Magento 2 Development Cookbook

Magento 2 is an open source e-commerce platform with innumerable functionalities that gives you the freedom to make on-the-fly decisions, which can help with the challenges of growing an online business. It allows you to customize multiple levels of security permissions and enhance the look and feel of your website, and thus give you a personalized experience of promoting your business.

This book will take the readers through the intermediate to advanced features of Magento 2 and teach them how to build beautiful, highly-customized online stores.

What this book will do for you...

- Install a Magento 2 shop with sample data
- Upgrade the data in a Magento 1 shop to a Magento 2 shop
- Manage the look and feel of the shop with custom themes
- Extend the shop with custom functionality, such as forms, grids, and more
- Accelerate your store with some performance tools
- Build and structure your own shipping module
- Test your shop with automated tests and manage your product display

Inside the Cookbook...

- A straightforward and easy-to-follow format
- A selection of the most important tasks and problems
- Carefully organized instructions to solve problems efficiently
- Clear explanations of what you did
- Solutions that can be applied to solve real-world problems

Magento 2 Development Cookbook

Over 60 recipes that will tailor and customize your experience with Magento 2

Bart Delvaux
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Upgrading from Magento 1'
- A synopsis of the book’s content
- More information on *Magento 2 Development Cookbook*
About the Author

**Bart Delvaux** is an experienced web developer with several years of experience in the PHP world. He has worked with the most important frameworks in PHP, such as Drupal and Zend Framework, but Magento is his specialization.

Bart has obtained all the Magento developer certifications: Front End Developer, Developer, as well as Developer Plus. He currently works for ISAAC Software Solutions, a company that specializes in software solutions such as web shops, apps, system integrations, and more.

Bart finished a large variety of Magento projects in his Magento career that started in 2010 with the principle "quality above quantity". Having gone from handling a basic shop to shipping modules and large, complex Magento stores, Magento holds no secrets from him.

Bart has also worked on *Magento 1.8 Development Cookbook, Packt Publishing*. Now that Magento 2 is out, it is time for the next one!
Magento is one of the most popular e-commerce platforms on the market. It contains a lot of e-commerce functionality, it is stable, and it is free. This means that a lot of people choose Magento for their online business.

The first stable version of Magento was released in 2008. The later releases were based on the first version of Magento. Technology changes quickly and Magento needed a big update—a big release Magento 2 is now ready.

Developing in Magento is not as easy as you would expect. Even if you have knowledge of Magento 1, a good guide with practical examples that shows you the best practice is a must have, and this is exactly what this book will do.

With Magento 2 Development Cookbook, we will cover the most important topics that will help you become a good Magento 2 developer. We will start with the basics and we will end with the more advanced topics.

This book is divided into several recipes, which show you which steps to take to complete a specific action. In each recipe, we have a section that explains how everything works.

We will start this book with the creation of a good development environment. For a good development environment, we need the right tools. We will install Magento and we will discuss how we can migrate data from a Magento 1 to a Magento 2 shop. Next, we will see some functional stuff. You will learn how the catalog system works, which product types are available, and a lot more.

After this, you will learn how we can create a Magento theme to change the look and feel of the Magento shop. But the main focus of this book will be the development part. We will create a custom module that we will extend with a lot of common features that are used in Magento projects, such as extra controller pages, database integrations, custom shipping methods, and extra backend interfaces.

At the end of this book, we will see how we can improve the performance of a Magento shop. Finally, we will see some debugging techniques, such as Xdebug and creating unit tests using the Magento test framework.
What this book covers

Chapter 1, Upgrading from Magento 1, provides an introduction to how you can install and migrate the data from a Magento 1 to a Magento 2 shop. We will also prepare our development environment in this chapter.

Chapter 2, Working with Products, gives you a more functional information about the possibilities of displaying products in your Magento shop.

Chapter 3, Theming, explains how you can customize the look and feel of your webshop using a custom Magento theme.

Chapter 4, Creating a Module, describes how to create a basic Magento module; how to extend that module with custom configurations, such as a custom page, translations, and blocks; and how to change behavior of standard Magento classes.

Chapter 5, Databases and Modules, demonstrates how you can extend a Magento module with database interactions, such as install and upgrade scripts, a custom entity that represents a database table.

Chapter 6, Magento Backend, shows you how to integrate a Magento module with the backend, such as adding configuration pages, creating overview pages, and extending the admin menu.

Chapter 7, Event Handlers and Cronjobs, describes how the event-driven architecture is implemented in Magento and how to integrate this in your module. Later in this chapter, you will learn how to create cronjobs and how to test them.

