Instant PhpStorm Starter

Learn professional PHP development with PhpStorm

Włodzimierz Gajda
Instant PhpStorm Starter

Learn professional PHP development with PhpStorm

Włodzimierz Gajda

PACKT PUBLISHING
BIRMINGHAM - MUMBAI
Credits

Author
Wlodzimierz Gajda

Reviewer
Timon Royer

Acquisition Editor
Usha Iyer

Commissioning Editor
Priyanka Shah

Technical Editor
Jalasha D'costa

Project Coordinator
Joel Goveya

Proofreader
Lydia May Morris

Production Coordinator
Prachali Bhiwandkar

Cover Work
Prachali Bhiwandkar

Cover Image
Sheetal Aute
Włodzimierz Gajda is a very experienced lecturer and highly passionate teacher. In the last twenty years, he has conducted numerous classes and courses on very diverse subjects—from programming in C language and TCP/IP networking to building LEGO robots and developing web applications with PHP.

His teaching and learning activities for the last ten years focus around web development and web design. Currently he works in the Institute of Mathematics and Computer Science at The Catholic University of Lublin, where he teaches web development, programming in PHP, and web design.

Włodzimierz occasionally contributes to open source projects on GitHub (https://github.com/gajdaw) and regularly writes books for Helion—the largest IT publishing house in Poland. He has published 12 books including:

- **GIMP. Praktyczne projekty (GIMP. Practical projects)**
- **HTML, XHTML i CSS. Praktyczne projekty (HTML, XHTML and CSS. Practical projects)**
- **PHP. Praktyczne projekty (PHP. Practical projects)**
- **jQuery. Poradnik programisty (jQuery for programmers)**
- **Symfony w przykładach (Symfony by examples)**
- **Zend Framework od podstaw (Zend Framework from scratch)**
- **Git. Rozproszony system kontroli wersji (Git. Distributed Version Control System)**

Tatra mountains trekking, playing blues (http://www.youtube.com/user/gajdaw), and reading horrors are his preferred leisure activities.

About the Reviewer

Timon Royer is a graphic designer with more than 13 years of experience in the industry. His company specializes in Drupal Frontend Development and UX Design. Timon is married and lives in Germany.
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.
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](http://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

**Instant PhpStorm Starter**
- So, what is PhpStorm? 3

**Installation**
- Step 1 – What do I need? 5
- Step 2 – Downloading PhpStorm 5
- Step 3 – Proceeding with the installation 6
- Step 4 – Entering the license key 6
- Step 5 – Licenses 6
- And that’s it!! 7

**Quick start – your first PHP application**
- Step 1 – Unzipping and opening files 8
- Step 2 – Enabling Symfony2 operations 10
- Step 3 – Generating HelloBundle 11
- Step 4 – Customizing the action 13
- Step 5 – Opening your browser 14

**The PhpStorm interface**
- Tool windows 18
- Favorite files 24
- Opening existing files 24

**Top features you need to know about**
- Editing 27
  - General editing operations 27
  - Context-sensitive code editing 28
  - File templates 40
  - File properties 44
  - The editor’s appearance 45
- High-level programming operations for the PHP language 46
  - Accessing documentation 47
  - Generators 48
Table of Contents

Inheritance diagrams 51
Autoformatting 52
Inspections 54
VCS 57
  Initiating a new Git repository – a new empty project 58
  Ignoring files 59
  Adding new files to the project 60
  Committing 61
  Log analysis 63
  Branches 64

People and places you should get to know 70
  Official sites 70
  Articles and tutorials 70
  Community 71
Welcome to *Instant PhpStorm Starter*. The efficiency of your work heavily depends on the tools you use. In order to achieve high productivity you have to reconsider your workbench. How much of your time is lost by performing simple yet unavoidable operations? Searching in the manual for a function you just called in your code, typing the same fragment of code for the hundredth time, preparing API documentation, or formatting the code to comply with standards—all of these tasks, simple though they are, will consume a lot of your time. What would you say to a magic wand that just performs them, with a single click or a keyboard shortcut? Marvelous?

For PHP programmers, this magic wand is called PhpStorm. The IDE created by professionals for professionals.

This book contains the following sections:

*So, what is PhpStorm?* will start with a brief introduction, underlining why you should try PhpStorm and what it can do for you.

*Installation* explains how you can get PhpStorm up and running on your system. The application is platform independent—you can follow instructions appropriate for your environment.

*Quick start – your first PHP application*

Impatient? Let’s stop talking and get to work. Without any further ado we will create a fully-fledged web application using Symfony2 framework. Then we will discuss basic features of the PhpStorm interface.

*Top features you need to know about* will dive into the most productive options that make PhpStorm shine. Its unsurpassed capabilities in guessing your intentions as far as possible include not only autocompletion and hinting but also code injection and live templates. These are context-sensitive operations. In some cases you don’t have to type the code at all, as it can be generated automatically. Generators discussed in this section will lift the burden of type setters, getters and magic functions from your shoulders.
We will also dive into different ways to access documentation here. Detailed instruction will show you how to display PHP manual pages, API and inheritance diagrams with a single shortcut or click.

Finally, we will learn how to improve the quality of PHP code using autoformatting and inspections. This section will be finished with a short introduction presenting how to use Git for version control.

*People and places you should get to know* will help you to find other sources for exploration so that you may find additional information about PhpStorm's amazing capabilities. So don't stop once you have learned the basics!
So, what is PhpStorm?

PhpStorm is an Integrated Development Environment (IDE) designed specifically to facilitate the development of Internet applications written in PHP. Its main features can be split into the following categories:

- Navigation
- Editing and re-factorization
- Files synchronization (deployment)
- VCS
- Debugging

Modern web applications consist of many files, libraries, modules, and classes stored in nested folders. PhpStorm will help you to navigate through all of them. Opened files are presented on tabs; you can switch them with mouse and keyboard shortcuts. Three controls: Project view, Framework MVC Structure, and Favorites are very useful when selecting files to be opened. Project view presents a tree-like control with folders and files, Framework MVC Structure presents the project organized into controllers and view. The last control, Favorites, contains labeled groups of files. Each group can consist of arbitrary files and thus you can create easily accessible sets of files stored in many different projects' folders.

PhpStorm's embedded editor is context sensitive and features customizable templates, code autocompletion, autoformatting, and folding. The same text ——depending on the context—can be expanded into:

```markdown
foreach
font-family
<fo></fo>
```

The first replacement will happen if the current context is PHP, the second replacement in the context of CSS, and third inside the HTML code. Moreover, templating, autocompletion, and autoformatting can be tailored to your specific requirements globally and on a per-project basis, giving you full control of the inserted code.

Internet applications are usually developed, tested, and deployed on different machines. Thanks to built-in synchronization operations, the project's files can be sent to or downloaded from arbitrary FTP, FTPS, or SFTP servers. Thus the basic deployment process is an internal PhpStorm operation that doesn't require additional tools.

