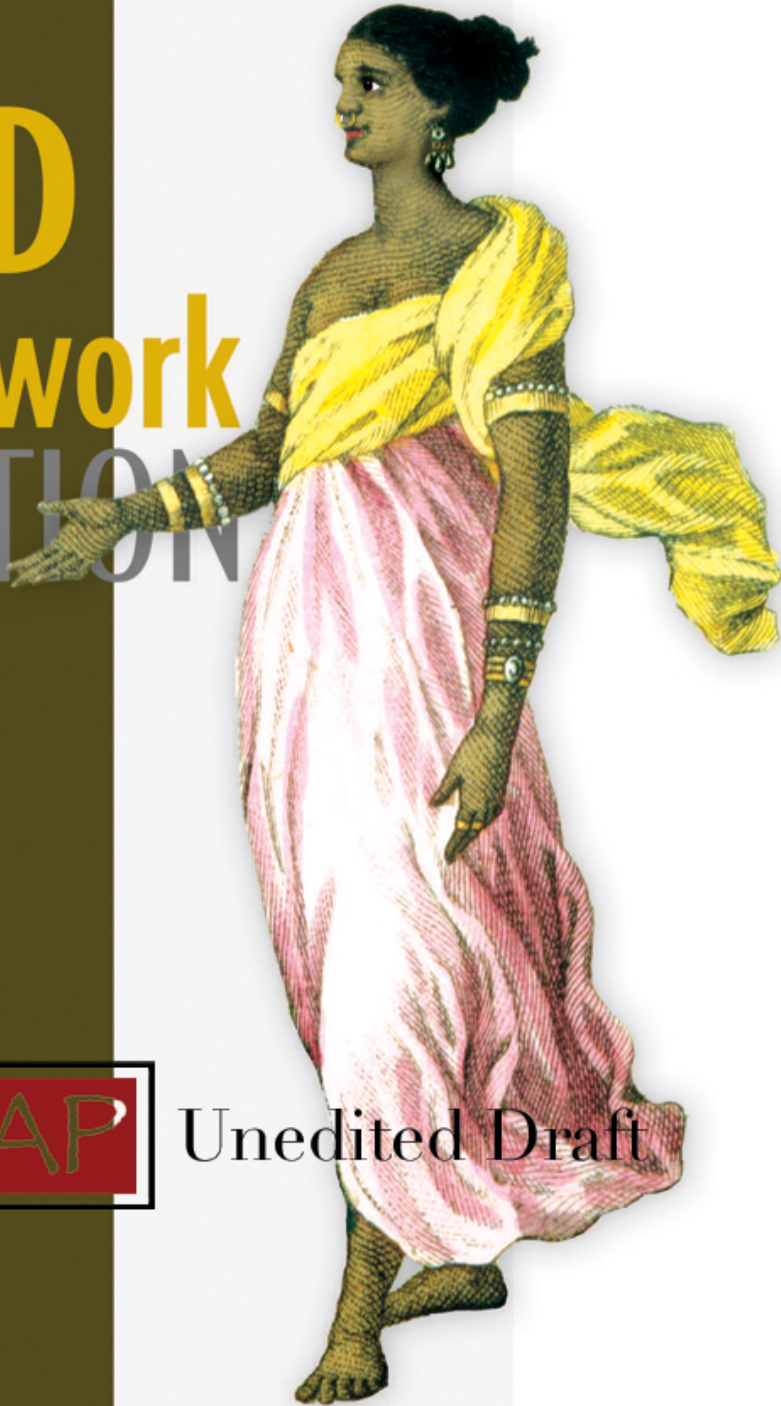


ZEND Framework IN ACTION

Rob Allen
Nick Lo



Unedited Draft





**MEAP Edition
Manning Early Access Program**

Copyright 2007 Manning Publications

For more information on this and other Manning titles go to
www.manning.com

Please post comments or corrections to the Author Online forum at
<http://www.manning-sandbox.com/forum.jspa?forumID=329>

Contents

Part 1: The Essentials

Chapter 1 - Introducing the Zend Framework

Chapter 2 - Hello Zend Framework!

