nuovo studi meridionalismo

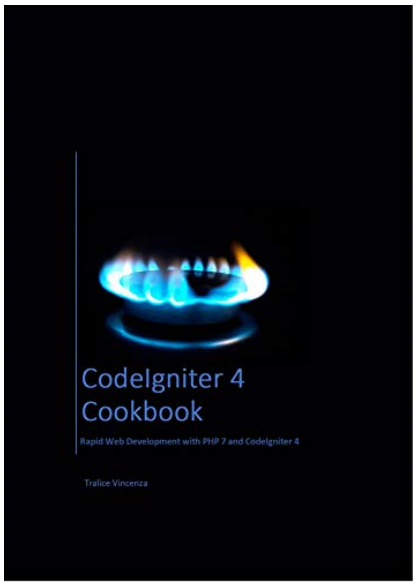
Anno VI - n. 11/Ottobre 2020
ISSN (2464-9279)
RECENSIONI

Vincenza Tralice, *CodeIgniter 4 Cookbook: Rapid Web Development with PHP 7 and CodeIgniter 4 (English Edition),* Amazon, Seattle, 2020

ASIN: B089XMXDCN

Pagine: 219

di Diego Santoro



This book is about web development with CodeIgniter 4 and PHP 7. PHP has been the preferred choice for web development for years and many statistics report that PHP is installed on the 80% of the servers in the world. The reason of PHP popularity lies in the ease of its usage: you can easily create a website putting two or three PHP scripts together. I call this approach to the web development the "script-approach" which has proven not to be well suitable for middle and large web applications. The script-approach does not help developers with code maintenance. In the latest decade, we assisted to a continuous emerging of new web frameworks. Most of them have a high learning curve and a big footprint and for such reasons they quickly lost popularity. Among those frameworks, CodeIgniter emerged for its simplicity providing to developers the right tool for fast prototyping and rapid web development.

The book is organized is six chapters as follows:

- Chapter 1 is an introduction to the web development with PHP and CodeIgniter;
- Chapter 2 walks you through the preparation of the development environment;

- Chapter 3 introduces PHP syntax fundamentals for non-PHP developers to easily understand the examples presented in the book;
- Chapter 4 presents routes, controllers and CodeIgniter core features such as logging and dependency injection;
- Chapter 5 goes deep into the details of routes and controllers presenting requests and responses objects (views, json objects);
- Chapter 6 presents CodeIgniter database API: Query Builder, Model and Entity classes.

An extract[1]:

Chapter 1. Getting Started with CodeIgniter 4

This book is about CodeIgniter 4, an MVC-based framework for building web applications. The framework has always been appreciated since its first release by the PHP community for its high performance and low learning curve. CodeIgniter allows you to build and maintain high-quality web applications without feeling the weight of a heavy framework. This chapter first presents you the old and new approach of doing web development with the PHP programming language and then it introduces the CodeIgniter 4 framework and its features.

1.1 Web Development with PHP

Web development with PHP has changed a lot over the years. In the past you could easily bump into web applications written as a bunch of poorly designed scripts. Those scripts would contain repeated pieces of code with a lot of file imports. If you were lucky then scripts would contain business logic right at the beginning preparing associative arrays or collections of objects holding model data. Then, those data structures were formatted some lines of code later and presented to the user. In the worst case, PHP code was completely mixed with the HTML code and you could easily find yourself to maintain scripts that open a database connection right before the opening HTML table tag.

Maintaining and debugging old PHP applications was and remains a nightmare.

The old approach of doing web development is to be avoided whenever possible since it poses serious limitations to the size and robustness of your applications with real risk to the security of your business and clients.

Writing web applications as a bunch of scripts is simple and fast, but the application design phase is basically skipped. It seems to be the right way for small projects, but codebase can grow big in no time when you are busy with the collection of client requirements and project development. You can easily end up with a pretty messed-up project.

Thinking of your applications as a set of scripts does not help you make a clear separation of concerns about aspects and components of your application. It surely leads to a complete lack of design, especially when it is applied to large-scaled projects.

Nowadays web development keyword is Framework.