Chapter 8, Creating a Shipping Module, shows you how to create a module with the configurations that are required for a new shipping method.

Chapter 9, Creating a Product Slider Widget, will cover how to create a module with a custom widget, how to build the backend interface, and how to provide a good UI in the frontend of that widget.

Chapter 10, Performance Optimization, describes how to benchmark a site to explore the limits and how to improve the performance using different techniques such as Redis and Memcached.

Chapter 11, Debugging and Unit Testing, shows you how to use the PHP debugger Xdebug and how we can create automated tests using the Magento 2 testing framework.
1

Upgrading from Magento 1

In this chapter, we will cover:

- Creating a Magento 1 website with sample data
- Creating a Magento 2 website
- Preparing an upgrade from Magento 1
- Upgrading the database
- Using an IDE
- Writing clean code with PHP MD and PHP CS

Introduction

Magento is one of the most complete e-commerce platforms on the open source market. With a default Magento installation, all the common e-commerce features, such as catalog navigation, promotion rules, tax settings, online payments, and so on are available.

The first version of Magento was released in 2008 after one year of development. Magento was initially designed as an e-commerce system that could be used for a wide range of uses. In later years, Magento became very popular as an out-of-the-box e-commerce system and a lot of minor versions of the 1.x series have been released in the last few years.
To be future proof, Magento started the development of a major upgrade of the system, also known as Magento 2. Magento 2 is a big improvement on every part of Magento. Every aspect is analyzed and rewritten with up-to-date technologies to be ready for the future. Everything, including the developer experience, maintainability, performance, and technologies will be improved.

In this chapter, we will upgrade the data of a Magento 1 installation to a Magento 2 installation. We will also prepare some tools that we can use in the following chapters of this book.

Creating a Magento 1 website with sample data

To start a Magento 2 upgrade, we need a Magento 1 webshop with some data. In this recipe, we will install the latest Magento version, 1.9, with the sample data for the new responsive theme.

Getting ready

To install a Magento 1 website, we need the following stuff:

- A web server (Linux, Apache2, PHP, or MySQL)
- The Magento 1.9 codebase
- The Magento 1.9 sample data

The Magento 1.9 codebase and sample data can be downloaded from the Magento site at http://www.magentocommerce.com/download.

The following stuff is recommended for the installation:

- Command-line access
- A virtual host (domain name) that is going to be your web root

We recommend that you use a test server that is on your development machine. If you use a Linux or a Mac operating system, you can install the webserver on your local machine. If you have a Windows machine, you can use a virtual Linux server for your development.
How to do it...

1. Extract the Magento code archive in your webroot (the directory of the virtualhost). An `ls -la` command should give you the following output:

```
api.php
app
cron.php
cron.sh
downloader
ersors
favicon.ico
get.php
includes
index.php
index.php.sample
install.php
js
lib
LICENSE_AFL.txt
LICENSE.html
LICENSE.txt
mage
media
php.ini.sample
pkginfo
RELEASE_NOTES.txt
shell
skin
var
```

2. Extract the sample data archive to a different folder from the webroot. Copy the contents of the `media` and `skin` folders to the `media` and `skin` folders in your webroot. We can do this by using the following `cp` command:

```
cp -R <path_to_sampledata_folder>/media/* <path_to_magento_folder>/media/
cp -R <path_to_sampledata_folder/skin/* <path_to_magento_folder>/skin/
```
3. Create a database for the Magento 1 installation and name it `magento1`. We can do this by running the following commands:

   ```bash
   mysql -u <username> -p
   create database magento1;
   exit;
   ```

4. Import the SQL file that is in the sample data directory. This file contains a database that we will import into the `magento1` database. We can do this by running the following command:

   ```bash
   mysql -u <username> -p magento1 < "path_to_sample_data.sql"
   ```

   To avoid permission problems, ensure that all files and folders have the right permissions. For security reasons, it is recommended that all files have just enough permissions so that only the right users can access the right files. When you give all the rights (777), you don’t have permission problems because each user can read, write and, execute each file of your application. More information about file permissions can be found at [http://devdocs.magento.com/guides/m1x/install/installer-privileges_after.html](http://devdocs.magento.com/guides/m1x/install/installer-privileges_after.html).

