Getting Started with PhpStorm. Exploring the IDE

Welcome to PhpStorm! This short guide aims to help you get a grip on the IDE.

Before You Start

If you are going to run and debug an application directly on a remote host, the only thing you need is register access to this host in PhpStorm to enable synchronization.

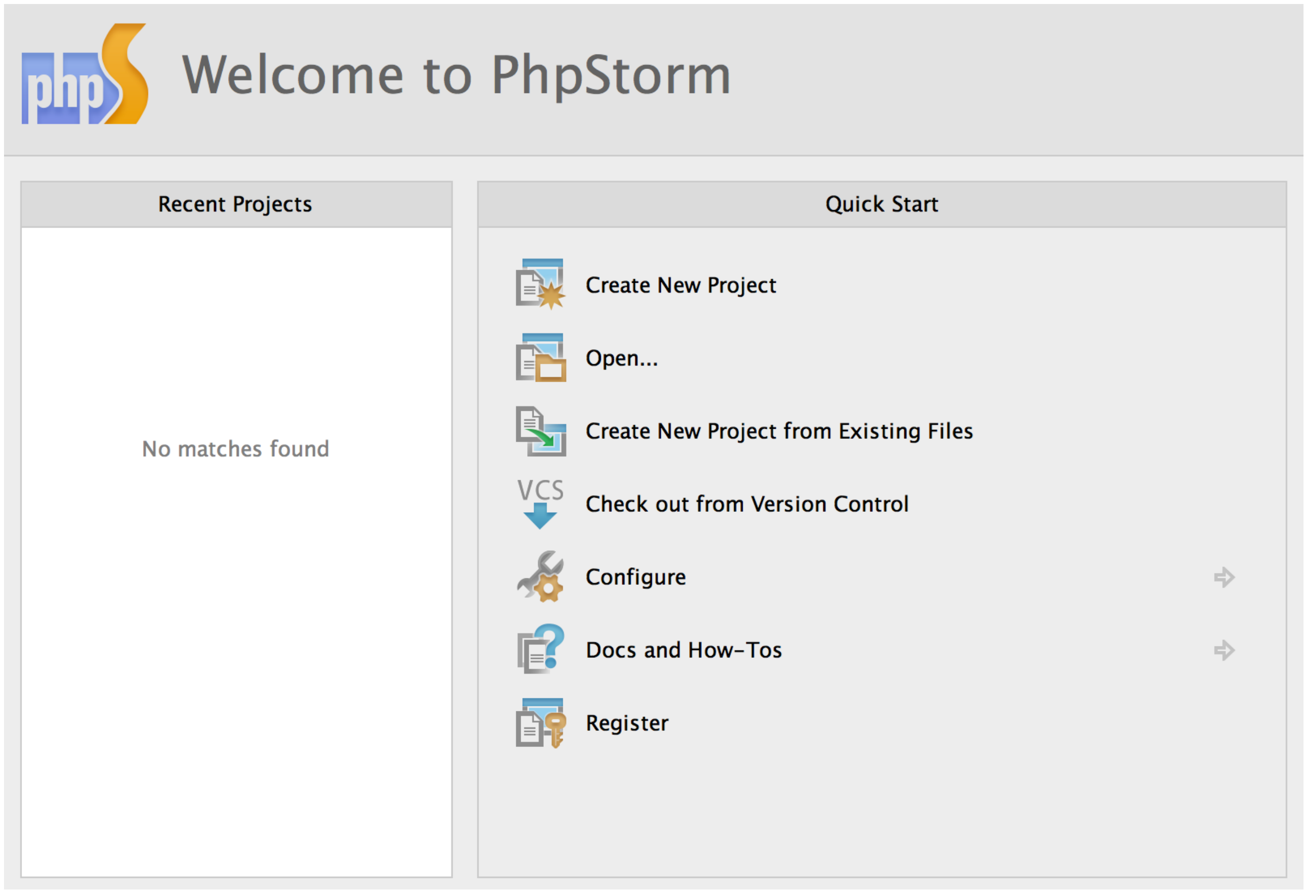
If you plan to launch the application locally, you need a PHP engine installed and registered in PhpStorm and a Web server installed, configured, and integrated with PhpStorm. You can install these components separately or use an AMP package (Apache, MySQL, PHP solution stack). PHP engine version 5.4 and higher comes with built-in web server, which is supported by PhpStorm.

