Community Experience Distilled

Zend Framework 2 Application Development

Explore the Zend Framework 2 and create your own superb social network

Christopher Valles

[PACKT] PUBLISHING open source* community experience distilled

Zend Framework 2 Application Development

Explore the Zend Framework 2 and create your own superb social network

Christopher Valles

[PACKT] PUBLISHING
open source
community experience distilled

BIRMINGHAM - MUMBAI

Zend Framework 2 Application Development

First published: October 2013

Production Reference: 1211013

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78216-210-0

www.packtpub.com

Cover Image by Tim Green (thisistimgreen@gmail.com) via Flickr InMail

Credits

Author
Christopher Valles

Reviewers
Richard Ainger
Gabriel García Fernández
Doug Johnson
Brent Shaffer

Acquisition Editor
James Jones

Lead Technical Editor
Madhuja Chaudhari

Technical Editors
Shashank Desai
Krishnaveni Haridas
Ankita Thakur

Project Coordinator
Amey Sawant

Copy Editors
Mradula Hegde
Sayanee Mukherjee
Kirti Pai
Adithi Shetty
Laxmi Subramaniam

Proofreaders
Chrystal Ding
Clyde Jenkins

Indexer
Monica Ajmera Mehta

Graphics
Sheetal Aute

Production Coordinator
Manu Joseph

Cover Work
Manu Joseph

About the Author

Christopher Valles is a Software Engineer from Barcelona, Spain, currently based in London, UK. He started developing when he was seven using a Vtech kid laptop that was strangely shipped with a simple version of the BASIC programming language. Since then, he has explored more than 16 different programming languages ranging from Assembler to PHP, Python, and GO.

Chris also stepped into the sysadmin role and has been managing systems since he started working in this industry. He has taken care of servers right from simple webservers to infrastructures on the Cloud and internal Mac infrastructures. He is an Apple Certified Support Professional and Apple Certified Technical Coordinator.

His desire to learn and experiment has driven him to explore other fields, such as machine learning and robotics. He currently owns close to five robots and has built more than 20 over the past years. If you don't find him on the computer, he is probably spending time in the kitchen cooking delicious recipes.

The sectors where Chris has worked ranges from adult content websites and payment processors to social networks and the gaming industry. Presently, he's working as a Software Engineer at Hailo Networks, Ltd.

Acknowledgments

First of all, I want to thank my lovely girlfriend for supporting me along the way of this journey. I know for sure that this would have not been possible without her, and I will be always grateful to her.

I also thank my friends, Gabriel Garcia and Tasos Bitsios, who where there answering my questions and sharing with me their thoughts about the book, chapters, and the code. Last but not least, I want to thank all the reviewers who helped me with their comments, specially Brent Shaffer for his patience and help on the implementation of OAuth 2.0 and for letting me use his library in this book.

I want to mention some people who, over the years, have contributed somehow to the person I am today and have encouraged me to follow my dreams, keep exploring new stuff, pursue new challenges, and supported me with all my crazy ideas. My parents Vicente Vallés Serrat and Manuela Ramos González, my brother Bryan Vallés Ramos, my first boss Òscar Martínez Ciuró, and Stuart Helen Overton-Smith for getting on the plane with me, teaching me how to fly, and helping me accomplish my dream of being a pilot. A special mention goes for my cats, Schrödinger and Bones, for giving me company and staying with me at 4 a.m. while I was writing.

Finally, I also want to thank all the people who made efforts and invested time researching the polyphasic sleep cycles and shared their knowledge for free on the Internet. To write this book, I switched to everyman sleep cycle that allowed me to stay awake 19 hours per day for the last six months.

About the Reviewers

Richard Ainger is a Software Engineer and all-round computer enthusiast residing in Australia.

Since becoming passionate about computers at an early age, Richard has explored a variety of programming languages including PHP, Java, C, and MIPS ASM. Richard has also branched outside of the more traditional desktop development and smartphone environments and has been an active developer of homebrew software for the PSP gaming system.

More recently, Richard has been enjoying developing software solutions for **The Cyber Institute** (**TCI**), a leader in the e-learning space within Australia. TCI is responsible for creating learning systems and content for some of the biggest names in Australia.