5. When the files are in the right place and the database is imported, we can run the Magento installer. Open your browser and go to the domain that is configured for your website. You should see the installer as in the following screenshot:
6. Continue with the installation process by accepting the terms and conditions.
7. On the next screen, choose the correct language, locale, and currency for your store.
8. On the configuration page, fill in the form with the right data:
   - **Database Type**: MySQL.
   - **Host**: Enter the hostname or IP address of your database server (localhost if it is on the same machine).
   - **Database name**: Enter `magento1` in this field (or another name if you have a different name for your database).
   - **User name**: Enter your database username.
   - **User password**: Enter your database password.
   - **Tables prefix**: Leave this field empty (the string in this field will be used to prefix all tables of your database).
   - **Base URL**: Enter the URL of your website in this field.
   - **Admin path**: Enter `admin` in this field. This will be the path of the backend.
   - **Enable charts**: For development, it is recommended that this be unchecked.
   - **Skip Base URL Validation Before the Next Step**: When checked, the wizard will check for a valid URL when processing this form.
   - **Use Web Server (Apache) rewrites**: Check this when the apache module `mod_rewrite` is enabled.
   - **Use Secure URL's (SSL)**: This checkbox must be unchecked if you don't use HTTPS.

9. Submit this form and we will be forwarded to the next step. In this step, you can configure the administrator account. Fill in the right data and remember the username and password because this is required to manage the store. Leave the **encryption key** field empty.
Upgrading from Magento 1

10. After submitting this form, the installation is complete. Optionally, you can submit the Magento survey. At the bottom of the page, there are buttons to navigate to the frontend and backend. When going to the frontend, you can see a demo shop with sample data as in the following screenshot:
11. The layout is responsive. When scaling your browser to a smaller width, the website will switch to the mobile layout like in the following screenshot:

How it works...

We have just created a fully functional Magento 1 store. The webshop is fully configured and filled with data about products, customers, and orders, just the data we need to migrate to Magento 2 (in the upcoming recipes).

When installing a new shop, you have to follow the installer. This interface creates a configuration file app/etc/local.xml. If the file doesn't exist, Magento will launch the installer wizard. If the file is there, Magento will run the shop.

With a valid local.xml file, it is technically possible to install a new Magento shop, but this is not recommended because some settings such as a backend user, time zone, and currency are not set. These are actions that you have to do manually when choosing for this method.
Creating a Magento 2 website

In the previous recipe, we created a Magento 1 website with sample data that we will use for an upgrade. In this recipe, we will do the same, but we will create a Magento 2 website with the sample data for Magento 2.

Getting ready

To install Magento 2, we need the newest tools to run that application. Make sure your webserver has the following stuff installed:

- PHP 5.5 or higher
- MySQL 5.6 or higher
- Apache 2.2 or higher
- Command line access
- Composer

We can install Magento 2 in different ways. In this recipe, we will install Magento 2 using Composer. The advantage of this is that we can use GIT to add version control to our custom development.

How to do it...

1. We will install Magento 2 with Composer. For this, we need authentication keys. With an account on the magento.com site, go to Developers | Secure keys in the My Account section. On this page, you can generate public and private keys that will be your username and password in the next step.

2. To install Magento 2 with composer, we have to run the following command:

```
composer create-project --repository-url=https://repo.magento.com magento/project-community-edition <installation_dir>
```
3. You will be prompted for a username and password. The username is the public key and the password is the private key that we generated in the previous step. When the command has run, the installation directory will have the following structure:

```
app
bin
CHANGELOG.md
composer.json
composer.lock
CONTRIBUTING.md
CONTRIBUTOR_LICENSE AGREEMENT.html
COPYING.txt
dev
.gitignore
Gruntfile.js
.htaccess
.htaccess.sample
index.php
lib
LICENSE_AFL.txt
LICENSE.txt
nginx.conf.sample
package.json
.php_cs
php.ini.sample
pub
README.md
setup
.travis.yml
update
var
vendor
```

Check that the user and group of these files are the same as your Apache user. One recommendation is to execute all the commands as your apache user.
4. We have installed the codebase with composer. Now we can run the installation wizard. Open your browser and enter the URL of your site. You should see the following welcome screen:

![Magento Welcome Screen](image)

5. Hit the **Agree and Setup Magento** button and start the environment check.

6. Click on **Next** and enter your database information as follows:
   - **Database Server Host**: The hostname or IP address of the database server
   - **Database Server Username**: The username of the database account
   - **Database Server Password**: The password for the account
   - **Database Name**: The name of the database
   - **Table Prefix**: Optionally, you can give a prefix for each table

7. Go to the next step and check if the right information is filled for the URL part. In the advanced section, you can optionally configure HTTPS, apache rewrites, and your encryption key. For our test environment, we can leave these settings as they are configured.

