

MASARYK UNIVERSITY
FACULTY OF INFORMATICS



Refactoring of Kdyby packages

BACHELOR'S THESIS

Filip Procházka

Brno, Spring 2017

MASARYK UNIVERSITY
FACULTY OF INFORMATICS



Refactoring of Kdyby packages

BACHELOR'S THESIS

Filip Procházka

Brno, Spring 2017

This is where a copy of the official signed thesis assignment and a copy of the Statement of an Author is located in the printed version of the document.

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Filip Procházka

Advisor: RNDr. Jaroslav Bayer

Acknowledgement

This is the acknowledgement for my thesis, which can span multiple paragraphs.

Abstract

This is the abstract of my thesis, which can span multiple paragraphs.

Keywords

package, kdyby, nette, doctrine, orm, composer, packagist

Contents

1	Introduction	1
2	Background for understanding the Kdyby	3
2.1	<i>A brief history of Kdyby</i>	3
2.2	<i>Technologies used</i>	3
2.2.1	Git and Github	3
2.2.2	Continuous integration	4
2.2.3	Composer	4
2.2.4	OAuth 2	5
2.2.5	Nette Framework	5
2.2.6	dibi	5
2.2.7	Doctrine 2 ORM	5
2.2.8	Symfony Framework	5
2.2.9	Monolog	6
2.2.10	RabbitMQ	6
2.2.11	ElasticSearch	6
2.2.12	Redis	6
2.2.13	PhpStan	6
2.2.14	Nette\Tester	6
2.3	<i>Techniques and design patterns</i>	7
2.3.1	Dependency Injection	7
2.3.2	Aspect Oriented Programming	7
2.3.3	Event Dispatcher	7
3	Current state of the Kdyby	9
3.1	<i>State of the project</i>	9
3.2	<i>State of each package</i>	9
3.2.1	Doctrine	9
3.2.2	Console	10
3.2.3	Events	10
3.2.4	Annotations	11
3.2.5	DoctrineCache	11
3.2.6	DoctrineMagicAccessors	11
3.2.7	DoctrineCollectionsReadonly	11
3.2.8	DoctrineCollectionsLazy	11

3.2.9	DoctrineDbalBatchImport	11
3.2.10	DoctrineForms	11
3.2.11	Curl	11
3.2.12	CurlCaBundle	12
3.2.13	Autowired	12
3.2.14	FormsReplicator	12
3.2.15	Translation	12
3.2.16	Validator	12
3.2.17	RabbitMq	12
3.2.18	Money	12
3.2.19	DoctrineMoney	12
3.2.20	Aop	12
3.2.21	Clock	12
3.2.22	Redis	13
3.2.23	ParseUseStatements	13
3.2.24	RedisActiveLock	13
3.2.25	TesterParallelStress	13
3.2.26	Monolog	13
3.2.27	ElasticSearch	13
3.2.28	DoctrineSearch	13
3.2.29	Geocoder	13
3.2.30	CsobPaygateNette	13
3.2.31	CsobPaymentGateway	13
3.2.32	Wkhtmltopdf	14
3.2.33	FakeSession	14
3.2.34	RequestStack	14
3.2.35	StrictObjects	14
3.2.36	Facebook	14
3.2.37	Google	14
3.2.38	Github	14
3.2.39	NettePhpServer	14
3.2.40	TesterExtras	14
3.2.41	HtmlValidatorPanel	14
3.2.42	BootstrapFormRenderer	15
3.2.43	PayPalExpress	15
3.2.44	PresentersLocator	15
3.2.45	SvgRenderer	15
3.2.46	QrEncode	15