Every spare moment, Richard devotes extending his knowledge into new areas of software development trying to keep up with the constantly evolving field of software and web development.

I would like to thank the talented people I work with daily at TCI from whom I have learned a lot. I would also like to thank Christopher Valles and the people at Packt Publishing for giving me the opportunity to review and provide feedback during the production of this book.

Gabriel García Fernández is a Computer Engineer based in Barcelona working as Project Manager and CTO at Urban Ventures S.L. He has been working as a Web Developer for many years before becoming a tech manager.

He has an entrepreneurial mind, proactive, with a strong focus on team management and employees' motivation. He is very adaptable, self-motivated, and easily develops empathy with people and is always ready to learn. He is passionate about Agile methodologies and web development.

He spends his free time with his girlfriend, his family, going out with friends, or reading (literature is one of his passions).

Doug Johnson first discovered a knack for programming at Dartmouth College while tracking down a programming error in an orbital simulation that led to a citation in Physics. He has programmed applications in many languages, including C, C#, Delphi, PHP, and JavaScript and taught programming and scripting at the college level.

As the founder and owner of an accounting software consultancy, he developed customized solutions for business problems in the SMB market, primarily in medicine and manufacturing. Working with several companies, he has written health-care- and electronic-health-record software and served for several years on the IHE standards committee for eye care, co-writing one of the profiles.

He is currently developing tax software for Intuit, and he also develops websites and web applications using Drupal, Zend Framework, and AngularJS, primarily in the not-for-profit space.

He likes to cook and is developing a website called CookLoose that aims to teach how to go from being a non-cook to a competent cook.

I'd like to thank my wife who builds systems into our household that allows me to live comfortably while not paying any real attention to how she does it. She has designed a kitchen that anyone can work in. She doesn't understand a thing about what I do for a living but is still willing to listen to me going on and on about the latest, coolest thing I've discovered as long as I make her coffee in the morning.

Brent Shaffer is a Computer Scientist and musician living in Salt Lake City, Utah. He works for Adobe Systems, Inc. in the Web Services team for the Adobe Marketing Cloud platform. The Web Services team oversees all API Framework responsibilities for the Adobe Marketing Cloud, such as third-party integrations, authentication with OAuth 2.0, rate limiting, throttling, and traffic monitoring.

Brent also works remotely, performing with his band More Hazards More Heroes, a folk duo from Nashville, TN. His band has performed across the country their music has been featured in GoPro commercials and Hollywood films.

Brent is a longtime PHP and Symfony Framework advocate, and has worked closely with the community, writing plugins and official documentation for both the Symfony 1.x and 2.x projects.

www.PacktPub.com

Support files, eBooks, discount offers and more

You might want to visit `www.PacktPub.com` for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

`http://PacktLib.PacktPub.com`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why Subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print and bookmark content
- On demand and accessible via web browser

Free Access for Packt account holders

If you have an account with Packt at `www.PacktPub.com`, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface

Nowadays, the programming scene is full of frameworks, and they move really fast, providing more and more functionalities. When we have to choose a framework, every developer has different requirements. But usually in the business environment we want to choose something that assures somehow a continuity on the framework, stability, and a good community behind.

Zend Framework is well known for being a really stable framework chosen by a lot of companies from really big to small ones. They have a huge community behind and a lot of people contribute in a way or another.

The first version of ZF was released in 2007, and since then they continue updating the project. Right now the code is old and tied to solutions that made sense in the past; the technology has advanced and new techniques and approaches have appeared. In this situation it makes perfect sense to try to incorporate them to the framework, but integrating these with the old code is just a nightmare because too many things have to be changed. In order to get the most from the new techniques, the solution was to do a new framework from scratch and implement the new concepts from the core and build everything around it.

That's what happened in September 2012, the first release of ZF2 was launched after a lot of work, discussions, and lines of code, and now we have it available to build the next generation of online services.

This new version was created from scratch using the best approaches for every problem and is really fresh and flexible. All the components have been revised and a lot of them rewritten, they also solved well-known problems from ZF1.