   ![Tips](image)
   
   Make sure that the `mod_rewrite` option is enabled for the apache server. When not enabled, the URL rewrites will not work correctly.
8. In the next step, you can configure your time zone, currency, and default language.

9. In the last step, you can configure your administration account. After clicking on the Next button, you are ready to install. Click on the Install Now button and the installer will start. This will take some time because the installer will add the sample data during the installation. You can open the Console Log to see what is currently happening.

10. When the installer is ready, you will see the following success message:

```
Success
Please keep this information for your records:

Magento Admin Info:
Username: admin
E-Mail: user@example.com
Password: ******
Your Store Address: http://magento2.local/
Magento Admin Address: http://magento2.local/admin/
Encryption Key: df71fd2e7e3816f61dd37162ea667a20

Database info:
Database Name: magento2
Username: root
Password: ******

For security, remove write permissions from these directories: /var/www/magento2/app/etc

Launch Magento Admin
```

11. Run the following commands in your Magento installation directory to configure the sample data:

```
php bin/magento sampledata:deploy
composer update
php bin/magento setup:upgrade
```
12. The preceding commands will download and install the sample data packages. Because they contain a lot of images, this could take some time. The `setup:upgrade` command will install the sample data, and this also takes some time.

13. The installation of the webshop is now complete. You now have an up-and-running Magento 2 webshop. When you navigate to the category Gear | Bags, you should see something like in the following screenshot:

![Magento 2 screenshot](image)

How it works...

We have now installed a Magento 2 website. Like we did in the previous recipe for Magento 1.9, we downloaded the codebase (using composer), created a database, and installed Magento.
For Magento 2, we used composer to download the codebase. Composer is a PHP dependency manager. All the dependencies are set in the composer.json file. For this recipe, there are the Magento and the magento-sample-data dependencies in the composer.json file. There is also a composer.lock file generated. In that file, the versions of the installed dependencies are stored.

When working with GIT, we only have to commit the composer.json, composer.lock, and .gitignore files for a working Magento 2 project. When another person does a Git clone of the repository and runs the composer's install command, Magento 2 will be installed with the version that is in the composer.lock file.

The sample data for Magento 2 is now a script that will be executed after the installation of Magento. That script will add products, customers, orders, CMS data, and more configurations to populate the shop.

The shop is installed and the configuration settings (database, encryption key, and so on) are now stored in app/etc/env.php instead of in the app/etc/local.xml file in Magento 1.

There's more...

When installing Magento 2, here are some common issues that can occur and their fixes:

- When you don't see CSS in your browser, you have to check the following things:
  - Make sure the pub/ folder is writable
  - Run the command php bin/magento setup:static-content:deploy to generate the static content
- You forget to install the sample data:
  - You can install the sample data after the installation of Magento with the command php bin/magento sampledata:deploy
- The installation is not responding anymore:
  - This could be caused by an Apache timeout. If this occurs, you can maybe try the command-line installation. This works as follows:

To run the Magento installer from the command line, we can use the command php bin/magento setup:install. We have to add the following required parameters to the command to configure the installation:

- base-url: The base URL, for example http://magento2.local/
- db-host: The database host or IP address
Preparing an upgrade from Magento 1

The differences between Magento 1 and Magento 2 are huge. The code has a whole new structure with a lot of improvements but there is one big disadvantage. What do I do if I want to upgrade my Magento 1 shop to a Magento 2 shop?

Magento created an upgrade tool that migrates the data from a Magento 1 database to the right structure for a Magento 2 database.

The custom modules in your Magento 1 shop will not work in Magento 2. It is possible that some of your modules will have a Magento 2 version, and depending on the module, the module author will have a migration tool to migrate the data that is in the module.

Getting ready

Before we get started, make sure you have an empty (without sample data) Magento 2 installation with the same version as the Migration tool that is available at:

1. In your Magento 2 version (with the same version as the migration tool), run the following commands:
   
   ```bash
   composer config repositories.data-migration-tool git https://github.com/magento/data-migration-tool-ce
   composer require magento/data-migration-tool:2.0.0
   ```

2. Install Magento 2 with an empty database by running the installer. Make sure you configure it with the right time zone and currencies.