4	Designing roadmap of refactoring	17
4.1	<i>Common requirements</i>	17
4.2	<i>Roadmap for each package</i>	17
4.2.1	Doctrine	17
4.2.2	Console	17
4.2.3	Events	17
4.2.4	Annotations	17
4.2.5	DoctrineCache	17
4.2.6	DoctrineMagicAccessors	17
4.2.7	DoctrineCollectionsReadonly	17
4.2.8	DoctrineCollectionsLazy	18
4.2.9	DoctrineDbalBatchImport	18
4.2.10	DoctrineForms	18
4.2.11	Autowired	18
4.2.12	FormsReplicator	18
4.2.13	Translation	18
4.2.14	Validator	18
4.2.15	RabbitMq	18
4.2.16	Money	18
4.2.17	DoctrineMoney	18
4.2.18	Aop	19
4.2.19	Clock	19
4.2.20	Redis	19
4.2.21	ParseUseStatements	19
4.2.22	RedisActiveLock	19
4.2.23	TesterParallelStress	19
4.2.24	Monolog	19
4.2.25	ElasticSearch	19
4.2.26	DoctrineSearch	19
4.2.27	Geocoder	19
4.2.28	CsobPaygateNette	20
4.2.29	CsobPaymentGateway	20
4.2.30	Wkhtmltopdf	20
4.2.31	FakeSession	20
4.2.32	RequestStack	20
4.2.33	StrictObjects	20
4.2.34	Facebook	20
4.2.35	Google	20

4.2.36	Github	20
4.2.37	NettePhpServer	20
4.2.38	TesterExtras	21
4.2.39	HtmlValidatorPanel	21
5	The refactoring process of the Kdyby	23
5.1	<i>Doctrine</i>	23
5.2	<i>Console</i>	23
5.3	<i>Events</i>	23
5.4	<i>Annotations</i>	23
5.5	<i>DoctrineCache</i>	23
5.6	<i>DoctrineMagicAccessors</i>	23
5.7	<i>DoctrineCollectionsReadonly</i>	23
5.8	<i>DoctrineCollectionsLazy</i>	24
5.9	<i>DoctrineDbalBatchImport</i>	24
5.10	<i>DoctrineForms</i>	24
5.11	<i>Autowired</i>	24
5.12	<i>FormsReplicator</i>	24
5.13	<i>Translation</i>	24
5.14	<i>Validator</i>	24
5.15	<i>RabbitMq</i>	24
5.16	<i>Money</i>	24
5.17	<i>DoctrineMoney</i>	25
5.18	<i>Aop</i>	25
5.19	<i>Clock</i>	25
5.20	<i>Redis</i>	25
5.21	<i>ParseUseStatements</i>	25
5.22	<i>RedisActiveLock</i>	25
5.23	<i>TesterParallelStress</i>	25
5.24	<i>Monolog</i>	25
5.25	<i>ElasticSearch</i>	25
5.26	<i>DoctrineSearch</i>	26
5.27	<i>Geocoder</i>	26
5.28	<i>CsobPaygateNette</i>	26
5.29	<i>CsobPaymentGateway</i>	26
5.30	<i>Wkhtmltopdf</i>	26
5.31	<i>FakeSession</i>	26
5.32	<i>RequestStack</i>	26

5.33	<i>StrictObjects</i>	26
5.34	<i>Facebook</i>	26
5.35	<i>Google</i>	27
5.36	<i>Github</i>	27
5.37	<i>NettePhpServer</i>	27
5.38	<i>TesterExtras</i>	27
5.39	<i>HtmlValidatorPanel</i>	27
6	Conclusion	29

List of Tables

List of Figures

1 Introduction

The Kdyby is an Open-source software (OSS) [1] project that I, Filip Procházka, lead and maintain. It is a set of PHP [2] libraries, that aim to ease writing of web applications.

Through my carer, I've been able to use the Kdyby in core business applications of companies such as Damejidlo.cz and Rohlik.cz. A lot of people consider my work useful enough, to incorporate it to their own applications as well.

As of writing this, the more popular libraries have hundreds of thousands of downloads. Five of Kdyby libraries have over quarter million downloads and one is approaching half a million with staggering amount of 470 thousands of downloads [3]. In conclusion, a sober estimate would be, that Kdyby libraries are used in hundreds of real production applications.

If I'll account only for the two biggest projects that I can confirm are using the Kdyby packages, over a billion Czech crowns [4] has literary flowed through the Kdyby. That is a big responsibility.

Over the years, I've had problems keeping up with the demand and the packages began to get obsolete. I wanna use this thesis as way to fix the situation.

I'm going to review the state of each library and decide it's future. Which means I'll either deprecate it and provide the users a suggestion for a better alternative, or fix the problems and refactor the library.

