. Search engine components (Padilla 2009, 318).....	61
Picture 18. Attributes for Fields in <i>Zend_Search_Lucene_Documents</i> (Lyman 2009, 172)	62
Picture 19. E-Mail sending code sample (Allen 2007, 14).....	63

1 INTRODUCTION

1.1 Motivation and background

The computer science has the reputation among the experts to pose issues regarding the mastering of large software projects. These issues also occur at the level programming level. In order to overcome these issues, a number of mastering techniques were developed. At the programming level the techniques that were developed include structured programming, logical programming or object-oriented programming. (Van der Aalst & Beisiegel & Van Hee & König & König & Stahl 2012, 1.) This research work concentrates on the concept of framework, which in the web development environment qualifies as computer software or web application framework. Moreover, the web application frameworks implement a consistent number of the mastering techniques discussed above, in order to overcome the issues occurring in large software projects. A web application framework is a collection of source codes “organized in a certain architecture that can be used for rapid development of web applications” (Porebski & Przystalski & Nowak 2011, 2). Specifically, this research concentrates on the Zend Framework software, a web application software created and developed by Zend Technologies Inc. The reason why the Zend Framework was chosen for the purpose of this research is due to its capabilities. The capabilities of the Zend Framework can help provide an enhanced perspective of the benefits that web application frameworks in general can provide to software development projects. The Zend Framework has its limitations similarly to competitive web application frameworks. However, the Zend Framework has the benefit of being a hybrid implementation suitable for small to large and complex software development projects. (Evans 2008, 17.)

The Zend Framework allows for developing web services and applications using the version 5.2 and later of PHP Hypertext Protocol (hereinafter PHP). The Zend Framework is an open source (hereinafter OS) application and it is integrally built by using Object-Oriented Programming (hereinafter OOP) code. Additionally, the Zend Framework allows the use of the components that form the libraries individually. (Zend Technologies Inc. 2010.)

One of the reasons why this research is done is on the grounds that it is important for the web developers and project leaders to understand the methodology and the use of web frameworks. In the same time that a web application is developed, the source code grows and the same parts of code may be reused in different parts of the application making it difficult to maintain. (Pack 2008, 5.) For that reason, this research aims to provide new perspectives to web development teams and project leaders from the technology point of view, a perspective that can help them provide reliable, efficient and fast software solutions.




















Another motivation of this research is the interest manifested by Zend Technologies in this research from the marketing point of view. I have personally made contact with Zend Technologies through their Marketing department. The representative of Zend technologies answered positively onto my request for support. The Zend Technologies Marketing department representative stated that Zend Technologies is interested in the outcomes of this research work and that they will use the outcomes for promotional purposes within the Internet environment. Zend Technologies, through the contact person, did not make any requirements regarding the contents of this research work. (Solomin 2012.)

An additional motivation aspect originates from my personal experience in web development. Despite the fact that I did not work in a company or organization in the software development area, I have personally developed from simple to complex web applications, some of them being currently used. I have experience of working with PHP for more than four years and I have never used a web application framework to create and develop my projects. The reason why I did not use a web application framework is due to the fact that I did not have sufficient information regarding web applications' benefits and how to use them. This research provides consistent data sufficient to be able to create a clear picture about web application frameworks and a starting point for using web application frameworks.

1.2 Objectives

The objective of this research is to emphasize the benefits and limitations of framework applications by emphasizing the Zend Framework functionality. In this way software

developers can understand if it is in their benefit to use a framework in the project they work on. This research also aims to help business organizations and educational institutions, such as Information Technology universities, to find out if the use of frameworks in their environment could contribute to their development. A list of different PHP framework applications and a comparison of features they offer is presented in Figure 1.

PHP Framework	PHP4	PHP5	MVC	Multiple DB's	ORM	DB Objects	Templates	Caching	Validation	Ajax	Auth Module	Modules	EDP
Akelos 	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
ash.MVC 	-	✓	✓	-	-	✓	✓	-	✓	-	✓	✓	-
CakePHP 	✓	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	-
CodeIgniter 	✓	✓	✓	✓	-	✓	✓	✓	✓	-	-	-	-
DIY 	-	✓	✓	-	✓	✓	✓	✓	-	✓	-	-	-
eZ Components 	-	✓	-	✓	-	✓	✓	✓	✓	-	-	-	-
Fusebox 	✓	✓	✓	✓	-	-	-	✓	-	✓	-	✓	-
PHP on TRAX 	-	✓	✓	✓	✓	✓	-	-	✓	✓	-	✓	-
PHPDevShell 	-	✓	✓	-	✓	✓	✓	✓	✓	✓	✓	✓	-
PhpOpenbiz 	-	✓	✓	✓	✓	✓	✓	-	✓	✓	✓	-	-
Prado 	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
QPHP 	✓	✓	✓	✓	-	✓	✓	-	✓	✓	✓	✓	✓
Seagull 	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
Symfony 	-	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	-
WACT 	✓	✓	✓	✓	-	✓	✓	-	✓	-	-	✓	-
WASP 	-	✓	✓	-	-	✓	✓	-	✓	✓	✓	✓	-
Yii 	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Zend 	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
ZooP 	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓	-	-

- **MVC**: Indicates whether the framework comes with inbuilt support for a Model-View-Controller setup.
- **Multiple DB's**: Indicates whether the framework supports multiple databases without having to change anything.
- **ORM**: Indicates whether the framework supports an object-record mapper, usually an implementation of ActiveRecord.
- **DB Objects**: Indicates whether the framework includes other database objects, like a TableGateway.
- **Templates**: Indicates whether the framework has an inbuilt template engine.
- **Caching**: Indicates whether the framework includes a caching object or some way other way of caching.
- **Validation**: Indicates whether the framework has an inbuilt validation or filtering component.
- **Ajax**: Indicates whether the framework comes with inbuilt support for Ajax.
- **Auth Module**: Indicates whether the framework has an inbuilt module for handling user authentication.
- **Modules**: Indicates whether the framework has other modules, like an RSS feed parser, PDF module or anything else (useful).
- **EDP**: Event Driven Programming. **New!**

Figure 1. PHP Frameworks features comparison (PHPFrameworks.com 2012)

Due to the fact that there are many available frameworks for developing PHP applications (Figure 1.), some software developers may find it difficult to make decisions regarding the choice of a framework or write their own framework. Making the wrong choice in this case can lead the software developer to a rather difficult task. Working with a framework that does not suit the project would require the software developer to write additional code, an aspect that would extend the time frame the

software development project would need to be executed in. Moreover, in software development projects there are situations that may suggest the use of a framework. However, the use of a framework may actually complicate the development process more than needed. (Ahamed & Pezewski, Alex & Pezewski, Al 2004.) For that reason, this research work provides an extensive analysis over the benefits and limitations web application frameworks can provide. In order to provide an accurate analysis that covers the variety of available web application this research concentrates on a hybrid web application framework: Zend Framework. (Evans 2008, 17.)

1.3 Thesis structure

To logically follow the objectives of this research the structure of this papers is as follows. The scope of this research is outlined in Chapter 2, among the core questions, research methodology and theoretical framework justification. The Chapter 3 introduces the software development business and presents the role of frameworks in software development projects. The Chapter 4 is entirely dedicated for justifying and analysing the results of the survey among web developers. The main point of this research is Chapter 5 where the Model View Controller and its implementation in the Zend Framework is analysed. The Chapter 6 concludes this research, discusses the results of this research and outlines suggestions on further research.

2 RESEARCH TOPIC, QUESTIONS AND METHODOLOGY

2.1 Research scope

The web developers represent a critical aspect within the web development process. Thus, the tools that the web developers use have a great influence on the development process. Therefore, it is important that the web developers have relevant information regarding the benefits and limitations of the tools they use in the development process. For that reason, this research aims to provide the information necessary to choose the right web application framework for the purpose of creating and developing web applications. This research does not seek to outline that the Zend Framework is the right web application framework to use for every web development project. What this research work aims to emphasize are the benefits of using a web application framework. Additionally, this research aims to outline in what cases a web application framework is suitable for use and also in what projects the Zend Framework is suitable. These cases are not specifically outlined throughout the chapters of this research mainly due to the variations of web development projects and customer requests.

Aside the analysis of the Zend Framework web application framework, this research also aims to emphasize the interest of potential, novice and experienced web developers of participating in training sessions where web application frameworks are subjects of. Through this academic aspect, this research aims to provide necessary information for academic organizations, such as universities, to expand their curricula with elective courses through which they would improve the skills of students that are interested in participating such courses.

This research is an in-depth analysis of the features that the Zend Framework web application framework provided through its Model View Controller. Through the analysis of the Model View Controller this research emphasizes the core libraries, directly connected components and the methodologies that the Zend Framework uses to enable its functionality. This research does not aim to teach the reader how to specifically use the libraries of the Zend Framework. However, a minimum emphasis on the methodology of implementing the Zend Framework in web application projects is presented. Thus, this research does not include source codes that would work by

themselves first and foremost considering that this is not a research work that can be defined as a tutorial. This research work includes pieces of code that are necessary to exemplify the use of the components of the web application.

2.2 Research questions

This research emphasizes the work-flow methodology of the Zend Framework and it also highlights extensively the most important and critical libraries of the Zend Framework. At the same time, this research puts into consideration different types of projects in which the Zend Framework would be suitable and the methodology to follow in choosing the Zend Framework for a software development project. This research concentrates on reviews of related literature and emphasizes PHP software projects which are ongoing, completed or even ended.

The aim of this research is to emphasize the benefits or, the contrary, the drawbacks of using a framework for developing web software applications, the Zend Framework in this case. Moreover, the information resulted from this research emphasizes the types of projects for which the use of the Zend Framework is recommended. Moreover, a small study on the percentage of software developers who use a framework is also conducted. The motivation to answer the research questions, the methods which were used to answer to them and, of course, the research questions themselves are discussed in the following paragraphs.

1. What are the advantages and disadvantages of using a framework in web applications development?

The advantages and disadvantages of using a framework is one of the main questions which needs to have a clear answer at the end of this research. From my experience, a large number of web software developers do not use and even do not know what a framework is. Hence, highlighting the advantages and disadvantages of using a framework provides the necessary information about this concept. Moreover, answering this question also enables decision makers to analyze the need and the appropriate web application framework for their projects.

2. What is the work-flow methodology of the Zend Framework?

Through work-flow methodology the functionalities, algorithms and data structure of the software is emphasized. The answer to this question emerges from exploring and experimenting on the Zend Framework software. There is no actual application built, but PHP scripting code samples are used in order to exemplify the Zend Framework functions.

2.3 Research methodology

This research is conducted based on the exploratory research methodology and it is based on the analysis of literature and article reviews. The use of exploratory research methodology is appropriate for this research as it helps creating a clear picture for software developers and decision makers about the use of frameworks in web applications development. To be more specific, in this research the Zend Framework software is studied in depth in order to provide clear understanding about the Zend Framework software and the framework as a concept. (McDaniel & Gates & Sivaramakrishnan 2008, 43.)

From my personal experience in web development and the web developers who start a career in software development field have little knowledge about the framework concept. Moreover, the situation is the same for prospective web developers. Hence, this research is targeted especially to web developers and people planning to start a new career in web development. Moreover, this research is structured in three stages. The first stage includes the exploration of the available information from different secondary sources. The functionality of web application frameworks is described within the second stage and the third stage experiments the functionality of the Zend Framework software based on the information gathered in the previous steps. Despite the fact that the research is structured in three stages, these stages are not clearly outlined; these stages are used as a composite.

Using the explanatory method, this research identifies the concept of a framework for the researcher. Further, this research provides practical opportunities for web developers

and organizations according with their needs. (Hair & Wolfinbarger & Money & Samouel & Page 2011, 147.)

2.4 Theoretical framework

Theoretical framework represents an important aspect of a research for the reason that it emphasizes graphically or in a narrative form the key aspects to be studied and the hypothetical relationships between them. Practically, it is an ensemble of assumptions, beliefs, expectations, concepts and theories used to inform and support the research. (Miles & Huberman 1994, 18.) In this research the Services-Oriented Architecture Framework Governance (hereinafter SOAFG) is used. For the purpose of providing an enhanced understanding of the SOAFG concept, this research provides explanations regarding the components of the concept, Service-Oriented Architecture (hereinafter SOA), SOA and SOA Framework respectively, in the following paragraphs.

SOA is an archetype for the management and utilization of distributed capabilities which might be controlled by different entities, such as people or organizations. From the perspective of businesses, entities create capabilities to support the needs of other entities. This aspect translated into distributed computing, represents one or more computers supporting the needs of other computers or systems. (OASIS Open 2006, 8.)

An appropriate view of SOA Governance is as the accession of the Business Governance and IT Governance to the SOA Governance and extending them (Figure 2). This way, it is ensured that the SOA benefits are reached. (Technical Standard - SOA Governance Framework 2009.)

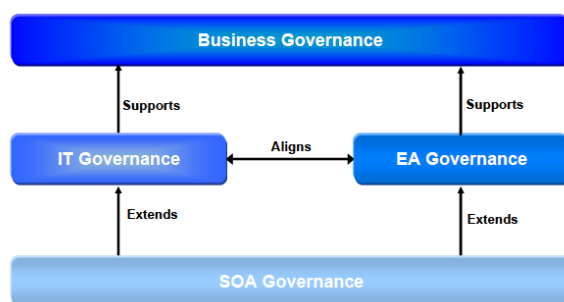


Figure 2. SOA Governance Relationships (The Open Group 2009, 9)

The purpose of SOA Framework is to enable organizations with the possibility to create their own customized and focused SOA Governance Model to prevent the creation of governance regimes that duplicates areas already covered (The Open Group 2009, 9). The Zend Framework has SOA implemented and that is the main reason why I choose to use the technical standards of SOA to analyse this software. The way how the SOA policies are covered by the Zend Framework from the different points of SOA key leverage structure, shown in Figure 3, such as infrastructure, architecture technology, project execution or information (Afshar 2007, 4) are to be highlighted throughout this research. Considering that the Zend Framework implements the SOA Governance Framework and it is also offered as a service to customers, this research follows the SOA Governance Framework to analyse the components of the Zend Framework. This research does not intend to show that the Zend Framework implements SOA Framework, but rather the methodology used to implement the SOA Framework. As it can be seen in Figure 3, SOA Framework has clear areas defined for organizations that provide software as a service to follow and some of these particular areas are studied.

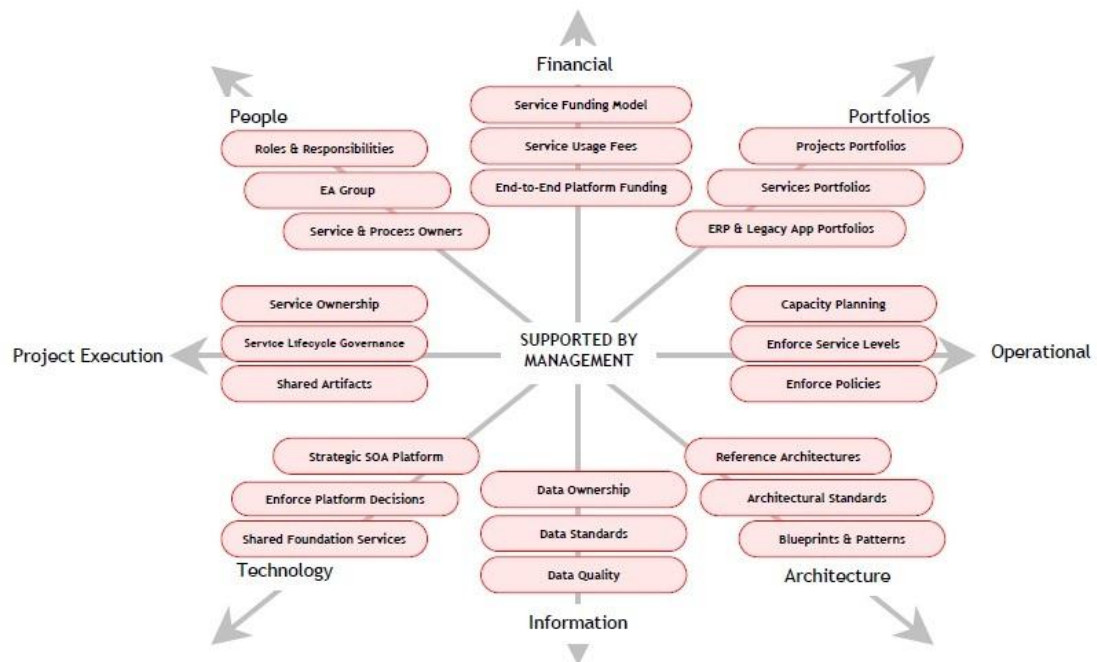


Figure 3. Key Leverage Points for Policies (SOA Governance) (Afshar 2007, 5)

The SOA Governance Framework consists of two main components. The first component is an SOA Governance Reference Model (hereinafter SGRM) which is meant to be a starting point. The last component of the SOA Governance Framework is SOA Governance Vitality Method (hereinafter SGVM). This last component implements an improvement process in order to provide focused Governance Regimen. (The Open Group 2009, 11.)