3. When these steps are done, you can test the tool by running the following command:
   
   ```bash
   php bin/magento migrate:data --help
   ```

4. The next thing is creating the configuration files. Examples of the configuration files are in `vendor/magento/data-migration-tool/etc/<version>`. We can create a copy of this folder where we can set our custom configuration values. For a Magento 1.9 installation, we have to run the following `cp` command:
   
   ```bash
   cp -R vendor/magento/data-migration-tool/etc/ce-to-ce/1.9.1.0/ vendor/magento/data-migration-tool/etc/ce-to-ce/packt-migration
   ```

5. Open the `vendor/magento/data-migration-tool/etc/ce-to-ce/packt-migration/config.xml.dist` file and search for the `source/database` and `destination/database` tags. Change the values of these database settings to your database settings like in the following code:
   
   ```xml
   <source>
       <database host="localhost" name="magento1" user="root"/>
   </source>
   <destination>
       <database host="localhost" name="magento2_migration" user="root"/>
   </destination>
   ```

6. Rename that file to `config.xml` with the following command:
   
   ```bash
   mv vendor/magento/data-migration-tool/etc/ce-to-ce/packt-migration/config.xml.dist vendor/magento/data-migration-tool/etc/ce-to-ce/packt-migration/config.xml
   ```
How it works...

By adding a composer dependency, we installed the data migration tool for Magento 2 in the codebase. This migration tool is a Magento console command that will handle the migration steps from a Magento 1 shop.

In the \texttt{etc} folder of the migration module, there is a sample configuration of an empty Magento 1.9 shop.

If you want to migrate an existing Magento 1 shop, you have to customize these configuration files so it matches your preferred state.

In the next recipe, we will learn how we can use the script to start the migration.

Upgrading the database

In the previous recipe, we configured the database migration tool. In this recipe, we will run the migration tool so that we can migrate parts from a Magento 1 shop to a Magento 2 shop.

Getting ready

You need a Magento 1 website and a Magento 2 website. The Magento 2 website needs to have the database migration tool installed and configured as described in the previous recipe.

In this recipe, we will do a migration from a clean Magento 1 site, to a Magento 2 site without sample data.

We did a migration from a clean Magento 1 database with some test products. Make sure you have a cleanly installed Magento 1 shop with some test data (products, orders, and so on) in it.

How to do it...

1. First we need to make sure that the database settings are correct in the \texttt{vendor/magento/data-migration-tool/etc/ce-to-ce/packt-migration/config.xml} file. Open that file and check that the database credentials are correct.

   We created this file in the previous recipe:

   ```xml
   <source version="1.9.1">
   <database host="localhost" name="magentol_migration"
   user="root"/>
   </source>
   ```
<destination version="2.0.0.0">
<database host="localhost" name="magento2_migration" user="root"/>
</destination>

If you have a database prefix in your source or destination database, you can optionally configure source_prefix and dest_prefix in the <options> section of the same configuration file.

Test the migration first with a clean Magento 1.9 database. The mapping that we will use in this recipe is for a clean Magento 1.9 installation. With an existing shop, you will have custom attributes and entities that need more configuration to make the migration work.