2 Background for understanding the Kdyby

2.1 A brief history of Kdyby

At 2006, few months after I've started attending my high school, I've began learning PHP and a friend introduced me to a concept called Content Management System (CMS) [5]. I've immediately started working on my own CMS, as a way to learn PHP and also as a way to make some money.

I've managed to make a working prototype, that was used in production on few websites. The oldest preserved version is archived at <https://github.com/fprochazka/kdyby-cms-old>. And as expected, it is full of security holes and badly written code.

Then the concept of Open-source software (OSS) [1] was introduced to me and I've decided to start working on everything openly, under a free license [6]. Sadly, since then, no new working version of the Kdyby CMS was ever released, because I've rewritten it from scratch exactly 10 times.

Around year 2012, I've realized this is not a good way to continue and split all useful code to separate libraries, that could be used more or less independently and have their own release cycle.

2.2 Technologies used

2.2.1 Git and Github

Git is a Version Control System, that is decentralized and very fast. [7] Github is a collaboration platform for software development using Git.

Each project has a page on Github called a repository, that can be used to inspect the Git history, files and other metadata. On the project repository page, there are issues and pull requests. Pull requests are a way to ask the maintainer of the repository, to incorporate provided code patch to the repository. It can be a bugfix, or new feature.

There are tools around pull requests that allow collaboration, code review and discussion about the provided code, so that the maintainers can help the contributors to provide the best code possible.

Kdyby is hosted and developed on Github.

2.2.2 Continuous integration

Continuous integration (CI) is the practice of merging all developer working copies to a shared mainline several times a day, to prevent merging conflicts. [8] But now-days, when somebody speaks about CI, they usually mean CI servers, that run prepared task on the provided code.

In practice, it means that as the developer is working on a feature or bugfix, they push the work in progress code into a repository, the code is then picked up by a CI server, that executes the tests, checks coding style and runs various other tasks, to verify that the code is not broken and works.

When the work is finished and all the task on CI server completed with success, the code can be probably safely integrated, provided that the tests for new or changed functionality were added.

Some popular CI services are Travis CI¹, CircleCI² and GitLab CI³. The Kdyby is using the Travis CI.

2.2.3 Composer

Composer is a tool for dependency management [9] in PHP. It allows you to declare the libraries your project depends on and it will manage (install or update) them for you [10].

Packages are usually published using Github with metadata in a file named `composer.json`, that is written in JSON [11] format.

Composer is decentralized, but has a single main metadata repository Packagist⁴. It stores and provides all the package metadata like available versions and where to download them.

All Kdyby libraries are published as Composer packages on Packagist and installing them using the Composer is the only officially supported installation method.

1. Travis CI <https://travis-ci.org/>

2. CircleCI <https://circleci.com/>

3. GitLab CI <https://about.gitlab.com/features/gitlab-ci-cd/>

4. Packagist <https://packagist.org/>

2.2.4 OAuth 2

OAuth is a protocol for authentication and authorization, that can be implemented into a web service. It allows for secure exchange of user information, allowing third party websites to implement a login and registration process, that simplifies these tasks for the user. Essentially allowing him or her to login or register to a services through the OAuth 2 provider with two clicks.

Kdyby provides packages for integrating Nette Framework with OAuth 2 providers, such as Facebook⁵, Google⁶ and Github⁷.

2.2.5 Nette Framework

Nette Framework is an OSS framework for creating web applications in PHP [12].

2.2.6 dibi

Dibi is a Database Abstraction Library for PHP. It supports a lot of significant databases: MySQL, PostgreSQL, SQLite, MS SQL, Oracle, Access and generic PDO and ODBC. [13]

2.2.7 Doctrine 2 ORM

Doctrine ORM is an Object-Relation Mapper (ORM) [14], which means it allows the programmer to create PHP classes called entities, that represent relational data in a database and are used to actually map the data from the database to the classes and back. In conclusion, it allows the programmer to write a fully Object-oriented (OOP) [15] applications.