PhpStorm is cross-platform and works on Windows, Mac OS X, and Linux.

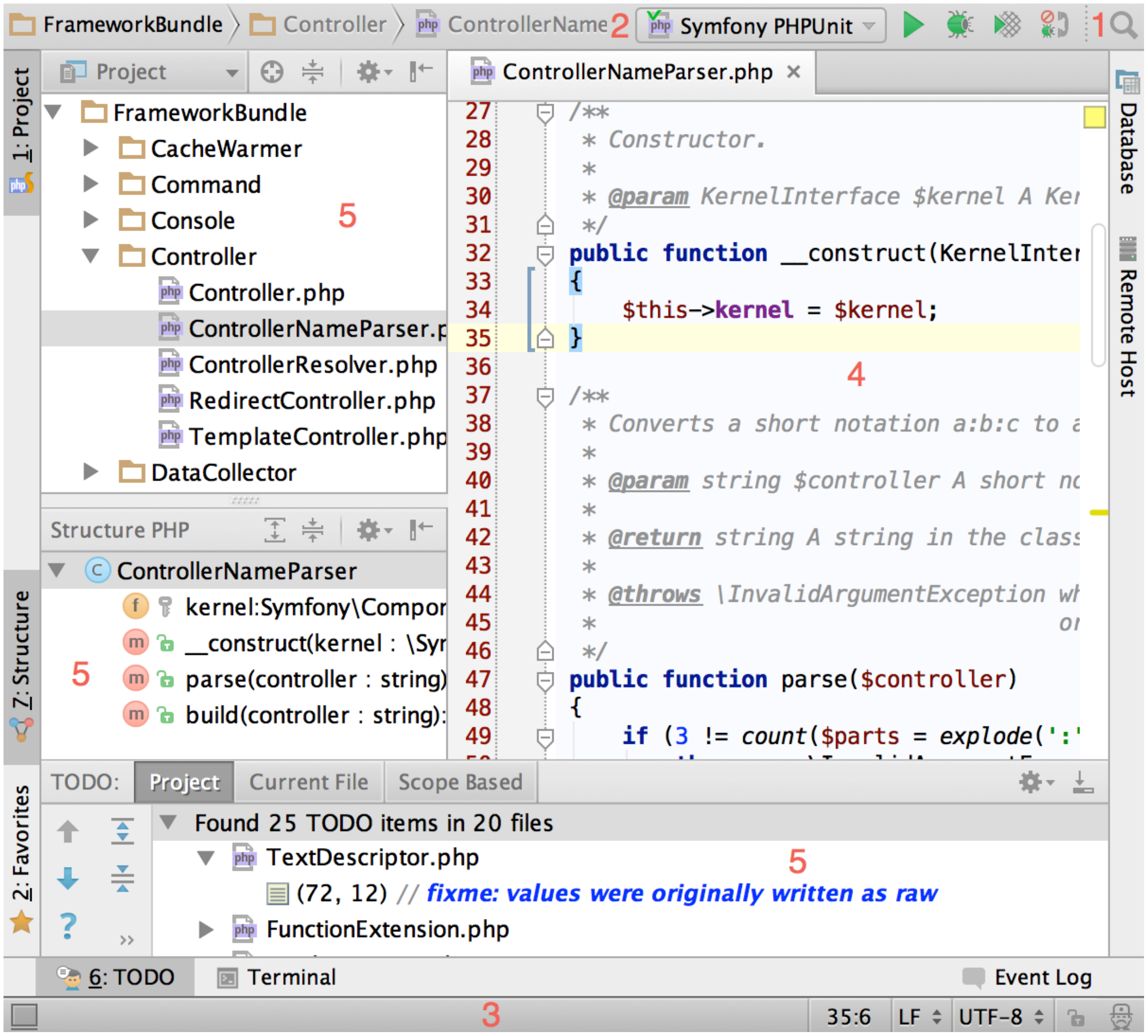
See also: Installing and configuring XAMPP
Installing and configuring MAMP

Look at the IDE

When you launch PhpStorm for the very first time or there is no open project, you see the Welcomescreen, which gives you the main entry points into the IDE (creating or opening a project, checking out a project from version control, viewing documentation, and configuring the IDE):