In this book we will review all the changes made in ZF2. The book is written with a hands-on approach, so you will learn the concepts while programming and building something that is actually usable. We will create a social network from scratch using 90 percent of all the components available in ZF2. So, we will cover a huge part of it, and definitely all you will need in the future is to write your applications.

What this book covers

Chapter 1, *Let's Build a Social Network*, explains how we can create or build a social network. It also covers how we can architect a social network.

Chapter 2, *Setting Up the Environment*, is important because if you do not understand it you will not be able to do anything! Let's see how to use `Vagrant` and virtual machines.

Chapter 3, *Scratching the Surface of Zend Framework 2*, is essential to get the knowledge of the basic components of the framework and also to help understand concepts introduced later on.

Chapter 4, *The First Request/Response – Building the User Wall*, is essential in a social network. It is where the users can see the content, and where we can see how the API-centric approach works.

Chapter 5, *Handling Text Content – Posting Text*, is the simplest action a user can do. We will learn how to create forms and validate them.

Chapter 6, *Working with Images – Publishing Pictures*, is something people love to do on social networks. We will learn how to handle images and file uploads.

Chapter 7, *Dealing with URLs – Posting Links*, is essential to share things we found on the Internet. This is the perfect excuse to see how to scrap contents of other websites and create custom validators.

Chapter 8, *Dealing with Spam – Akismet to the Rescue*, explains how to protect ourselves from the spammers in the Internet.

Chapter 9, *Let's Read Feeds – A News Reader*, is something we use almost every day. Why don't we integrate it and learn how to deal with feeds on our social network?

Chapter 10, *Sign Up*, is the basic thing people have to do to start using a service that will store data tailored to them. And, as usual, we will have to send the typical confirmation e-mail. How do we accomplish that task?

Chapter 11, *Log In*, is something we are used to doing and our users will need to do. We will see how to be sure that our users are who they say and authenticate them.

Chapter 12, *Sending E-mails*, is a link everyone has used at least once. We will cover in-depth the e-mail sending mechanism and we will see how to organize the views for e-mails.

Chapter 13, *OAuth 2.0 Protocol Securing our API*, is essential if we don't want people to use it in the Web for which it was not designed. As we will expose the API to the world, we should learn how to protect it.

What you need for this book

For this book, you will need a computer, of course. In terms of software required, you don't have to be worried; all the components we use are open source, and everything is provided on the virtual machine we will set up in *Chapter 2*, *Setting Up the Environment*. The only thing you will need to do manually is install the last version of VirtualBox available for your operating system; the rest will be done almost automatically by the `Vagrant` scripts we provide.

Who this book is for

This book is for developers that want to start using Zend Framework 2 and want to build applications of an important size. We will cover a different approach on application architecture so this will also be helpful for you.

You don't need to have any experience with the framework or with Zend Framework 1, but if you have, it will be easier for you; but remember, it's not required. A good and solid knowledge of PHP 5.3 and an object-oriented paradigm is required because the whole framework is based on that and is strongly object oriented.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text are shown as follows: "On implementing this functionality, we will see how the new `Zend\Form` element handles the upload of files and how to process them."

A block of code is set as follows:

```
config.vm.box = "ubuntu-12.04"
config.vm.box_url = "http://files.vagrantup.com/precise64.box"
config.vm.network :private_network, ip: "192.168.56.2"
```