3 THE APPLICATION FRAMEWORK CONCEPT

Following theoretical framework that is used to conduct this research, SOA Governance Framework respectively, the indications theoretical framework provides does not put emphasis on the structure of research works that use SOA Governance Framework. However, the SOA Governance Frameworks provide standards that are used by the majority of software development organizations as the standards for their software development practices. Thus, this chapter defines the framework concept and highlights the role of frameworks in software development projects.

3.1 Defining the framework concept

The English dictionary available on the Internet (Cambridge Dictionary Online 2011), defines the concept of “framework” as ”a supporting structure around which something can be built” (Cambridge Dictionaries Online 2012). In the IT field, a framework is an ensemble of technologies, tools and languages which work together in order to build and deploy robust enterprise level applications in an accessible way. (Porebski et al. 2011, 2.)

When referring specifically to the web application frameworks the web developers with relevant experience in this area are aware of the fact that all web applications have common features. These features include aspects such as users who can authenticate, forms validation and security and data processing with databases. In individual applications these tasks can be done in simple and various ways. However, the development process could be simplified and accelerated if, by using abstraction, similar processes would be used not only in one web application, but also in future web applications. (Porebski et al. 2011, 37.)

This research does not intend to highlight that the Zend Framework represents the solution of all the phases of the development process, but rather how the Zend Framework can handle these phases. This way, decision makers may be able to have an enhanced perspective of when to use the Zend Framework. Moreover, the decision makers may also be able to consider using other web application framework for their projects than the Zend Framework.

3.2 Frameworks in Software Projects

Most of the business applications today have a software program attached to them as a critical component. The characteristics of the business application, such as the quality and the effectiveness, can make a difference between the success and the failure of the business solutions. Therefore, an important number of businesses acknowledge that if they want their business to stay on the market, they need to develop software applications that support the critical activities of the business. (Nienaber & Cloete 2003, 1.)

Within the last decades, software development projects have had a general tendency to fail, in terms of satisfying the user expectations, meeting the delivery times or staying within the assigned budget. This failure is still valid today and this issue is making the software developers and project leaders become increasingly concerned and active in order to change the situation. As a result, improvements were made in most of the areas of Software Project Management (hereinafter SPM) field. (Nienaber & Cloete 2003, 1.)

Software development does not only include technical details. Compliant software development practices involve issues that have managerial, organizational and cultural characteristics. These issues cover aspects which start from the structure of an organization, continue with the process of software development and ends with the standardization for a more efficient collaboration. (De Cesare & Lycett & Macredie & Patel & Paul 2010, 1.)

Software development practices have an important role in SPM and in software development. They are critical for the success or failure of a project. In order to build software applications which are successful, at a low cost and in a short period of time without making compromises regarding the quality of the software solution, specific techniques have to be emphasized. The aspects previously stated can be fulfilled by developing the process and the quality of the production regarding the software application. (Sharp & Ryan 2010, 8.)

A framework represents a tool or a set of tools that provides solutions that can be used in different problems regarding the development of software applications (Ahamed et al.

2004, 1). Hence, by providing reusable solutions, these tools or set of tools also provides the possibility to reduce the time frame in which the application is developed. The framework is an important and essential tool if the maximum utilization of the OOP features is taken into account, not only in an SPM, but in any software application development. (Ahamed et al. 2004, 2.)

Development of web applications makes no exception regarding the demand of characteristics such as low cost, reliability and fast production. The PHP is a widely used scripting language for developing software applications for the use on the web. The practice among the majority of the web developers, i.e. software developers for web software applications, is that they develop a User Interface (hereinafter UI) for the dynamic content. The UI is then combined with a back-end, an interface that the user does not see nor interact with, in order to provide dynamic content. Usually, the back-end interface is represented by a dedicated computer which handles the process of creating the dynamic content. (Trent et al. 2008, 166.)

According to the official website of PHP (The PHP Group 2012), there are two versions of the scripting language which are available and supported: version 5.3.11 and version 5.4.1. However, from my knowledge and experience as web developer of more than 4 years, there are still Internet Services Providers (hereinafter ISPs) and software developers that use older versions of PHP. These older versions of PHP are known among software developers and web experts to have security threats.

The PHP scripting language frameworks which are available on the Internet provide the software developers with the possibility to enhance the functionality of the scripting language with code libraries. The code libraries are collections of programming codes that can be reused in programming other web software applications, written in the C programming language code. (Trent & Tatsubori & Suzumura & Tozawa & Onodera 2008, 165.) For the web development purpose, there is a variety of PHP scripting language frameworks available over the Internet and some of them are provided free of charge, open source and also for a fee. As it was discussed before, this research focuses on the Zend Framework due to the fact that the Zend Framework provides a variety of scripting languages to be used making it suitable for large projects. The variety of scripting languages was a determinant factor in choosing the Zend Framework as a

research subject for this thesis. This factor was determinant because it makes the Zend Framework a hybrid web application framework which can cover a wider area of aspects than the competitive web application frameworks. (Evans 2008, 17.) The Zend Framework is provided for an annual fee and it is open source.

4 THE ZEND FRAMEWORK APPLICATION

The core of this research is discussed here i.e. the compound analysis from theoretical framework point of view, of the main components of the Zend Framework. Mainly analyzing the architecture aspect of SOA Governance Framework, this chapter also concentrates on other aspects of SOA Governance Framework, such as Project Execution, Operational aspect or Information aspect. However, these aspects are not specifically outlined throughout the chapter, but they can be identified in an accessible manner by the readers of this research work.

4.1 The Zend Framework Overview

The Zend Framework is a framework application which consists of broadly coupled components. The Zend Framework application is maintained and distributed by Zend Technologies USA, Inc. (hereinafter Zend Technologies) and the first version was released in 2005. Despite the fact that the Zend Framework is proprietary software, Zend Technologies is not the only company sustaining the development of the Zend Framework. After the first release, important companies, such as IBM, Google, Microsoft or Adobe Systems, joined the Zend Technologies in sponsoring the Zend Framework (Merkel 2010, 258.)

Following the information given by the Zend Framework project (Zend Technologies Ltd. 2012), the design characteristics and the benefits of using the Zend Framework include a rigorously tested code base, an extensible architecture and an increased level of interdependence between the components. Beside the previously outline features, the Zend Framework also provides the following:

- Web 2.0 technologies support, such as Asynchronous JavaScript and XML (hereinafter AJAX) and Web Services (Merkel 2010, 258)
- Continuous development of the components in the key areas, such as Model-View Controller (hereinafter MVC), database interaction or authentication, authorization and session management (Merkel 2010, 258)
- The use of coding standards and best practices through unit testing and the analysis of code coverage (Merkel 2010, 258)

- Extensive MVC support (Abeyasinghe 2009, 67)

Distinct separation between the presentation layer, business logic layer and data access layer. (Abeyasinghe 2009, 67).

4.2 The Zend Framework setup

At this point this research work outlined perhaps a clear idea about what the Zend Framework is capable of and how the prospective web developers of the Zend Framework could benefit them and in which context. However, regardless of the experience or expertise of the prospective users of the Zend Framework, there is one important step left which was not emphasized in the previous chapters, i.e. how does one start to use the Zend Framework and what are the requirements for it. (Padilla 2009, 1).

The primary step to be taken in starting using the Zend Framework is to deploy a web server that will be used to build and test the web application. There are several options to choose from, depending on the OS. A common option for the previously discussed OSs and also Mac OS and IBM I OS is the option provided by Zend Technologies, which incorporates the Zend Framework: Zend Server. At the time of writing this research, there are three versions available: Production Solution, Single Server and Community Edition. (Padilla 2009, 3.) A comparison of features that are provided with each version is presented in Picture 1. For the purpose of this chapter, the Community Edition (hereinafter CE) is used to demonstrate the setup process on a Windows OS.

The first step to start deploying Zend Server CE is to download the installation suite from the official site of Zend Technologies, download options available at <http://www.zend.com/en/products/server/downloads>. Continuing the process, the person deploying the Zend Server has the option to choose between a typical installation of the Zend Server and a custom installation. If the custom installation option is chosen, the person deploying the Zend Server can choose to install only the components that are needed. However, regardless of the components chosen to install, the following components are required to be installed:

- PHP

- Common Extensions
- Additional Extensions
- Zend Framework
 - Base
 - Extras
- MySQL server. (Padilla 2009, 3).

Zend Server Features	Production Solution	Single Server	Community Edition
Certified PHP	✓	✓	✓
Zend Framework	✓	✓	✓
Built-in DB connectivity (Oracle, DB2, MySQL,...)	✓	✓	✓
Java connector	✓	✓	✓
Debugger interface	✓	✓	✓
Bytecode acceleration	✓	✓	✓
Data Caching API	✓	✓	✓
Run Zend Guard encoded files	✓	✓	✓
Technical support	✓	✓	✗
PHP hot fixes	✓	✓	✗
Security updates	✓	✓	✗
Page caching	✓	✓	✗
Application deployment and rollback	✓	✓	✗
Application monitoring (alerting)	✓	✓	✗
Application problem diagnostics	✓	✓	✗
Code tracing	✓	✓	✗
Job queue	✓	✓	✗
Zend Studio integration for enhanced development	✓	✓	✗
Zend Download Server (Linux only)	✓	✓	✗
High availability	✓	✗	✗
Easy application scalability	✓	✗	✗
Multi-server management and monitoring	✓	✗	✗
Highly available job queue	✓	✗	✗
Cloud integration	✓	✗	✗
Metered billing options	✓	✗	✗

Picture 1. Zend Server versions comparison (Zend.com 2012)

Further, after choosing the needed components and clicking the *Next* button, the Apache installation follows. This is a critical step and choosing the right connection port is important. Generally, the default communication port 80 should be used. However,

there might be circumstances when the default port cannot be used as long as it is being used by another application. It is advisable that in case there are applications running on communication port 80 to be closed and the Zend Server installation to be continued. When the installation process ends, the default web browser of the OS is automatically opened and the server administration panel. The server requires the creation of a password for accessing the web server console. By the time the password is created Zend Server and the Zend Framework are ready to use. (Padilla 2009, 5.) Another option to set up the Zend Framework is to independently install the following components:

- Apache 2.2 or Microsoft IIS 5 or later
- MySQL 5.1 or later
- PHP 5.2.4 or later
- Zend Framework (Padilla 2009, 1).

4.3 MVC implementation

4.3.1 Introduction to MVC

When software applications are being built, the software developers encounter different types of difficulties, such as the decisions on suitable UI design, the right algorithms to use for different sections of the application or the modules to be used. The same difficulties may be faced by the web developers as well, considering the ever increasing use cases of PHP. Moreover, the increasing interest of different organizations in using the Internet for the benefit of their business also contributes to working difficulties of web developers. Accordingly, there is also about the same probability that there is at least one solution available to solve most of these difficulties. This fact gives the opportunity to web developers to benefit from those existing solutions instead of creating a new one, this way saving precious time. (Abeysinghe 2009, 7-29.)

One possibility which allows web developers to use the solutions to decision making difficulties within the web applications they build is the Model-View-Controller (hereinafter MVC) architecture. The MVC is an architectural pattern (Galloway & Haack & Wilson 2011, 2) utilized in the computer science having the most use cases in

software applications that involve UIs (Abeyasinghe 2009, 29). Initially named “Thing-Model-View-Editor” (Galloway et al. 2011, 2), the MVC has the role of dividing the concerns in the UI into three main tiers, as it can be seen in Figure 4 below..

The Model tier of the MVC, also known as the Data part (Lyman 2009, 22), provides the necessary rules for UI information management and manipulation and it consists of an assembly of classes emphasizing the characteristics of the respective information. This first separation of the MVC contains information regarding data retrieval from different media, such as a database or the information contained in an Extensible Markup Language file (hereinafter XML). (Padilla 2009, 55).

The View tier of the MVC, also referred to as the Presentation part (Lyman 2009, 22), outlines the methods to be used in presenting the UI. Considering that this research concentrates on web development, the View tier of the MVC would contain Hypertext Markup Language (hereinafter HTML) code, used to display web pages in web browsers. More than that, Cascading Style Sheets (hereinafter CSS) is used to describe the presentation semantics. The View tier of MVC in web development may also contain JavaScript code, a scripting language used for dynamic web pages, forms and pictures. (Padilla 2009, 55.)

Lastly, the Controller tier of MVC, also defined by some authors as the Business Logic (Lyman 2009, 22), governs the means of communication between the user, which represents the entity the UI is presented to, the application flow and the application-specific logic. (Galloway et al. 2011, 2.) Therefore, the Controller manages the View and the Model parts (Padilla 2009, 54).

The “entity” concept used in the previous phrase may not necessarily refer to a human being, but also to a virtual robot or computer software. The “entity” concept has the same definition throughout this research.

The MVC architecture illustrates the solutions for user interaction management. However, it does not provide in any way the process of how to operate the interaction with other application matters, such as data access. (Galloway et al. 2011, 2.)

The MVC architecture is becoming rapidly a standard in the area of developing web applications, with an important emphasis on development teams. The main reason why MVC becomes rapidly a standard architecture is due to its distinct separation of representation of information, making the development and maintenance of complicated web applications effective. Consequently, the members of development teams can concentrate on their area of expertise and not having to work with programming or scripting code not related with their areas. (Lyman 2009, 22.)

4.3.2 MVC life cycle model in web applications

In the previous subchapter the MVC architecture components were highlighted and described. The life cycle model of the MVC is discussed below and it is exemplified through a single request made by an entity to the web server. A web server is a dedicated computer which serves as a warehouse for Web site files used to make web sites available on the Internet. (Appu 2002, 3.)

As it can be seen in Figure 4, the MVC life cycle in web applications starts when an entity requests a web page from the web server by using a specific Uniform Resource Locator (hereinafter URL) (Padilla 2009, 56). In compliance with RFC 1738 (1994), a URL is a standard used in computer technology to provide a system with the possibility of performing a series of operations on web resources by abstracting the identity of the resource location.

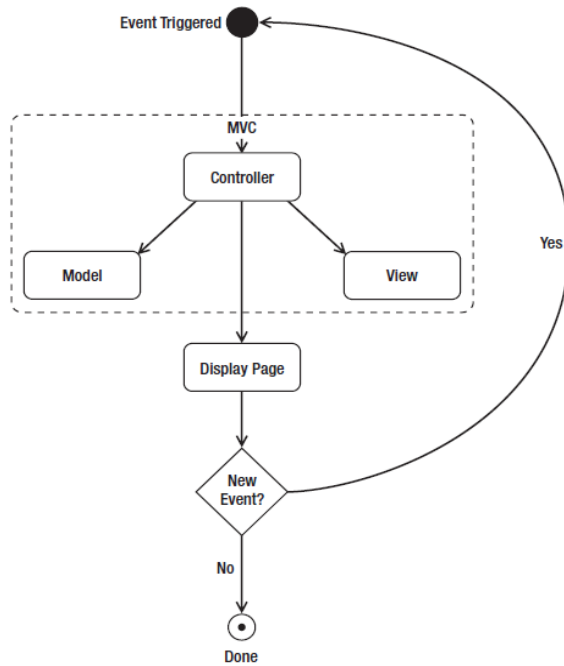


Figure 4. MVC Lifecycle Model (Padilla 2009, 56)

The MVC life cycle model illustrated in Figure 4 shows that the operation following the entity requesting a web page is the translation of the respective event into a controller request. This operation represents the application's state where the three main components of the MVC work together. The controller evaluates the type of the request the event-triggering entity made. In case that any data was submitted by the entity, the controller loads a pattern to process, format or transform the respective data. Otherwise, the controller considers the request from the entity as a request to visualize a web page. (Padilla 2009, 56.)