Part 2: A Core Application

Chapter 3 - Building a web site with the Zend Framework

Chapter 4 - Ajax

Chapter 5 - Managing the Database

Chapter 6 - User Authentication and Authorisation

Chapter 7 - Forms

Chapter 8 - Searching

Chapter 8 - Email

Chapter 10 - Deployment

Part 3: More Power to Your Application

Chapter 11 - Talking with Other Applications

Chapter 12 - Mash ups with Public Web Services

Chapter 13 - Caching: Making it faster

Chapter 14 - Internationalization and Localization

Chapter 15 - Creating PDFs

Chapter 16 - Integrating with other PHP libraries

Appendices

Appendix A - Stuff you (should) already know

Appendix B - System Specific Gotchas

Appendix C - Zend Framework Core Components Reference

1 Introducing the Zend Framework

PHP has been used to develop dynamic websites for over 10 years. Initially all PHP websites were written as PHP code interspersed within HTML on the same page. This works very well initially as there is immediate feedback and for simple scripts this appears to be all that is needed. PHP grew in popularity through versions 3 and 4, and so it was inevitable that larger and larger applications would be written in PHP. It became obvious very quickly that intermixing PHP code and HTML was not a long term solution for large websites.

The problems are obvious in hindsight: maintainability and extensibility. Whilst PHP intermixed with HTML allows for extremely rapid results, in the longer term it is hard to continue to update the website. One of the really cool features of publishing on the web is that it is dynamic with content and site layouts changing. Large websites change all the time. The look and feel of the site is updated regularly. New content is added and content is regularly re-categorized as the needs of the users (and advertisers!) change. Something had to be done!

The Zend Framework was created to help ensure that the production of PHP based websites is easier and maintainable in the long term. It contains a rich set of reusable components containing everything from a set of Model-View-Controller application components to PDF generation. Over the course of this book, we will look at how to use all the components within the context of a real website.

1.1 Introducing structure to PHP websites

The solution to this tangled mess of PHP code and HTML on a website is structure. The most obvious introduction to structured applications within PHP sites is applying the concept of “separation of concerns”. This means that the code that does the display should not be in the same file as the code that connects to the database and collects the data as shown in Figure 1.1.

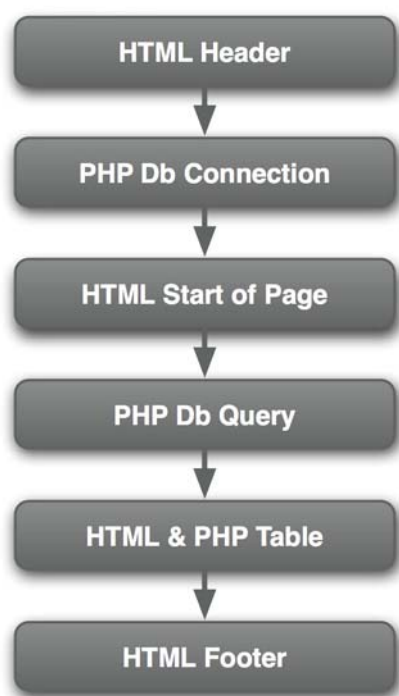


Figure 1.1: The organization of a typical PHP file created by a novice interleaves HTML and PHP code in a linear fashion as the file is created.

The first stages of introducing structure to a website's code happen by default for most developers; the concept of reusability dawns. Generally, this means that the code that connects to the database is separated into a file called something like `db.inc.php`. Then it seems logical to separate out the code that displays the common header and footer elements on every pages. Functions are introduced to help solve the problem of global variables affecting one another

As the website grows, common functionality is grouped together into libraries. Before you know it, the application is much easier to maintain and adding new features becomes possible again. This stage lasts for a little while and the website continues to expand until it gets to the point where the supporting code is so large that you can't hold a picture of how it all works in your head.

PHP coders are used to standing on the shoulders of giants as our language provides easy access to libraries such as the GD image library, the many database client access libraries and even system specific libraries such as COM on Windows. It was inevitable that Object-Oriented Programming would enter the PHP landscape. Whilst classes in PHP4 provided little more than glorified arrays, PHP5 provides excellent support for all the things you'd expect in an object oriented language. Hence there are visibility specifiers for class members (public, private and protected) along with interfaces, abstract classes and support for exceptions.

The improved object-oriented support allows for more complicated libraries (known as frameworks) to evolve, such as the Zend Framework which supports a way of organizing web application files know as the MVC design pattern. This is shown in Figure 1.2.