2.2.8 Symfony Framework

Symfony is a PHP web application framework and a set of reusable PHP components/libraries, similar to Nette. [16]

5. Facebook <https://developers.facebook.com/docs/facebook-login/manually-build-a-login-flow>

6. Google <https://developers.google.com/identity/protocols/OAuth2>

7. Github <https://developer.github.com/v3/oauth/>

2.2.9 Monolog

Monolog is a logging library that sends your logs to files, sockets, inboxes, databases and various web services. This library implements the PSR-3 [17] interface that you can type-hint against in your own libraries to keep a maximum of interoperability. [18]

2.2.10 RabbitMQ

RabbitMQ is OSS message broker software (sometimes called message-oriented middleware) that implements the Advanced Message Queuing Protocol (AMQP). The RabbitMQ server is written in the Erlang programming language and is built on the Open Telecom Platform framework for clustering and failover. [19]

2.2.11 Elasticsearch

Elasticsearch is a search engine based on Lucene. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents. Elasticsearch is developed in Java and is released as open source under the terms of the Apache License. It is the most popular enterprise search engine . [20]

2.2.12 Redis

Redis is an in-memory database OSS project, that is networked, in-memory, and stores keys with optional durability. [21]

2.2.13 PhpStan

PHPStan focuses on finding errors in your code without actually running it. It catches whole classes of bugs even before you write tests for the code. [22]

2.2.14 Nette\Tester

Nette\Tester is an unit testing [23] framework for the PHP. [24]

2.3 Techniques and design patterns

2.3.1 Dependency Injection

Inversion of control is a design principle in which custom-written portions of a computer program receive the flow of control from a generic framework.

Dependency injection is a technique whereby one object supplies the dependencies of another object. Passing the service to the client, rather than allowing a client to build or find the service, is the fundamental requirement of the pattern. [25]

2.3.2 Aspect Oriented Programming

In computing, aspect-oriented programming (AOP) is a programming paradigm that aims to increase modularity by allowing the separation of cross-cutting concerns. It does so by adding additional behavior to existing code (an advice) without modifying the code itself, instead separately specifying which code is modified via a "pointcut" specification, such as "log all function calls when the function's name begins with 'set'". This allows behaviors that are not central to the business logic (such as logging) to be added to a program without cluttering the code core to the functionality. [26]

2.3.3 Event Dispatcher

The Event Dispatcher is a pattern for writing modular code. It allows to create extension points in the library or application, that another library or application can hook into and change or extend the behavior.

Typically, the extension points are called hooks or events, and the new functionality is provided with objects called listeners.

3 Current state of the Kdyby

To be able to lay out the roadmap, first we have to know the current state of each Kdyby package, the original purpose and the current requirements. We shall only review those packages, that actually made it to production and at least one usable version was released.

Few years back I was really eager to solve all the problems around developing web applications in PHP and I've created few GitHub repositories as a reminder for me, to start working on those problems. And I've actually started to work on some, for example DoctrineForms is one of them, but it was never "officially released". The rest I've not even started working on.

3.1 State of the project

As of 28.4.2017, there are still 68 open pull requests with 622 of them resolved, and 217 open issues, with of them 401 resolved. There is no coding standard being enforced automatically on any package. No static analysis tool is checking the code. But most of the packages have unit and integration tests and linter checking the code for multiple versions of PHP.

Almost all of the packages try to be compatible with PHP 5.4, but PHP 5.4 had end of life at 3.9.2015¹ and is no longer supported by PHP developers.

3.2 State of each package

Let's review each relevant package separately.

3.2.1 Doctrine

Kdyby\Doctrine (or simply Doctrine) is an integration of Doctrine 2 ORM into Nette Framework.

1. PHP 5.4 had end of life at 3.9.2015 <http://php.net/eol.php>

3. CURRENT STATE OF THE KDYBY

Doctrine 2 itself is separated into several packages, mainly doctrine/orm², doctrine/common³, doctrine/annotations⁴, doctrine/cache⁵ and doctrine/collections⁶. What started as a monolith integration in Kdyby, got separated into Events 3.2.3, Console 3.2.2, Annotations 3.2.4 and DoctrineCache 3.2.5 for reusability.

Over the years, it cumulated a lot of responsibilities, that don't belong to it. I have already started extracting few of them in the past, for example an entity prototyping tool 3.2.6, collection utilities 3.2.8, 3.2.7 and helper for loading big SQL scripts to the database 3.2.9.

There is a big issue Chop up the package⁷ that discusses what other parts should be separated and dropped completely.

New versions of Nette and Doctrine 2 ORM were released and completely new versions are being prepared, which the integration cannot be currently used with.