If you intend to cooperate with other developers, you should consider VCS systems. PhpStorm supports all major solutions: CVS, Subversion, Git, Mercurial, Perforce, and TFS. A simplified VCS interface allows easy cooperation with other programmers, while the built-in editor helps tracing changes in files.
The most advanced features of PhpStorm concern object-oriented programming. The IDE can easily navigate through a namespaced OOP code (which include classes, methods, and parameters). You can switch from a method call to its implementation, find all places where a method or class is used, and find all the methods and properties defined in a class. Type hierarchy and diagrams present the inheritance relation in a very intuitive and easy-to-read manner. On the other hand, when overriding methods, you don't have to analyze ancestors because automatic hinting takes care of all inherited interfaces as well as the magic methods. Automatic generators of setters, getters, and constructors will increase your productivity further, while refactoring operations will be indispensable during renaming and moving of classes, methods, and properties.

If you compliment the described features with advanced debugging capabilities, you will get a product that will satisfy the most demanding users.
Installation

In five easy steps, you can install PhpStorm and get it running on your system.

Step 1 – What do I need?
Before you install PhpStorm, you will need to check that your working environment complies with the following requirements:

- Disk space: 300 MB free (minimum).
- Memory: 512 MB (minimum), 2 GB (recommended).
- Screen: 1024 x 768 (minimum).
- Oracle (Sun) JDK 1.6 or higher.


PhpStorm is available for three platforms: Windows, Mac, and Linux. Additional requirements depend on the platform:

- Windows
  - Processor: Intel Pentium III/800 MHz or higher
  - Microsoft Windows 7 (including 64-bit)/Vista/2003/XP/2000

- Mac
  - Mac OS X 10.5 or higher

- Linux
  - Processor: Intel Pentium III/800 MHz or higher

Step 2 – Downloading PhpStorm
Installation packages of PhpStorm for all platforms can be found at http://www.jetbrains.com/phpstorm/download/index.html. Keep in mind that it is a commercial software and you can use it for free only during the first 30 days after installation. After this time you will have to purchase a license key.

At the time of writing *Instant PhpStorm Starter*, the latest available version of PhpStorm was 5.0.4. All figures in this book were created using PhpStorm 5.0.4.
Step 3 – Proceeding with the installation
Depending on your platform, choose the appropriate installation instruction.

- **Windows**
  Run the downloaded executable file, for example, PhpStorm-5.0.4.exe. The installation wizard will present six dialog boxes. The only decisions you are required to make involve the choice of destination folder, desktop icons, and the start menu folder name. You can safely leave the default values as they are.

- **Mac**
  Mount the downloaded Mac OS X Disk image file, for example, PhpStorm-5.0.4.dmg, as another disk in your system and copy PhpStorm to your applications folder.

- **Linux**
  Unpack the downloaded file, PhpStorm-5.0.4.tar.gz, using the following command:
  
  ```
  tar xzf PhpStorm-5.0.4.tar.gz
  ```
  
  After that, run PhpStorm.sh from the bin subdirectory.

Step 4 – Entering the license key
Using the links **Buy now** or **Apply now**, available at the address http://www.jetbrains.com/phpstorm/buy/index.jsp, purchase or obtain for free a license key. Next start PhpStorm and display the dialog box **Help/Register**. Inside this dialog box enter received license key.

Step 5 – Licenses
JetBrain offers five different kinds of licenses:

- **Commercial License**
- **Personal License**
- **Academic License**
- **Classroom License**
- **Open Source Project License**
Individual developers working on commercial projects should choose Personal License. The last two licenses—Classroom License and Open Source Project License—are issued for free. In order to get one of them, you will have to send a special request. The form for a request is linked at the Buy & Upgrade tab on PhpStorm home page (http://www.jetbrains.com/phpstorm/buy/index.jsp). The links are titled Apply now.

Complete comparison of all the licenses is available at http://www.jetbrains.com/phpstorm/buy/license-matrix.jsp.

And that's it!!
By this point, you should have a working installation of PhpStorm.
Quick start – your first PHP application

Once you've completed the installation, you can proceed with your first application. PHP is mainly used for developing web applications, thus we will use PhpStorm to create a PHP-driven website. Moreover, to make life easier one of the most popular PHP frameworks, that is, Symfony2 will be adopted.

Symfony2 is currently a very hot topic for PHP programmers. It sets a new standard for PHP development:

- A Symfony2 code is object-oriented, namedpaced, and standards-compliant
- The quality of the code is asserted with unit tests
- Third-party libraries are managed with the composer—the PHP dependency manager
- For code management and testing, Symfony2 has adopted two of the most popular version control and continuous integration solutions: GitHub and Travis CI


In order to run this example, the user needs PHP installed.

Step 1 – Unzipping and opening files

Visit the address http://symfony.com/download and download the latest version of Symfony2.

1. Choose the file labeled as Symfony Standard 2.*.* (.zip). Its name should conform to the pattern: Symfony_Standard_Vendors_2.*.*.zip.
2. Unzip this archive to the folder symfony-2-hello-world/, then start PhpStorm and choose the File/New project from existing files main menu option:
3. You will be asked about your server configuration. Select the last choice, which is, *Source files are in a local directory, now Web server is yet configured.*

4. In the last dialog box, select the folder `symfony-2-hello-world/`.

5. After this operation you will see the main window of PhpStorm as shown in the following screenshot:
Instant PhpStorm Starter

PhpStorm stores the configuration of the project inside the `.idea` subfolder created inside the project's root folder. Thus after the first step the `symfony-2-hello-world` folder will contain the `.idea` subfolder.

**Step 2 – Enabling Symfony2 operations**

Open the Event Log tool window (View | Tool Windows | Event log). There you will find the information that the current project uses the Symfony2 framework. Click on the Configure hyperlink as shown in the following screenshot:

Inside the dialog box press the button **Apply** and then **OK**:
After that, close the dialog box.

After this operation your project will contain two new tool windows: Framework and Command Line Tools Console. You will find these windows in the main menu. Navigate to View | Tool Windows | Framework and then Tools | Run command. First the tool window (that is, Framework) will help you to navigate through the project, the second will be used to execute Symfony2 commands. We will see how to do that in Step 3 – Generating HelloBundle.

The Command Line Tools Console window is displayed inside Tools Windows only after you run the operation using the Run command window in the Tools tab.

Step 3 – Generating HelloBundle

The main building block of applications developed with Symfony2 is a bundle. A bundle can be compared to modules or plugins in other frameworks. To create a new bundle, go to Tools | Run command and execute the following command:

```
php app/console generate:bundle
   --namespace=My/HelloBundle
   --dir=src
   --no-interaction
```

The previous command should be written without new line characters. New line characters are inserted to increase readability of the command.

The command will create the src/My/HelloBundle folder and will add the following inside app/AppKernel.php:

```php
   new My\HelloBundle\MyHelloBundle(),
```
It will add the following line inside `app/config/routing.yml`:

```yaml
my_hello:
    resource: "@MyHelloBundle/Controller/"
    type:       annotation
    prefix:    /
```

To avoid conflicts between bundles coming from different vendors, the bundle name is two-dimensional. It contains the vendor’s name (that is, My) and the actual bundle name (that is, HelloBundle).