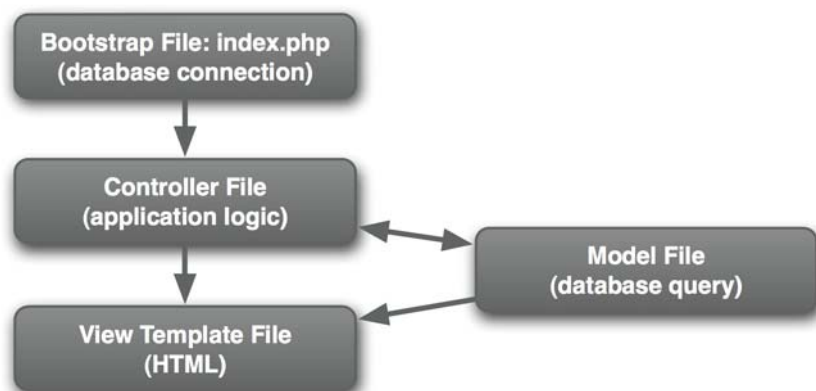


Figure 1.2: The organization of a typical MVC application

An application designed using MVC principles results in more files. Each file is specialized in what it does which makes maintenance much easier. For example, all the code that makes database queries is stored in classes known as Models. The actual HTML code is known as the View (which may also contain simple PHP logic) and the Controller files handle the connection of the correct models to the correct views to display the desired page.

The Zend Framework isn't the only option for organizing a website based on MVC principles; there are many others in the PHP world. Let's look at what the Zend Framework contains and why it should be considered.

1.2 Why use the Zend Framework?

As you have this book in your hands, you probably want to know why you'd be interested in the Zend Framework over all the other PHP frameworks out there. In a nutshell, the Zend Framework introduces a standardized set of components that allow for easy development of web applications. These applications can be easily developed, maintained and enhanced.

The key features of the Zend Framework are:

- Everything in the box
- Modern design
- Easy to learn
- Full documentation
- Simpler Development
- Rapid development

1.2.1 Everything in the box

The Zend Framework is a comprehensive full stack framework that contains everything you need to develop your application. This includes a robust MVC component to ensure that your website is structured according to best practices. Accompanying the MVC component, there are components for authentication, searching, localization, PDF creation, email and connecting to web services, along with a few other more esoteric items as shown in Figure 1.2.



Figure 1.3: The Zend Framework can be divided into ten main modules

That's not to say that the Zend Framework doesn't "play nice" with other libraries; it does that too. A core feature of the design of the framework is that it is easy to use just those bits you want to use with the rest of your application or with other libraries such as PEAR, the Doctrine ORM or the Smarty template library.

1.2.2 Modern design

The Zend Framework is written in object-oriented PHP5 using the modern design techniques, known as design patterns. Software design patterns are recognized high level solutions to design problems and, as such, are not a specific implementation of the solution. The actual implementation depends on the nature of the rest of the design. The Zend Framework makes use of many design patterns and its implementation has been carefully designed to allow the maximum flexibility for application developers without making them do too much work!

The framework recognizes the PHP way and doesn't force you into using all the components, so you are free to pick and choose between them. This is especially important as it allows you to introduce specific components into an existing site. The key is that each component within the Framework has very few dependencies on other components.

1.2.3 Easy to learn

If you are anything like me, learning how a vast body of code works is hard! Fortunately the Zend Framework is modular and has a design goal of simplicity which makes it easy to learn, one step at a time. Each component doesn't depend on lots of other components and so is easy to study. The design of each component is such that you do not need to understand how it works in its entirety before you can use it and benefit from it. Once you have some experience of using the component, building up to use the more advanced features is straight-forward as it can be done in steps. This is key to reducing the barrier to entry for most users.

For example, the configuration component `Zend_Config` is used to provide an object oriented interface to a configuration file. It supports two advanced features: section overloading and nested keys, but neither of these features need to be understood in order to use the component. Once the user has a working implementation of `Zend_Config` in their code, confidence increases and so using the advanced features is a small step.

1.2.4 Full documentation

No matter how good the code is, lack of documentation can kill a project through lack of adoption. The Zend Framework is aimed at developers who do not want to have to dig through all the source code to get their job done and so the Zend Framework puts documentation on an equal footing with the code. This means that the core team will not allow new code into the framework unless it has accompanying documentation.