2. If these settings are correct, we can run the upgrade tool. Run the following command:

   php bin/magento migrate:data --help

3. This gives us the following output:

4. To start or test a migration, we have to run the following command:

   php bin/magento migrate:data vendor/magento/data-migration-tool/etc/ce-to-ce/packt-migration/config.xml
5. The migration will start and will give the following output:

```
[2015-10-25 13:51:49][INFO][mode: data][stage: integrity check][step: EAV Step]: started
[2015-10-25 13:51:49][INFO][mode: data][stage: integrity check][step: Customer Attributes Step]: started
[2015-10-25 13:51:49][INFO][mode: data][stage: integrity check][step: Map Step]: started
[2015-10-25 13:51:49][INFO][mode: data][stage: integrity check][step: Log Step]: started
[2015-10-25 13:51:49][INFO][mode: data][stage: integrity check][step: Ratings Step]: started
[2015-10-25 13:51:49][INFO][mode: data][stage: integrity check][step: ConfigurablePrices step]: started
[2015-10-25 13:51:49][INFO][mode: data][stage: setup triggers][step: Stage]: started
[2015-10-25 13:51:49][INFO][mode: data][stage: data migration][step: EAV Step]: started
100% [==================================] Remaining Time: 1 sec
[2015-10-25 13:51:49][INFO][mode: data][stage: volume check][step: EAV Step]: started
100% [==================================] Remaining Time: 1 sec
[2015-10-25 13:51:49][INFO][mode: data][stage: data migration][step: Customer Attributes Step]: started
100% [==================================] Remaining Time: 1 sec
[2015-10-25 13:51:49][INFO][mode: data][stage: volume check][step: Customer Attributes Step]: started
100% [==================================] Remaining Time: 1 sec
[2015-10-25 13:51:49][INFO][mode: data][stage: data migration][step: Map Step]: started
100% [==================================] Remaining Time: 1 sec
[2015-10-25 13:51:49][INFO][mode: data][stage: volume check][step: Map Step]: started
100% [==================================] Remaining Time: 1 sec
[2015-10-25 13:51:50][INFO][mode: data][stage: data migration][step: Url Rewrite Step]: started
100% [==================================] Remaining Time: 1 sec
[2015-10-25 13:51:50][INFO][mode: data][stage: volume check][step: Url Rewrite Step]: started
100% [==================================] Remaining Time: 1 sec
[2015-10-25 13:51:50][INFO][mode: data][stage: data migration][step: Log Step]: started
100% [==================================] Remaining Time: 1 sec
[2015-10-25 13:51:50][INFO][mode: data][stage: volume check][step: Log Step]: started
100% [==================================] Remaining Time: 1 sec
[2015-10-25 13:51:50][INFO][mode: data][stage: data migration][step: Ratings Step]: started
100% [==================================] Remaining Time: 1 sec
[2015-10-25 13:51:50][INFO][mode: data][stage: volume check][step: Ratings Step]: started
100% [==================================] Remaining Time: 1 sec
[2015-10-25 13:51:50][INFO][mode: data][stage: data migration][step: ConfigurablePrices step]: started
100% [==================================] Remaining Time: 1 sec
[2015-10-25 13:51:50][INFO][mode: data][stage: volume check][step: ConfigurablePrices step]: started
100% [==================================] Remaining Time: 1 sec
[2015-10-25 13:51:50][INFO][mode: data][stage: data migration][step: OrderGrids Step]: started
100% [==================================] Remaining Time: 1 sec
[2015-10-25 13:51:50][INFO][mode: data][stage: volume check][step: OrderGrids Step]: started
100% [==================================] Remaining Time: 1 sec
[2015-10-25 13:51:50][INFO][mode: data][stage: data migration][step: Migration completed]
```

6. The migration is now complete. If you check your database for the Magento 2 website, you will see that the data (products, categories, and so on) is migrated from Magento 1.

If you want to rerun the migration tool, you have to remove the `var/migration-tool-progress.lock` file.

7. We can also migrate the settings from the Magento 1 website. To do this, you have to replace the `data` parameter in the command using `settings`.

```
```
8. To check if the upgrade works, you have to look at the data of the Magento 2 installation. We can check the following things in the backend:
   - The orders ([Sales | Orders])
   - The products ([Products | Catalog])
   - The customers ([Customers | All Customers])

9. You can also check in the database if you look at the following tables:
   - `sales_order`
   - `customer_entity`
   - `catalog_product_entity`
   - `url_rewrite`

**How it works...**

When the migration tool starts, it starts checking all the configurations that are in the configuration files of the migration tool. If there are more things available in the Magento 1 database than the things that are configured, the migration tool will give a notification and stop the migration.

It's likely that every existing Magento 1 shop works with custom attributes, custom entities, and so on. Each entity, attribute, and so on needs to be declared in the configuration files.

The most time-consuming part of a migration is to create a good configuration file so that the migration tool won't fail on missing stuff. It is on you to decide what to ignore and what to migrate. If the configuration files are valid, the migration will start and the data will come into the Magento 2 database. The same principle applies when migrating the settings, but you have to think about whether you want it.

With the migration tool, it is only possible to migrate data and settings. The code of Magento 1 modules will not work in Magento 2. So for your modules, you need to see if there is a Magento 2 version/alternative available.

**There's more...**

In this recipe, we did a migration of a clean Magento 1 installation to a clean Magento 2 installation. However almost every running Magento 1 shop is not clean. It contains custom attributes, custom modules, and a custom configuration.
Upgrading from Magento 1

When migrating such a shop to a new shop, the migration is a bit more complex. The first question is: What needs to be migrated? With the tool, you can migrate every entity, from products, customers, and orders to reviews, settings, and more.

If you want to skip data that must be migrated, you can use the map.xml file. If you open the file vendor/magento/data-migration-tool/etc/ce-to-ce/packt-migration/map.xml, you see that a lot of entities are ignored in the map/source/document_rules tag.

If you want to change something in the map.xml file, you have to make sure that the right map.xml file is loaded. This file is configured in the config.xml file (where you did your database configuration). In that file, you have to look for the XML tag config/options/map_file.

If you have an error such as Source documents not mapped, you have to add the configuration for these entities in the map/source/document_rules tag of the map.xml file. If the error is something like Destination documents not mapped, you have to add configuration in the map/destination/document tag of the map.xml file.