**Bundles, controllers, actions, and templates**

The code of Symfony2 application is structured into bundles. You can think about them as modules or plugins. Bundles contain controllers that—in default settings—are classes. Each controller (that is, class) contains methods. We will call these methods as actions. In a typical scenario, each action, when executed, will produce some output. This output will be rendered as HTML using a template file called `view`.

Our application contains a bundle named `My/HelloBundle`. My is the name of a vendor, and HelloBundle is the name of the bundle. The full name `My/HelloBundle` is unique in the whole application.

Inside our bundle you will find one controller, `src/My/HelloBundle/Controller/DefaultController.php`. This controller contains only one action named `indexAction` as shown:

```php
class DefaultController extends AbstractController
{
    // ...

    public function indexAction($name)
    {
        // ...
    }
}
```

The view of the action is stored in the file, `src/My/HelloBundle/Resources/views/Default/index.html.twig`.

Using the **Framework** window, you can easily navigate through controllers, their actions and views:
Step 4 – Customizing the action

1. Open the file src/My/HelloBundle/Controller/DefaultController.php
   You can go to View | Tool Windows | Project. Select the filename in the project window
   and double-click on it:
2. Change the code of `indexAction` (the method inside the `DefaultController` class) as shown in the following code:

```php
class DefaultController extends Controller
{
    /**
     * @Route("/")
     * @Template()
     */
    public function indexAction()
    {
        return array();
    }
}
```

3. Next, open the view `src/My/HelloBundle/Resources/views/Default/index.html.twig` and change its contents to the following code:

```twig
{% extends '::base.html.twig' %}

{% block title %}
   Hello, world!
{% endblock %}

{% block body %}
   <h1>Hello, world!</h1>
{% endblock %}
```

**Step 5 – Opening your browser**

Now start the browser and visit its address:

http://localhost/symfony-2-hello-world/web/

Note that the actual address you use depends on your directory structure and configuration of your server.

You should see a page that looks a similar to the following screenshot:
The PhpStorm interface

When you completed the first application, it’s time to get familiar with PhpStorm’s interface. In this section we will analyze PhpStorm’s main window and its components. PhpStorm contains the following controls:

- Main menu (1)
- Toolbar (2)
Instant PhpStorm Starter

- Navigation bar (3)
- Tool buttons (4)
- Editor window (5)
- Project Tool Window (6)
- Status bar (7)

Main menu, toolbar, and status bar are standard controls available in almost all windowed applications. Tool buttons, tool windows, and navigation bar are specific to PhpStorm.

The following table illustrates and provides the path to these controls, which are specific to PhpStorm:

<table>
<thead>
<tr>
<th>Name</th>
<th>Illustration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toolbar</td>
<td><img src="image1.png" alt="Toolbar Illustration" /></td>
</tr>
<tr>
<td>Path from the main menu:</td>
<td>View</td>
</tr>
<tr>
<td>Tool Buttons</td>
<td><img src="image2.png" alt="Tool Buttons Illustration" /></td>
</tr>
<tr>
<td>Path from the main menu:</td>
<td>View</td>
</tr>
</tbody>
</table>
### Status Bar

Path from the main menu:
**View | Status Bar**

### Navigation Bar

Path from the main menu:
**View | Navigation Bar**

Tool buttons can be turned on and off by clicking on the button shown, available in the left-bottom corner of the status bar:
Instant PhpStorm Starter

Tool windows

Tool buttons, which were previously presented, allow easy access to Tool windows. You can access every tool window using the main menu:

- View | Tool Windows | Project
- View | Tool Windows | Favorites
- View | Tool Windows | TODO
- View | Tool Windows | Structure
- View | Tool Windows | Database
- View | Tool Windows | Event log
- View | Tool Windows | Remote host

Or you can access them using Tool Buttons:
Each tool window has two context menus. You can enter them with a right mouse click on a tool button:

Or a title bar of a tool window:

Both the context menus contain the same options:

- **Pinned Mode, Docked Mode, Floating Mode, Split Mode** – these options change the mode of the window
- **Show Views as Tabs** – this option changes the appearance of sub options
- **Move to, Resize** – these options can be used to change the size and placement of the window
- **Hide** – this option hides the window (shortcut Shift + Esc)
The options concerning the mode of a window are also available after clicking on the gear icon in the top-right corner of each tool window:

Split mode changes the position of a button on the toolbar. TODO and Event Log buttons, visible when the split mode is turned on and off will be aligned to the left or right:
Using the **Move to** operations, you can place all the buttons on the same side of the window. Here is PhpStorm with all the buttons moved to the left with **Move to/Left**: 
**Instant PhpStorm Starter**

When you set the mode of the windows to **Pinned Mode** and **Dock Mode**, then you will get four regions—each of them capable of holding one window:

![Phabricator](image)

If **Pinned Mode** and **Dock Mode** are turned off then only one window can be visible:
If we activate any other tool window, for example, Favorites, Project Tool Window—the one that is currently visible—will be hidden.

Using **Floating Mode** and **Pinned Mode**, you can configure tool windows to be displayed at arbitrary locations:
Favorite files
Contemporary web applications usually consist of many files. Our first project contains more than 5000 files and 1000 folders. One of the very useful PhpStorm features is the ability to group arbitrary files for easier access. When you open the Favorites window (View | Tool Windows | Favorites), you will find there two buttons labeled + and -. Using these buttons you can create and delete new groups of the favorite files. Once the group is created, you can drag-and-drop any project files. The following is a group titled files modified in the first example, containing all the files that were modified in the Hello, world! example:

Opening existing files
The following are the four most useful methods to open files:

1. Using the standard File | Open operation:
2. By double-clicking on the name of the file in **Project window**:
3. Using the navigation bar:

4. Using the Ctrl + Shift + N shortcut:
Top features you need to know about

PhpStorm is a fairly-sophisticated IDE that goes far beyond standard editing operations, simple autocompletion, and hinting. It offers highly customizable templates that can be used in context-sensitive matters. You will learn to use them in the Editing section. Next, in the High-level programming operations for PHP language section, we will examine generators, autoformatting, and inspections. Finally, the VCS section will show you how to use embedded version control operations.

Editing

Once you know how to start a new project you will certainly need to edit some files. The following is what you will learn:

- To use general operations: copy and paste, change indentation, toggle case, duplicate code and open web pages with arbitrary browser
- To use code hinting, autocompletion live templates, and code injection
- To use and customize file templates
- To set basic files properties: encoding and the read-only attribute
- To change the appearance of the editor window

General editing operations

Before we proceed with templates and autocompletion, it’s worth taking the time to learn the operations you will need most frequently: copy and paste, and indentation.

- Copy and paste

  PhpStorm supports multiple copy/paste buffers. You can list all the available pastes with Edit | Paste from history. The shortcut for this operation is Ctrl + Shift + V (Mac: command + shift + V).