When a project is opened, you will see the main window divided into several logical areas:



```
/**
 * Constructor.
 *
 * @param KernelInterface $kernel A Ker
 */
public function __construct(KernelInter
{
    $this->kernel = $kernel;
}

/**
 * Converts a short notation a:b:c to a
 *
 * @param string $controller A short nc
 *
 * @return string A string in the class
 *
 * @throws \InvalidArgumentException wh
 *                                    or
 */
public function parse($controller)
{
    if (3 != count($parts = explode(':'
```

```
Found 25 TODO items in 20 files
    TextDescriptor.php
        (72, 12) // fixme: values were originally written as raw
    FunctionExtension.php
```

1. Menus and toolbars, which help you execute various commands.
2. Navigation bar for navigating through the project.
3. Status bar contains the various information about the entire IDE, the current project or file in the editor, information, warnings, and error messages.
4. The Editor, where you actually create your code.
5. Tool windows. They are numerous, and perform different functions: help you explore and navigate through the projects and file structures, view search and inspection results, run, debug and test applications, work in interactive consoles, and more.

Why Do We Need a Project?

Everything you do in PhpStorm, is done within the context of a project. A project represents a complete software solution and defines project-wide settings.
This makes the basis for coding assistance, bulk refactoring, coding style consistency, etc.

See also: Project

Import Your Code Into PhpStorm Project

With PhpStorm, you can set up a project in three ways:

- Download the sources from a remote host and arrange them in a PhpStorm project
- Clone a version control repository and create a project around the downloaded sources
- Start a project from scratch

Create a Project from Downloaded Sources

One of the most widely used workflows is updating an already existing application. In this case you need to download the application sources and arrange them in a PhpStorm project.

1. First of all, configure access to the remote host where the sources are located (Tools | Deployment | Configuration).
2. Start the New Project from Existing Sources Wizard (File | New Project from Existing Files, then specify the method to access the remote host (FTP/SFTP/FTPS, local or mounted folder).

See also: Creating a project from downloaded sources

Create a Project Around the Sources from a Version Control Storage

You can also download sources from VCS storages and repositories such as GitHub.

Choose VCS | Checkout from Version Control | <your vcs>). When asked by PhpStorm, type your credentials to access the storage.

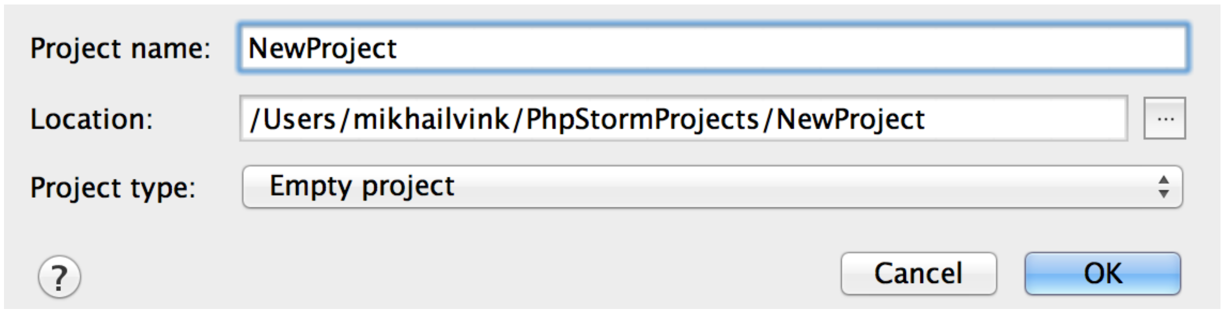
See also: Cloning a repository from GitHub
Setting up a local Git repository
Checking out files from an SVN repository

Start a Project from Scratch

You can start your project from scratch developing an application from the very beginning.

1. On the Welcome screen, click “Create New project”.

2. In the “Create New Project” dialog box that opens, enter the project properties such as project name, parent folder and project type.



Project name: NewProject

Location: /Users/mikhailvink/PhpStormProjects/NewProject

Project type: Empty project

3. Next, when the project opens, let’s create a PHP file in it. To do that, place the caret at the project root directory in the Project tool window, point to New on the context menu, choose PHP File, and name it MyFile in the Create New PHP File dialog that opens.
4. In the new file that opens in the editor, type your code and save the file by pressing `Ctrl+S`.

