Learn by doing: less theory, more results

CouchDB and PHP Web Development

Get your PHP application from conception to deployment by leveraging CouchDB's robust features

Beginner's Guide

Tim Juravich

[PACKT] PUBLISHING open source community experience distilled

CouchDB and PHP Web Development Beginner's Guide

Get your PHP application from conception to deployment by leveraging CouchDB's robust features

Tim Juravich

[PACKT] open source
community experience distilled
PUBLISHING

BIRMINGHAM - MUMBAI

CouchDB and PHP Web Development Beginner's Guide

First published: June 2012

Production Reference: 1150612

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

ISBN 978-1-849513-58-6

www.packtpub.com

Cover Image by Parag Kadam (paragvkadam@gmail.com)

Credits

Author

Tim Juravich

Reviewers

Gonzalo Ayuso

David Carr

Wenbert Del Rosario

Acquisition Editor

Sarah Cullington

Lead Technical Editors

Arun Nadar

Chris Rodrigues

Technical Editor

Lubna Shaikh

Project Coordinator

Leena Purkait

Proofreader

Kevin McGowan

Indexer

Monica Ajmera Mehta

Graphics

Manu Joseph

Valentina D'silva

Production Coordinator

Arvindkumar Gupta

Cover Work

Arvindkumar Gupta

About the Author

Tim Juravich is an experienced product, program, and technology leader who has spent the past decade leading teams through a variety of projects in PHP, Ruby, and .NET. After gaining experience at several Fortune 500 companies, Tim discovered entrepreneurship, founded three of his own startups, and helped dozens of other startups open their doors.

Tim currently serves as the Director of Program Management for Thinktiv, a venture accelerator. When not at work, Tim actively mentors engineers, contributes to open source projects, and works on a variety of side projects.

Check out Tim's blog at `http://juravich.com`, and be sure to follow him on Twitter `@timjuravich`

I would like to thank my loving parents, my older (but smaller) brother Jon, and my wife Leigha. Without Leigha's support and love through our first year of marriage, this book, and much more, would not have been possible.

I would also like to thank my clients and colleagues who have provided invaluable opportunities for me to shape my career, my life, and my perspective on technology.

About the Reviewers

Gonzalo Ayuso is a web architect with more than 10 years of experience in web development, specializing in open source technologies. He has experience in delivering scalable, secure, and high-performing web solutions to large scale enterprise clients. He has a varied background, always related to Linux and the Internet. He is mainly focused on Internet technologies, databases, and programming languages (mostly PHP, Python, and JavaScript). You can check his blog at `gonzalo123.wordpress.com` or follow him on Twitter `@gonzalo123`.

Wenbert Del Rosario is from Cebu, Philippines. He started his career as a web developer in college, learning PHP and Adobe Photoshop. He works with open source technologies – Zend Framework, Code Igniter, MySQL, jQuery, and Wordpress are some of the tools he has up his sleeve. He also works with Django (Python) and Ruby on Rails.

In his free time, he loves to work on personal projects. He also does some freelance and consulting. He knows he has a lot to learn, but his experience has taught him to solve real-world and business problems. He is very passionate and shares some of his thoughts and day-to-day encounters through his blog (`http://blog.ekini.net`).

Wenbert's latest employer is Norwegian Pacific Offshore. He also has worked for Lexmark Research and Development Corporation in Cebu.

I would like to thank my family and my wife, Noeme, for all their support and encouragement.

---

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

PHP and CouchDB Web Development will teach you the fundamentals of combining CouchDB and PHP to create a full application from conception to deployment. This book will direct you in developing a basic social network, while guiding you through some of the common pitfalls that are frequently associated with NoSQL databases.

What this book covers

Chapter 1, *Introduction to CouchDB*, provides a quick definition of NoSQL and an overview of CouchDB.

Chapter 2, *Setting up your Development Environment*, sets up your computer for developing an application with PHP and CouchDB.

Chapter 3, *Getting Started with CouchDB and Futon*, defines CouchDB documents and shows how to manage them both from the command-line and within Futon – CouchDB's built-in administration utility.

Chapter 4, *Starting your Application*, creates a simple PHP framework to house your application and publishes this code to GitHub.

Chapter 5, *Connecting your Application to CouchDB*, connects your application to CouchDB using a variety of methods, and ultimately picks the right solution for your application.

Chapter 6, *Modeling Users*, creates users within your application and handles document creation and authentication with CouchDB.

Chapter 7, *User Profiles and Modeling Posts*, perfects your user profile using Bootstrap and posts content to CouchDB.

Chapter 8, *Using Design Documents for Views and Validation*, explores CouchDB's exclusive use of Design Documents to improve the quality of your application.