The last process in the MVC life cycle model is the controller displaying to the entity the data, in compliance with the type of the request the entity made, in a HTML format. At this process stage, the request of the entity is dropped and a new request from the entity is expected. The life cycle starts all over again if the entity requests a new page and it is repeated by the time the entity terminates the application. (Padilla 2009, 56.)

5 THE ZEND FRAMEWORK ARCHITECTURE

5.1 The Zend Framework controllers

The way that the MVC architecture previously outlined translates in the Zend Framework is discussed in this chapter. The role that the controller has in the Zend Framework is also emphasized. The MVC architecture of the Zend Framework follows the same pattern as it was outlined in the previous subchapters, having three main tiers, as it is shown in Figure 5.

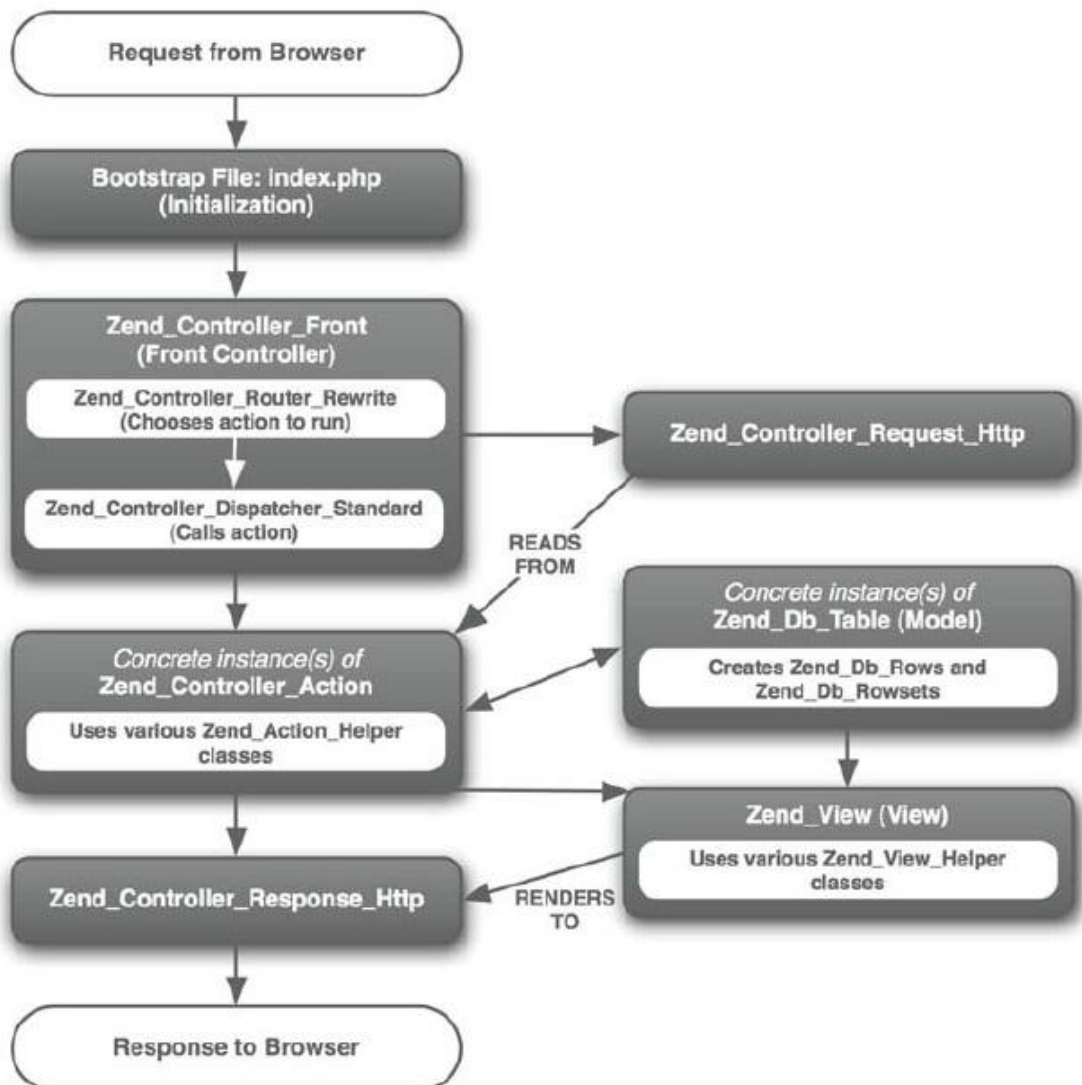


Figure 5. The Zend Framework MVC controller (Allen & Lo 2007, 11)

Zend_Config

One of the most important goals that web developers have in building web applications is to eliminate the action of repeating certain code logic within different parts of the application code. (Gilmore 2009, 127.) The need to eliminate the redundant code logic is at least necessary in the context of creating and using data at the configuration level in a web application. For example, database authentication parameters, the main e-mail address of the website and other parameters used to run the web application correctly might be subjects for redundancy. The web developers can choose to specify these parameters in every script they create. This type of practice is known as hard coding. The issue of this practice occurs when the web developer or the website administrator decides to change some of the parameters, such as the e-mail address of the website or the database authentication details. These parameters might have been used in tens or maybe hundreds of individual scripts located in different files. For this matter, the Zend Framework provides the web developers with an integrated component called *Zend_Config*. The *Zend_Config* component is meant to be used for parameters and other code types management, such as the ones previously discussed, in a single location. Practically, this single location is represented by a file named *config.ini*, located in the *application* directory of any web application built using the Zend Framework. (Gilmore 2009, 130.)

In the case of web applications that have a large scale there might be multiple environments of the web applications, also known as stages: the development stage and the production stage. For every one of the stages previously outlined the parameters for database authentication, for example, or the main e-mail address might be slightly different. The Zend Framework provides the means to manage the configuration data through the *Zend_Config* adapter. The configuration data can be stored in an *.ini* file formats, through *Zend_Config.ini* adapter or *.xml* file format, through *Zend_Config.xml* handler. Moreover, the configuration data can be stored in a PHP native file as an array. In this case, the Zend Framework handler used as a class object called *Zend_Config*. (Evans 2008, 128.) Additionally, using the *Zend_Config* feature to avoid redundancy of the code logic provides the web developers to switch between the *development* and *production* stages of the web application by simply switching the configuration files or modifying the single *config.ini* file (Gilmore 2009, 130).

To illustrate the difference between the two different environments of a web application and to show the usefulness of such a feature made available by the Zend Framework, suppose a student registration for re-examination at Kemi-Tornio University of Applied Sciences by e-mail address. Based on the examples emphasized by Gilmore (2009, 130), the differences between the *development* and the *production* stages are highlighted in Figure 6 and Figure 7, respectively.

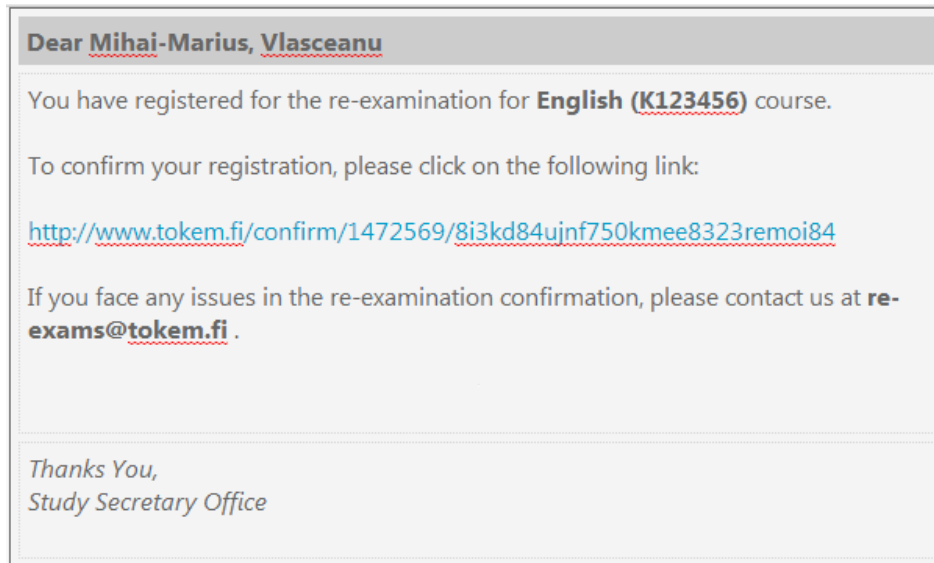


Figure 6. Confirmation E-Mail in development stage

As it visible in Figure 6, the hard coding practice is used within the *development* stage for testing purposed. However, the web developers must change every occurrence of hard coded parameters so that the script performs as it should be within the *production* stage. (Gilmore 2009, 131.)

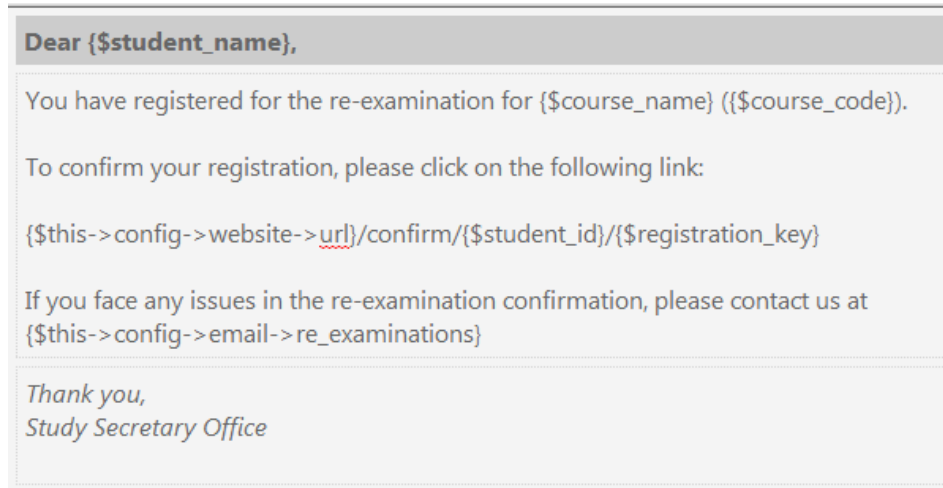


Figure 7. Confirmation E-Mail in Production stage

The Zend Framework application is capable to identify which configuration file to load or which set of configuration settings to use based on the value of a variable defined within the *bootstrap.ini* file, namely *APPLICATION_ENVIRONMENT*, as it is emphasized in Figure 8.

```
defined('APPLICATION_ENVIRONMENT')
    or define('APPLICATION_ENVIRONMENT', 'development');
```

Figure 8. Application environment variable (Gilmore 2009, 131)

Additionally, for the Zend Framework to identify the location of the *config.ini* file and to register the *\$config* variable into the registry of the website, two more code lines need to be added in the *bootstrap.ini* file. (Gilmore 2009, 131.) These code lines and their sequence are represented in Figure 9. The second code line in Figure 9 enables the use of the configuration variables within later parts of the website by retrieving them from the *Zend_Registry* component by using the command *\$this->config = Zend_Registry::get('config');* (Gilmore 2009, 131.)

```
$config = new Zend_Config_Ini('/home/webadmin/html/application/config.ini',
    APPLICATION_ENVIRONMENT);
Zend_Registry::set('config', $config);
```

Figure 9. Bootstrap config (Gilmore 2009, 131)

The *config.ini* file serves as the main file to store the configuration data for the web application built using the Zend Framework. The procedure to name the configuration variables or parameters is left up to the web developer. However, it is a common practice and it is also recommended to identify the parameters by using a dotted notation, e.g. *database.params.dbname = reexams_registrations*. Further, when the web developers need to refer to the respective parameter, they have to append *\$this->config* at the beginning of the command and replace the dots within the parameter with a right-facing arrow (*->*). Hence, the web developer would refer to the previously exemplified database parameter, *database.params.dbname = reexams_registrations*, by using the command *\$this->config->database->params->dbname*. (Gilmore 2009, 132.)

Within the *config.ini* file every parameter or variable is listed two times, each listing corresponding to the web application stage emphasized above. These occurrences are clearly delimited by two titles or sections within the *config.ini* file. The *production* stage parameter is marked within the *config.ini* file with the *[production]* title, whereas the *development* stage is marked with the *[development]* title. The *Zend_Config* component automatically selects the appropriate configuration data with respect to the *APPLICATION_ENVIRONMENT* variable setting. (Gilmore 2009, 132.) A possible content of a *config.ini* file is sampled in Figure 10. According to Gilmore (2009, 133) the implementation of this specific component into the early stage of a project is capable of providing time saving and decrease the level of inconveniences to the project.

```

; Production site configuration data
[production]
website.params.url = http://www.token.fi
database.params.host = mysql.token.fi
database.params.username = token_webadmin
database.params.password = token_encrypted
database.params.dbname = students
email.params.support = contact@token.fi

; Development site configuration data
[development]
website.params.url = http://test.token.fi
database.params.host = mysql.token.fi
database.params.username = token_web_dev
database.params.password = token_encrypted
database.params.dbname = students_dev
email.params.support = http://bugzilla.token.fi

```

Figure 10. *config.ini* contents sample

Zend_Db_Table

The Model tier from the general MVC architecture translates in the Zend Framework through a class object named *Zend_Db_Table*. However, this component can be represented by any data. (Lyman 2009, 22.) The *Zend_Db_Table* component of the Zend Framework represents a critical side of the software for the reason that it provides an effective, powerful and flexible connection with the database in order to integrate it with a website (Gilmore 2009, 135-139.)

Through the *Zend_Db_Table* component the Zend Framework provides the web developers with the possibility to avoid writing Structured Query Language (hereinafter SQL) statements, but leaving them the choice to do so if they need to. The *Zend_Db_Table* component provides the web developers an object-oriented interface which allows them to perform any kind of database operations, such as retrieving or inserting data. (Gilmore 2009, 139.)

The *Zend_Db_Table* component offers support for various database engines. Beside MySQL, database engines such as Oracle, Microsoft SQL Server PostgreSQL and SQLite are also supported. (Gilmore 2009, 139.) The *Zend_Db_Table* component also provides the business logic of the application which in most of the web applications is translated through the database.(Allen & Lo 2007, 11.)

In almost every web application development, regardless the complexity, the web developer needs to use a database in order to manage the data. This fact has an important impact on the time factor in a project. The web developers have to spend perhaps more time on creating specific ways to access and manage data from the database as they would spend on working on other parts of the application. In order to access and manage the data from a database, the SQL language must be used. Hence, the web developers must mix at least two different programming languages within the application, in this case PHP and SQL, in the same time. Considering the programming languages compound, the work efficiency of web developers is affected. Moreover, the purpose of using MVC, to distinguish the application into three tiers, interferes with the logic of the application. In order to overcome these obstacles, the *Zend_Db* component makes available to the web developers a programming strategy called Object-Relational

Mapping (hereinafter ORM). Besides avoiding the obstacles previously discussed, ORM is an intuitive and powerful strategy to use while not keeping the web developer from using the database at its entire capabilities. (Gilmore 2009, 137.)

The ORM provides the web developer with the possibility to interact with the database by using an object-oriented language written in the same programming language as the website, in this case PHP. Moreover, this programming strategy also allows the web developer to extend the basic database interaction behaviour, such as selecting, updating, inserting or deleting data, with more complex and custom SQL queries. In this way, the web developer can concentrate on the primary programming language that is used to power the web application. More than that, the isolation of the actions related with the database and enhancing the efficiency maintenance of the web application over time are provided. (Gilmore 2009, 137.)