3.2.2 Console

Kdyby\Console (or simply Console) is an integration of Symfony Framework Console Component, that allows for writing interactive cli applications. Doctrine 3.2.1 depends on this package and is the reason this package exists.

There are tasks, that are better suited for console interaction, than a web interface. Among others, Doctrine 2 ORM has tools for generating a database schema from the entities metadata and there is a console command for it, that is written using Symfony Console.

3.2.3 Events

Kdyby\Events (or simply Events) provides an event dispatcher 2.3.3 implementation for Nette Framework.

It started as an integration of Doctrine 2 ORM EventManager, but then it evolved into a standalone system with support for lazy initial-

2. doctrine/orm <https://github.com/doctrine/doctrine2>

3. doctrine/common <https://github.com/doctrine/common>

4. doctrine/annotations <https://github.com/doctrine/annotations>

5. doctrine/cache <https://github.com/doctrine/cache>

6. doctrine/collections <https://github.com/doctrine/collections>

7. Chop up the package <https://github.com/Kdyby/Doctrine/issues/238>

ization of listeners and it also contains a naive bridge for Symfony Framework EventDispatcher Component.

Creating such interchangeable eventing system turned out to be a mistake, because it is a maintenance hell. The systems should have stayed separate.

3.2.4 Annotations

Kdyby\Annotations (or simply Annotations) is an

3.2.5 DoctrineCache

Lorem ipsum.

3.2.6 DoctrineMagicAccessors

Lorem ipsum.

3.2.7 DoctrineCollectionsReadonly

Lorem ipsum.

3.2.8 DoctrineCollectionsLazy

Lorem ipsum.

3.2.9 DoctrineDbalBatchImport

Lorem ipsum.

3.2.10 DoctrineForms

Lorem ipsum.

3.2.11 Curl

Lorem ipsum.

3. CURRENT STATE OF THE KDYBY

3.2.12 CurlCaBundle

Lorem ipsum.

3.2.13 Autowired

Lorem ipsum.

3.2.14 FormsReplicator

Lorem ipsum.

3.2.15 Translation

Lorem ipsum.

3.2.16 Validator

Lorem ipsum.

3.2.17 RabbitMq

Lorem ipsum.

3.2.18 Money

Lorem ipsum.

3.2.19 DoctrineMoney

Lorem ipsum.

3.2.20 Aop

Lorem ipsum.

3.2.21 Clock

Lorem ipsum.

3.2.22 Redis

Lorem ipsum.

3.2.23 ParseUseStatements

Lorem ipsum.

3.2.24 RedisActiveLock

Lorem ipsum.

3.2.25 TesterParallelStress

Lorem ipsum.

3.2.26 Monolog

Lorem ipsum.

3.2.27 ElasticSearch

Lorem ipsum.

3.2.28 DoctrineSearch

Lorem ipsum.

3.2.29 Geocoder

Lorem ipsum.

3.2.30 CsobPaygateNette

Lorem ipsum.

3.2.31 CsobPaymentGateway

Lorem ipsum.

3. CURRENT STATE OF THE KDYBY

3.2.32 Wkhtmltopdf

Lorem ipsum.

3.2.33 FakeSession

Lorem ipsum.

3.2.34 RequestStack

Lorem ipsum.

3.2.35 StrictObjects

Lorem ipsum.

3.2.36 Facebook

Lorem ipsum.

3.2.37 Google

Lorem ipsum.

3.2.38 Github

Lorem ipsum.

3.2.39 NettePhpServer

Lorem ipsum.

3.2.40 TesterExtras

Lorem ipsum.

3.2.41 HtmlValidatorPanel

Lorem ipsum.

3.2.42 BootstrapFormRenderer

Lorem ipsum.

3.2.43 PayPalExpress

Lorem ipsum.

3.2.44 PresentersLocator

Lorem ipsum.

3.2.45 SvgRenderer

Lorem ipsum.

3.2.46 QrEncode

Lorem ipsum.

4 Designing roadmap of refactoring

In this chapter, I'm going to lay out the plan for the refactoring itself and set some specific goals for each package and for the project itself.

4.1 Common requirements

Lorem ipsum.

4.2 Roadmap for each package

4.2.1 Doctrine

Lorem ipsum.

4.2.2 Console

Lorem ipsum.