  To select vertical columns of the text you can use two methods. The first, is a special option, Edit | Column selection, which is available in the main menu. When activated, this option will produce the following result when you select some text:

  ![Column selection example]

  The same result can be achieved with the Alt key.
Indentation

The following are three options that will help you indent your code:

- **Edit | Indent selection (Tab)**
- **Edit | Unindent selection (Shift + Tab)**
- **Edit | Convert indents**

Changing case

The **Toggle case** option (**Edit | Toggle case**) toggles uppercased letters to lowercase and vice versa. The shortcut for this operation is **Ctrl + Shift + U** (Mac: **command + shift + U**).

Duplicates

In order to duplicate a selected block or current line, use the **Duplicate** operation (**Ctrl + D**, Mac: **command + D**). Depending on the context, you may use **Edit | Duplicate line** (when no block is selected) or **Edit | Duplicate block**.

Web preview

If you are editing HTML files, PhpStorm can automatically preview them with an arbitrary browser. The **Web preview** option (**View | Web preview**, shortcut: **Alt + F2**) opens the context menu:

![Web preview screenshot]

Context-sensitive code editing

1. Start a new project in PhpStorm. Create an empty file and name it **index.php** with following content:

```php
<?php

?>
<!DOCTYPE html>
<html>
<head>
<title></title>
<meta charset="UTF-8" />
</head>
<body>

</body>
</html>
```
The editor is context sensitive, which means that the result of commands can be different depending on the position of the cursor in the document.

2. Place the cursor between PHP tags:
   ```php
   <?php
   |
   ?>
   
   And then type `file` and press the tabulator. You will get the following result:
   ```php
   <?php
   file();
   ?>
   
3. Next, move the cursor inside CSS tags:
   ```html
   <style>
   |
   </style>
   
   And again type `file` and press the tabulator. This time the result will be as follows:
   ```html
   <style>
   color-profile
   </style>
   
4. Finally, place the cursor between the `body` tags:
   ```html
   <body>
   |
   </body>
   
   Then type `file` and press the tabulator. You will get:
   ```html
   <body>
   <file></file>
   </body>
Instant PhpStorm Starter

5. Depending on the context, the same operation produces different autocompleted texts:

- Inside PHP code the result will be: `file()`
- Inside CSS code the result will be: `color-profile`
- Inside HTML code the result will be: `<file></file>

Comments

Two of the most useful operations for programmers are comment and uncomment. You can access them with the main menu by navigating to **Code > Comment with...**, but after a while you will surely prefer shortcuts `Ctrl + /` (for line comment) and `Ctrl + Shift + /` (for block comment). The Mac equivalents are `command + /` and `command + shift + /`. Execute both of them in HTML, CSS, and PHP files to ascertain that they are context sensitive.

If you select the `foreach` instruction as shown:

```php
<?php
foreach (file('books.txt') as $line) {
    $e = explode('|', trim($line));
    echo $e[0] . "<br/>";
}
?>
```

And then activate the **Comment with line comment** option under the **Code** menu, you will get the following code:

```php
//foreach (file('books.txt') as $line) {
//    $e = explode('|', trim($line));
//    echo $e[0] . "<br/>";
//}
```

If you perform the same operation when the `ul` list is selected as shown:

```html
<ul>
    <li>red</li>
    <li>green</li>
    <li>blue</li>
</ul>
```

You will see the following result:

```html
<!--<ul-->-->
    <!--li>red</li-->-->
    <!--li>green</li-->-->
    <!--li>blue</li-->-->
<!--</ul-->-->
```
Autocompletion and code hints

Autocompletion removes the burden of remembering many long function and variable names. If you type `file_g` inside the PHP code editor, it will automatically guess your intention to type the `file_get_contents` function name. In the same manner you can type `list-` inside CSS rule and get the hint containing all the attributes:

```php
p {
    list-style: ;
    list-style-position: ;
    list-style-image: ;
    list-style-type: ;
}
```

To get the value for any of the previous attributes, use arrows and Enter. If you press the Esc key, the hints will disappear. The Ctrl + Space bar shortcut will recall the hints.

Now select `list-style-position` and put the cursor after the colon:

```css
ul {
    list-style-position: |;
}
```

Now press Ctrl + Space bar. You will get the hints: `inherit, inside, outside`. No need to check the CSS documentation! Typical hints in the PHP include the names of variables. Type the code and position the cursor below the `$b` variable as shown in the following code:

```php
<?php

$a = 1;
$b = 2;

|

?>
```

Now Ctrl + Space bar will produce the list of all variables in the current scope.
Live templates

Live templates offer another method of entering arbitrary large code fragments. They are defined inside the Settings window in the Live templates dialog box as shown in the following screenshot:

Here is definition of the `foreach` live template shown in a previous dialog box:

```php
foreach ($ITERABLE$ as $$$VAR_KEY$ => $$$VAR_VALUE$) {
    \$END$
}
```

This live template is available inside the PHP context. Information about it is displayed in the lower part of the window as: Applicable in PHP. Change. If you want to execute this live template, perform the following steps:

1. Open the PHP file, and place the cursor somewhere between the PHP tags `<?php` and `?>`.
2. Type `foreach`, and click on Tab.
   You will get following code:
   ```php
   foreach ( as $ => $) {
   \$END$
   ```
3. Similarly, when you type `pubsf` and press Enter, you will get the following code:

```php
public static function (){
}
```

The list of all live templates available in the current context is available in main menu at `Code | Insert live template` (Ctrl + J, Mac: command + J).

A large number of live templates are defined for CSS and HTML. They are available under Zen CSS and Zen HTML. Open the HTML file and then type `html:5` and press Enter. You will get the contents of the `html:5` live template.

4. If you want to define a new PHP live template, go to `Settings | Live templates`. The dialog box will open, then select the **PHP** category and click on the plus icon as shown in the following screenshot:

5. Then in the dialog box, enter the following values:

   - **Abbreviation**: cls
   - **Description**: class with constructor

And the following code:

```php
/**
 * $NAME$ class allows...
 */
```
class $NAME$
{
    /**
     * Constructor.
     */
    public function __construct($PARAMETERS$)
    {
        $END$
    }
}
6. The last step is to set the context for the defined template. Click on the **Change** link and choose **PHP** as shown in the following screenshot:

7. After you complete the definition, close the previous dialog box by clicking on the **OK** button, and open any PHP file. Place the cursor between the PHP tags `<<?php` and `?>`. If you type `cls` and then press the tabulator, you will get your defined template. If you start typing now, for example `Abc`, you will change both the class name and the PhpDoc comment into this:

```php
/**
 * Abc
 * ...
 */

class Abc
{
  ...
}
```
8. Live templates can contain variables. In the previous example we defined a variable \$NAME\$. Let's try with another example. Using the dialog box shown in step 5, define the live template with abbreviation for\(i\) and the following template text:

```php
for ($NAME = 0; $NAME < 10; $NAME++) {
    echo $NAME;
}
```

The previous template contains the variable $NAME$ in four places. If you insert the template into your PHP document and start typing the text $TheVariable$, you will get the following code:

```php
for ($TheVariable = 0; $TheVariable < 10; $TheVariable ++) {
    echo $TheVariable;
}
```

9. If the template uses multiple variables, as, for example, with the foreach template, you will get the following code:

```php
foreach ($ITERABLE$ as $$VAR_KEY$ => $$VAR_VALUE$) {
    $$END$
}
```

10. You can switch between them with the Tab key. The special variable $END$ denotes the cursor's position in a template.