In order to be able to perform different actions with a database, the first essential step to perform is to create a connection link between the web application and the database system. One way to perform this step is the method emphasized above, namely through the *config.ini* file. Within the *config.ini* file the parameters needed to create the connection link with the database system can be defined. In frequent cases there are four parameters needed to be defined in order to create the connection link. These parameters are the *hostname*, which represents the logical address of the database, the database *username* and the *password* needed for authentication and *database name*, which defines the name of the database to be used. As it is discussed above, since the *config.ini* file contains two sections, these parameters should be defined in both *[production]* and *[development]* sections. An example of a possible parameters definition within the *config.ini* file is presented in Figure 11. (Gilmore 2009, 139.)

```

[production]
database.params.host = localhost
database.params.username = tokem_admin
database.params.password = tokem_db_pw
database.params.dbname = tokem_edu

[development]
database.params.host = localhost
database.params.username = tokem_dev
database.params.password = tokem_dev_pw
database.params.dbname = tokem_edu_dev

```

Figure 11. Database parameters example

After the database connection parameters were defined, the next step is to use these parameters to create the connection link. The database connection link has to be created within a file called *bootstrap.php* and a possible way to achieve that is highlighted in Figure 12. (Gilmore 2009, 139.)

```

01 $config = new Zend_Config_Ini('/home/webadmin/html/application/config.ini',
02     APPLICATION_ENVIRONMENT);
03 ...
04
05 // Instantiate the database
06 $db = Zend_Db::factory('PDO_MySQL', array(
07     'host' => $config->database->params->host,
08     'username' => $config->database->params->username,
09     'password' => $config->database->params->password,
10     'dbname' => $config->database->params->dbname
11 ));
12
13 Zend_Db_Table_Abstract::setDefaultAdapter($db);

```

Figure 12. Database connection link in *bootstrap.php* (Gilmore 2009, 139)

The *bootstrap.php* file represents an important file within the Zend Framework due to the fact that the file itself contains the initialization of the all the components and modules the application uses. (Padilla 2009, 32.) The 01 line occurring in Figure 13 indicates the physical location of the *config.ini* file for the application to use and it also indicates which section of the *config.ini* file to use, in this case the *[application]* section. Further, the piece of code between the lines 06 and 11 represents the actual connection link. Within those lines the type of the database to which the connection link is created is transmitted to the Zend Framework. In the example emphasized in Figure 13 the connection link is created to a MySQL database system, fact indicated through

PDO_MySQL value of the first parameter of the *Zend_Db::factory* function. The second parameter of the *Zend_Db::factory* function represents an array consisting of the connection parameters. (Gilmore 2009, 140.)

The last line occurring in Figure 13, line 13, indicates to the Zend Framework that the connection link created above to be used as the default connection link for all the database interactions following that line of code. This particular line it is not mandatory to be used. However, using the respective function provides the web developer with the possibility of omitting the identification of the database connection link every time an interaction with database is needed within code. (Gilmore 2009, 140.)

Beside the parameters within the associative array in Figure 13, other optional parameters can be included in the same array for the connection's options. Such parameters include any of the following:

- Port

The “port” parameter can be optionally specified if the database server provides the options to run on a different connection port than the default one and the administrator chooses to change it

- Options

This particular parameter represents an array of generic options that are specific to all *Zend_Db_Adapter* classes, and it may contain the following values:

- *Zend_Db::FETCH_ASSOC*, which returns the data in an associative array
- *Zend_Db::FETCH_NUM*, which returns the data in an indexed array
- *Zend_Db::FETCH_BOTH*, which is a combination of the previous values and which returns the data in an array of arrays
- *Zend_Db::FETCH_COLUMN*, which returns the data in an array of values where each array element represents the value returned by each column in the result set
- *Zend_Db::FETCH_OBJ*. This option returns the data in an array of *stdClass* objects
- *Zend_Db::CASE_FOLDING*. This option has the capability to control the way the array keys is returned. It can take three valid values:

Zend_DB::CASE_NATURAL, which is the default value, *Zend_Db::CASE_UPPER* and *Zend_Db::CASE_LOWER*

- *Zend_Db::AUTO_QUOTE_IDENTIFIERS*. This option enables the possibility to use identifiers for column names, table names and aliases that represent reserved keywords within SQL syntaxes. (Evans 2008, 75.)

The *Zend_Db* adapter offers various methods of fetching data from the database, depending on the needs of the web developer. The following methods are available;

- *fetchAll()*

This method retrieves all the data from the database table and it returns it into an array of arrays. The root array has its index value increased by one for every row, but the index value is not related with the database table. An output example is shown in Picture 2.

```

1 Array
2 (
3     [0] => Array
4     (
5         [id] => 4
6         [emailAddress] => cal@example.com
7         [userPassword] => calevans
8         [firstName] => Cal
9         [lastName] => Evans
10    )
11    [1] => Array
12    (
13        [id] => 5
14        [emailAddress] => kathy@example.com
15        [userPassword] => kathyevans
16        [firstName] => Kathy
17        [lastName] => Evans
18    )
19 )

```

Picture 2. *fetchAll()* output example (Evans 2008, 78.)

- *fetchAssoc()*

This method is similar to the *fetchAll()* method. However, the root array takes the value of the primary key of the table. A possible output of the *fetchAssoc()* methods is emphasized in Picture 3.

```

1 Array
2 (
3   [4] => Array
4     (
5       [id] => 4
6       [emailAddress] => cal@example.com
7       [userPassword] => calevans
8       [firstName] => Cal
9       [lastName] => Evans
10    )
11   [5] => Array
12     (
13       [id] => 5
14       [emailAddress] => kathy@example.com
15       [userPassword] => kathyevans
16       [firstName] => Kathy
17       [lastName] => Evans
18     )
19 )

```

Picture 3. *fetchAssoc()* output example (Evans 2008, 79.)

- *fetchCol()*

The *fetchCol()* method is meant to return a single column. The data is returned in an array of which the index key is an incrementing integer unrelated with the database.

- *fetchPairs()*

This particular method assumes that the data which is fetched is from exactly two columns. The value from the first column, usually but not necessarily, the primary key, is placed in the array as the key, while the values fetched from the second column are set as the value in the array. An example output of the *fetchPairs()* method is shown in Picture 4.

```

1 Array
2 (
3   [4] => Cal
4   [5] => Kathy
5 )
6
7 OR
8
9 Array
10 (
11   [Evans] => Kathy
12 )

```

Picture 4. *fetchPairs()* output example (Evans 2008, 80.)

- *fetchRow()*

The *fetchRow()* method retrieves data only from the first row of the result set. If the result set has multiple rows, only the first row is returned. The result is output in a simple associative array, where the key is represented by the column name in the database and the value is the value within the column.

- *fetchOne()*

This last method is meant to return the value of a single row from a single column. The result is not output in an array. The result is a scalar value of the value of the specified column. (Evans 2008, 81.).

Profiler

The *Zend_Db* controller offers another functionality feature that may help web developers discover the performance issues within their web application and also debug SQL queries. This feature is the database profiler. The performance of the database profiler can be seen in big web applications where there are many SQL transactions and where the task of finding where a performance issue originates can be challenging. (Evans 2008, 81.) In such big web applications performance issues can occur when the web developer does not consider the decreased load performance of the web application (Lyman 2009, 203).

To enable the database profiler in the Zend Framework the *config.ini* file previously discussed must be opened and the line showed in Picture 5 written in. It is recommended, for security reasons, that the profile should be enabled only within the *[development]* environment section. (Lyman 2009, 203.)

```
1 database.params.profiler = true
```

Picture 5. Turning on the database profiler

Once the database profiler was enabled, the web developers need to create a view script for it and include it in the scripts they construct. In order to create a view script for the profiler, the web developers need to create a new file called *profiler.phtml* within the *application/views/scripts/util/* directory. Once the file was created the view script

retrieves the adapter of the current database and verifies if the profiler is enabled or not. The view script is executed only if the database profiler is enabled, doing nothing otherwise. (Lyman 2009, 204.) An example of a view script is shown in Picture 6.

```

echo "<hr /><table border='1'><tr><th>Query #</th><th>Time</th><th>Query</th></tr>";
$profiler = $db->getProfiler();
$totalTime = $profiler->getTotalElapsedSecs();
$queryCount = $profiler->getTotalNumQueries();
$longestTime = 0;
$longestQuery = null;
$count = 0;

foreach ($profiler->getQueryProfiles() as $query) {
    echo "<tr>";
    echo "<td>".$count++."</td>";
    echo "<td>".$query->getElapsedSecs()."</td>";
    echo "<td>".$query->getQuery()."</td>";
    if ($query->getElapsedSecs() > $longestTime) {
        $longestTime = $query->getElapsedSecs();
        $longestQuery = $query->getQuery();
    } // if ($query->getElapsedSecs() > $longestTime)
    echo "</tr>";
} // foreach ($profiler->getQueryProfiles() as $query)
echo "</table>";

echo 'Total Queries Executed : ' . $queryCount . "\n";
echo 'Total Time : ' . $totalTime . ' seconds' . "\n";
echo 'Average query length : ' . $totalTime / $queryCount . ' seconds' . "\n";
echo 'Queries per second : ' . $queryCount / $totalTime . "\n";
echo 'Longest query length : ' . $longestTime . "\n";
echo "Longest query : \n" . $longestQuery . "\n";
echo '</pre>';

```

Picture 6. Database profiler view script (Evans 2008, 82)

As a result of executing the view script within a web browser, the script outputs the data with respect to the construction of the view script. The table lines represented in Picture 7 are a possible example of such output.

Database Profiling Report

Total queries executed: 15

Total elapsed time: 0.011603116989136

#	Query	Time
(1)	connect	0.0011670589447021
(2)	DESCRIBE `pages`	0.0013909339904785
(3)	SELECT `pages`.* FROM `pages` WHERE (name = 'Installing Zend Server') LIMIT 1	0.00041699409484863
(4)	DESCRIBE `pages`	0.0011181831359863
(5)	SELECT `pages`.* FROM `pages` WHERE (((`pages`.`id` = 5)))	0.00032520294189463
(6)	DESCRIBE `content_nodes`	0.0012180805206299
(7)	SELECT `content_nodes`.* FROM `content_nodes` WHERE (`page_id` = 5)	0.0003509521484375
(8)	DESCRIBE `menu_items`	0.0012729167938232
(9)	SELECT `menu_items`.* FROM `menu_items` WHERE (menu_id = 3) ORDER BY `position` ASC	0.00031900405883789
(10)	DESCRIBE `pages`	0.0010108947753906
(11)	SELECT `pages`.* FROM `pages` WHERE (((`pages`.`id` = 5)))	0.00022411346435547
(12)	DESCRIBE `content_nodes`	0.00096702575683594
(13)	SELECT `content_nodes`.* FROM `content_nodes` WHERE (`page_id` = 5)	0.00035595893859863
(14)	DESCRIBE `menu_items`	0.001140832901001
(15)	SELECT `menu_items`.* FROM `menu_items` WHERE (menu_id = 5) ORDER BY `position` ASC	0.00032496452331543

Picture 7. Database profiling report (Lyman 2009, 206)

After the profiling view script was created, the web developers need to render it in the main layout of the web application. To perform this action, the web developers have to edit the *application/layputs/scripts/layout.phtml* file and add the lines shown in Picture 8. Ideally, the lines should be added immediately after the closing `</body>` tag. (Lyman 2009, 205.)

```

1 </body>
2 <?php
3     $this->addScriptPath(APPLICATION_PATH . '/views/scripts');
4     echo $this->render('util/profiler.phtml');
```

Picture 8. Render the profiler (Lyman 2009, 205.)

Another way of using the database profiler of the Zend Framework is through *Zend_Log*. The *Zend_Log* component enables the possibility that every SQL queries to be logged within the web browser's web developer add-ons, such as Firebug. The security recommendation of enabling the database profiler only for the development environment is still valid. However, for the *Zend_Log* component, it is not necessary to

edit the *config.ini* file. In this case, the file that needs to be edited is the *bootstrap.php*. As shown in Picture 10 a protected method will be created in the *bootstrap.php* file which will initiate the profiler. In the first instance, the script excludes the possibility that the current environment is the production one. Thereafter, the script calls the database adapter and then the profiler is added to the adapter with the constructor instruction to profile all the database queries. (Pope 2009, 113.)

```
protected function _initDbProfiler()
{
    $this->_logger->info('Bootstrap ' . __METHOD__);
    if ('production' !== $this->getEnvironment()) {
        $this->bootstrap('db');
        $profiler = new Zend_Db_Profiler_Firebug(
            'All DB Queries'
        );
        $profiler->setEnabled(true);
        $this->getPluginResource('db')
            ->getDbAdapter()
            ->setProfiler($profiler);
    }
}
```

Picture 9. Enabling *Zend_Log* (Pope 2009, 113)

Zend_Tool

Another important component of the Zend Framework is *Zend_Tool*. The *Zend_Tool* component is mostly important and it provides its best capabilities when it is used to create a project. Not only that it facilitates the work of the web developer by providing ease in coding, but it also helps the web developer to implement Rapid Application Development (hereinafter RAD) when the web developers create a project. (Padilla 2009, 27.)

RAD is an approach, or a development lifecycle, which provides the possibility to develop computer systems and important systems for organizations, from the strategic point of view, in a relatively short period of time. The development process is shortened by using powerful development techniques and methodologies. (Bellalcheru 2008, 3.)

By using the RAD lifecycle when creating a project in the Zend Framework, *Zend_Tool* helps the web developers to concentrate on the coding work, allowing him or her to let the controller create the foundations of a project, such as structures and files, automatically and free of bugs by using then command line tool. The available commands for using *Zend_Tool* are emphasized in Table 1. The *Zend_View* controller can create the following items:

- Directory structure of the project
- Actions
- Controllers
- Views
- Bootstrap file
- Details of the project. (Padilla 2009, 27).

Type	Parameters	Example Usage
Controller	create [name, indexActionIncluded=true]	zf create controller Account
Action	create [name, controllerName=index, viewIncluded=true]	zf create action list Account
Profile	show	zf show profile
View	create [controllerName,actionName]	zf create view Account list
Test	create [libraryClassname]	zf create test my_foo_baz zf disable test zf enable test
Project	create [path=null, profile='default']	zf create project /path/to/project

Table 1. *Zend_Tool* commands (Padilla 2009, 28)

When the web developers need to add functionality to the application they construct, *Zend_Tool* can be used to do it in a straightforward process. The process of adding new functionality to an application must be preceded by the creation of a new action and the associated view. While this action can be done manually by the web developer, *Zend_Tool* provides the possibility of doing it by using the commands emphasized in Table 1. For example, to create a controller, the web developer must open the command-line tool, such as *cmd* for Windows operating systems or Terminal for Linux

operating systems. Further, the navigation to the root directory of the project through the command-line tool is required. Supposing the new controller is called “students”, web developers have to enter the following command to create the controller: *zf create controller students*. Following this command, *Zend_Tool* controller creates a folder named *StudentsController*, which has the purpose of being used to hold the views of the new controller, as well as the index script for action/view. A default controller which holds all the main variables and parameters exists on within the root of the project and it is called *IndexController*. (Lyman 2009, 62.)

Further, supposing that need for registering new students, it is now needed to create an action for the controller that was previously created to handle this action. To create the registration action for the Students controller, the web developer must enter the following command into the command-line tool: *zf create action create students*. (Lyman 2009, 62.)

The *Zend_Tool* controller is provided within the Zend Framework latest releases. However, it is not automatically made available on the system the Zend Framework is installed and further action is required in order to be used. To enable *Zend_Tool* different actions need to be done, depending on the operating system. For Windows installations of the Zend Framework, the web developer or the system administrator needs to navigate to the *bin* directory of the download archive of the Zend Framework and locate the files *Zf.bat*, *Zf.php* and *Zf.sh*.

For Windows operating systems, the *zf.bat* and *zf.php* files are needed and it is recommended that the two files be placed or copied within the *PHP_HOME* directory. However, these files can be placed in any directory of the hard disk where the web developer has the execute permissions. For Linux operating systems, only the *zf.sh* and *zf.sh* files are needed and the placement recommendation stands. The next step of enabling *Zend_Tool* is to specify to *Zend_Tool* the location on the hard disk of the Zend Framework’s installation. This action can be done through several ways and also depending on the operating system. For Windows operating systems, it is recommended that the installation location of the Zend Framework is specified by setting an environment variable specific for the Zend Framework, called *ZEND_TOOL_INCLUDE_PATH*. However this also can be done by copying the Zend

Framework library into the *PHP_INCLUDE* directory. The downside of this second method is that web developers will have to maintain two copies of the Zend Framework. (Padilla 2009, 28.)