4.2.3 Events

Lorem ipsum.

4.2.4 Annotations

Lorem ipsum.

4.2.5 DoctrineCache

Lorem ipsum.

4.2.6 DoctrineMagicAccessors

Lorem ipsum.

4.2.7 DoctrineCollectionsReadonly

Lorem ipsum.

4.2.8 DoctrineCollectionsLazy

Lorem ipsum.

4.2.9 DoctrineDbalBatchImport

Lorem ipsum.

4.2.10 DoctrineForms

Lorem ipsum.

4.2.11 Autowired

Lorem ipsum.

4.2.12 FormsReplicator

Lorem ipsum.

4.2.13 Translation

Lorem ipsum.

4.2.14 Validator

Lorem ipsum.

4.2.15 RabbitMq

Lorem ipsum.

4.2.16 Money

Lorem ipsum.

4.2.17 DoctrineMoney

Lorem ipsum.

4.2.18 Aop

Lorem ipsum.

4.2.19 Clock

Lorem ipsum.

4.2.20 Redis

Lorem ipsum.

4.2.21 ParseUseStatements

Lorem ipsum.

4.2.22 RedisActiveLock

Lorem ipsum.

4.2.23 TesterParallelStress

Lorem ipsum.

4.2.24 Monolog

Lorem ipsum.

4.2.25 ElasticSearch

Lorem ipsum.

4.2.26 DoctrineSearch

Lorem ipsum.

4.2.27 Geocoder

Lorem ipsum.

4.2.28 CsobPaygateNette

Lorem ipsum.

4.2.29 CsobPaymentGateway

Lorem ipsum.

4.2.30 Wkhtmltopdf

Lorem ipsum.

4.2.31 FakeSession

Lorem ipsum.

4.2.32 RequestStack

Lorem ipsum.

4.2.33 StrictObjects

Lorem ipsum.

4.2.34 Facebook

Lorem ipsum.

4.2.35 Google

Lorem ipsum.

4.2.36 Github

Lorem ipsum.

4.2.37 NettePhpServer

Lorem ipsum.

4.2.38 TesterExtras

Lorem ipsum.

4.2.39 HtmlValidatorPanel

Lorem ipsum.

5 The refactoring process of the Kdyby

This chapter documents what I've accomplished with each package in detail.

5.1 Doctrine

Lorem ipsum.

5.2 Console

Lorem ipsum.

5.3 Events

Lorem ipsum.

5.4 Annotations

Lorem ipsum.

5.5 DoctrineCache

Lorem ipsum.

5.6 DoctrineMagicAccessors

Lorem ipsum.

5.7 DoctrineCollectionsReadonly

Lorem ipsum.

5.8 DoctrineCollectionsLazy

Lorem ipsum.

5.9 DoctrineDbalBatchImport

Lorem ipsum.

5.10 DoctrineForms

Lorem ipsum.

5.11 Autowired

Lorem ipsum.

5.12 FormsReplicator

Lorem ipsum.

5.13 Translation

Lorem ipsum.

5.14 Validator

Lorem ipsum.

5.15 RabbitMq

Lorem ipsum.

5.16 Money

Lorem ipsum.

5.17 DoctrineMoney

Lorem ipsum.

5.18 Aop

Lorem ipsum.

5.19 Clock

Lorem ipsum.

5.20 Redis

Lorem ipsum.

5.21 ParseUseStatements

Lorem ipsum.

5.22 RedisActiveLock

Lorem ipsum.

5.23 TesterParallelStress

Lorem ipsum.

5.24 Monolog

Lorem ipsum.

5.25 ElasticSearch

Lorem ipsum.

5.26 DoctrineSearch

Lorem ipsum.

5.27 Geocoder

Lorem ipsum.

5.28 CsobPaygateNette

Lorem ipsum.

5.29 CsobPaymentGateway

Lorem ipsum.

5.30 Wkhtmltopdf

Lorem ipsum.

5.31 FakeSession

Lorem ipsum.

5.32 RequestStack

Lorem ipsum.

5.33 StrictObjects

Lorem ipsum.

5.34 Facebook

Lorem ipsum.

5.35 Google

Lorem ipsum.

5.36 Github

Lorem ipsum.

5.37 NettePhpServer

Lorem ipsum.