To solve errors such as Source fields not mapped you have to add configuration in the map-eav.xml file.

See also

Migrating configuration files is the most time consuming part of a data migration. If you want more information on the migration tool, you can have a look at the Magento Migration Whitepaper, available at http://magento.com/resources/magento-2-migration-whitepaper.

Using an IDE

Writing good code starts with a good development environment. An Integrated Development Environment (IDE) is the main part of a good development environment. NetBeans is a free and open source PHP editor that can be used for Magento development. In this recipe, we will set up a Magento 2 project in NetBeans.

Getting ready

Install the latest version of NetBeans IDE on your computer. You can download it from the following URL:

https://netbeans.org/downloads/

For PHP development, you need to download the HTML5 & PHP bundle.
How to do it...

1. To create a new project, open NetBeans and navigate to File | New Project.
2. A window like the one in the following screenshot will appear on your screen. Click on PHP and PHP Application with Existing Sources.

3. Click on Next and configure the following settings:
   - **Source Folder**: This field is set to the location of your Magento code (like /var/www/html/magento2/)
   - **Project Name**: The NetBeans project name is entered in this field
   - **PHP Version**: This field is set to PHP 5.5
   - **Default Encoding**: This field is set to UTF-8
4. In the next screenshot, you can see how everything is configured:

When you are working with a version control system like GIT, it is recommended that you check the checkbox. Put NetBeans metadata into a separate directory. If not checked, a .nbproject folder is created in your Magento root, and you don't want to have that folder in your version control system. Another possibility is to add the .nbproject folder in the .gitignore file.

5. Click on Next and configure the final settings:

- **Run as**: If you are developing on a local PC, choose Local Web Server
- **Project URL**: The URL of your website
- **Index file**: Set this to index.php
Chapter 1

The settings are shown in the following screenshot:

6. Click on the Finish button and your NetBeans project is ready. You can now start developing.

There's more...

In this recipe, we used the free code editor NetBeans, but there are also some other good alternatives on the market, such as:

- PHPStorm
- Eclipse with PDT (PHP Development Tools)
- Zend Studio
Writing clean code with PHP MD and PHP CS

Maintaining clean code is much more efficient than maintaining spaghetti code, but writing clean code is not as easy as it sounds. These days there are some tools that help you with writing clean code, such as PHPMD and PHP_CodeSniffer.

PHPMD stands for PHP Mess Detector; this tool will check your code on complexity and how variables are used and will detect some possible bugs. It goes a bit further than the syntax check in your IDE.

PHP_CodeSniffer or PHPCS checks your code on coding standards such as PSR-1 and PSR-2.

Getting ready

We will install PHPMD and PHP_CodeSniffer in our development environment. Make sure you have command-line access to your development environment.

How to do it...

1. Before installing PHPMD and PHP_CodeSniffer, we have to make sure that PHP is installed on our development machine. Especially if you are developing on a remote server, it could be that PHP is not installed.

2. Download and install PHPMD. Depending on your OS, the protocol could be different. You can find instructions at: http://phpmd.org/download/index.html

3. Download and install PHP_CodeSniffer. You can find the installation instructions at: https://github.com/squizlabs/PHP_CodeSniffer

4. Everything is installed, so we can run a test for PHPMD. For the PHPMD command, these are the required options:
   - Filename or directory
   - The format of the report
   - The ruleset

5. Let's run the following command to check the file on clean code and output text:
   ```
   phpmd app/code/Magento/Cms/Model/Observer.php text cleancode
   ```
6. It gives us the following output:

```
/var/www/magento2/app/code/Magento/Cms/Model/Observer.php:7
0  Avoid using static access to class
   \Magento\Cms\Helper\Page' in method 'noCookies'
/var/www/magento2/app/code/Magento/Cms/Model/Observer.php:7
1  Avoid using static access to class
   \Magento\Store\Model\ScopeInterface' in method
   'noCookies'.
/var/www/magento2/app/code/Magento/Cms/Model/Observer.php:7
7  The method noCookies uses an else expression. Else is
never necessary and you can simplify the code to work
without else.
```

7. There are a lot of errors, but Magento 2 defines its own rules for PHPMD. To run a test
with these rules, we can run the following command:

```
phpmd app/code/Magento/Cms/Model/Observer.php text
dev/tests/static/ testsuite/Magento/Test/Php/_files/phpmd/
ruleset.xml
```