For Linux operating systems, the creation of a symbolic link within the *PHP_INCLUDE* directory pointing to the current location of the Zend Framework's library is recommended. (Padilla 2009, 29.)

Zend_View

The View component of the MVC implementation in the Zend Framework is translated through a class object of the Zend Framework called *Zend_View* (Lyman 2009, 24). In most cases where the *Zend_View* is used in web applications, *Zend_Controller*, the third component of the MVC implementation of the Zend Framework which is emphasized below, automatically creates a new instance of the *Zend_View* class object (Evans 2008, 74). The respective new instance renders the object class methods from the *Zend_Controller* to be used in processing the action request from the entity. HTML responses are then generated by *Zend_View* by using variables of which data is assigned from the model. *Zend_View* also includes in the response a series of tools and filters in order to improve the efficiency of the process. (Lyman 2009, 24.)

The *Zend_View* controller of the Zend Framework can be seen as a template system which separates the display logic away from the business logic. *Zend_View* can render, besides HTML code, Portable Document Format (hereinafter PDF) documents. Moreover, *Zend_View* can output data in other different formats, such as XML, or JavaScript Object Notation (hereinafter JSON). It is also possible to use an independent template system within the Zend Framework. (Lyman 2009, 74.)

Contrary to some of the previously discussed controllers, such as *Zend_Db*, where the initialization of the respective controller was done manually in *bootstrap.php* file, *Zend_View* controller is initialized automatically by the Zend Framework. To access this controller, the command *\$this->view* is used and parameters and values can be assigned to it. (Evans 2008, 59.)

To exemplify how to use the *Zend_View* controller, such as how to create variables and assign parameters to them, the two lines of code shown in Figure 13 creates two variables. *student_name* and *student_email* and assigns values to them. (Evans 2008, 60.)

```
$this->view->student_name = $student_name ;  
$this->view->student_email = $student->student_email ;
```

Figure 13. Create parameters and assign values to *Zend_View*

View Scripts

Within the *Zend_View* controller, view scripts represent PHP files containing the display logic of the template which are used by the controller to render the data processed by the actions of the controller or by the model created by the web developer. (Evans 2008, 60.)

The process of rendering the data from the view scripts includes two separate, but linked steps. Within the first step, a new instance of *Zend_View* is created with the data loaded from the action method. (Lyman 2009, 30.) An URI for the view script is being constructed by the *Zend_View* controller by examining the name of the controller and the name of the action, merging them together with the location (Evans 2008, 75).

When the action method has finished processing the data, the second step occurs, where *Zend_View*, triggered by the view renderer helper, renders the proper view scripts, returning the response to the response segment of the controller (Lyman 2009, 30).

For an enhanced illustration of this process, suppose the address <http://www.tokem.fi/student/login> is accessed by a student. The *Zend_View* controller will merge the default location for the view scripts, which is *apps/view/scripts*, with the name of the controller, which in this case is “student”, and the name of the action, in this case “login”. Further, the default file extension “.phtml” is appended which results in the following URI: *apps/view/scripts/member/login.phtml*. (Evans 2008, 60.)

Besides the automatic method of rendering emphasized above, there is also the manual alternative. The manual script rendering can be done by calling the *render()* method, which takes as a parameter the filename of the script to be called. A possible example of such a method is shown in Figure 14. (Evans 2008, 60.)

```
$this->view_render('student.php');
```

Figure 14. Manual rendering of a view script

The location of the *student.php* script is not random and yet very important. The web developer must place custom scripts, such as *student.php*, within the script paths where the *Zend_View* controller will search for them. Alternatively, the web developer can create custom directories within the scripts paths. In this case, the web developer must use some of the *Zend_View* methods to manipulate the controller so it searches the custom scripts in the custom directories. These methods are: (1) *getScriptPaths()*, (2) *setScriptPath()* and (3) *addScriptPath()*.

The second method, *setScriptPaths()*, allows the web developer to set where the *Zend_View* controller will search for view scripts, by changing the entire path. The third method, *addScriptPath()*, adds a new directory into the stack to which the *Zend_View* controller will search for view scripts, while the first method returns the current setting of the view scripts path. (Evans 2008, 60-61.)

As the data and parameters for *Zend_View* controller using dynamic content or content set by the web developer and *Zend_View* also outputs data in HTML format and other languages discussed in previous chapters, both the input and output data needs to be filtered and escaped, respectively. This happens for security reasons and to avoid eventual errors or anomalies. The *Zend_View* controller provides a method that enables the data filtering and escaping. That is the *escape()* method and, by default, it uses the *htmlentities()* PHP function to provide the output escaping. However, *Zend_View* also offers the possibility to manually configure the escaping behaviour when the *escape()* method is called. In order to benefit from this feature, the web developer must call the *setEscape()* method, where the function must receive two arguments: (1) a reference two an object and (2) the name of a method. As an example, for escaping the name of a

student from the class “Student”, the web developer would execute the following command: `$this->view>setEscape($Student, “escapeStudentName”)`, where `escapeStudentName` represents the name of the method where the web developer created a custom escaping procedure. (Evans 2008, 61.)

View Helpers

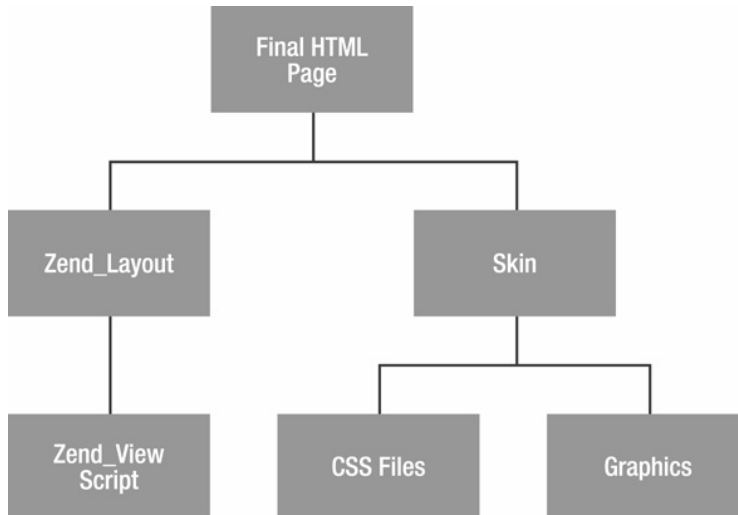
In addition to scripts, the `Zend_View` controller also offers helpers. Practically, the view helpers are class objects built by the developers of the Zend Framework. These classes help web developers to avoid repeatedly writing the same code over and over. (Lyman 2009, 38.) In fact, these helpers help the web developer create complex methods and widgets which they can later use within their code and also reduce the amount of code within the application. The Zend Framework offers by default a couple of view helpers which can be used immediately, such as the following:

- `formButton()`
- `formCheckbox()`
- `formFile()`
- `declarVars()`
- `formHidden()`
- `formLabel()`
- `formRadio()`
- `formReset()`
- `formReset()`
- `formSubmit()`
- `htmlList()`
- `url()`
- `formTextarea()`
- `formText()` (Evans 2008, 62.).

Zend_Layout

Despite the fact that the purpose of the Zend Framework (at least one of them) and its components which also emphasized throughout this research is to avoid using the same code over and over again, there is an issue within the Zend Framework which was not covered yet. The issue that might be noticed by some of the readers is within the view script. All the view scripts have the same header contents, at least the minimum of it. These contents might be the *DOCTYPE*, header scripts and so on and they repeat themselves in every view script. However, the Zend Framework offers a solution to this issue to. The solution is represented by the *Zend_Layout* component and this component is meant to solve a common issue which is known as *two-step views*, in older versions of the Zend Framework, and the *three-step views* in the latest versions. (Lyman 2009, 18.) What this component practically does is to maintain a consistent look of the web application design. The *Zend_Layout* component provides the means of defining default settings of the web application design which are built from common items that all the web application pages have without making changes to the underlying code base. (Evans 2008, 181.)

Within the Zend Framework, *Zend_Layout* component has its own MVC implementation which includes the front controller and a number of action helpers. (Lyman 2009, 18.) The use of *Zend_Layout* implies the use of so called “skins” (Lyman 2009, 18) which gives the possibility to change the skin of the web application, such as colours, without changing the layout. Picture 10 highlights the methodology of creating pages using the *three-step views* approach. (Lyman 2009, 18.)



Picture 10. Three-step views with Zend_View, Zend_Layout, and skins (Lyman 2009, 19)

The first step in creating a skin for a web application, the web developer needs to create a new folder under the *public/skins* folder which is recommended to be named with a short name for the new skin. As it can be seen in Picture 11, since a skin consists of graphics and CSS files, it is also recommended to create two other folders, such as *images* and *css*, to organize the files of the skin more efficient. The two folders will keep the CSS files and the graphics separate. (Lyman 2009, 31.)

Another required file for a skin is a *skin.xml* file contained under the directory of the skin. This file is used by a helper of the *Zend_Layout* component of the Zend Framework to get information regarding the skin. The helper is the *loadSkin()* method and what it basically does is to get information regarding what CSS file to load and also help switching between styles without the need to edit the *skin.xml* file every time, creating a dynamic layout this way. (Lyman 2009, 31.) A content sample of such *skin.xml* file is shown in Picture 11.

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <skin>
3    <stylesheets>
4      <stylesheet>layout.css</stylesheet>
5      <stylesheet>text.css</stylesheet>
6    </stylesheets>
7  </skin>
  
```

Picture 11. *skin.xml* contents sample (Lyman 2009, 31.)

As it is discussed above, the representation of data in a skin is done by using CSS language. Thus, the subsequent action to take after defining the skin data as shown in Picture 12 is to define the data representation elements through CSS. (Lyman 2009, 31.)

5.2 Building Projects in Zend Framework

To start building a project in the Zend Framework, the first step to follow is to create a new project. Usually, to create a new project in the Zend Framework, as it was pointed out earlier, the command-line interface is used. Once the command-line interface is started, the web developer needs to navigate to the root directory of the server where all the publicly accessible scripts are located. To create a project, the web developer needs to execute a simple command: `zf create project <project_name>`, where `<project_name>` represents the name of the new project. At this point, the new project is created and it can be accessed on the web server through the web browser by appending the name of the project to the address of the server (e.g. http://localhost/new_project/). A default page is displayed by accessing the respective address. To simplify the working environment, it is recommended that the Document Root, which is the default directory which the server will access when the root address is requested (e.g. <http://localhost/>), to be changed to the directory of the new project. To proceed with changing the Document Root, a server variable needs to be changed. This Document Root server variable is located in the `httpd.conf` file under the `apache2/conf/` directory. Picture 12 highlights the parts of the `httpd.conf` file that should be modified. After performing the modification of the server variable, a server restart is needed and the project can then be accessed by pointing the default address of the server. (Lyman 2009, 7.)

```
<Directory "{absolute path to apache}\Apache2\htdocs\zf_cms">
    Options Indexes FollowSymLinks
    AllowOverride All
    Order allow,deny
    Allow from all
</Directory>
```

Picture 12. Document Root server variable (Lyman 2009, 7)

The command that was previously executed also created the default directory structure for a project, besides creating a directory for the project. The default directory structure for a project in the Zend Framework consists of three main folders: *public*, a folder which will hold the future files that will be publicly accessible, *library*, which holds the library files of the Zend Framework and other custom libraries, and *application* directory. The *application* directory represents the core location of the project and within this directory all the view scripts, models and controllers will be stored. However, this is the standard, but not mandatory required, directory structure of a project and the web developers or the team of web developers can customize it as they need to. (Lyman 2009, 8.)

The public folder of the default directory structure of the Zend Framework is the Document Root of a project and it should contain only those files that must be accessible through the web browser. One of these files is the *.htaccess* file. The *.htaccess* is not directly accessible for users of the website and its' contents cannot be viewed by users. However, this file can be used to independently set and configure the *application environment*. Other aspects of the web server, such as redirects or URL rewrite rules can be configured within the *.htaccess* file. (Lyman 2009, 9.)

Another important file that has to be located and accessible within the *public* directory of a project is the index file. Within the index file, the *application* environment should be defined for the reason that, depending on the type of the request made to the web server, the *.htaccess* file will not be loaded every time. Besides defining the *application environment* the index file also ensures that the Zend Framework's library is loaded and application path is set. At the end, the index file contains a new instance of the bootstrap and it starts the application. Additional directories can be created within the *public* directory if needed. (Lyman 2009, 9.)

5.3 Security

The first concern when using a framework to build a project should be the security aspect. The tools that are used for creating an easy-to-use and manage a web application can become the target of hackers and if those tools are not properly secured the damages caused by a successful attack can be substantial. The Zend Framework has substantially

tested background and the developers working to this software treat security very seriously. Using the components that were created by the developers of the Zend Framework help web developers write more secure code than they would do without it. Among these security tools there are two components that play an important role not only in the Zend Framework, but also in many other applications in which security is an issue. These two components are *Zend_Auth* and *Zend_Acl*. (Lyman 2009, 145.)

The *Zend_Auth* component has the role of handling the user authentication and its persistence while *Zend_Acl* provides the means of managing the users, roles and their access to different resources. (Lyman 2009, 145.) Authentication represents a process through which the identity of someone or something is confirmed. (Pope 2009, 244.)

The use of *Zend_Auth* can become difficult or complicated for the reason that it allows the web developer to define a custom authentication method using custom adapters. (Evans 2008, 88.) the Zend Framework provides, by default, six authentication adapters from which the web developer can choose and, if needed customize:

- *Zend_Auth_Adapter_DbTable*
- *Zend_Auth_Adapter_Digest*
- *Zend_Auth_Adapter_Http*
- *Zend_Auth_Adapter_Ldap*
- *Zend_Auth_Adapter_OpenId*
- *Zend_Auth_Adapter_InfoCard*. (Pope 2009, 244).

Despite any adapter the web developer chooses, all of them return the same result which in this case it is stored in the *Zend_Auth_Result* instance, when the *authenticate()* method is called (Evans 2008, 88). The *Zend_Auth_Result* stores the authentication data even if the authentication process was not successful (Pope 2009, 244).

As it was emphasized earlier, besides handling the authentication process, the *Zend_Auth* also manages the persistence of the authentication, which translates into the time period for which the authentication is valid. (Pope 2009, 244.) The default authentication persistence time frame is the value of PHP session component, which, according with the manual of PHP, (PHP Group 2012) has a default value of 1440 seconds. The PHP session component has its own component in the Zend Framework

which is *Zend_Session*. The two components, *Zend_Session* and *Zend_Auth* are used together in the Zend Framework to keep the authentication persistence across multiple page requests. (Allen 2007, 12.)

The second component that was emphasized at the beginning of this subchapter is *Zend_Acl*. The *Zend_Acl* component represents the implementation of Access Control Lists (hereinafter ACL) on the Zend Framework. (Lyman 2009, 164.) ACLs, when referred to web development, provide the security means of defining the specific access rights of a user to the different interfaces and resources of a web application and also let the user act upon those specific access rights. (Evans 2008, 87.)

Zend_Acl is used in conjunction with *Zend_Session* and thus with *Zend_Auth* on the grounds that it uses the authentication information stored in *Zend_Session* to make decisions about granting access to restricted resources of a web application. The *Zend_Acl* decisions about granting access rights to restricted resources of a web application are based on a lists system, also known as Role Based Access Control Lists system. (Allen 2007, 12.) The Role Based Access Control Lists system consists of two types of lists: (1) Resources List and (2) Roles List (Lyman 2009, 164).

The items that may fall within the resources list are various. These items can be pages within the application, a database record, a text document, a picture or any other resource that can be accessed through a web server. (Allen 2007, 12.)

Under the Role List, in the case of a web application, the items that generally occur are the entities that were authenticated beforehand through *Zend_Auth* (Allen 2007, 12) and, within this list, access roles, such as administrator, are assigned to these entities (Lyman 2009, 164). The Resource List and the Role list and the relationship between them are under the management of *Zend_Acl* which provides a method, called *isAllowed()*, used to interrogate the lists.(Lyman 2009, 165.)

The first step in using the *Zend_Acl* component is to create roles. As it was previously pointed out, the roles generally represent the entities that use a web application. To create or add a role the *addRole()* method is used and it takes as an argument a new instance of *Zend_Acl_Role* class object. It is also notable that the roles within *Zend_Acl*

support inheritance, which means that a particular user role, such as the administrator role, can inherit and extend the access rights of a lower level role, such a user. (Lyman 2009, 165.)

The second step in implementing *Zend_Acl* within a web application is to pass the resources that are subject to controlling the access to *Zend_Acl*. This process uses the *add()* method of *Zend_Acl* and it takes as argument a new instance of the *Zend_Acl_Resource* class object. Further, the roles that have access rights or are restricted for accessing the resource have to be defined. Defining the roles access rights to resources implies the use of two methods of the same *Zend_Acl* component: *allow()* and *deny()*. Picture 13 demonstrates a possible implementation of *Zend_Acl* in a web application. (Lyman 2009, 165.)