Chapter 9, *Adding Bells and Whistles to your Application*, leverages existing tools to simplify and improve your application.

Chapter 10, *Deploying your Application*, shows your application to the world, and teaches you how to launch your application and database using a variety of Cloud services.

Bonus Chapter, *Replicating your Data*, finds out how to use CouchDB's replication system to scale your application as it grows.

You can download the *Bonus Chapter* from `http://www.packtpub.com/sites/default/files/downloads/Replicating_your_Data.pdf`.

What you need for this book

You'll need a modern computer with Mac OSX. *Chapter 1*, *Introduction to CouchDB,* will provide the setup instructions for Linux and Windows machines, and the code written in this book will work on any machine. However, the majority of the command-line statements and applications that we'll use in this book are Mac OSX-specific.

Who this book is for

This book is for beginners and intermediate PHP developers, who are interested in using CouchDB development in their projects. Advanced PHP developers will appreciate the familiarity of the PHP architecture, and can easily learn how to incorporate CouchDB into their existing development experiences.

Conventions

In this book, you will find several headings appearing frequently.

To give clear instructions of how to complete a procedure or task, we use:

Time for action – heading

1. Action 1
2. Action 2
3. Action 3

Instructions often need some extra explanation so that they make sense, so they are followed with:

What just happened?

This heading explains the working of tasks or instructions that you have just completed.

You will also find some other learning aids in the book, including:

Pop quiz – heading

These are short multiple choice questions intended to help you test your own understanding.

Have a go hero – heading

These set practical challenges and give you ideas for experimenting with what you have learned.

You will also 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: " It's difficult to standardize the `install` methods for Linux, because there are many different flavors and configurations."

A block of code is set as follows:

```
<Directory />
  Options FollowSymLinks
  AllowOverride None
  Order deny,allow
  Allow from all
</Directory>
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
<Directory />
  Options FollowSymLinks
  AllowOverride All
  Order deny,allow
  Allow from all
</Directory>
```

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

```
sudo apt-get install php5 php5-dev libapache2-mod-php5 php5-curl php5-
mcrypt
```

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: "Start by opening **Terminal**".

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 through 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.

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/support`, 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 to our website, or added to any list of existing errata, under the Errata section of that title.

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 to address it.

1
Introduction to CouchDB

Welcome to CouchDB and PHP Web Development Beginner's Guide. In this book, we will learn the ins and outs of building a simple but powerful website using CouchDB and PHP. For you to understand why we do certain things in CouchDB, it's first important for you to understand the history of NoSQL databases and learn CouchDB's place in database history.

In this chapter we will:

- Go over a brief history of databases and their place in technology
- Talk about how databases evolved into the concept of NoSQL
- Define NoSQL databases by understanding different classifications of NoSQL databases, the CAP theorem and its avoidance of the ACID model
- Look at the history of CouchDB and its main contributors
- Talk about what makes CouchDB special

Let's start by looking at the evolution of databases and how NoSQL arrived on the scene.

The NoSQL database evolution

In the early 1960s, the term **database** was introduced to the world as a simple layer that would serve as the backbone behind information systems. The simple concept of separating applications from data was new and exciting, and it opened up possibilities for applications to become more robust. At this point, databases existed first as tape-based devices, but soon became more usable as system direct-access storage on disks.

In 1970, Edgar Codd proposed a more efficient way of storing data – the relational model. This model would also use SQL to allow the applications to find the data stored within its tables. This relational model is nearly identical to what we know as traditional relational databases today. While this model was widely accepted, it wasn't until the mid 1980s that there was hardware that could actually make effective use of it. By 1990, hardware finally caught up, and the relational model became the dominant method for storing data.

Just as in any area of technology, competition arose with **Relational Database Management Systems** (**RDBMS**) . Some examples of popular RDMBS systems are Oracle, Microsoft SQL Server, MySQL, and PostgreSQL.

As we moved past the year 2000, applications began to produce incredible amounts of data through more complex applications. Social networks entered the scene. Companies wanted to make sense of the vast amounts of data that were available. This shift brought up some serious concerns about the datastructure, scalability, and availability of data that the relational model didn't seem to handle. With the uncertainty of how to manage this large amount of ever-changing data, the term NoSQL emerged.

The term **NoSQL** isn't short for "no SQL;" it actually stands for "not only SQL". NoSQL databases are a group of persistent solutions, which do not follow the relational model and do not use SQL for querying. On top of that, NoSQL wasn't introduced to replace relational databases. It was introduced to complement relational databases where they fell short.

What makes NoSQL different

Besides the fact that NoSQL databases do not use SQL to query data, there are a few key characteristics of NoSQL databases. In order to understand these characteristics, we'll need to cover a lot of terminology and definitions. It's not important that you memorize or remember everything here, but it's important for you to know exactly what makes up a NoSQL database.