See also: Creating a project from scratch

VCS

Surely, you keep your sources under version control, right? Git? SVN? Mercurial? With PhpStorm, it’s easy to set up, just click the Version Control node in the Settings dialog (File | Settings | Version Control). If your project is set up around a cloned repository, PhpStorm already knows which VCS is applied and automatically puts the downloaded sources under control of this system. You can also define the behavior of VCS in the parts that are common for all of them, such as: confirmation on creating or deleting files, tasks performed in the background, ignoring unversioned files and more.

Local history

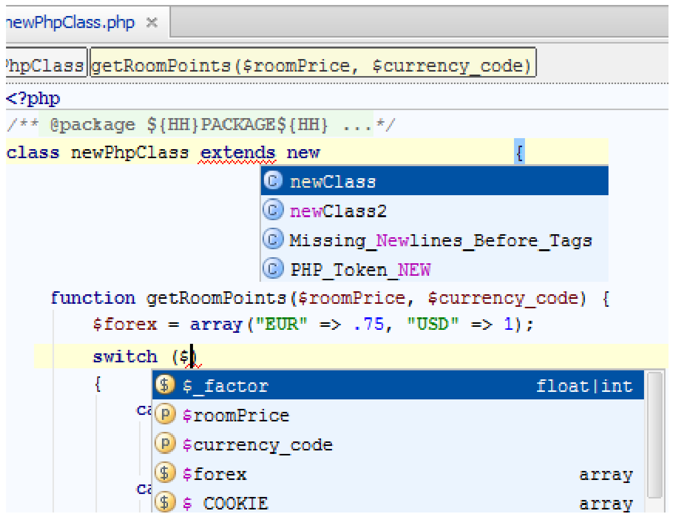
In addition to a traditional version control system you can use local history. With Local History, PhpStorm automatically tracks changes you make to the source code, results of refactoring, and state of the source code based on a set of predefined events (testing, deployment, commit or update). Local history is always enabled.

Write Code Smartly

What makes PhpStorm stand out from the numerous IDEs, is its full-featured editor. Whatever you do for developing your source code, PhpStorm is always at hand, helping you create error-freeapplications. Here is a brief outline of the smart PhpStorm’s coding assistance:

- At every stage of development, use code completion (`Ctrl+Space`), which takes into account the current context.

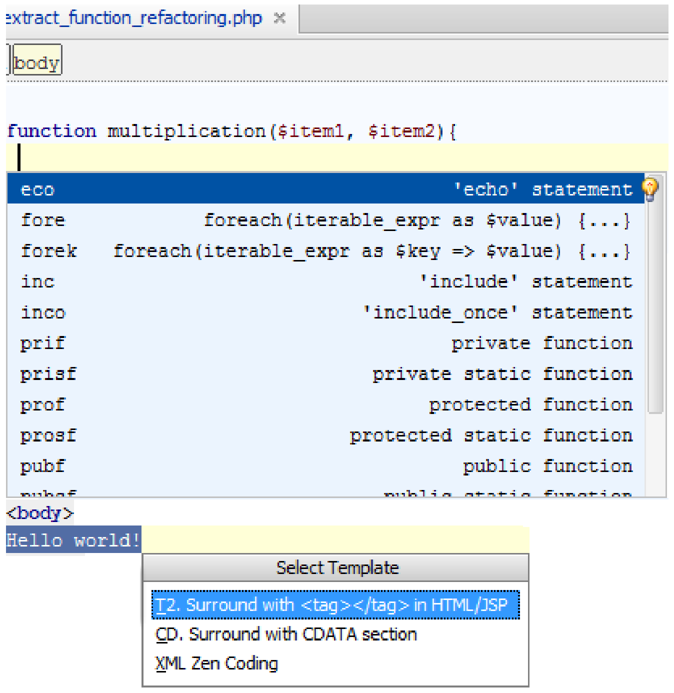
For example, depending on the place where you invoke code completion, you can complete keywords or code blocks, infer types, or complete methods and properties.



```
newPhpClass.php ×
hpClass getRoomPoints($roomPrice, $currency_code)
<?php
/** @package ${HH}PACKAGE${HH} ...*/
class newPhpClass extends new                    {
                              newClass
                              newClass2
                              Missing_Newlines_Before_Tags
                              PHP_Token_NEW
    function getRoomPoints($roomPrice, $currency_code) {
        $forex = array("EUR" => .75, "USD" => 1);
        switch ($)
        {           $_factor                     float|int
            ca      $roomPrice
                    $currency_code
                    $forex                           array
            ca      $_COOKIE                         array
```