Live templates can be used to surround the selected text. Inside any HTML file, somewhere between the body tags, type the following text:

```html
<body>
Lorem ipsum dolor sit amet
</body>
```

11. Select this paragraph and then go to Code | Surround with live template (Ctrl + Alt + J, Mac: command + alt + J):
12. From the context menu, select the first option: **T2. Surround with \texttt{<tag></tag>} in HTML/JSP.** And then type a single letter \texttt{p}. You will get the following code:
\[
\texttt{<p>Lorem ipsum dolor sit amet</p>}
\]

13. The next thing to do is select \texttt{ipsum}, press \texttt{Ctrl + Alt + J}, choose XML Zend Coding, and type \texttt{str}. This time you will get the following code:
\[
\texttt{<p>lorem <strong>ipsum</strong> dolor sit amet</p>}
\]

14. The reverse operation is also implemented. You can remove the tags surrounding any fragment of text. Go to the following code:
\[
\texttt{<p>lorem <strong>ipsum</strong> dolor sit amet</p>}
\]

15. Place the cursor inside \texttt{ipsum}, between \texttt{p} and \texttt{s}. Then go to **Code | Unwrap | Remove** (\texttt{Ctrl + Shift + Delete}, Mac: \texttt{command + shift + delete}). You will get the context menu, which will offer the following four operations:

- Remove Enclosing Tag \texttt{strong}
- Remove Enclosing Tag \texttt{p}
- Remove Enclosing Tag \texttt{body}
- Remove Enclosing Tag \texttt{html}

You can use these operations to remove one of the surrounding HTML tags.

**Code injection**

As you have probably noticed, the templating operations are context sensitive. You can use HTML templates inside HTML context and PHP templates inside PHP context. Sometimes you will probably need to use live templates out of the current context.
Instant PhpStorm Starter

For example, inside a PHP file, you will need to define a string containing HTML code:

```php
$html = "<h1>Hello, <strong>world</strong>!</h1>";
```

1. By default, inside a PHP string you can access PHP live templates:

   ![PHP code with live templates]

   If you type `$html = "h1";` and press the Tab key, nothing interesting happens. In order to access HTML live templates inside a PHP string, you can use the code injection technique. Place the cursor between quotation marks and click on the bulb icon as shown in the following screenshot:

   ![Code injection example]

2. From the context menu, click on **Inject Language** and then on **HTML**:

   ![Injected HTML code]

3. After this operation you can access HTML live templates inside a PHP string variable. You can check by going to Code | Insert Live Template (Ctrl + J, Mac: command + J):

4. If you now type $html = "h1"; and press the Tab key, you will get this:

   $html = "<h1>\</h1>";

   This means that you can access HTML templates.

Some special constructs can be used to define automatic injections without the need to run the Inject Language operation. Inside the PHP code this automatic code injection will happen if you use heredoc strings denoted with HTML, CSS, JS, SQL, XML, or REGEX.

If you type the following PHP code, and place the cursor at | position:

```php
<?php
$str = <<<HTML
One, two,
|$HTML;
?>
```

Then if you type file and press Enter you will get the following code:

```php
<?php
$str = <<<HTML
One, two,
<file></file>
HTML;
?>
```

It means that inside a string the autocompletion features work in the HTML context.
Similarly, to test the behavior for the SQL language embedded within a PHP string, type the code as follows:

```php
<?php
$sql = "SELECT * FROM |";
?>
```

If the cursor is placed at the `|` position, then the Ctrl + Space bar shortcut will produce the autocomplete list for the SQL language.

All automatic code injections are defined at Settings | Language Injections.

File templates

When you need to add a new file to your project, go to File | New.... It is available in the main menu and in the project's context menu:
The last option inside the **New...** submenu is **Edit File Templates**. Using this option you can define and customize file templates. Here is the procedure used to define an HTML5 template containing a meta tag and embedded CSS:

1. **Select the available, predefined** HTML5 **template (A).**
2. **Create a duplicate named** Copy of HTML5 (**B**).
3. **Select** Copy of HTML5 **template (C).**
4. **Insert the following new HTML code (D):**
   ```html
   <!DOCTYPE html>
   <html lang="en">
   <head>
     <title></title>
     <meta charset="UTF-8" />
     <style>
       ...
     </style>
   </head>
   <body>
     ...
   </body>
   </html>
   ```
5. **Apply your changes (E).**
Instant PhpStorm Starter

After you define a new Copy of Html5 template, it is available inside File | New... as shown in the following screenshot:

The definition of a template can contain the following variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>${NAME}</td>
<td>Current filename</td>
</tr>
<tr>
<td>${USER}</td>
<td>Current user system login name</td>
</tr>
<tr>
<td>${DATE}</td>
<td>Current system date</td>
</tr>
<tr>
<td>${TIME}</td>
<td>Current system time</td>
</tr>
<tr>
<td>${YEAR}</td>
<td>Current year</td>
</tr>
<tr>
<td>${MONTH}</td>
<td>Current month</td>
</tr>
<tr>
<td>${DAY}</td>
<td>Current day of the month</td>
</tr>
<tr>
<td>${HOUR}</td>
<td>Current hour</td>
</tr>
<tr>
<td>${MINUTE}</td>
<td>Current minute</td>
</tr>
<tr>
<td>${PRODUCT_NAME}</td>
<td>Current IDE name</td>
</tr>
<tr>
<td>${PROJECT_NAME}</td>
<td>Name of the current project</td>
</tr>
</tbody>
</table>

Template files are written in Velocity Template Language (VTL). Documentation of VTL is available at [http://velocity.apache.org/engine/releases/velocity-1.6.4/user-guide.html](http://velocity.apache.org/engine/releases/velocity-1.6.4/user-guide.html).