There are two types of documentation supplied with the framework: API and end-user. The API documentation is created using `PHPDocumenter` and is automatically generated using special “docblock” comments in the source code. These comments are typically found just above every class, function and member variable declaration. The key advantage of using docblocks is that IDEs such as `PHPIDE` in `Eclipse` or `Zend’s Studio` are able to supply auto-completion tool tips whilst coding and so improve developer productivity.

The Zend Framework also supplies a full manual as part of the download and also available online at <http://framework.zend.com/manual>. The manual provides details on all components of the framework and shows what functionality is available. Examples are provided to help you get started in using the component in an application. More importantly, in the case of the more complicated components (such as `Zend_Controller`), the theory of operation is also covered, so that you can understand why the component works the way it does.

The documentation provided with the framework does not explain how to fit all the components together to make a complete application. As a result, a number of tutorials have sprung up on the web by community members to help developers get started on the framework. These have been collated on a web page on the framework’s wiki at <http://framework.zend.com/wiki/x/q>. The tutorials, whilst a useful starting point, do not tend to go depth with each component or show how it works within a non-trivial application, which is why this book exists.

1.2.5 Simpler development

As we have noted, one of PHP’s strengths is that developing simple dynamic web pages is very easy. This has enabled millions of people to have fantastic websites who may not have had them otherwise. The ability of PHP programmers range from people who are beginners to programming through to enterprise developers needing to meet their deadlines. The Zend Framework is designed to make development simpler for every type of developer.

So how does it make development simpler? The key feature that the framework brings to the table is tested, reliable code that does the “grunt” work of an application. This means that the code you write is the code you need for your application. The code that does the “boring” bits is taken care of for you and is not cluttering up your code.

1.2.6 Rapid Development

The Zend Framework makes it easy to get going on your web application or add new functionality to a current website. As the framework provides many of the underlying components of an application, you are free to concentrate on the core parts of your application, rather than on the underlying foundation. Hence, it is easy to get started quickly on a given piece of functionality and immediately see the results.

Another way the framework speeds up development is that the default use of most components is the common case. In other words, you don't have to worry having to set lots of configuration settings for each component just so that you can get started using it. For example, the most simplistic use of the whole MVC is bootstrapped with just:

```
require_once('Zend/Loader.php');
Zend_Loader::registerAutoload();
Zend_Controller_Front::run('/path/to/controllers');
```

Once up and running, adding a new page to your application can be as easy as adding a new function to a class, along with a new template file in the correct directory. Similarly, `Zend_Session` provides a multitude of options that can be set so that you can manage your session exactly how you want to, however none need to be set in order to use the component for most use-cases.

1.2.7 Structured code is easy to maintain

As we have seen earlier, separating out different responsibilities makes for a structured application. It also means finding what you are looking for is easier whilst bug fixing. Similarly, when you need to add a new feature to the display code, the only files you need to look at are related to the display logic. This avoids bugs created inadvertently by breaking something else. The framework also encourages you to write object oriented code, which helps to ensure that maintenance of your application is simpler.

1.3 What is the Zend Framework?

The Zend Framework is a PHP library for building PHP web application. It provides a set of components to enable you to build PHP applications more easily which will be easier to maintain and extend over the lifetime of the application. That rather simple description doesn't tell the whole story though, so we will look at where this framework came from and then have a brief look at what it actually contains

1.3.1 Where did it come from?

Frameworks have been around for years. The very first web framework I used in a real project was Fusebox which was originally written for ColdFusion. Many other frameworks have come along since then, with the next major highlight being Struts, written in Java. A number of PHP clones of Struts were written, but didn't translate well to PHP. The biggest difference being that Java web applications run in a virtual machine that runs continuously, so the startup time of the application is not a factor for every web request. PHP initializes each request from a clean slate and so the large initiation required for Struts clones made them relatively slow as a result. Recently, a new framework entered the world based on a relatively unknown language called Ruby. Rails (or Ruby on Rails as it is also known) promoted the concept of convention over configuration and has taken the web development world by storm. Shortly

after Rails came along, a number of direct PHP clones have appeared, along with a number of frameworks that are inspired by Rails, rather than direct copies.

In late 2005, Zend Technologies, a company that specializes in PHP, started their PHP Collaboration project to advance the use of PHP. There are three strands to the project: an eclipse IDE plugin called PDT, the Zend Framework and the Zend Developer Zone website. The Zend Framework is an open source project that provides a web framework for PHP and is intended to become one of the standard frameworks that PHP applications of the future will be based on.