Frameworks provide a standard and solid way to build and deploy web applications much faster than you could if you wrote code from scratch. It is worth noting that development from scratch is bug prone as well as being a complete waste of time for two main reasons: you are not respecting simple code reuse principles such as the DRY (Don't Repeat Yourself) principle and you are forced to prepare and run test campaigns for the same application logic in all new projects. The framework frees you from writing common application logic that you would write anyway for any new project, and, at the same time, it allows you to be completely focused on the business logic.

There are a lot of PHP frameworks available on the market but sometimes they forget the programming language they are intended to be used with: PHP.

[1] https://www.amazon.it/CodeIgniter-Cookbook-Rapid-Development-English-ebook/dp/B089XMXDCN/ref=sr_1_1?__mk_it_IT=ÅMÅŽÕÑ&dchild=1&keywords=vincenza+tralice&qid=1600758217&s=books&sr=1-1

PHP is a programming language born for web development. The time that the PHP interpreter processes the first line code of your script, it has already created and prepared the entire environment you need to serve the incoming request. If you already had experience with PHP for web development then you know that the PHP interpreter creates environment variables containing all request data, such as cookies, user sessions, uploaded files and user input. Everything is ready and available; you just need to prepare the response. The ease of writing web applications with PHP is one of the reasons that leads the script-approach to be preferred to proper application design.

Many frameworks introduce complex programming models that wrap native PHP functionalities. Of course, the result is a worsening of application performance mainly caused by the loading of these wrapped functionalities at bootstrap time. It is worth noting that a bootstrap of the framework is required every time a request is sent to the server. Serving a request must be as fast as no framework would be in the middle between environment creation and application logic.

Large PHP applications just need a guided track to keep the code organized.

Code organization and route management are the two most important things that will likely get out of hands as your codebase grows.

Writing well-organized large PHP applications can be achieved with simple code organization rules provided by specific patterns for web development. Keeping code well-organized is also achieved by decoupling request URLs from the piece of code that serves them. In old PHP applications, changing the script name meant to change the request route. If users save the application URL in the browser bookmark and the route is changed then it is impossible for them to reach the same application page again.

Most of all modern web frameworks implement the well-known pattern called Model-View-Controller (MVC) which has proven to be well-suited for web development over the years.

The pattern was first used in the late '70 and today it is widely used by different technologies such as Java, Ruby and .NET. The MVC pattern helps keep separated three important components of any web application:

- The Model, which represents domain application data. It holds the application data and implements the business logic.
- The View, which represents the templates according to which application data must be presented to the user. The logic to put in a view must be strictly necessary to present the data to the user.
- The Controller. The controller role is to handle the incoming request, execute the model and return the model result to the view.

A slightly different interpretation of the MVC pattern is sometimes offered to developers. This other version recommends that the model is just a container of application data and that all business logic lies in the controller. The MVC pattern has been interpreted in different ways over the years and there are many versions of it. Of course, all of them are well accepted provided that the context in which they are implemented is well defined.

Figure 1.1 shows a schema representing the interactions between the three components.

In detail, the controller first receives the request, then invokes the model. The model result is passed to the view. The view renders the model according to the defined format (generally HTML). Finally, the response is sent to the browser.

Many PHP frameworks available on the market, such as CodeIgniter, Laravel or Slim, implement the MVC pattern.

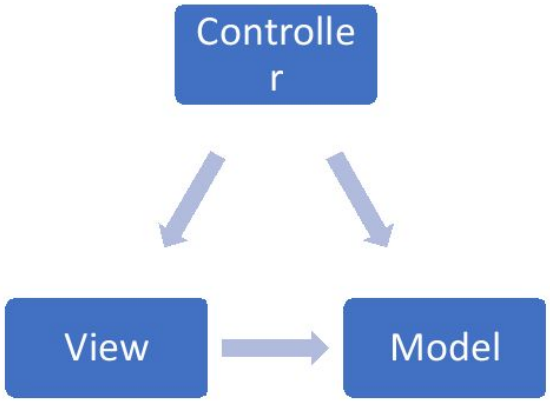


Figure 1.1. Model-View-Controller schema

1.2 CodeIgniter 4

CodeIgniter is an MVC-based framework released by EllisLab in 2006. Since then, the framework went through many improving changes which were also made possible thanks to newer versions of PHP. In the 2014 CodeIgniter was acquired by the British Columbia Institute of Technology and then it was officially declared as a community-maintained project.