- Use live templates/snippets (choose Code | Insert Live Template or press `Ctrl+J`) and surround templates (choose Code | Surround with Live Template or press `Ctrl+Alt+J`) to produce entire code constructs. PhpStorm comes with a wide range of ready-to-use live templates, or snippets, which you can explore in theSettings dialog (File | Settings | Editor |Live templates). If you see that you are lacking something especially important for your development goals, extend this set of snippets with your own ones.

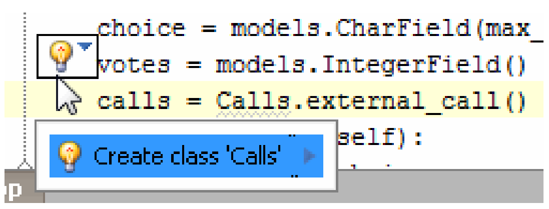
  Don’t also miss the possibility to surround with complete code constructs (chooseCode | Surround With or press `Ctrl+Alt+T`).



```
extract_function_refactoring.php ×
body
function multiplication($item1, $item2){
|
eco                                     'echo' statement
fore            foreach(iterable_expr as $value) {...}
forek   foreach(iterable_expr as $key => $value) {...}
inc                                  'include' statement
inco                            'include_once' statement
prif                                    private function
prisf                            private static function
prof                                  protected function
prosf                          protected static function
pubf                                     public function
<body>
Hello world!
```

Select Template
T2. Surround with <tag></tag> in HTML/JSP
CD. Surround with CDATA section
XML Zen Coding

- Almost like a pair programmer, PhpStorm keeps an eye on what you are currently doing, and comes up with smart suggestions, or intention actions, which are marked with a light bulb sign. If you want to know exactly what is there under the light bulb, click it, or press `Alt+Enter`. This way you can, for example, auto-create a new method that you are already using in the code, but have not yet declared.



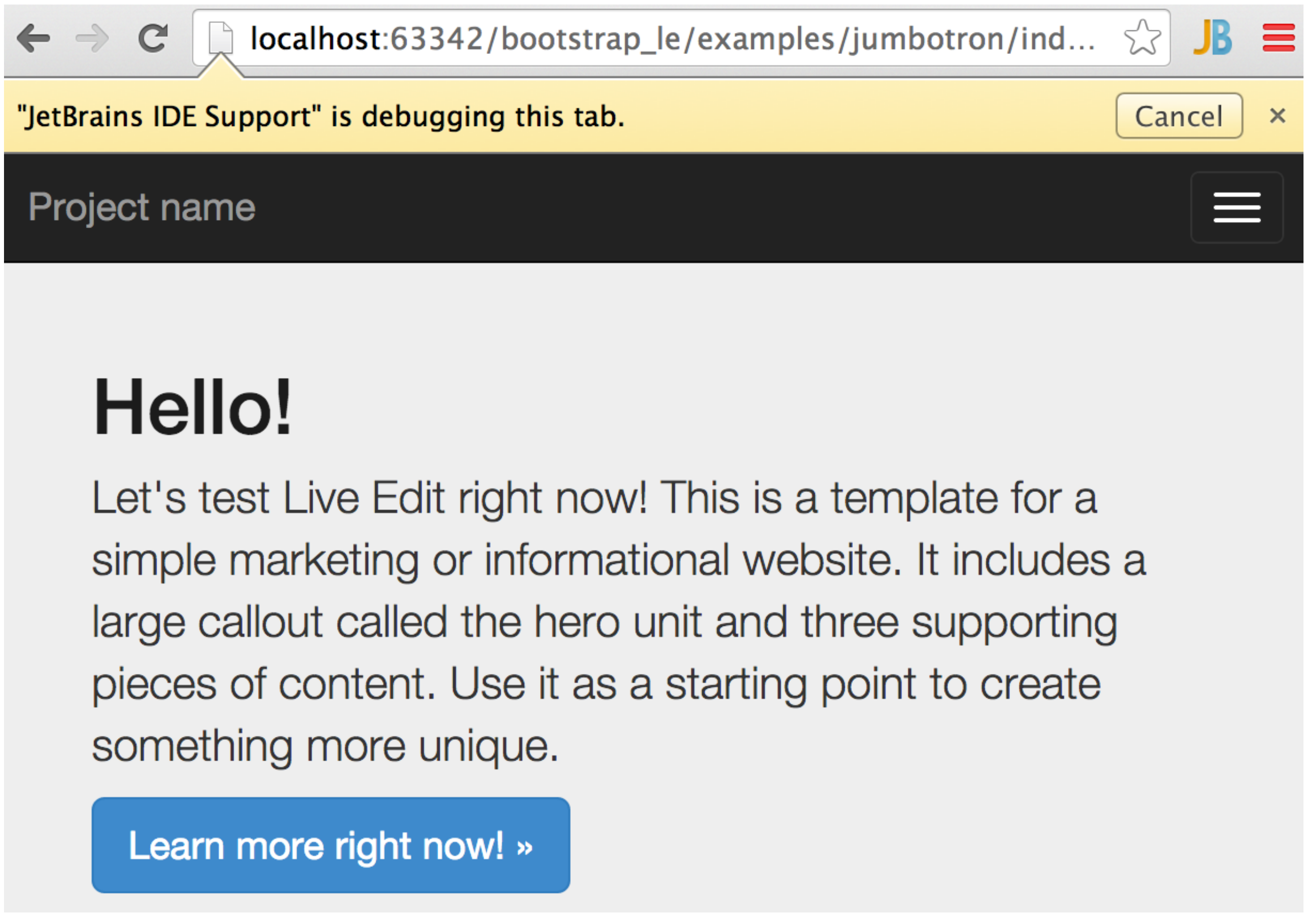
```
choice = models.CharField(max_
votes = models.IntegerField()
calls = Calls.external_call()
                          self):
```