1.3.2 What's in it?

The Zend Framework is composed of many distinct components grouped into a set of top level modules. As a complete framework, you have everything you need to build enterprise ready web applications. However, the system is very flexible and has been designed so that you can pick and choose to use those bits of the framework that are applicable to your situation. Following on from the high level overview in Figure 1.3 shown earlier, Figure 1.4 gives a good overview of all the components within the framework.

<p>Core:</p> <ul style="list-style-type: none"> Zend_Controller Zend_View Zend_Db Zend_Config Zend_Filter & Zend_Valdiate Zend_Registry <p>Authentication and Access:</p> <ul style="list-style-type: none"> Zend_Acl Zend_Auth Zend_Session <p>Internationalization:</p> <ul style="list-style-type: none"> Zend_Date Zend_Locale Zend_Measure <p>Http:</p> <ul style="list-style-type: none"> Zend_Http_Client Zend_Http_Server Zend Uri 	<p>Inter-application communication:</p> <ul style="list-style-type: none"> Zend_Json Zend_XmlRpc Zend_Soap Zend_Rest <p>Web Services:</p> <ul style="list-style-type: none"> Zend_Feed Zend_Gdata Zend_Service_Amazon Zend_Service_Flickr Zend_Service_Yahoo <p>Advanced:</p> <ul style="list-style-type: none"> Zend_Cache Zend_Search Zend_Pdf Zend_Mail/Zend_Mime <p>Misc!</p> <ul style="list-style-type: none"> Zend_Measure
---	---

Figure 1.4: The Zend Framework contains lots of components that cover everything required to build an enterprise application.

Each section of the framework consists of a number of components, which is usually the name of the main class too. For example, Zend_View is the concrete view class used by applications. Each component also contains a number of other classes too that are not listed in Figure 1.4. The classes that are actually used within your application are discussed as we go through the book and learn about each component.

The Core Components

The core components provide a full-features Model-View-Controller (MVC) system for building applications that separate out the view templates from the business logic and controller files. There are three families of classes that make up the MVC system: `Zend_Controller` (Controller), `Zend_View` (View) and `Zend_Db` (Model). Figure 1.5 shows the basics of the Zend Framework's MVC system.

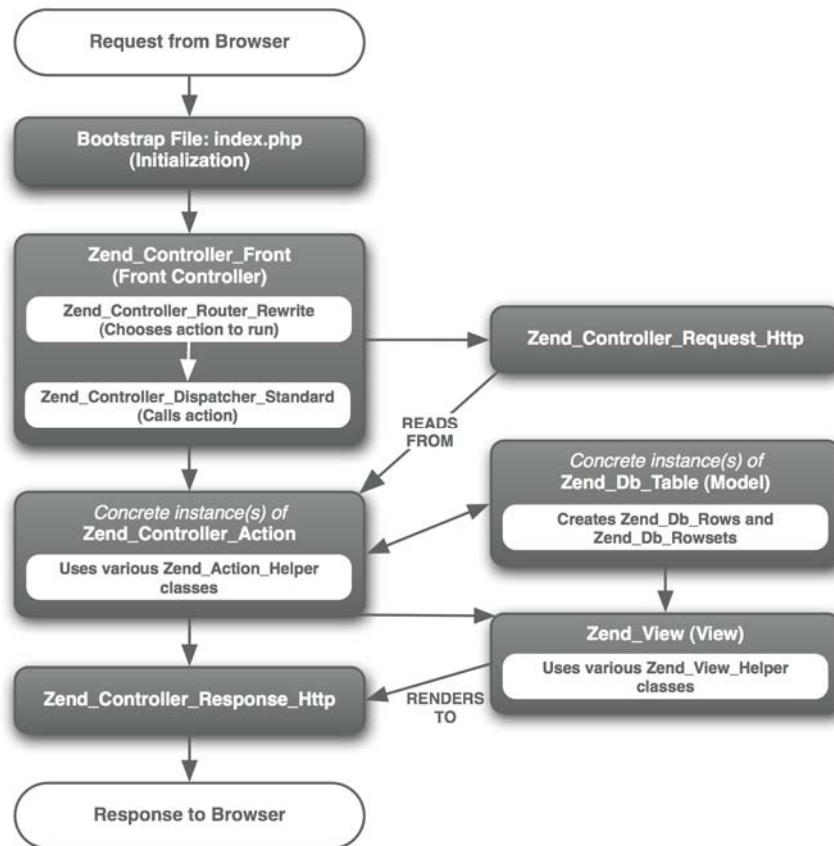


Figure 1.5: MVC: the Zend Framework way

The `Zend_Controller` family of classes provides a front controller design which dispatches requests to controller actions (also known as commands) so that all processing is centralized. As you'd expect from a fully featured system, the controller supports plug-ins at all levels of the process and has built in flex-points to enable you to change specific parts of the behavior without having to do too much work.