The CodeIgniter 4 is a complete renewed framework compared to its first releases. It was written by using the latest features of PHP 7. Programming with CodeIgniter has always been easy and full of fun since its first version. Web applications development with CodeIgniter 4 is a real pleasure.

CodeIgniter was written with the purpose of dealing with performances demanded by real-world scenarios and it has always proven to be extremely fast compared to other frameworks on the market.

CodeIgniter is also characterized by a low learning curve and it also requires a small footprint. It is ideal for developers that do not like to be forced to follow complex programming models introduced by frameworks.

Examples of the CodeIgniter high performances can be easily seen by analyzing what its competitors got completely wrong. Generally, CodeIgniter competitors hold model results in memory before those are rendered to the browser. Forcing developers to pass an object representation of the response to the controller is not nice since keeping data in memory results in performance loss. Too much memory is allocated by their internal components throughout the request processing. CodeIgniter has nothing of all that. CodeIgniter allows us to make data available to users as easy as printing them with the echo function. In addition, other PHP frameworks load all features and functionalities at the bootstrap making them available to the developer at the time the application starts. Again, CodeIgniter does not do that. If your code needs extra functionalities from the framework then you must explicitly load them. CodeIgniter also provides you a mechanism to auto-load features the time they are really needed saving precious resources such as memory and processing time.

CodeIgniter does not stack useless functionalities on top of what the PHP interpreter already provides by default. It basically gets the best out of PHP and provides exactly what developers need to write fast-growing applications. It completes what PHP has left out and delegated to developers. It requires zero configuration, and even when you really must configure something it makes it easy for you by providing well-defined configuration classes.

The latest version of CodeIgniter also managed to adhere to some recommendations, better known as PSRs and discussed in the next paragraph, issued by a group of professionals born from the PHP community with the purpose of making the framework interoperable with other vendors, frameworks and tools.

CodeIgniter also takes the security of your application very seriously. It provides several features and techniques to enforce basic security principles and practices.

CodeIgniter guarantees to respect and follow the OWASP (which stands for Open Web Application Security Project) recommendations.

This attention to the OWASP recommendations makes it possible to prevent and mitigate the effect of many web attacks, such as SQL Injection, Cross Site Scripting and Cross Site Request Forgery, making your application secure for the happiness of your clients.

1.3 PSR Compliance

CodeIgniter 4 is compliant to some recommendations issued by the Framework Interop Group (FIG), also referred to as PSR-FIG where PSR stands for PHP Standards Recommendations.

FIG is a group of framework developers created in 2009 at the php|tek conference. Over the years, several new members have joined, and it counts more than 20 professionals at the time of writing. PSR-FIG's goal is to help interoperability between frameworks by ratifying interfaces, style guides and applications aspects.

It is worth noting that FIG is not an official PHP group, but its work is really appreciated by the PHP community since it is intended to make the developers job easier especially when it comes to the integration with other libraries and vendors. It promotes code reuse.

CodeIgniter is not a member of the FIG but it still tries to be compliant to some of their recommendations such as:

- PSR-1. This recommendation is about basic coding standards and covers naming conventions for files, classes and methods. CodeIgniter 4 styles guide guarantees to meet this recommendation and to add even more of its own.
- PSR-2. This recommendation is about coding style guides such as whitespace, indentation and alignment.
- PSR-3. Logger Interface. This recommendation is about rectifying logging interfaces. The CodeIgniter Logger interface is compliant to it.
- PSR-4. Autoloading Standard. This recommendation covers files and namespaces organization with the purpose of making standardized methods for classes autoloading. It is one of the most important recommendations since it promotes interoperability and integrations of vendors packages. CodeIgniter 4 is compliant to it.
- PSR-6. Caching Interface. CodeIgniter is not compliant to this recommendation since it is considered by the CodeIgniter development team to go further the scope of the recommendation itself.
- PSR-7. HTTP Message Interface. This recommendation is very important and tries to standardize the interface of the HTTP request and response. Other frameworks like Slim meet this recommendation but CodeIgniter 4 does not.