Here is an example that shows how to use HTML comments to add information about the author:

```html
<!DOCTYPE html>
<!-- by ${USER} -->
<html>
<head>

...</head>
```

42
In order to include the same header information inside many templates, use the **Includes** tab as shown in the following screenshot:

![file templates](image)

The included information is embedded inside a template with following command:

```php
#parse("the name of the include")
```

Each defined include can be used in an arbitrary number of templates. Thus the same file prefix is defined for the following templates:

♦ PHP file
♦ PHP class
♦ PHP interface
♦ PHP trait

The template for a PHP class looks like this:

```php
<?php
#parse("PHP File Header.php")

class ${NAME} {
}
```

File templates are available in PHP settings. Go to **File | Settings | File Templates.**
File properties

PhpStorm's status line presents basic information about the current file:

The following table lists the components of the status line:

<table>
<thead>
<tr>
<th>Arrow labelled with</th>
<th>Shows</th>
<th>When clicked</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>The current line and column number</td>
<td>Opens the Go to line dialog box</td>
</tr>
<tr>
<td>B</td>
<td>The file's encoding</td>
<td>Changes the encoding</td>
</tr>
<tr>
<td>C</td>
<td>The read-only attribute</td>
<td>Sets or unsets the read-only attribute</td>
</tr>
<tr>
<td>D</td>
<td>The inspection icon</td>
<td>Opens the inspector dialog box</td>
</tr>
</tbody>
</table>

Using the File encodings option (File | Settings | Project Settings | File Encodings, Mac: PhpStorm | Preferences | Project Settings | File Encodings), you can set the default encoding for all new files in the project.
The editor's appearance

If you want to customize the editor appearance go to the Settings dialog box and analyze the following dialog boxes under Editor:

✦ Appearance
✦ Colors & Fonts
✦ Editor Tabs

Appearance contains checkboxes that allow you to set:

✦ block caret
✦ line numbering
✦ margins
✦ showing white characters

Some of these options are available in the main menu under View | Active Editor as shown in the following screenshot:
Instant PhpStorm Starter

The **Colors & Fonts** dialog box uses predefined schemes. If you want to create a customized scheme titled "my", follow this procedure:

1. Copy the **Default** scheme (1)
2. Set the name of a newly created scheme to **Large Font Scheme** (2).
3. Apply the changes (3).

4. Next change the fonts properties (for example, font size and color). All the modifications will be stored in a new scheme. If you want to reload another set of predefined editor properties, simply change the scheme name using the drop-down list denoted by 4.

High-level programming operations for the PHP language

PhpStorm offers large number of high-level programming operations that will help you to write and inspect the PHP code. The following list provides a few examples of such operations:

- Accessing documentation
- Generating code
- Generating class diagrams
- Autoformatting
- Inspections
Accessing documentation

PhpStorm offers four different operations that will help you to access the documentation: Quick Definition, Quick Documentation, Parameter Info, and External Documentation. The first one, Quick Definition, presents the definition of a given symbol. You can use it for a variable, function, method, or class. Quick Documentation allows easy access to DocBlocks. It can be used for all kinds of symbols: variables, functions, methods, and classes. The next operation, Parameter Info, presents simplified information about a function or method interface. Finally, External Documentation will help you to access the official PHP documentation available at php.com.

Their shortcuts are as follows:

- Quick Definition (Ctrl + Shift + I, Mac: alt + Space bar)
- Quick Documentation (Ctrl + Q, Mac: F1)
- Parameter Info (Ctrl + P, Mac: command + P)
- External Documentation (Shift + F1, Mac: shift + F1)

The Esc (Mac: shift + esc) hotkey will close any of the previous windows.

If you place the cursor inside the parenthesis of $s = str_replace(); and run Parameter Info (Ctrl + P, Mac: command + P), you will get the hint showing all the parameters for the `str_replace()` function. If that is not enough, place the cursor inside the `str_replace` function name and press Shift + F1 (Mac: shift + F1). You will get the manual for the function.

If you want to test the next operation, open the project created in the Quick start – your first PHP application section and place the cursor inside the class name Controller in the `src/My/HelloBundle/Controller/DefaultController.php` file. The place where you should place the cursor is denoted with bar | in the following code:

```php
class DefaultController extends Controller
{
}
```

The Quick Definition operation will show you the class definition:
The **Quick Documentation** operation will show you the documentation defined with PhpDoc blocks:

It is a formal standard for commenting on the PHP code. The official documentation is available at [http://www.phpdoc.org](http://www.phpdoc.org).

**Generators**

PhpStorm enables you to do the following:

- Implement magic methods
- Override inherited methods
- Generate constructor, getters, setters, and docblocks

All of these operations are available in **Code | Generate** (Alt + Insert, Mac: command + N). Perform the following steps:

1. Create a new class `Foo` and place the cursor at the position of `|`:

   ```php
   class Person
   {
   }
   ```
2. The **Generate** dialog box will contain the following operations:

![Implement Methods dialog box]

The **Implement Methods** dialog box contains all available magic methods:

![Choose methods to implement]

3. Create the class with two private properties:

   ```php
   class Lorem
   {
       private $ipsum;
       private $dolor;
   }
   ```

4. Then go to **Code | Generate | Getters and Setters**. In the dialog box select both properties:

![Choose Fields]

5. Then press OK. PhpStorm will generate the following methods:

```php
class Lorem
{
    private $ipsum;
    private $dolor;

    public function setDolor($dolor)
    {
        $this->dolor = $dolor;
    }

    public function getDolor()
    {
        return $this->dolor;
    }

    public function setIpsum($ipsum)
    {
        $this->ipsum = $ipsum;
    }

    public function getIpsum()
    {
        return $this->ipsum;
    }
}
```

6. Next, go to Code | Generate | DocBlocks and in the dialog box select all the properties and methods:
7. PhpStorm will generate docblocks for each of the selected properties and methods, for
example:

```php
/**
 * @param $dolor
 */
public function setDolor($dolor)
{
    $this->dolor = $dolor;
}
```

**Inheritance diagrams**
The Context menu, which is available inside a PHP code, contains two options
concerning diagrams:

- **Diagrams | Show Diagram** (Ctrl + Alt + Shift + U)
- **Diagrams | Show Diagram Popup** (Ctrl + Alt + U)

If you right-click inside the file `src/My/HelloBundle/Controller/DefaultController.php`, which was discussed previously, you will get the context menu as shown in the following screenshot:
Instant PhpStorm Starter

With them you can generate the diagram in the following screenshot for the DefaultController class:

![Diagram of DefaultController class](image)

The diagram can contain all the properties and methods. The default appearance for the diagram can be set in the Diagrams dialog box (Settings | Diagrams, Mac: Preferences | Diagrams).

**Autoformatting**

You can easily reformat your code by going to Code | Reformat Code (Ctrl + Alt + L, Mac: command + alt + L). Create a new PHP file and type the code without indents:

```php
class Abc
{
    public static function printNumbers($n)
    {
        for ($n = 0; $n < 10; $n++) {
            echo $n . "\n";
        }
    }
}
```
Then after you execute **Reformat Code** you will get this:

```php
class Abc
{
    public static function printNumbers($n)
    {
        for ($n = 0; $n < 10; $n++) {
            echo $n . "\n";
        }
    }
}
```

Coding standards are defined in the **Code Style** dialog box in **Settings**. PhpStorm allow you to define arbitrary coding styles for each language:

You can define your own format code or you can use one of the predefined formats:

- PEAR
- Zend
- Drupal
- PSR1/PSR2
- Symfony2
Instant PhpStorm Starter

If you want to use predefined format, go to Set from... | Predefined Style as shown in the following screenshot:

Conformance of your code to a specific standard can be automatically verified with the phpcs tool discussed in next section.