The view template system is called `Zend_View` which provides a PHP based template system. This means that, unlike Smarty or PHPTAL, all the view templates are written in PHP. `Zend_View` provides a helper plug-in system to allow for creation of reusable display code. It is designed to allow for overriding for specific requirements, or even replacing entirely with another template system such as Smarty.

`Zend_Db_Table` implements a table row gateway pattern to form the basis of the model within the MVC system. The model provides the business logic for the application which is usually database-based in a web application. Supporting `Zend_Db_Table` is `Zend_Db` which provides object oriented, database

independent access to a variety of different databases, such as MySQL, Postgres, SQL Server, Oracle and SQLite.

The most simplistic setup of the MVC components can be done with the very simple code:

```
require_once 'Zend/Controller/Front.php';  
Zend_Controller_Front::run('/path/to/your/controllers');
```

It is more likely, however, that a more complicated bootstrap file will be required for a non-trivial application as we will explore in chapter 2 when we build a complete Hello World application in Zend Framework.

Working with the MVC classes are a couple of separate classes that are used to create the core of a complete application. The framework encourages convention over configuration, however some configuration is invariably required (such as database login details). `Zend_Config` allows for reading configuration data in either INI or XML formats and includes a useful inheritance system for supporting different configuration settings on different servers, such as production, staging and test.

Security is very much on the minds of every PHP developer worth his salt. Input data validation and filtering is the key to a secure application. `Zend_Filter` and `Zend_Validate` are provided to help the developer ensure that input data is safe for use in the application.

The `Zend_Filter` class provides a set of filters that typically remove or transform unwanted data from the input as it passes through the filter. For example, an numeric filter would remove any characters that were not numbers from the input and an HTML entities filter would convert the “<” character to the sequence “<”. Appropriate filters can then be set up to ensure that the data is valid for the context it will be used in.

`Zend_Validate` provides a very similar function to `Zend_Filter`, except that it provides a yes/no answer to the question “is this data what I expect?”. Validation is generally used to ensure that the data is correctly formed, such as the data provided as an email address is actually an email address. In the case of failure, `Zend_Validate` also provides a message indicating why the input failed validation so that appropriate error messages can be provided back the end user.

Authentication and Access Components

Not every application needs to identify their users, but it is a surprisingly common requirement. Authorization is the process of providing access to a given resource, such as a web page, to an authenticated user. That is, authentication is the process of identifying and entity, usually via a token such as a username/password pair, but could equally be via a fingerprint. Authorization is the process of deciding if the authenticated entity is allowed to have access to, or perform operations on, a given resource, such as a record from a database.

As there are two separate processes required, the Zend Framework provides two separate components: `Zend_Acl` and `Zend_Auth`. `Zend_Auth` is used to identify the user and is typically used in conjunction with `Zend_Session` to hold that information across multiple page requests (known as token persistence). `Zend_Acl` is then uses the authentication token to provide access to private information using the Role Based Access Control List system.

As is becoming a watchword around here, flexibility is a key design decision within the `Zend_Auth` component. There are so many ways to authenticate a user, that the `Zend_Auth` system is built with the intention that the user will provide their own. The most common scenario of HTTP digest authentication

is provided out of the box, but for any other method, you must create a class that extends `Zend_Auth_Adapter`. Fortunately, this is not difficult as we will see in chapter 6.

As `Zend_Acl` is an implementation of the Role Based Access Control List system, the manual talks in lots of abstract terms. This is because RBACL is a generic system that can provide access to anything by anyone and so specific terms are discouraged. Hence we talk about Roles requesting access to Resources. A Role is anything that may want to access something that is under the protection of the `Zend_Acl` system. Generally, for a web application, this means that a Role is a user that has been identified using `Zend_Auth`. A Resource is anything that is to be protected. This is generally a record in a database, but could equally be an image file stored on disk. As there is such a wide type of resources, the `Zend_Acl` system provides for us to create our own very simply by implementing `Zend_Acl_Role_Interface` within our class.