The first thing that makes NoSQL databases different is their data structure. There are a variety of different ways in which NoSQL databases are classified.

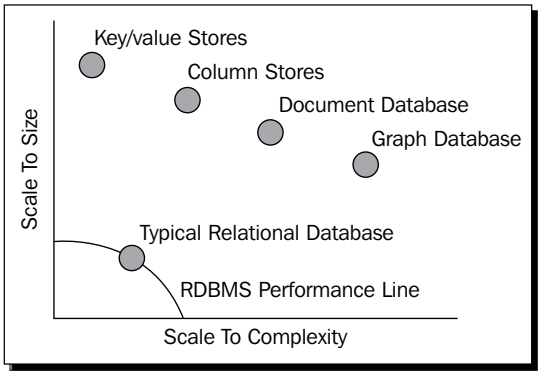
Classification of NoSQL databases

NoSQL databases (for the most part) fit into four main data structures:

- **Key-value stores**: They save data with a unique key and a value. Their simplicity allow them to be incredibly fast and scale to enormous sizes.
- **Column stores**: They are similar to relational databases, but instead of storing records, they store all of the values for a column together in a stream.

- **Document stores**: They save data without it being structured in a schema, with buckets of key-value pairs inside a self-contained object. This datastructure is reminiscent of an associative array in PHP. This is where CouchDB lands on the playing field. We'll go much deeper into this topic in *Chapter 3*, *Getting Started with CouchDB and Futon*.
- **Graph databases**: They store data in a flexible graph model that contains a node for each object. Nodes have properties and relationships to other nodes.

We won't go too deeply into examples of each of these types of databases, but it's important to look at the different options that are out there. By looking at databases at this level, it's relatively easy for us to see (in general) how the data will scale to size and complexity, by looking at the following screenshot:



If you look at this diagram, you'll see that I've placed a **Typical Relational Database** with a crude performance line. This performance line gives you a simple idea of how a database might scale in size and complexity. How is it possible that NoSQL databases perform so much better in regards to high size and complexity of data?

For the most part, NoSQL databases are scalable because they rely on distributed systems and ignore the ACID model. Let's talk through what we gain and what we give up through a distributed system, and then define the ACID model.

When talking about any distributed system (not just storage or databases), there is a concept that defines the limitations of what you can do. This is known as the CAP theorem.

CAP theorem

Eric Brewer introduced the CAP theorem in the year 2000. It states that in any distributed environment, it is impossible for it to provide three guarantees.

- **Consistency**: All the servers in the system will have the same data. So, anyone using the system will get the latest data, regardless of which node they talk to in the distributed system.
- **Availability**: All of the servers will always return data.
- **Partition-tolerance**: The system continues to operate as a whole, even if an individual server fails or cannot be reached.

By looking at these choices, you can tell that it would definitely be ideal to have all three of these things guaranteed, but it's theoretically impossible. In the real world, each NoSQL database picks two of the three options, and usually develops some kind of process to mitigate the impact of the third, unhandled property.

We'll talk about which approach CouchDB takes shortly, but there is still a bit to learn about another concept that NoSQL databases avoid: ACID.

ACID

ACID is a set of properties that apply to database transactions, which are the core of traditional relational databases. While transactions are incredibly powerful, they are also one of the things that make reading and writing quite a bit slower in relational databases.

ACID is made up of four main properties:

- **Atomicity**: This is an all or nothing approach to dealing with data. Everything in the transaction must happen successfully, or none of the changes are committed. This is a key property whenever money or currency is handled in a system, and requires a system of checks and balances.
- **Consistency**: Data will only be accepted if it passes all of the validation in place on the database, such as triggers, data types, and constraints.
- **Isolation**: Transactions will not affect other transactions that are occurring, and other users won't see partial results of a transaction in progress.
- **Durability**: Once the data is saved, it is safe against errors, crashes, and other software malfunctions.

Again, as you read through the definition of ACID, you are probably thinking to yourself, "These are all must haves!" That may be the case, but keep in mind that most NoSQL databases do not fully employ ACID, because it's near impossible to have all of these restrictions and still have blazing fast writes to data.

So what does all of that mean?

I've given you a lot of definitions now, but let's try to wrap it together into a few simple lists. Let's talk through the advantages and disadvantages of NoSQL databases, when to use, and when to avoid NoSQL databases.

Advantages of NoSQL databases

With the introduction of NoSQL databases, there are lot of advantages:

- You can do things that simply weren't possible with the processing and query power of traditional relational databases.
- Your data is scalable and flexible, allowing it to scale to size and complexity faster, right out of the box.
- There are new data models to consider. You don't have to force your data into a relational model if it doesn't make sense.
- Writing data is blazing fast.