Any command-line input or output is written as follows:

```
cd /var/source-api
php composer.phar self-update
php composer.phar install
cd /var/source-client
php composer.phar self-update
php composer.phar install
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "This validator belongs to the **Email** field."

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase to address it.

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at `http://www.packtpub.com`. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from `http://www.packtpub.com/support`.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best

1
Let's Build a Social Network

In this chapter, we will see an overview of what we are going to build through this book, a social network. Also, we will cover the technical solution chosen to build it from scratch and thereby getting the most out of Zend Framework 2.

By the end of this chapter, you should have a good understanding of how the API-centric approach works and a clear picture of the project that we will build through this book.

Why ZF2?

Zend Framework 2, as you might know, is a framework that allows you to build applications using the components provided. The framework uses the best practices of the industry and the components are extensively tested and proven to be good. This means that you will build your application on top of a robust base.

The benefits of using a Zend framework against using your own framework is that you benefit from the knowledge of all the contributors of the project. Also, it will be easier to get more developers on board as it's a known framework with the documentation available online, and there are a lot of people who already know it or have worked with it.

What are we going to build?

The objective of this book is to show you as many components of **Zend Framework** 2 (**ZF2**) as possible. In order to achieve this, we will build a basic social network that will allow the user to post text, pictures, links, comments, and so on. This will cover all the basics of the framework and also demonstrate the usage of the majority of the components in common real-life scenarios.

Let's see all the functionalities and sections this social network will offer to the users.

Building a user profile

The user wall will be the main section where users can share content. Users will be able to post text, images, and URLs.

By building this section, we will experiment with the first request and building blocks of ZF2, and in the meantime we are going to see the technical approach chosen for the project in action.

Posting text

Users will be able to post text on their wall and also on their friends' walls using a simple form. To accomplish this, we will have a look at how forms work and how to use filters and validators to ensure that the data is secure and correct.

Uploading pictures

Pictures are one of the forms of media shared extensively by users of social networks. In our case, we will give users the opportunity to do this. On implementing this functionality, we will see how the new `Zend\Form` element handles the upload of files, and how to process them.

Sharing links

URLs are the last thing users will be able to share on the profile. A description will appear automatically with the link. This functionality will allow us to discover how to crawl contents from remote websites, filter them, and store them.

Posting comments

People also enjoy commenting on the content of others people and we will give the user a way to do this on our social network. An interesting part of ZF2 will be used in this section to fight spam. We will see how to use third-party services using Akismet.

Building news reader

Building news reader is a big section on the project. From here, the user will be able to add, remove, read, and organize RSS feeds. We will use `Zend\Feed` to do basic RSS actions and provide the frontend with data.

Registering and logging in

A basic action that users should be able to do is register and login to the social network. We will provide them with the forms to do this. We are going to integrate the session handling through `Zend\Session` and the login functionality using `Zend\Authentication` on this project.

E-mails

Another functionality that social networks provide is keeping the user updated about what is happening while he/she is away. We will implement a notification system and also a way to recover the password if you lose it. On building this section of the project, we will see the `Zend\Mail` component in action and how to use it to send e-mails.