Inspections

The quality of your code can be asserted with inspections (Code | Inspect Code). This operation is available for many different languages: PHP, HTML, CSS, JavaScript, and so on. This is a context-sensitive operation, which works differently for every language.

We will analyze how it works for the PHP code. If you run Code | Inspect code when editing the DefaultController.php file, you will first see the dialog box that sets the scope of inspection. Select a single file, as show in the following screenshot:
By default, the inspection informs, about various problems which include syntax errors, unused variables and classes, mistyped keywords, dead code, and so on. Here is the result of inspecting DefaultController.php. The result informs us of two unnecessary use instructions:
The inspections can be tailored to your specific needs for each language independently. Open the **Inspections** dialog box in **Settings**, and then go to **PHP** as shown at following screenshot:

![Inspections dialog box](image)

Using this option you can define inspections for PHP code. Every type of inspection is activated or deactivated independently, for example:

- **Under General:**
  - Class hierarchy checks
  - Deprecated
  - Dynamic method called as static

- **Under Code smell:**
  - Inconsistent return points
  - Usage of silence operator

In the same manner you can turn on or off specific inspections for other languages: HTML, CSS, or JavaScript to name a few.
A quite popular tool to inspect if the PHP code conforms to a given standard is named pphpcs. It is available at PEAR package [http://pear.php.net/package/PHP_CodeSniffer/redirected. PhpStorm] and allows you to use the phpcs tool as an inspection. To define such an inspection, first go to Settings | PHP | Code sniffer and enter the path to the phpcs tool. Next navigate to Settings | Inspections | PHP and check PHP Code Sniffer and open that validation option:

After this, when you run the inspection, your code will be checked with phpcs.

VCS
Nowadays, everyone is working using Version Control System. PhpStorm includes embedded plugins which allow you to work with all major VCS systems: Git, Mercurial, Subversion, CSV and with most popular open source hosting platforms such as GitHub. In this section you will learn to work with Git and GitHub from within PhpStorm. You will see:

- How to initiate a new repository in PhpStorm
- How to commit changes
- How to analyze a log of the repository
- How to create and switch branches
- How to merge branches
- How to resolve conflicts
In order to proceed with the following examples you need to install Git. This software is available at http://git-scm.com/downloads.

**Initiating a new Git repository – a new empty project**

Start a new PhpStorm project called `git-start-afresh` and then select **VCS | Import into Version Control | Create Git Repository**... as shown in the following screenshot:

You will be asked to set the destination for a `.git` folder. Confirm the default settings:

The last dialog box will ask the question, **Do you want to add this directory as a VCS root?**. After confirmation your project is ready.
Now your project is stored as a local Git repository and its main branch is called **master**:

![Image of Git master branch](image)

**Ignoring files**
Open source projects usually do not include IDE settings inside repositories. If you want to remove the `.idea/` folder from your repository, follow this procedure:

1. Go to **Version Control | Ignored Files** (A).
2. Add a new rule (B).
3. The rule should ignore all files under (C), select the `.idea/` folder (D).

![Image of PhpStorm settings for ignored files](image)

The list of ignored files that we manage with the dialog box is stored in the PhpStorm configuration and not in the `.git` folder. Thus the files are not ignored by Git commands run within the command line. If you want to ignore files using built-in Git capacities, use `.gitignore` or `.git/info/exclude`. Visit [https://help.github.com/articles/ignoring-files](https://help.github.com/articles/ignoring-files) for further information.
Instant PhpStorm Starter

Adding new files to the project

Add a new `index.html` file to your project. You will be asked whether the file should be added to the repo, to which you should answer Yes. Next, add another file `foo.html`, but this time do not add it to the repo and do so by answering No.

PhpStorm uses colors to denote the state of files in the repo. The green color means that the new file was added to the repo but not committed yet (that is, it is staged). The red color means that the new file was not staged, that is, prepared to be included in the next revision.

Add another new file `bar.html` and then use the Ignored Files dialog box to ignore it. Its color will change to gray. Red, green, and gray colors represent three states: staged new files, unstaged new files, and ignored files:

![Image of file colors]

If you want to add all the files to the repo, click on VCS | Git | Add (Ctrl + Alt + A, Mac: command + alt + A). All the files will be staged.
Committing

In order to commit changes, execute VCS | Commit Changes (Ctrl + K, Mac: command + K). Using the button shown with the arrow you can analyze all the changes:
In order to complete a commit, you have to insert a commit message in the **Comment** box shown in the previous screenshot. After a successful commit, modify one of the stored files, for example, `foo.html`. Its appearance will change. The name of the file will be blue and the editor will contain stripes showing exactly what parts of the file were modified. The diverse methods to track modifications are shown in the following screenshot:

![Commit Changes Screenshot](image.png)

The horizontal arrows (→) show the changes in the editor window, while vertical ones, pointing downward show changes in the **Changes** tool window.

After another commit the file will be saved in the repo and changes will disappear. Notice that the tool window titled **Changes** shows information about modified files. If it doesn't contain any files, it means that all the changes were committed. All you have to do to verify it is take a look at the **Changes** tool window!
Log analysis

After some commits you can analyze the history of your project using the Log tab of the Changes dialog box:

![Diagram of the Changes dialog box with labels for different buttons and actions]

The following table will describe the previous screenshot:

<table>
<thead>
<tr>
<th>Button labeled with</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Opens the Changes tool window</td>
</tr>
<tr>
<td>B</td>
<td>Opens Log</td>
</tr>
<tr>
<td>C</td>
<td>Filters commits by commit message</td>
</tr>
<tr>
<td>D</td>
<td>Filters commits by branch</td>
</tr>
<tr>
<td>E</td>
<td>Filters commits by author</td>
</tr>
<tr>
<td>F</td>
<td>Filters commits by modified files</td>
</tr>
<tr>
<td>G</td>
<td>Selects a given commit</td>
</tr>
<tr>
<td>H</td>
<td>Shows commit’s message</td>
</tr>
<tr>
<td>I</td>
<td>Shows list of modified files</td>
</tr>
</tbody>
</table>
**Branches**

Branches are managed through a dialog box at **VCS | Git | Branches**. You can select it from the main menu, context menu, or by using a button in the status bar. If you click on the **New Branch** option a dialog box will pop up as shown in the following screenshot:

![Create New Branch dialog box](image)

Enter the name for a new branch. Your repository will switch to a newly created branch.

Create four branches: `dev`, `one`, `two`, `three`. Each of them will appear inside the **Git Branches** menu:

![Git Branches menu](image)

Notice that the **Git Branches** window presents the name of the current branch. If you want to switch to another branch select the branch name and perform the **Checkout** operation.
Now switch to the branch `one`, and create a `one.txt` file and add it to the repo. Inside this file type `One, one, one` and commit the change. The comment should contain a text `one1`. This operation is illustrated as follows:

Now change the current branch to `two`. The `one.txt` file will disappear. Using exactly the same procedure as previously, create—in the branch `two`—a new commit. The commit should contain one file, `two.txt`, containing the text `Two, two, two`.