5.38 TesterExtras

Lorem ipsum.

5.39 HtmlValidatorPanel

Lorem ipsum.

6 Conclusion

We made it!

Bibliography

- [1] Wikipedia. *Open-source software* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: https://en.wikipedia.org/w/index.php?title=Open-source_software&oldid=776201239.
- [2] Wikipedia. *PHP* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: <https://en.wikipedia.org/w/index.php?title=PHP&oldid=775984544>.
- [3] *Packages from kdyby - Packagist*. 2017. URL: <https://packagist.org/packages/kdyby/>.
- [4] Matt Jedlička. *Rohlik.cz loni prodal zboží za miliardu, letos chce konečně zisk*. 2017. URL: <http://tyinternety.cz/e-commerce/rohlik-cz-loni-dosahl-na-miliardovy-obrat-letos-chce-konecne-zisk/>.
- [5] Wikipedia. *Content management system* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: https://en.wikipedia.org/w/index.php?title=Content_management_system&oldid=775559667.
- [6] Wikipedia. *Free software license* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: https://en.wikipedia.org/w/index.php?title=Free_software_license&oldid=776336813.
- [7] Scott Chacon and Ben Straub. *Pro Git*. Apress, 2014. ISBN: 1484200772. URL: <https://www.amazon.com/Pro-Git-Scott-Chacon/dp/1484200772>.
- [8] Wikipedia. *Continuous integration* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 28-April-2017]. 2017. URL: https://en.wikipedia.org/w/index.php?title=Continuous_integration&oldid=777232210.
- [9] Wikipedia. *Package manager* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: https://en.wikipedia.org/w/index.php?title=Package_manager&oldid=775128084.
- [10] Composer Community. *Introduction - Composer*. 2017. URL: <https://getcomposer.org/doc/00-intro.md>.

BIBLIOGRAPHY

- [11] Wikipedia. *JSON* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: <https://en.wikipedia.org/w/index.php?title=JSON&oldid=775860167>.
- [12] Wikipedia. *Nette Framework* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: https://en.wikipedia.org/w/index.php?title=Nette_Framework&oldid=766339218.
- [13] David Grudl. *Dibi is Database Abstraction Library for PHP 5 - Homepage*. 2017. URL: <https://dibiphp.com/>.
- [14] Wikipedia. *Object-relational mapping* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: https://en.wikipedia.org/w/index.php?title=Object-relational_mapping&oldid=769454010.
- [15] Wikipedia. *Object-oriented programming* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: https://en.wikipedia.org/w/index.php?title=Object-oriented_programming&oldid=775231020.
- [16] Wikipedia. *Symfony* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: <https://en.wikipedia.org/w/index.php?title=Symfony&oldid=769598534>.
- [17] PHP Framework Interop Group. *PHP Standards Recommendations*. 2017. URL: <http://www.php-fig.org/psr/>.
- [18] Jordi Boggiano. *Monolog - Logging for PHP*. 2017. URL: <https://seldaek.github.io/monolog/>.
- [19] Wikipedia. *RabbitMQ* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: <https://en.wikipedia.org/w/index.php?title=RabbitMQ&oldid=774630105>.
- [20] Wikipedia. *Elasticsearch* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: <https://en.wikipedia.org/w/index.php?title=Elasticsearch&oldid=776540688>.
- [21] Wikipedia. *Redis* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: <https://en.wikipedia.org/w/index.php?title=Redis&oldid=774326024>.
- [22] Ondřej Mirtes. *PHPStan - PHP Static Analysis Tool*. 2017. URL: <https://github.com/phpstan/phpstan>.
- [23] Wikipedia. *Unit testing* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: https://en.wikipedia.org/w/index.php?title=Unit_testing&oldid=773620225.

BIBLIOGRAPHY

- [24] David Grudl. *Nette Tester – enjoyable unit testing*. 2017. URL: <https://tester.nette.org/en/>.
- [25] Martin Fowler. *Inversion of Control Containers and the Dependency Injection pattern*. 2004. URL: <https://martinfowler.com/articles/injection.html>.
- [26] Wikipedia. *Aspect-oriented programming* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2016. URL: https://en.wikipedia.org/w/index.php?title=Aspect-oriented_programming&oldid=751829822.