View Your Changes Instantly

You can open an HTML, JavaScript, or CSS file in the browser and monitor how the changes you make to it in the editor are rendered without refreshing the page and even without leaving PhpStorm. By the way, completion look-up is also live. As you browse through the suggestion list, the page in the browser changes its appearance just as if you have already accepted the current suggestion.

There are some prerequisites to working with Live Edit. The Live Edit plugin should be installed inFile | Settings | Plugins | Install JetBrains Plugins | Live Edit | Install Plugin.

Live Edit works using a browser plugin, and currently only Google Chrome, Chrome Canary and Dartium are supported. The Chrome extension will be installed automatically if the Chrome Browser is not running during the first launch of the IDE with the Live Edit plugin installed. The Google Chrome Extension JetBrains IDE Support is available in Chrome Web Store.



localhost:63342/bootstrap_le/examples/jumbotron/ind...

"JetBrains IDE Support" is debugging this tab.

Hello!

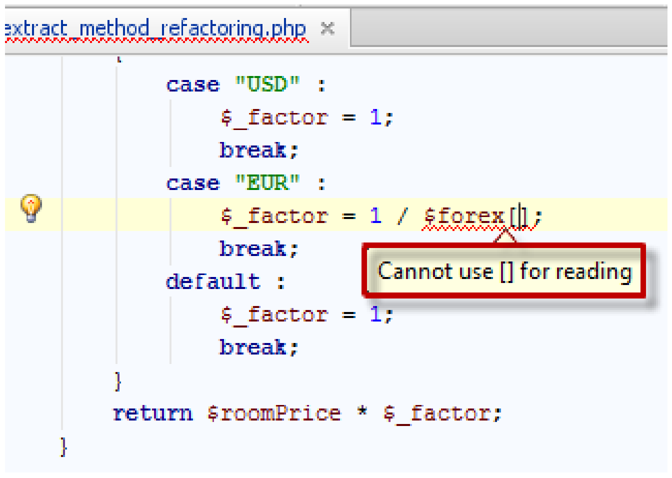
Let's test Live Edit right now! This is a template for a simple marketing or informational website. It includes a large callout called the hero unit and three supporting pieces of content. Use it as a starting point to create something more unique.