Internationalization Components

As we live in a multi-cultural world with multiple languages, the framework provides a rich set of functionality to allow for localizing your application to match your target users. This covers minor issues like ensuring that the correct currency symbol is used through to full support for changing all the text on the page to the correct language. Date and time routines are also provided with a simple object oriented interface to the multitude of ways that we humans display the calendar.

Http Components

The Zend Framework provides a component to read data from other websites. `Zend_Http_Client` makes it easy to collect data from other web sites and services and then present it on your site. A server component is also provided to allow for PHP based serving for web pages. Obviously this component is intended for development and other specialized requirements rather than general web page serving, as proper web-servers like Apache are orders of magnitude faster!

Inter-application Communication Components

When you need to communicate with another application over HTTP, the most common transfer format is one of two flavors of XML: XML-RPC and SOAP. As you would expect from an enterprise-class framework, the Zend Framework provides components to allow for easy processing of both XML-RPC and SOAP protocols. More recently, the lightweight JSON protocol is gaining favor, mainly due to how easy it is to process within the JavaScript of an Ajax application. `Zend_Json` provides a nice solution to both creating and reading JSON data.

Web Services Components

The Zend Framework provides a rich set of functionality to allow access to services provided by other suppliers. These components cover generic RSS feeds along with specific components for working with the public APIs from Google, Yahoo! and Amazon. RSS has come a long way from its niche amongst the more technologically minded bloggers and is now used by the majority of news sites. `Zend_Feed` provides a consistent interface to reading feeds in the various RSS and atom versions that are available without having to worry about the details.

Google, Yahoo! and Amazon have provided public APIs to their online services in order to encourage developers to create extended applications around the core service. For Amazon, the API revolves around providing access to the data on `amazon.com` in the hope that the new application will encourage more sales! Similarly, Yahoo! provides API access to their Flickr photo data in order to allow additional

services for Flickr users, such as the print service provided by moo.com. The traditional Yahoo! properties such as search, news and images are also available via Zend_Service_Yahoo.

Google has a number of online applications that allow for API access which are supported by the Zend_Gdata component. The Zend_Gdata component provides access to Google's Blogger, Calendar, Base and CodeSearch applications. In order to provide consistency, the Zend_Gdata component provides the data using Zend_Feed, so if you can process an RSS feed, then you can process Google Calendar data too.

Advanced Components

There are a set of other components provided with the Zend Framework that do not fit easily into any category, so I have rather lazily grouped them together into the advanced category. This potpourri of components includes caching, searching, pdf creation, email and the rather esoteric measurement class.

Everyone wants a faster website and caching is one tool that can be used to help speed up your website. Whilst not a sexy component, the Zend_Cache component provides a generic and consistent interface to cache any data in a variety of back end systems such as disk, database or even with APC's shared memory. This flexibility ensures that you can start small with Zend_Cache and as the load on your site increases, the caching solution can grow up with you to help ensure you get the most out of the server hardware.

Every modern website provides a search facility. Most provide a terrible search facility; so terrible that the site's users would rather search Google than use the site's own system. Zend_Search is based on the Apache Lucene search engine for Java and provides an industrial strength text search system that will allow your users to find what they are looking for. As required by a good search system, it supports ranked searching so that the best results are at the top, along with a powerful query system.

Another component that I've lumped into the advanced category is Zend_Pdf which covers the creation of PDF files programmatically. PDF is a very portable format for creating documents intended for printing. This is because you can control the position of everything on the page with pixel-perfect precision without having to worry about differences in the way web browsers render the page. Zend_Pdf is written entirely in PHP and can create new PDF documents or load existing one for editing.

Email Components

The Zend Framework provides a strong email component to allow for sending emails in plain text or HTML. As with all Zend Framework components, emphasis has been placed on flexibility combined with sensible defaults. Within the world of email, this means that the component allows for sending email using SMTP or via the standard PHP mail() command. Additional transports can be easily slotted into the system by writing a new class that implements Zend_Email_Transport_Interface. When sending email, a simple object-oriented interface is used:

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

This snippet also shows a fluent interface where each member function returns an object to itself so that they can be chained together to make the code more readable.

1.4 Zend Framework design philosophy

The Zend Framework has a number of published goals that make up the design philosophy of the framework. If these goals do not mesh with what your view on developing PHP applications then the Zend Framework is unlikely to be a good fit for your way of doing things. Let's look at what makes the Zend Framework unique.