```

1 <?php
2 $acl = new Zend_Acl();
3 // create the user role
4 $acl->addRole(new Zend_Acl_Role('user'));
5 // create the admin role, which inherits all of the user's permissions
6 $acl->addRole(new Zend_Acl_Role('admin'), 'user');
7 // add a new resource
8 $acl->add(new Zend_Acl_Resource('cms'));
9 // set access rules
10 $acl->allow('admin', 'cms');
11 $acl->deny('guest', 'cms');
12 // query acl
13 // this will print 'allowed'echo $acl->isAllowed('admin', 'cms') ? 'allowed' : 'denied';
14 // this will print 'denied'
15 echo $acl->isAllowed('guest', 'cms') ? 'allowed' : 'denied';

```

Picture 13. Sample *Zend_Acl* Usage (Lyman 2009, 165)

In addition to adding roles for a specific resource permissions can be also defined. The permissions can be applied not only for the resource as a whole, but also for parts of it and they either allow or deny the access to that part of the resource. For example, in the case of a bank, the clients are allowed to enter the building and access the services desks, but they are not allowed to access the safes. In order to define permissions to different parts of a resource, a third argument in array format has to be passed to the *allow()* or *deny()* methods when access rights are granted. Picture 14 demonstrates how to specify the permissions option when assigning role permissions to resources. (Pope 2009, 253.)


```
1 <?php
2 $acl->allow($visitor, $serverRoom, array('view'));
3 $acl->deny($visitor, $serverRoom, array('cabinet'));
4
5 $acl->isAllowed($admin, $serverRoom, 'view');
6 // returns true
7 $acl->isAllowed($visitor, $serverRoom, 'view');
8 // returns true
9 $acl->isAllowed($visitor, $serverRoom, 'cabinet');
10 // returns false
11 ?>
```

Picture 14. Sample *Zend_Acl* permissions usage (Pope 2009, 253)

The implementation of ACLs within the web application is a process that needs to be planned beforehand. Aspects such as the places where in the MVC the ACLs are applied, such as the Controller, Model or Domain layers, need to be considered and above that, the need of ACLs within the web application must be an aspect of planning. All the aspects of ACLs, such as roles and permissions are all managed in one place, meaning that ACLs are centralized. The reason why the implementation of ACLs in a web application needs to be planned beforehand is due to future implementations. Installing new modules can conflict with the current ACLs. (Pope 2009, 253.)

The common strategy used to implement ACLs is to control the access at the application level while using a centralized implementation of ACLs. The implementation process of centralized ACLs is commonly done at the stage where the bootstrap process is done. Thus, all the roles, resources and rules are created in the *bootstrap.php* file giving the possibility to place them in the registry of the Zend Framework or invoke them in the Front Controller. In either of these two cases, every resource request is routed through the Front Controller and then inspected to observe if the respective request passed through the ACL verification. No matter how convenient the implementation of this method looks like, it also has several drawbacks. One such drawback is that the lists can become large by time and thus hard to manage. Moreover, there are no means provided to implement ACLs in modules and the use of the Domain layer of the MVC in a web service would imply a new implementation of the ACLs. (Pope 2009, 254.)

Another strategy of implementing ACLs in a web application is to create an ACL for every module. As in the case of centralized ACLs, the process of implementing module specific ACLs also implies the use of the *bootstrap.php* file. The difference in this case is that every module would have its own ACL in the *bootstrap.php* file and that instead of using the Front Controller to examine every resource request an Action Helper would be used. While this implementation strategy of ACLs solves a number of drawbacks of the previous strategy, it does not come with a solution for the re-implementation of ACLs when the Domain is used in a web service environment. (Pope 2009, 254.)

A more complex strategy than those that were emphasized in the previous paragraphs, but which solves the issue of re-implementing the ACLs when the Domain needs to be used in a context outside of the controller architecture is to implement ACLs at the domain Layer. As in the case of module specific strategy, each module would have its own ACL. However, the management of ACLs in this case would be the responsibility of the Model component of the MVC and thus the ACLs would not be centralized anymore as long as the rules would be added to the ACLs by the Models. (Pope 2009, 255.)

5.4 Advanced components

Besides the components that were emphasized and described in the previous subchapters, the Zend Framework also offers several components that were not suitable to be described in the previous subchapters, but which they worth to be emphasized in this research. This subchapter aims to highlight several Zend Framework components that can prove helpful for web developers. (Allen 2007, 13.)

Zend_Cache

The first component of Zend Framework that worth to be highlighted is *Zend_Cache*. Generally, the *Zend_Cache* component provides the means of creating web applications that run fast, in terms of loading time. In order to provide this feature, *Zend_Cache* utilizes system resources, such as hard disks and shared memory, and even databases to cache data. (Allen 2007, 13.)

Caching represents a technology that uses different storage technologies to store resources, such as pages or page objects, which are frequently accessed by entities. These stored resources are placed in the proximity of the entities for fast access and improved experience. (Silicon Press, 1.)

The cache responsibilities are divided in *Zend_Cache* into two components: (1) *Zend_Cache_Frontend* and (2) *Zend_Cache_Backend*. (Lyman 2009, 208.) The *Zend_Cache_Frontend* component is responsible for caching general data, functions, classes and similar data. (Padilla 2009, 359.) The complete list of cache types available in *Zend_Cache* is available in Picture 15.

For the *Zend_Cache_Frontend* component there is a class object which helps to manage the resources that will be cached and also to provide information regarding the physical storage space available for caching. This class is the *Zend_Cache_Core* class and it provides a variety of methods available for cache management means. These methods a brief description is available in Appendix 2.

Cache Type	Class	Description
Core	<i>Zend_Cache_Core</i>	Parent class to all other <i>Zend_Cache_FrontEnd</i> classes; caches general data.
Class	<i>Zend_Cache_Frontend_Class</i>	Caches classes (objects) with their methods and properties.
File	<i>Zend_Cache_Frontend_File</i>	Caches files that will expire when the file is updated.
Function	<i>Zend_Cache_Frontend_Function</i>	Caches function calls.
Output	<i>Zend_Cache_Frontend_Output</i>	Caches any content displayed between the <i>Zend_Cache_Frontend_Output</i> methods <i>start()</i> and <i>end()</i> .
Page	<i>Zend_Cache_Frontend_Page</i>	Caches an entire page using the <i>Zend_Cache_Frontend_Page</i> <i>start()</i> method at the beginning of the content to cache. Impossible to use <i>end()</i> .

Picture 15. Front-end Cache Types (Padilla 2009, 360)

The second area of responsibility in terms of caching in *Zend_Cache* is represented by the *Zend_Cache_Backend*. While the other component, *Zend_Cache_Frontend*, specifies to *Zend_Cache* the data which will be cached, the *Zend_Cache_Backend*

specifies the methodology that is used to cache the respective data. As the *Zend_Cache_Frontend* does, *Zend_Cache_Backend* also consists of a number of six cache types, each having a corresponding class object. These cache types and their corresponding class objects are highlighted in Picture 16. (Padilla 2009, 361.)

Cache Record Type	Class	Description
File	Zend_Cache_Backend_File	Caches data onto the local file system.
SQLite	Zend_Cache_Backend_Sqlite	Integrates Zend_Cache into SQLite.
Memcached	Zend_Cache_Backend_Memcached	Integrates Zend_Cache into memcached.
Apc	Zend_Cache_Backend_Apc	Integrates Zend_Cache into APC.
Xcache	Zend_Cache_Backend_Xcache	Integrates Zend_Cache into Xcache.
ZendPlatform	Zend_Cache_Backend_ZendPlatform	Integrates Zend_Cache into ZendPlatform.

Picture 16. Cache Record Types (Padilla 2009, 362)

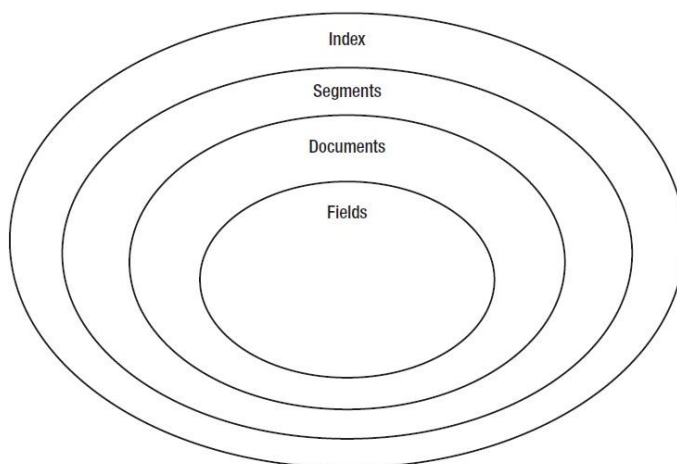
Zend_Search

Another component of the Zend Framework which worth to be emphasized is *Zend_Search*. Based on the Search Engine library of Apache software, also known as Lucene Search Engine, *Zend_Search* is a set of class objects that can help implementing a search engine in a web application. (Padilla 2009, 317.) The *Zend_Search* component offers the functionality and capabilities of widely known and commonly used search engines, such as Google or Yahoo. Likewise the popular previously discussed search engines *Zend_Search* provides support for ranked results, meaning that the results of search requests are returned according with their relevance with the search terms. (Allen 2007, 14.)

The Lucene search engine was developed originally in Java programming language as a full text search engine. Subsequently, the Lucene search engine was translated to other programming languages. (Lyman 2009, 192.) There are some aspects which differentiate the *Zend_Search* component from the popular search engines pointed out above through what *Zend_Search* is capable of doing. The first aspect that stands out of *Zend_Search*'s capabilities is the method of indexing the content of web application.

Zend_Search is not capable of automatically indexing the content of the web application. Web developers need to manually add resources to the index. This aspect does not necessarily represent a drawback, rather could be an asset as long as it provides flexibility in creating the index and also provides more choices about data sources from which indexes can be created than in the case of automatically creating them. (Lyman 2009, 192.)

However, a search engine, thus *Zend_Search* also, does not consist only of indexes. A search engine consists of a complex of several components which provide the proper functionality of the search engine. The index represents the main component of a search engine and it practically translates into a file consisting of a collection of documents of which data entities can search through. Further, indexes consist of segments, or sub-indexes, which can be independently searched. The segments extend the functionalities of documents, meaning that from the hierarchical point of view documents have a lower level than segments and thus than indexes. There is one more component of a search engine that is the lowest level from the hierarchical point of view. That component is strictly related to documents for the reason that the component is represented by the fields of the documents and they consist of content branched in items. A clear view of the components of *Zend_Search* is illustrated in Picture 17. (Padilla 2009, 317.)



Picture 17. Search engine components (Padilla 2009, 318)

In order to create a search index for the search engine, *Zend_Search* provides a straightforward process which involves the use of a static class method provided by the

class itself. The method that creates an index is the *create()* method to which the path of the directory in which the new search index is stored. (Lyman 2009, 171.)

Since *Zend_Search* utilizes documents as data to search through, after a new index was created, a new document must be created. The creation of a new document implies the instantiation of a class object called *Zend_Search_Lucene_Document*. While the document, which is the searchable data, consists of fields, the next step would be to add fields in the new document. The action of adding new fields in a document is done by using the *addField()* class method. The fields of a document differentiate themselves in five types: (1) keyword, (2) un-indexed, (3) binary, (4) text and (5) un-stored and each of these different categories can have one or a combination of attributes. These attributes are: “Stored”, which means that the fields are returned with the search results and they are stored in the index, “Indexed”, meaning that the fields are searchable and indexed, “Tokenized”, where the fields are broken in individual words, and “binary”, attributes which shows that fields are capable of storing binary data. Picture 18 highlights the field types and the possible attributes they can have. (Lyman 2009, 172.)

Field Type	Stored	Indexed	Tokenized	Binary
Keyword	Yes	Yes	No	No
Unindexed	Yes	No	No	Yes
Binary	Yes	No	No	Yes
Text	Yes	Yes	Yes	No
Unstored	No	Yes	Yes	No

Picture 18. Attributes for Fields in *Zend_Search_Lucene_Documents* (Lyman 2009, 172)

Besides the possibility of adding fields in a *Zend_Search* document, the same Zend Framework component also provides the ability to update, delete and optimize documents. The complete suite of methods that enables these additional functionalities is emphasized in Appendix 3. (Padilla 2009, 321.)

Zend_Mail

The *Zend_Mail* component of the Zend Framework is a powerful feature that offers the possibility to send e-mails in plain text or HTML format. (Allen 2007, 14.) *Zend_Mail* provides the accessible capabilities to use that would need the expertise of an experienced web developer to develop and implement. These capabilities include:

- Sending e-mails through SMTP and Sendmail
- Receiving e-mails through IMAP or POP3
- Using file attachments
- Support for all e-mail functionalities. (Padilla 2009, 228.)

Beside sending e-mail through SMTP and Sendmail *Zend_Mail* also allows for adding other e-mail transport protocols through the *Zend_Email_Transport_Interface* class object. A code sample of sending an e-mail through *Zend_Mail* is shown in Picture 19. (Allen 2007, 14.)