For Live Edit to work correctly, you should run a JavaScript Debug Session or Node.js Debug Configuration. In order to start a JavaScript Debug Session, a Run/Debug configuration has to be created first. This can be done automatically using the *Debug file_name* context menu, or manually if you want to specify additional options. Alternatively, open the page in Chrome by selecting *Open in Browser* from the context menu of the html file. When the page opens, select *Inspect in PhpStorm*from the context menu.

You can also refresh the page and view the changes without leaving PhpStorm, just choose Run | Reload in Browser on the main menu when the JavaScript Debug Session is active.

See also: Live Edit in PhpStorm tutorial

Analyze Code Transparently



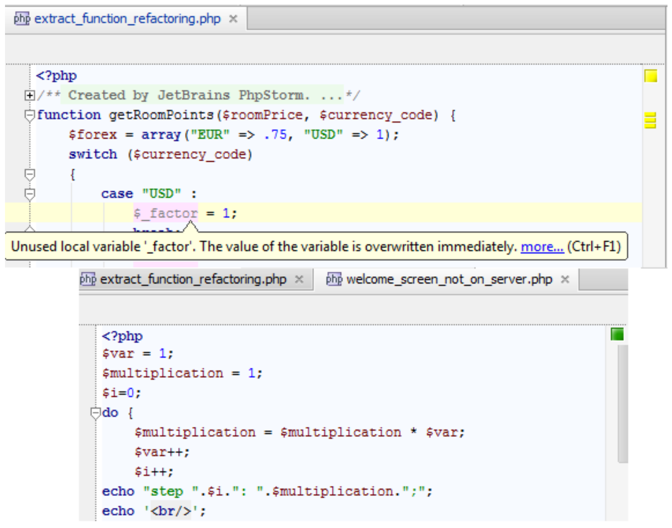
```
        case "USD" :
            $_factor = 1;
            break;
        case "EUR" :
            $_factor = 1 / $forex[];
            break;
        default :
            $_factor = 1;
            break;
    }
    return $roomPrice * $_factor;
}
```

Cannot use [] for reading

PhpStorm gives you numerous hints and prompts to help you avoid errors, or correct them, if they occur.

First, as you type, you immediately have all syntactical errors overlined with a wavy line. If you place the caret at an error location, you will see a brief description of the problem at the tooltip, and also in the left side of the Status bar.

The next level is static code analysis, or code inspections: your code is analyzed without actually executing it. Actually, PhpStorm inspects code in the current file on-the-fly, and shows inspection results in the marker bar as colored stripes. If you see that the right side of your IDE frame is bright with red stripes, beware — it means that your code contains serious errors. Less important things, recommendations to improve code, or warnings, are displayed as yellow stripes. Information for the current file is summarized in a colored indicator on top of the marker bar, which works as traffic lights.


extract_function_refactoring.php

```
<?php
/** Created by JetBrains PhpStorm. ...*/
function getRoomPoints($roomPrice, $currency_code) {
    $forex = array("EUR" => .75, "USD" => 1);
    switch ($currency_code)
    {
        case "USD" :
            $_factor = 1;
```

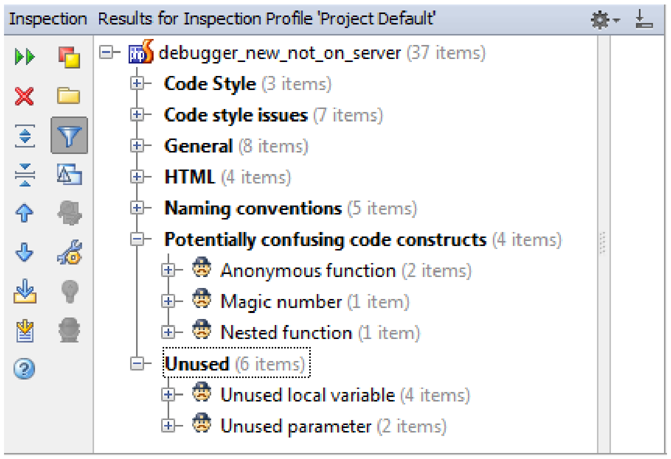
Unused local variable '_factor'. The value of the variable is overwritten immediately. more... (Ctrl+F1)

extract_function_refactoring.php
welcome_screen_not_on_server.php

```
<?php
$var = 1;
$multiplication = 1;
$i=0;
do {
    $multiplication = $multiplication * $var;
    $var++;
    $i++;
echo "step ".$i.": ".$multiplication.";";
echo '<br/>';
```

However, you might want to look deeper into the code of your application. In this case, you have to inspect a whole project, or any of its parts (Code | Inspect Code), and explore results in the Inspection tool window.