1.4.1 Provide high quality components

All code within the Zend Framework library will be of high quality. This means that it will be written using PHP5's features and will not generate any messages from the PHP parser (i.e. it is E_STRICT compliant). This is good as it means that any PHP parser messages in your logs come from your code, not the framework; this will help debugging considerably! Zend also define high quality to include documentation, so the manual for a given component is as important as the code.

It is intended that it will be possible to develop entire applications with no external library dependencies (unless you want them). This means that the Zend Framework is intended to be a "full stack" framework (like Ruby on Rails or the Django Python framework) rather than a set of components. This will ensure that there will be consistency in they way you use all the components: how they are named, how they work and how the files are laid out in sub directories. Having said that, it is important to Zend that the Zend Framework is modular with few dependencies between modules. This ensures that it plays well with other frameworks and libraries and you can therefore use as much or as little as you want. For example, if you just want PDF generation, you don't have to use the MVC system.

1.4.2 Simple as possible

Another design goal for the framework is the mantra "Don't change PHP". The PHP way is simple, pragmatic solutions and so the Zend Framework is intended to reflect that simplicity in order to provide a simple solution for mainstream developers. It is also powerful enough to allow for specialized usage via extension. The core developers have done a great job in covering the common scenarios and providing "flex-points" to allow for easy changing of the default behavior by those who want something more powerful or specialized.

1.4.3 Clean IP

All contributors to the Zend Framework have signed a Contributor License Agreement. This is an agreement with Zend which defines intellectual property status of the contribution. That is, the contributor warrants that (to the best of her knowledge), she is entitled to make the contribution and that no one else's intellectual property rights are being infringed. This is intended to help protect all users of the framework from potential legal issues related to IP and copyright. The risk is minimal, but with relatively recent actions by SCO against AutoZone shows that a litigator going after the user of the allegedly copyright infringing code is a possibility. As with everything, it is better to be prepared.

1.4.4 Supported by Zend Technologies

An obvious but important consideration is that the Zend Framework is supported by the company Zend Technologies. This means that the framework is unlikely to “die” due to inactivity of the core developers or by lack of updating to the latest and greatest version of PHP. Zend Technologies also have the resources to provide for the “boring” bits of the framework, such as documentation, which (as we all know) few developers really *like* doing!

1.5 Alternative PHP frameworks

As usage of PHP is so broad, no one framework is going to suit everyone. In the PHP world there are many other frameworks vying for your attention and all have their strengths and weaknesses. I have rather arbitrarily picked four other frameworks which all have some traction in the community but these are by no means the only choices. I have listed what I see as their strengths and weaknesses in Table 1.1.

	Zend Framework	Cake	Code Igniter	Solar	Symfony
Uses PHP5 to full advantage	Yes	No	No	Yes	Yes
Prescribed directory structure	No (Optional recommended structure)	Yes	Yes	No	Yes
Official internationalization support	Yes	In progress for version 1.2	No	Yes	Yes
Command line scripts required to setup framework	No	No	No	No	Yes
Requires configuration	Yes (minor amount)	No	No	Yes	Yes
Comprehensive ORM provided	No	Yes	No	No	Yes (Propel)
Good documentation and tutorials	Yes	Yes	Yes	Yes	Yes
Unit tests for source available	Yes	No	No	Yes	Yes
Community support	Yes	Yes	Yes	Yes (some)	Yes
License	New BSD	MIT	BSD-style	New BSD	MIT

1.1 Key features matrix: Zend Framework, Cake, Code Igniter, Solar and Symfony

Whilst this book is all about the Zend Framework, the other frameworks are worth investigating to see if they match better with your requirements. If you still need to support PHP 4, then you will need to use either Cake or Code Igniter as the rest do not support PHP 4. In this day and age, however, it’s time to leave PHP 4 behind us!

1.6 Summary

In this chapter we have looked at what the Zend Framework is and why it is useful for writing web applications. It enables rapid development of enterprise applications by providing a full stack framework using best practices in object oriented design. The framework contains many components from an MVC controller through PDF generation to providing a powerful search tool.

This book is about providing real world examples and so will have Ajax technology built in wherever it is appropriate. Now, let's move onto the core topic of this book and look at a how to build a simple, but complete Zend Framework application in Chapter 2.