As you can see, there are some clear advantages of NoSQL databases, but as I mentioned before, there are still some negatives that we need to consider.

Negatives of NoSQL databases

However, along with the good, there's also some bad:

- There are no common standards; each database does things just a little bit differently
- Querying data does not involve the familiar SQL model to find records
- NoSQL databases are still relatively immature and constantly evolving
- There are new data models to consider; sometimes it can be confusing to make your data fit
- Because a NoSQL database avoids the ACID model, there is no guarantee that all of your data will be successfully written

Some of those negatives may be pretty easy for you to stomach, except for NoSQL's avoidance of the ACID model.

When you should use NoSQL databases

Now that we have a good take on the advantages and disadvantages, let's talk about some great use cases for using NoSQL databases:

- Applications that have a lot of writing
- Applications where the schema and structure of the data might change

- Large amount of unstructured or semi-structured data
- Traditional relational databases feel restricting, and you want to try something new.

That list isn't exclusive, but there are no clear definitions on when you can use NoSQL databases. Really, you can use them for just about every project.

When you should avoid NoSQL databases

There are, however, some pretty clear areas that you should avoid when storing data in NoSQL.

- Anything involving money or transactions. What happens if one record doesn't save correctly because of NoSQL avoidance of the ACID model or the data isn't 100 percent available because of the distributed system?
- Business critical data or line of business applications, where missing one row of data could mean huge problems.
- Heavily-structured data requiring functionality in a relational database.

For all of these use cases, you should really focus on using relational databases that will make sure that your data is safe and sound. Of course, you can always include NoSQL databases where it makes sense.

When choosing a database, it's important to remember that "There is no silver bullet." This phrase is used a lot when talking about technology, and it means that there is no one technology that will solve all of your problems without having any side effects or negative consequences. So choose wisely!

Introduction to CouchDB

For this book and for a variety of my own projects and startups, I chose CouchDB. Let's take a historical look at CouchDB, then quickly touch on its approach to the CAP theorem, and its strengths and weaknesses.

The history of CouchDB

In April 2005, *Damien Katz* posted a blog entry about a new database engine he was working on, later to be called CouchDB, which is an acronym for **Cluster Of Unreliable Commodity Hardware**. Katz, a former Lotus Notes developer at IBM, was attempting to create a fault-tolerant document database in C++, but soon after, shifted to the **Erlang OTP** platform. As months went by, CouchDB started to evolve under the self-funding of Damien Katz, and in February 2008, it was introduced to the Apache Incubator project. Finally, in November 2008, it graduated as a top-level project.

Damien's team, **CouchOne**, merged with the Membase team in 2011 to form a new company called **Couchbase**. This company was formed to merge **CouchDB** and **Membase** into a new product, and increase the documentation and visibility for the product.

In early 2012, Couchbase announced that it would be shifting focus from facilitating CouchDB and moving to create Couchbase Server 2.0. This new database takes a different approach to the database, which meant that it would not be contributing to the CouchDB community anymore. This news was met with some distress in the CouchDB community until Cloudant stepped in.

Cloudant, the chief CouchDB hosting company and creator of BigCouch, a fault tolerant and horizontally scalable clustering frameworking built for CouchDB, announced that they would merge their changes back to CouchDB, and take on the role of continuing development of CouchDB.

In early 2012, at the time of writing, CouchDB's most major release was 1.1.1 in March 31, 2011. But CouchDB 1.2 is looking to be released just around the corner!

Defining CouchDB

According to `http://couchdb.apache.org/`, CouchDB can be defined as:

- A document database server, accessible via a RESTful JSON API
- Ad-hoc and schema-free with a flat address space
- Distributed, featuring robust, incremental replication with bi-directional conflict detection and management
- Query-able and index-able, featuring a table oriented reporting engine that uses JavaScript as a query language.

You might be able to read between the lines, but CouchDB chose availability and partial-tolerance from the CAP theorem, and focuses on eventual consistency using replication.

We could go really deep into what each of these bullet points mean, because it will take the rest of the book until we've touched on them in depth. In each chapter, we'll begin to build on top of our CouchDB knowledge until we have a fully operational application in the wild.

Summary

I hope you enjoyed this chapter and are ready to take a deep dive into really learning the ins and outs of CouchDB. Let's recap everything we learned in this chapter.

- We talked about the history of databases and the emergence of NoSQL databases
- We defined the advantages and disadvantages of using NoSQL
- We looked at the definition and history of CouchDB

That's it for the history lesson. Fire up your computer. In the next chapter, we'll set everything up to develop web applications with CouchDB and PHP, and make sure that it's all set up correctly.