```
<?php
$mail = new Zend_Mail();
$mail->setBodyText('My first email!')
->setBodyHtml('My <b>first</b> email!')
->setFrom('rob@akrobat.com', 'Rob Allen')
->addTo('somebody@example.com', 'Some Recipient')
->setSubject('Hello from Zend Framework in Action!')
->send();
?>
```

Picture 19. E-Mail sending code sample (Allen 2007, 14)

There might be situations when the use of default SMTP settings may not be used. One such situation may occur when the web application is within the development stage and there are no e-mail servers available or installed. When such situation occurs it is recommended to use a remote e-mail server through the SMTP mail transport. (Lyman 2009, 202.)

6 FRAMEWORK USAGE SURVEY

With special emphasis on People, this chapter emphasize advanced aspects from the SOA Governance Framework which are discussed below. In software development process, which web development is also part of, people play the most important role. This chapter, through the survey that is conducted and analysed, aims to emphasize the information level and the opinion of the people involved in the web development process regarding the web frameworks. Both the information and the opinion aspect regard the web applications that people use in the web development process and also the other available web application frameworks. This is done by gathering and analysing data the respondents provide through the survey. Aside people, their knowledge regarding the technology available in web development, some aspects about respondents' portfolio and project execution in their cases are analysed.

7.1 Survey motivation

The focus of this research work on the functionality and application methodology of the Zend Framework might create certain ambiguity with respect to those the application addresses to. At the same time, questions are raised to and about the individual developers and organizations that already included the Zend Framework in their tools for developing web applications. These questions refer to the characteristics of those intending to use or already use the Zend Framework, such as age, experience, training, work environment or the complexity of their duties and even to their financial statute.

There are currently many web application frameworks available on the market that can compete with the Zend Framework. However, beyond the advantages and disadvantages that the competitors have, the Zend Framework has the advantage of being a hybrid application framework, an aspect which makes it usable in projects that are not limited to a relatively simple web application. (Evans 2008, 17.) Considering these advantages and disadvantages, it is important to stay updated with the current features of the framework applications. A clear distinction between how the framework applications performed in the past, including the preference of the web developers for framework applications, and how they perform in the present is highlighted for an enhanced

understanding of why, where and how frameworks are suitable. (Hamersveld & Van de Bont, 2008, 6.)

To connect the objectives of this research with the survey conducted in this chapter, the focus of the survey questionnaire falls on web developers. In an inquiry to the Zend Technologies Inc. Marketing department solicited by me, Zend Technologies Ltd manifested their interest into the results of the survey. The data I have provided to Zend Technologies Ltd via e-mail consisted of detailed information regarding the objectives of this research and the expected results, aspects highlighted in the very first chapter of this research.

The objective of this survey is to create a profile of the web developer that uses a framework in completing their duties. Moreover, this survey aims to find out if the web developers that do not use a web application framework within their work would be interested in using one. More than that, this survey aims to understand the interest of people starting a new career in web development of participating in trainings regarding the Zend Framework. The results of the survey also highlight the level of contentment that web developers and organizations that use the Zend Framework find their experience in using it and if they would recommend it to other web developers.

7.2 Survey methodology

The survey is an electronic based, closed web self-completion conducted type of questionnaire. This type of survey delivers the advantage of excluding the possibility that the answers provided by the respondents are biased. Moreover, the type of the survey makes it easier for the respondents to answer questions that might refer to aspects that are sensitive. A drawback of this survey methodology could be that there is the possibility that the respondents misunderstand the question. (Brace 2008, 29.) However, the questions composing the questionnaire have clear structures and they do not leave room for misinterpretation or ambiguity of the answers. Moreover, web based surveys have the advantage for the respondents that they can complete the questionnaire in their own time.

The web-based questionnaire is a common method of interviewing used by researchers at the time of this research work. Usually, the interviews are hosted on a website and

users are invited by different means to complete the questionnaire. (Brace 2008, 31.) The survey included in this research work follows the same strategy as stated in the previous phrase. The questionnaire uses the Google Forms service, a feature made available for free by Google Inc. at <http://www.google.com/google-d-s/forms/> and hosted on Google Docs service.

Brace (2008) paraphrases Kellner and Basi by emphasizing the idea that if there is no interviewer involved in the questionnaire process, the answers of the respondents is more accurate than in other types of surveys. Moreover, there is also less social desirability bias (Brace 2008, 32) in the answer of the respondents.

In order to reach the target population, the most convenient and accessible environment that was used was the Internet. It is widely known that the people can be easily reached through the online environment. Moreover, the web development communities reside within the online environment. Thus, the communication channels that were used to reach the web development communities were social media. Several social media networks, such as LinkedIn.com, Facebook.com, Stackooerflow.com and also several online forums were used to connect with the web developers and encourage them to participate in the survey.

7.3 Survey questions

The questionnaire contains a mixed type of questions. However, the majority of the questions are “closed” type with both multiple and single choice answers. Moreover, spontaneous types of questions are included, where the respondent is asked to give an answer in his own words. (Brace 2008, 45.) The survey questions can be viewed in Appendix 1. The first question of the questionnaire, “Are you a Web Developer”, has the role of a security question, also known as a screen question, to make sure that the respondent actually falls in the target sample of the questionnaire. Despite the fact that this questionnaire has a well-defined target and it is specifically delivered to the target sample by different means, it might happen that the respondent it is not actually a member of the target population sample. (Brace 2008, 38.)

The main questionnaire starts with basic questions, in order to make a profile of the respondent. The respondent is asked to select the range where his or her age falls into and the respondent is further asked to select his or her gender.

The questionnaire has a specific structure of in which order the questions are asked. The topics that the questions refer to have a specific type, starting from general questions, in this case about web application frameworks, to specific topic, such as Zend Technologies and the Zend Framework. (Brace 2008, 40.)

The topics of the questions are well-connected with the topic of this research and it also fulfils the marketing interest of Zend Technologies manifested in the inquiries specifically sought and obtained by the author of this research to the Marketing department of the Zend Technologies. The marketing aspect of this questionnaire is also reflected in the question where the respondents, regardless of their age or career point, are asked if they are interested to participate in trainings having the Zend Framework as a subject in an university environment: “Would you be interested in participating in trainings about the Zend Framework? “. From the marketing point of view, this question aims to emphasize if web developers or people starting a career in web development are interested in exploring the technology behind the Zend Framework.

7.4 Results

The initial consideration of taken before conducting the market research survey was that the necessary number of legitimate responses to the survey totalling a number between 150 and 200 responses can be met. However, despite the efforts put together to reach as much of the target audience as it was needed for this survey to reflect as relevant data as it can the objective was partially fulfilled. I have contacted various companies and magazines that might have been interested in this research, but unfortunately no answer was received and the final number of total responses to this survey was 81. However, despite the decreased number of responses gathered through the survey, some aspects and some general opinions and tendencies of the audience that did respond to this survey can be observed and it is less likely that in the case that the other half audience that did not respond to the survey would change the results in a dramatic manner. It is

likely that in the case of the questions that received answers close to even value of percentage, but they may also remain close one by another.

The analysis provided within the current subchapter uses the multivariate analysis technique for the reason that there are multiple variables within the questionnaire data to be analysed concerning the sample of observations. The questionnaire contains questions of which variable, such as the existence awareness of the Zend Technologies organization, represents a function for other variables contained in questions that succeed it. (Kothari 2004, 130.)

The question that opens the questionnaire is the question that makes sure that the respondent is positioned within the target audience of the survey: web developers. As it can be seen in Table 2, the rate of responses that came from web developers is 93, 83% which means that 76 of the respondents were actual web developers. Analysing the survey data from Appendix 4 it can be observed that the remaining percentage of the respondents which stated that they are not web developers or they are people working in a different field than software development.

<i>Are you a web developer?</i>		
Option	No. of answers	% of total answers
Yes	76	93,83
No	5	6,17

Table 2. Answers to question #1.

The second and third question of the questionnaire aims to draw a profile of the respondent. Analysing the questionnaire data it can be seen that most of the respondents were Males aged between 18 and 30 years. In case that the sample size response rate would have been met it is unlikely that this situation would change dramatically. The is because approximately 33% of respondents were aged between 18 and 24 years, followed closely by respondents aged between 25 and 30 years old, approximately 33%, with 96% of them being Males. Table 3 and Table 4 emphasize the entire data distribution of age and gender for all the respondents.

<i>Which of the following ranges includes your age?</i>		
Option	No. of answers	% of total answers
12-17	6	7.40
18-24	28	34.56
25-30	27	33.33
31-35	7	8.64
36+	13	16.04

Table 3. Age distribution of respondents.

<i>Please indicate your gender.</i>		
Option	No. of answers	% of total answers
Male	78	96
Female	3	4

Table 4. Gender distribution of respondents.

Question number 4 is another variable of the survey that aims to contribute to creating a profile of the respondent. The question number 4 aims to outline the working status of the respondent, the web developer respectively. Correlating this question with the previous profiling questions, related with the age and gender of the respondent, some clear aspects can be deducted. One clear aspect is that in most of the cases respondents are employees of a company, first and foremost due to the age group they belong to. Accordingly, the questionnaire data provided these facts indeed a consistent rate of respondents pointed out that they have more than one working statuses such as self-employed, volunteers, freelancers and students, for the younger respondents. Despite this variance, the majority of them have a common social working status, the fact that they are all employees and the distribution of answers of respondent's current positions is highlighted in Table 5.

<i>What is/are your current position(s)?</i>		
Option	No. of answers	% of total answers
Employed	43	53.08
Freelancer	28	34.56
Volunteer	6	7.40
Self-Employed	12	14.81
Student	27	33.33
Un-employed	4	4.93
Other	4	4.93

Table 5. Working status distribution among respondents

The questions 5, 6 and 7 aim to gather the data regarding the knowledge of the respondents about web application frameworks. The questions are dependent one by another and therefore the respondent has to answer the questions in the order they are displayed. Question number 5 aims to provide understanding if the respondent has any knowledge about existing popular web application frameworks including the Zend Framework. The web application frameworks that are textually displayed as an option to choose in the survey were elected randomly. The population sample from which the options were chosen is based on the list in Figure 1 in Chapter 1. Due to the fact that the question is a multiple choice type the answers also have variance. However, despite that most of the popular web application frameworks were listed, the respondents seem to be aware of the existence of every one of them. There is no option which did not get chosen within answers. Moreover, from the questionnaire data shown in Appendix 4 it can be observed that at least three of the options tend to be more popular than others. These three web application frameworks are the CakePHP, the CodeIgniter and the focus of this research, Zend Framework. With 70% or more awareness rate among the respondents of this research it is unlikely that the situation would change drastically if more respondents would have participated to this survey. Table 6 highlights the popularity of common web application framework among the respondents of the survey. The indicated numbers and percentages consist of all the answers that have the respective web application framework name in the respondent's answer.

Please check the boxes for the corresponding items in the following list if you know what it represents and what it is used for.

Option	No. of answers	% of total answers
CakePHP	76	93.82
CodeIgniter	80	98.76
Yii	45	55.55
DIY	26	32.09
the Zend Framework	75	92.59
Symfony	59	72.83
Ez Components	15	18.51
FuseBox	8	9.87
Don't know	9	11.11

Table 6. Awareness of web application frameworks

However, the questionnaire data shows that the respondents of the survey not only that they are aware about the existence of the respective web application frameworks, but it also clearly shows that most of them also use or have used these web application frameworks. With 70% of the respondents stating that they did use or used these web applications frameworks to perform their jobs indicates a consistent answer regardless the increasing number of respondents.

The question number 8 of the questionnaire has the purpose to provide data regarding the experience that the web developer has within the web development field. A comparison between the answers of this question and the data from the previous question concludes that the respondents with relevant experience in web development field also use a web application framework. The experience range of those who use or used a web application framework is between two and ten years. Thus, the experienced web developers use web application frameworks to perform their job. Table 7 shows the experience distribution of the respondents of this survey.

<i>How experienced are you in web development?</i>		
Option	No. of answers	% of total answers
Not experienced	3	4
Less than 1 year	6	7
1 to 2 years	4	5
2 to 4 years	24	30
4 to 10 years	29	36
More than 10 years	15	19

Table 7. Experience distribution of web developers responding to survey

Question number 9 aims to outline what respondents do know about frameworks in web development. The question is an open type where respondents can input their own answers. Generally, the respondent's opinion about web applications frameworks varies. Some of the respondents pointed out that through the frameworks they picture a tool which can be used to develop web application rapidly and efficiently. However, there are also opposite opinions. Some other respondents stated that the frameworks represent tools that are useless for web developers and using them is a threat against best practices. Other respondents showed their interest in using or learning to use web application frameworks. Thus, there is no unified definition opinion regarding web application framework from the web developer's point of view.

The five questions succeeding question number 9 are more focused on the topic of this research. They refer to the Zend Technologies Inc. and the Zend Framework. Question number 10 researches for the popularity of the Zend Technologies Inc. Given the gathered data, there is no doubt that Zend Technologies is a popular organization among web developers. To emphasize that, approximately 85% of the respondents to this survey stated that they know about Zend Technologies while 9% responded that they don't know about Zend Technologies and the rest of 6% were not sure.

Question number 11 focuses more on the Zend Framework. The respondents to this question were supposed to point out if they have used the Zend Framework. In the case if this question, question number 11, the gathered data is not quite conclusive due to the close values of the percentages. Therefore, 52% of the respondents of the survey stated that they never used the Zend Framework while the rest pointed out the opposite. Thus, this specific question might have other results if the target sample size would have been

reached. From this question, since questions 12, 13 and 14 relate with question 11, the same situation follows. Due to the fact that in questions 12, 13 or 14 there are no clear results and the percentages have close values, the results regarding these questions are inconclusive.

In the second last question, question number 15, the respondents were supposed to answer a question that is also closely related with the topic of this research. The respondents were supposed to state if they are interested in participating in trainings about Zend Framework. The gathered data may not be clear enough to provide a precise and definitive answer. However, given the percentage of 53 points can be observed a tendency of respondents to point out that they are indeed interested in participating to trainings regarding Zend Framework. Table 8 shows the distribution of answers regarding the interest in trainings about the Zend Framework.

<i>Are you interested in participating in trainings about the Zend Framework?</i>		
Option	No. of answers	% of total answers
Yes	43	53
No	23	28
Not sure	11	14
Other	4	5

Table 8. Answers distribution regarding trainings

The last question of the questionnaire was an open type question where the respondents were asked to add any further comments that they would like to regarding the questionnaire. Some of the respondents left some observations regarding the academic aspect of teaching programming languages, such as that it might be inconvenient to implement a new plan due to the time factor. Other respondents emphasized that implementing the Zend Framework within their organization's tools would be difficult due to the fact that the staff would need training as a result of basic knowledge of newly graduate students.

8 CONCLUSIONS

The Zend Framework is an entirely object-oriented framework. Thus, the Zend Framework utilizes a consistent number of object-oriented concepts, such as inheritance and interfaces. Moreover, considering the object-oriented implementation, the Zend Framework has most of its components extensible to some point. The Zend Framework also allows web developers to create custom variations of the components without the need to change the code base of the Zend Framework. These aspects allow the web developers to create unique functionality for a project, but which are not limited to that only project only. However, for the reason that in the Zend Framework there are only objects, has the downside of complexity in coding, at least for the un-experienced web developers. The Zend Framework can be viewed as a collection of low level interdependent class objects that can be used either as a whole or individually.