If you switch the branch to one, then you will get the `one.txt` file. When the current branch is `two`, the project will contain the `two.txt` file.
Merging branches

Our project contains two different files: `one.txt` and `two.txt`. The files are included in different branches. The next operation we will undertake is the joining of the branches with the merge command. Switch the current branch to `one` and then go to VCS | Git | Merge changes. The checkbox with the name of the branch `two` should be checked:

After this operation the branch `one` will contain both files: `one.txt` and `two.txt`. The branch `two` contains only one file `two.txt`. Open the Log window in Changes and refresh its contents with the refresh button as shown in the following screenshot:

Conflicts

The merge operation previously discussed was executed without any problems because we modified two different files. The `one.txt` file was modified in the branch `one`, and the `two.txt` file was modified in the branch `two`. Now we will prepare modifications that overlap.

1. Switch to the branch `one` and inside the `index.html` file, insert a title and a paragraph:
   ```html
   <!DOCTYPE html>
   <html>
   <head>
   <title>Lorem</title>
   ```
2. Use Zen coding live template `html:5`. Open a new, empty HTML file and then type:

   `<html>`
   `<head>`
   `<title>Lorem ipsum</title>`
   `</head>`
   `<body>`
   `<p>Lorem ipsum.</p>`
   `</body>`
   `</html>`

3. Press Enter. Next, change the title, move, cursor into the body, and type:

   `<p>`

4. Press Enter. Inside the `p` tags, type `Lorem ipsum`.

5. Commit the changes and checkout the branch `two`. There, inside the `index.html` file, type the following contents:

   ```html
   <html>
   <head>
   <title>Dolor?</title>
   </head>
   <body>
   <article>Dolor sit amet.</article>
   </body>
   </html>
   ```

6. Commit the changes and checkout the branch `one`.

   If you try to merge the branch `two` into the branch `one`, you will get the information about conflicts. You are in the branch `one`. You can accept the version from this branch using the **Accept Yours** button. The version from the branch `two` can be accepted with the **Accept Theirs** button. These buttons are shown in the following screenshot:
The Files Merged with Conflicts dialog box can be invoked manually with the Resolve conflicts main menu option (VCS | Git | Resolve Conflicts).

7. You can resolve the conflict manually. Close the Files Merged with Conflicts dialog box. The editor will present the conflict visually with special markers <<<<<, =====, >>>>>:

Notice that conflicting files are marked red and that the status bar contains a strange branch’s name: Git: Merging one. The name of the branch means that we are in the middle of a branch operation.

8. The version from the current branch (one) is denoted by the following code:

```
<<<<<<< HEAD
<title>Lorem</title>
</head>
<body>
  <p>Lorem ipsum.</p>
=======
```
9. The overlapping contents from the merged branch (two) is shown as follows:

```
=======
  <title>Dolor?</title>
</head>
<body>
  <article>Dolor sit amet.</article>
```

10. Change the contents of the index.html file to the following code:

```
<!DOCTYPE html>
<html>
<head>
  <title>Lorem</title>
</head>
<body>
  <p>Lorem ipsum.</p>
  <article>Dolor sit amet.</article>
</body>
</html>
```

11. Then, proceed with commit. This last step will complete the merging. The name of the branch will again become one.
Instant PhpStorm Starter

People and places you should get to know

If you need help with PhpStorm, here are some people and places that will prove invaluable:

Official sites
- Home page: http://www.jetbrains.com/phpstorm/
- Official plugins: http://plugins.intellij.net/?webide

Articles and tutorials
- Debugging remote CLI with phpstorm by Joshua Thijssen:
  http://www.adayinthelifeof.nl/2012/12/20/debugging-remote-cli-with-phpstorm/
- JetBrains PhpStorm – The Smart IDE for WordPress Development by Mikhail Vink:
  http://www.wpmayor.com/articles/jetbrains-phpstorm-the-smart-ide-for-wordpress-development/
- Setting up PHPUnit and Selenium with IntelliJ IDEA/PhpStorm in official Yii documentation:
- Setting up JetBrains PhpStorm for use as a Drupal IDE by Chris Cohen:
  http://tiger-fish.com/blog/setting-jetbrains-phpstorm-use-drupal-ide
- Code Completion for CodeIgniter in phpStorm by Jeff Behnke:
- Why I love JetBrains phpStorm II by Simon:
  http://www.speich.net/articles/2011/05/15/why-i-love-jetbrains-phpstorm-ii/
Community

- *Get Involved into IntelliJ IDEA Community:*
  

- *Official forum:*
  
  [http://devnet.jetbrains.net/community/wi?view=discussions Official IRC channel: #idea-users channel at freenode](http://devnet.jetbrains.net/community/wi?view=discussions)

- *Twitter:*
  
  [https://twitter.com/intellijidea](https://twitter.com/intellijidea)

- *You track:*
  
Thank you for buying
Instant PhpStorm Starter

About Packt Publishing
Packt, pronounced ‘packed’, published its first book "Mastering phpMyAdmin for Effective MySQL Management" in April 2004 and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today’s systems, applications, and frameworks. Our solution based books give you the knowledge and power to customize the software and technologies you’re using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern, yet unique publishing company, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website: www.packtpub.com.

Writing for Packt
We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to author@packtpub.com. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.
Selenium 2 Testing Tools: Beginner’s Guide
Learn to use Selenium testing tools from scratch
1. Automate web browsers with Selenium WebDriver to test web applications
2. Set up Java Environment for using Selenium WebDriver
3. Learn good design patterns for testing web applications

Python Testing Cookbook
Over 70 simple but incredibly effective recipes for taking control of automated testing using powerful Python testing tools
1. Learn to write tests at every level using a variety of Python testing tools
2. The first book to include detailed screenshots and recipes for using Jenkins continuous integration server (formerly known as Hudson)
3. Explore innovative ways to introduce automated testing to legacy systems
4. Written by Greg L. Turnquist – senior software engineer and author of Spring Python 1.1

Please check www.PacktPub.com for information on our titles
Selenium Testing Tools Cookbook

Over 90 recipes to build, maintain, and improve test automation with Selenium WebDriver

1. Learn to leverage the power of Selenium WebDriver with simple examples that illustrate real world problems and their workarounds

2. Each sample demonstrates key concepts allowing you to advance your knowledge of Selenium WebDriver in a practical and incremental way

3. Explains testing of mobile web applications with Selenium Drivers for platforms such as iOS and Android

Web Services Testing with soapUI

Build high quality service-oriented solutions by learning easy and efficient web services testing with this practical, hands-on guide

1. Become more proficient in testing web services included in your service-oriented solutions

2. Find, analyze, reproduce bugs effectively by adhering to best web service testing approaches

3. Learn with clear step-by-step instructions and hands-on examples on various topics related to web services testing using soapUI

Please check www.PacktPub.com for information on our titles