Public APIs

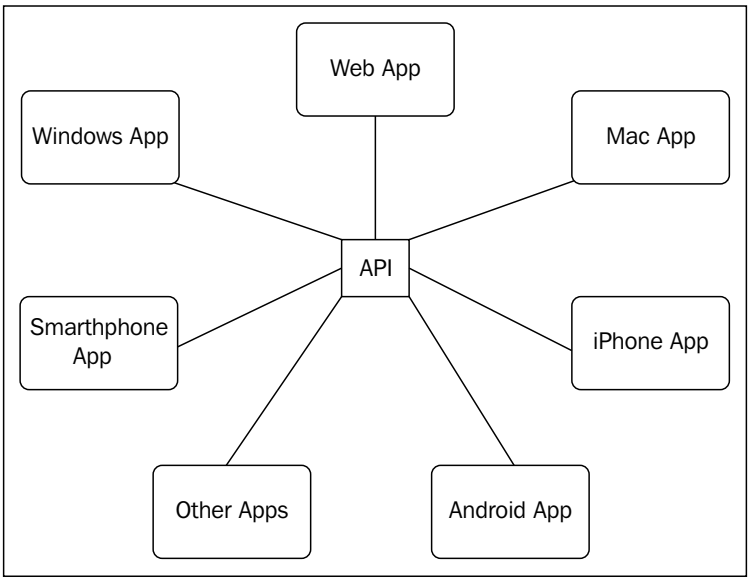
The last functionality we will provide in our project is the ability to integrate our data and functionalities in other projects. In order to accomplish this, we will expose our API to the outside world using OAuth 2.0. Here, we will learn how to put an OAuth 2.0 authentication mechanism with ZF2 in place.

The approach – API-centric app

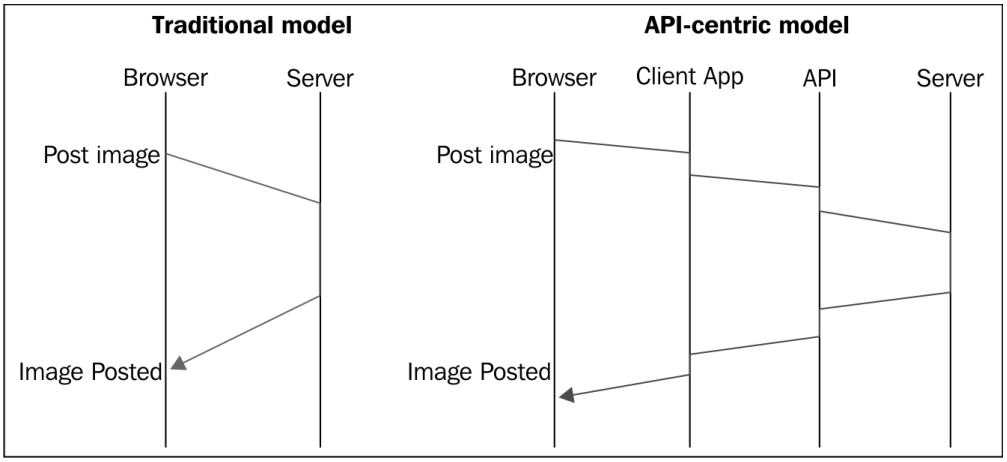
To build the social network, we will use an approach called API-centric. First of all, for those who are unfamiliar with the term API, it is an interface that we build to expose functionality. By doing this, we allow other applications to interact with us. For example, a news website can expose an API to allow people to retrieve articles from their archive by specifying the date of the article, the author, and so on.

An API-centric application is an application that executes all functionality that make calls to an internal API. For example, if a user on our social network is going to post a picture, our app will pass the image and details of the user to the API to execute the actual steps needed to store the image and publish it on the user profile.

The API-centric architecture looks like the following figure:



As you can see, API is the central point and everything else is built around it. The web app, the mac app, and so on are the clients that consume API. Now, let's compare the lifecycle of a request between an app-centric approach and a traditional model.



As you can see, in the traditional model you made requests directly to the server. The server in this case contains the logic of the client and also the business logic. In the API-centric model, the request is made from the browse to the client app; this application can be on the same server as the API or on a separate machine. Then, the client app will issue a request to the API. After this, the request will go back to the client app and finally to the user. In this case, we are separating the code of the client app from the code of the API. The client app acts as a proxy for the API that has the business logic. Note that the image doesn't represent the time spent on the request.

Since the explosive increase in the usage of smartphones, an increasing number of web applications have ended up with an application on the phone. Some of them just adapt the website to the phone screen size by removing or redesigning the interface, while others choose to build a native application that will run on the phone of the visitor.

The first benefit of this approach for our social network is that the core of the application is just an API and all the related clients will rely on it to use the functionality. This means we have a good separation of concerns, and we will have separated the business logic from the client logic. This will allow us to create a website to access the service and the possibility of building a native application for mobile phones or even a desktop program using the same API in future.

The second benefit is that the API is stateless; this means that the calls made to the API will not include anything about the session, and there is no session handling/management involved. This sounds wrong at the beginning but allows the developer to build a RESTful API that will not rely on the state of the user session or data stored on the session.

RESTful is the application of the REST architecture in web services. REST is an architectural style for designing networked applications. The idea behind it is to avoid complex mechanisms and use plain HTTP. A typical RESTful web service will use HTTP to do the four **CRUD** (**Create, Retrieve, Update, and Delete**) operations.

Another benefit is that the code can be tested further as you don't have to recreate the whole user session in order to test a functionality.

If we take a look at this approach from the server-side point of view, we can see some benefits; as we are separating the responsibilities of every component, we can assign different machines and resources to each of them. This way of organizing the servers allows us also to scale the components we need independent of the others.

One downside of the approach we can easily see is that if the API goes down, everything will go down and the clients will not be able to do anything.