8. This command gives empty output, which means that this file is valid.

9. We will now run a test on the same file with PHP_CodeSniffer. With the next
command, we will run a test on the same file we used for PHPMD.

```
phpcs app/code/Magento/Cms/Model/Observer.php
```

10. This test gives us the following output:

```
FILE: /var/www/magento2/app/code/Magento/Cms/Model/Observer.php
----------------------------------------------------------------------
FOUND 22 ERRORS AND 2 WARNINGS AFFECTING 12 LINES
----------------------------------------------------------------------

5 | WARNING | [ ] PHP version not specified
5 | ERROR   | [ ] Missing @category tag in file comment
5 | ERROR   | [ ] Missing @package tag in file comment
5 | ERROR   | [ ] Missing @author tag in file comment
5 | ERROR   | [ ] Missing @license tag in file comment
5 | ERROR   | [ ] Missing @link tag in file comment
10 | ERROR   | [ ] Missing @category tag in class comment
10 | ERROR   | [ ] Missing @package tag in class comment
10 | ERROR   | [ ] Missing @author tag in class comment
```
Upgrading from Magento 1

10  | ERROR  | [ ] Missing @license tag in class comment
10  | ERROR  | [ ] Missing @link tag in class comment
18  | ERROR  | [ ] Protected member variable "_cmsPage" must not be prefixed with an underscore
25  | ERROR  | [ ] Protected member variable "_scopeConfig" must not be prefixed with an underscore
27  | ERROR  | [ ] Missing short description in doc comment
28  | ERROR  | [ ] Missing parameter comment
28  | ERROR  | [x] Expected 27 spaces after parameter type; 1 found
29  | ERROR  | [ ] Missing parameter comment
42  | ERROR  | [ ] Missing parameter comment
42  | ERROR  | [x] Tag value indented incorrectly; expected 2 spaces but found 1
43  | ERROR  | [ ] Tag cannot be grouped with parameter tags in a doc comment
62  | ERROR  | [ ] Missing parameter comment
62  | ERROR  | [x] Tag value indented incorrectly; expected 2 spaces but found 1
63  | ERROR  | [ ] Tag cannot be grouped with parameter tags in a doc comment
78  | WARNING| [ ] Line exceeds 85 characters; contains 94 characters

------------------------------------------------------------------
PHPCBF CAN FIX THE 3 MARKED SNIFF VIOLATIONS AUTOMATICALLY
------------------------------------------------------------------

Time: 28ms; Memory: 3.75Mb

If the phpmd command is not working, you have to find the path to the phpmd executable and run it from there.
11. When we specify the ruleset of Magento 2, we have the following command:

```
phpcs app/code/Magento/Cms/Model/Observer.php --standard=
dev/tests/static/testsuite/Magento/Test/Php/_files/phpcs/
rulest.xml
```

12. This command gives us the following output:

```
FILE: /var/www/magento2/app/code/Magento/Cms/Model/Observer.php

----
FOUND 5 ERRORS AFFECTING 5 LINES
----
18 | ERROR | Missing variable doc comment
25 | ERROR | Missing variable doc comment
31 | ERROR | Missing function doc comment
45 | ERROR | Missing function doc comment
65 | ERROR | Missing function doc comment
----
```

Time: 35ms; Memory: 3.75Mb

**How it works...**

PHPMD and PHP_CodeSniffer are tools that checks PHP files on code style. These tools have defined their default rulesets for common usage.

Magento has created its own rulesets; they can be found in the directory `dev/tests/static/testsuite/Magento/Test/Php/_files/phpcs/ruleset.xml`.

When developing custom code in Magento 2, it is recommended that you configure these rulesets when working with PHPMD and PHP_CodeSniffer.

**There's more...**

Some IDE's have built-in support for PHPMD and PHP_CodeSniffer. These plugins will run a test when saving a file.

In NetBeans, you have the `phpcsmd` plugin that allows you to integrate these tools in your IDE. For more details visit the following URL:

```
http://plugins.netbeans.org/plugin/40282/phpmd-php-codesniffer-plugin
```
In **PHPStorm**, there is built-in support for PHPMD and PHP_CodeSniffer. If it is configured, there is a color indicator that says how clean your code is. More information can be found at [https://www.jetbrains.com/phpstorm/help/using-php-mess-detector.html](https://www.jetbrains.com/phpstorm/help/using-php-mess-detector.html).

💡 When configuring PHPMD and PHP_CodeSniffer in an IDE, these tools and PHP need to be installed on the machine on which the IDE is running.
Where to buy this book
You can buy Magento 2 Development Cookbook from the Packt Publishing website. Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.
Click here for ordering and shipping details.