There is no unified opinion among web developers regarding the use of web application frameworks in general, in many cases due to the fact that web application frameworks are not perfect as. Thus, to answer the first research question of this research work, “What are the advantages and disadvantages of using a framework in web applications development?”, the answer is not simple. The answer to the question depends on the size of the project or application. If the project or application is relatively small or simple it is not advisable to use a web application framework as it may complicate the development process unnecessarily. The benefits of using web application frameworks are visible in complex and large projects where they can reduce the development time and the energy of the web developers. Moreover, since most of the web application frameworks follow coding best practices and conventions, the web developers will produce clean and easy to maintain code. A web developer that knows a web application well is capable of to create and develop a web application faster than without it and also to meet the deadline within projects. However, beside these advantages, there are some disadvantages of using web application frameworks as well. One disadvantage is that the novice web developers need time and training to understand web application frameworks and assessing its functions in web development process. Another disadvantage is that often the web application frameworks need to be extended to fit the needs of a project, mainly because every project is unique.

One of the aspects the decision makers have to decide in web development business is the tools that will be used by web developers to work in creating and developing web applications. Decision makers should gather specific information regarding the projects they will work on and make the appropriate decisions regarding the tools that will be used. Such tools include the use of web application frameworks in projects. It might be the case that the use of a web application framework would make the work of both web developers and project managers complicated. Even though this research did not specifically outline to the type of projects, web application frameworks should be used since some ideas can be drawn for what was highlighted in this research. This research work suggests that web application frameworks are mostly suitable for big projects. This is because the code structure of the web applications can increase in size and become complicated to manage and a consistent number of code parts are repeated throughout the web application code.

Further, this research emphasized the characteristics of MVC architecture, which separates the information representation that the web application provides into three layers. The information is separated from the user interaction point of view and the three layers consist of the data layer, business logic and controller layer.

This research also conducted a survey among web developers which aimed to understand if, why and how web developers use web application frameworks. Furthermore, the survey emphasized the prospective interest of web developers in learning to use and implement a web application framework in an academic environment. Despite the fact that the respondent rate is lower than expected, some conclusions can be drawn from there. Generally, the web developers that use a web application framework are the ones with some experience in the web development field and also those who work in an organization.

Aside the included survey, the main topic of this research was the Zend Framework. The Zend Framework is a web application framework produced by Zend Technologies and it was used in this research to emphasize the characteristics of a web application framework and to outline not only the benefits, but also the downsides that web application frameworks provide. Web application frameworks include a suite of tools and libraries that some users find useless, but which they can provide benefits in large

projects. This research shows that web application frameworks can provide rapid development to web application projects and also efficiency. Moreover, web application frameworks enhance the code management possibilities and scalability of the web application code base.

While some aspects of web application frameworks' benefits are clear, the limitations and the controversial opinions of web developers' regarding this type of software is present. These controversial opinions can be overcome by conducting further research and increasing collaboration with organizations that develop such software. The further research should aim to solve some of the downsides of web applications to encourage web developers to use web application frameworks. Solving these downsides makes it possible to implement academia training programmes in web application frameworks.

From the MVC implementation point of view, the Zend Framework does not have its own Model implementation, such as the CakePHP, CodeIgniter or Yii have. This aspect can make it difficult for the web developers used with web application frameworks that have their own Model implementation, i.e. CakePHP, CodeIgniter, to start using the Zend Framework. The problem is that the web developers need to be able to create their own Model and they need prior knowledge for that.

The decoupled components nature of the Zend Framework provides the possibility to integrate other libraries almost effortlessly. Likewise, the Zend Framework also allows web developers to integrate the Zend Framework in other libraries as well. For example, it is possible to integrate the Zend Framework into the CakePHP.

The Zend Framework represents only one of many options of web application frameworks that can be implemented and adopted by web developers. Further research should provide more analysis of the most popular web applications to enable web developers and decision makers to make the right choice for their projects.

REFERENCES

- Abeysinghe, Samisa 2009. PHP Team Development: Easy and Effective Team Work Using MVC, Agile Development, Source Control, Testing, Bug Tracking, and More. Packt Publishing Ltd, Olton Birmingham, GBR.
- Afshar, Mohamad 2007. SOA Governance: Framework and Best Practices. An Oracle White Paper. Oracle Corporation, World Headquarters, Redwood Shores, USA. Downloaded September 13th, 2012.
<<http://www.oracle.com/us/technologies/soa/oracle-soa-governance-best-practice-066427.pdf>>
- Ahamed, Sheikh I. & Pezewski, Alex & Pezewski, Al 2004. Towards Framework Selection Criteria and Suitability for an Application Framework. Marquette University USA, Proceedings of the International Conference on Information Technology: Coding and Computing, Milwaukee, USA. Downloaded April 30th, 2012.
<<http://ieeexplore.ieee.org.ez.tokem.fi/stamp/stamp.jsp?tp=&arnumber=1286492>>
- Allen, Rob & Lo, Nick 2007. Zend Framework in Action. Manning Publications, Greenwich, Connecticut, USA.
- Appu, Ashok 2002. Administering and Securing the Apache Server. Premier Press Incorporated, NIIT (Corporation) Staff, Independence, KY, USA.
- Bellalcheru, Girish 2008. Rapid Application Development [RAD] in Solution Delivery, Beyond Prototypes. Downloaded October 29th, 2012.
<<http://www.lytecube.com/whitepapers/rad.pdf>>
- Berners-Lee, Masinter & McCahill, M. 1994. RFC 1738 Uniform Resource Locators (URL).Network Working Group, University of Minnesota, USA. Downloaded September 22, 2012.
<<http://www.ietf.org/rfc/rfc1738.txt>>
- Brace, Ian 2008. Questionnaire Design: How to Plan, Structure and Write Survey Material for Effective Market Research (2nd Edition). Kogan Page Ltd., London GBR.
- Bureau of Labor Statistics 2011. Current Employment and Wages from Occupational Employment Statistics (OES) survey. United States Department of Labor, July 2011. Downloaded September 8th, 2012.

<<ftp://ftp.bls.gov/pub/special.requests/oes/oesm11nat.zip>>

Cambridge Dictionaries Online 2012. Definition of framework noun from the Cambridge Advanced Learner's Dictionary & Thesaurus. Cambridge University Press. Downloaded May 3rd, 2012.

<<http://dictionary.cambridge.org/dictionary/british/framework?q=framework>>

De Cesare, Sergio & Lycett, Mark & Macredie, Robert D. & Patel, Chaitali & Paul, Ray 2010. Examining perceptions of agility in software development practice. Communications of the ACM Volume 53 Issue 6, ACM New York, NY, USA. Downloaded April 30th, 2012.

<http://dl.acm.org/ft_gateway.cfm?id=1743580&ftid=808221&dwn=1&CFID=96897677&CFTOKEN=99457834>

Evans, Cal 2008. PHP architect's Guide to Programming with Zend Framework. Marco Tabini & Associates, Inc., Toronto, Canada.

Galloway, Jon & Haack, Phil & Wilson, Brad 2011. Professional ASP. NET MVC 3. Wrox, Hoboken, NJ, USA.

Gilmore, W. Jason 2009. Easy PHP Websites with Zend Framework. W.J. Gilmore LLC, Columbus, Ohio, USA.

Hair, Joseph F. & Wolfinbarger, Celsi Mary & Money, Arthur H. & Samouel, Phillip & Page, Michael J. 2011. Essentials of Business Research Methods. M.E. Sharpe, New York, USA.

Hamersveld, Mario & Van de Bont, Cees 2008. Market Research Handbook (5th Edition). Wiley, Hoboken, NJ, USA.

Kothari, C.R. 2004. Research Methodology: Methods and Techniques. New Age International, Daryaganj, Delhi, IND

Lyman, Forrest 2009. Pro Zend Framework Techniques: Build a Full CMS Project. Apress, New York, USA.

McDaniel, Carl & Gates, Roger & Sivaramakrishnan, Subramanian 2008. Marketing Research Essentials, Canadian Edition. Wiley, Vancouver, Canada.

Merkel, Dirk 2010. Expert PHP 5 Tools. Packt Publishing Ltd, Olton Birmingham, GBR.

Miles, Mathey B. & Huberman, A. Michael 1994. Qualitative Data Analysis: An Expanded Sourcebook (2nd edition). Sage Publications, Inc, California, USA.

National Statistics Center 2012. Labour Force Survey. Ministry of Internal Affairs and Communications, Tokyo, Japan. Downloaded September 29, 2012.

<<http://www.stat.go.jp/english/data/shakai/2001/shousai/jikan/zuhyou/a001-2.xls>>

Nienaber, Rita & Cloete, Elsabe 2003. A Software Agent Framework for the Support of Software Project Management, University of South Africa Downloaded April 28, 2012.

<<http://dl.acm.org.ez.tokem.fi/citation.cfm?id=954014.954017&coll=DL&dl=ACM&CFID=79516628&CFTOKEN=41575649>>

OASIS Open 2005. OASIS Reference Model for Service Oriented Architecture 1.0 2006. Downloaded July 19th, 2012.

<<https://www.oasis-open.org/committees/download.php/16587/wd-soa-rm-cd1ED.pdf>>

Porebski, Bartosz & Przystalski, Karol & Nowak, Leszec 2011. Building PHP applications with Symfony™, CakePHP, and Zend® Framework. Wiley Publishing, Inc, Indianapolis, Indiana, USA.

Padilla, Armando 2009. Beginning Zend Framework. Apress, New York, USA.

Pack, Richard 2008. Choosing Your Java Based Web Framework: A Comparison. 2008 JavaOne Conference, Moscone Center, San Francisco, California USA. Downloaded September 21st, 2012.

<<http://download.hyperic.com/pdf/WebFrameworkComparison.pdf>>

PHPFrameworks 2012. Popular PHP Frameworks. Downloaded April 30, 2012.

<<http://www.phpframeworks.com/index.php>>

Pope, Keith 2009. Zend Framework 1.8 Web Application Development. Design, develop, and deploy feature-rich PHP web applications with this MVC framework. Packt Publishing Ltd., Olton, Birmingham,, UK.

Sharp, Jason H. & Ryan, Sherry D. 2010. A Theoretical Framework of Component-Based Software Development Phases. ACM SIGMIS Database, Volume 41 Issue 1, ACM New York, NY, USA. Downloaded April 28, 2012.

<<http://dl.acm.org/citation.cfm?id=1719055&dl=ACM&coll=DL&CFID=102052182&CFTOKEN=52058990>>

Solomin, Joshua 2012. Email. Senior Product Marketing Manager, Zend Technologies.

Re: Marketing Contact Form Submission. Email address joshua.s@zend.com 2.6.2012.

SP Silicon Press 2012. Web Caching. Information to understand Technology. Technology Brief. Silicon Press. Downloaded November 5th, 2012.

<<http://www.silicon-press.com/briefs/brief.webcaching/brief.pdf>>

Statistics Canada 2012. Employment by Industry. Statistics Canada, Ottawa, Canada. Downloaded September 29, 2012.

<<http://www.statcan.gc.ca/tables-tableaux/sum-som/l01/cst01/econ40-eng.htm>>

The Open Group 2009. Technical Standard - SOA Governance Framework. Downloaded July 19th, 2012. Requires signing in.

<<https://www2.opengroup.org/ogsys/publications/viewDocument.html?publicationid=12205&documentid=10455>>

The PHP Group 2012. PHP Downloads. Downloaded 2012.

<<http://php.net/downloads.php>>

Trent, Scott & Tatsubori, Michiaki & Suzumura, Toyotaro & Tozawa, Akihiko & Onodera, Tamiya 2008. Performance Comparison of PHP and JSP as Server-Side Scripting Languages, IBM Tokyo Research Laboratory, Shimotsuruma Yamato-shi, Japan. Downloaded April 30, 2012.

<http://delivery.acm.org.ez.token.fi/10.1145/1500000/1496961/p164-trent.pdf?ip=195.148.253.185&acc=ACTIVE%20SERVICE&CFID=79874044&CFTOKEN=50966476&acm_=1335833496_6047439164de8c53378386fe4c83d3fe>

Van der Aalst, Wil M. P & Beisiegel, Michael & Van Hee, Kees M. & König, Dieter & Stahl, Christian 2012. A SOA-Based Architecture Framework. Eindhoven University of Technology, Department of Mathematics and Computer Science, Eindhoven, The Netherlands. Downloaded December 5th, 2012.

<http://www.win.tue.nl/~cstahl/Papers/AalstBHKS2007_csrep.pdf>

Williams, Mark 2012. Business Register and Employment Survey, 2011, Statistical Bulletin. Crown, Office for National Statistics, London, United Kingdom. Downloaded September 29, 2012.

<http://www.ons.gov.uk/ons/dcp171778_280655.pdf>

Zend Corporation 2010. Zend and PHP: An Overview. Downloaded April 30, 2012.

<<http://www.zend.com/en/company/static.zend.com/topics/Solution-Brief-05-10-SA02.pdf>>

Zend Corporation 2012. About Zend Framework. Downloaded June 15th 2012.

<<http://framework.zend.com/about>>

SAMPLE OF THE QUESTIONNAIRE PROVIDED FOR THE PURPOSE OF THE
SURVEY AMONG WEB DEVELOPERS

Are you a Web Developer? *

- Yes
- No

Which of the following ranges includes your age? *

- 12-17
- 18-24
- 25-30
- 31-35
- 36+

Please indicate your gender. *

- Male
- Female

What is/are your current position(s)? *

- Employed
- Un-employed
- Student
- Freelancer
- Volunteer
- Self-employed
- Other:

Please check the boxes for the corresponding items in the following list if you know what it represents and what it is used for. *

- CakePHP
- CodeIgniter
- DIY
- Yii
- the Zend Framework
- Symfony
- FuseBox
- ez Components
- I don't have knowledge about any of the above

Have you ever used any of the products above? *

- Yes
- No
- Not sure

If yes, please specify which ones

How experienced are you in web development? *

- Not experienced
- Less than 1 year
- 1 to 2 years
- 2 to 4 years
- 4 to 10 years
- More than 10 years

When the concept of framework occurs to you, what main idea comes to your mind, when it is referred to web development? *

Have you ever heard about a company named Zend Technologies? *

- Yes
- No
- Not sure

Have you ever used their product called the Zend Framework? *

- Yes
- No

If yes, how did you use it?

- By myself
- In a team
- One time, for a project
- I am a frequent user of the Zend Framework
- Not the case
- Other:

How would you rate your experience with the Zend Framework?

1 2 3 4 5

Bad Excellent

Would you recommend it to other web developers? *

- Yes
- No
- Not sure

Are you interested in participating in trainings about the Zend Framework?*

- Yes
- No
- Not sure
- Other:

If you have any comments, please leave them here!

APPENDIX 2

Method	Description
setBackend(Zend_Cache_Backend)	Sets the Zend_Cache_Backend object for use; determines how to save the cached data.
getBackend()	Returns a Zend_Cache_Backend object.
setOption(String, Mixed)	Sets the front-end options. The first parameter is one of the option shown in Table 9-5; the second parameter is the value for the option.
getOption(String)	Returns the stored value for the specified option.
setLifetime(int)	Sets the lifetime for the front end in seconds.
load(String)	Returns the cached data with the ID specified by the string parameter.
test(String)	Determines whether the cache ID specified is available for use. If available, returns true; otherwise, returns false.
save(String, String, Array, int, int)	Saves data into the cache. The first parameter is the data to cache. The second parameter is the unique ID for the cached content. The third parameter is an array containing a list of strings for tagging the data. The fourth parameter is the time to keep the content in the cache. The fifth parameter contains the priority of the data, 0–10 (0 = very low; 10 = very high).
remove(String)	Removes a specific cached item containing the string ID. Returns true if successfully removed.
clean(String)	Removes all cached items specified by the clean mode string. Clean modes are shown in Table 9-8.
getIdsMatchingTags(Array)	Returns an array containing all the cache IDs that contain the matching tags.
getIdsNotMatchingTags(Array)	Returns an array containing all the cache IDs that do not contain the matching tags.
getIds()	Returns an array containing all the IDs cached.
getTags()	Returns an array containing all the tags cached.
getFillingPercentage()	Returns an int value between 0 and 100 that represents the percent of space taken up by the cache.
touch(String, int)	Adds extra life to the cached content containing the specified ID. The first parameter is an ID; the second parameter is the extra time to allocate to the cached content.

Table 1. *Zend_Cache_Core* Methods (Padilla 2009, 360)

Method	Parameter	Description
<code>__construct()</code>	<code>Zend_Search_Lucene</code> → (String directory, Boolean)	Creates/opens the index located at the directory supplied in the first parameter. If second parameter is false, opens index for updating. If second parameter is true, creates or overwrites index.
<code>create()</code>	<code>create(String directory)</code>	Creates a new index at the specified directory.
<code>open()</code>	<code>open(String directory)</code>	Opens the index at the directory specified for reading or updating.
<code>getDirectory()</code>	<code>getDirectory()</code>	Returns the directory path as a <code>Zend_Search_Lucene_Storage_Directory</code> object.
<code>count()</code>	<code>count()</code>	Returns the total number of documents within the index, including deleted documents.
<code>maxDoc()</code>	<code>maxDoc()</code>	Returns the total number of documents.
<code>numDocs()</code>	<code>numDocs()</code>	Returns the total number of non-deleted documents.
<code>isDeleted()</code>	<code>isDeleted(int document_id)</code>	Returns a Boolean value. If the document is deleted, it returns true; if not deleted, it returns false.
<code>hasDeletions()</code>	<code>hasDeletions()</code>	Returns a Boolean value. If the index has had documents deleted, it returns true, false otherwise.

Table 2. *Zend_Search_Lucene* Methods (Padilla 2009, 321)

(Continued)

setDefaultSearchField()	setDefaultSearchField(String fieldname)	Sets the field name that will be searched by default. An empty string marks a search to be done on all fields by default.
getDefaultSearchField()	getDefaultSearchField()	Returns a string representing the default search field.
setResultSetLimit()	setResultSetLimit(int)	Sets the total number of results to fetch when searching. Default is 0, which returns all.
getMaxBufferedDocs()	getMaxBufferedDocs()	Returns the total number of documents in memory that must be met to write the documents into new segment in the file system.
setMaxBufferedDocs()	setMaxBufferedDocs(int)	Sets the total number of documents in memory that must be met to write the documents into the file system. Default is 10.
find()	find(String query Zend_Search_Lucene_Search_Query)	Queries the search engine. Accepts either a String query or a Zend_Search_Lucene_Search_Query object.
getFieldNames()	getFieldNames(Boolean)	Returns an array of unique fields contained in the index. If true, it returns only indexed fields; if false, it returns all field names.
getDocument()	getDocument(int)	Returns a Zend_Search_Lucene_Document object of the document ID specified.
hasTerm()	hasTerm(Zend_Search_Lucene_Index_Term)	Returns a Boolean value. If the index contains the term specified, it returns true; otherwise false.
terms()	terms()	Returns an array containing all the terms in the index.
optimize()	optimize()	Merges all the segments into one segment to increase index quality.
commit()	commit()	Commits any changes made when deleting documents.
addDocument()	addDocument(Zend_Search_Lucene_Document)	Adds a document to the index.
delete()	delete(int)	Removes a document from the index.
docFreq()	docFreq(Zend_Search_Lucene_Index_Term)	Returns the total number of documents that contain the term.

Table 3. *Zend_Search_Lucene* Methods (Padilla 2009, 322)