PhpStorm comes with a wide range of pre-defined inspections; you can familiarize yourself, and configure them in theInspections page of the Settings dialog.



```
Inspection  Results for Inspection Profile 'Project Default'
debugger_new_not_on_server (37 items)
    Code Style (3 items)
    Code style issues (7 items)
    General (8 items)
    HTML (4 items)
    Naming conventions (5 items)
    Potentially confusing code constructs (4 items)
        Anonymous function (2 items)
        Magic number (1 item)
        Nested function (1 item)
    Unused (6 items)
        Unused local variable (4 items)
        Unused parameter (2 items)
```

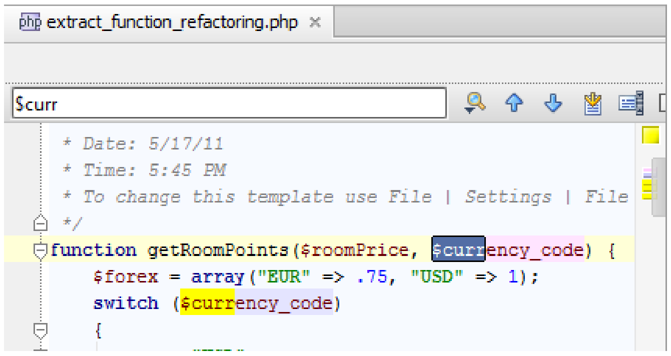
Find Your Way Through

Navigation facilities of PhpStorm fall into these major areas:

- Navigating within the source code
- Jumping between the IDE components

Source Code

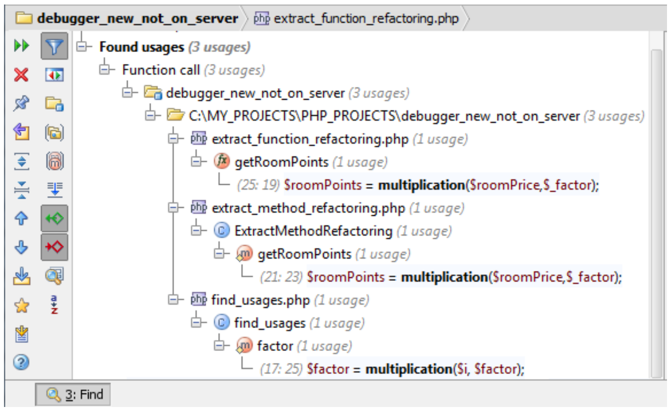
Let's start with finding fragments of text. One of the most basic means of navigation and search in the source code is our good old `Ctrl+F` command: start typing your search string, and get immediately to its occurrences in the current file.



```
extract_function_refactoring.php ×
$curr
 * Date: 5/17/11
 * Time: 5:45 PM
 * To change this template use File | Settings | File
 */
function getRoomPoints($roomPrice, $currency_code) {
    $forex = array("EUR" => .75, "USD" => 1);
    switch ($currency_code)
    {
```

But PhpStorm goes further, and helps you look for a certain string within a directory, or any arbitrary scope, or an entire project (`Ctrl+Shift+F`).

Besides that, PhpStorm suggests a more sophisticated approach, namely search for usages. For example, if you want to navigate from a symbol to one of its usages within your application, press `Alt+F7`, or choose Find Usages on its context menu.

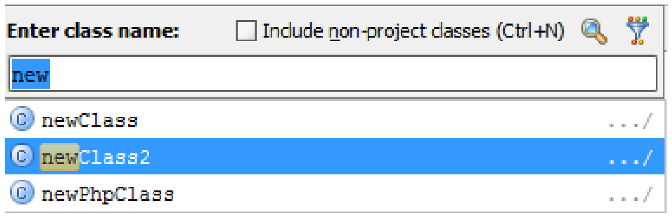


```
debugger_new_not_on_server > extract_function_refactoring.php
Found usages (3 usages)
    Function call (3 usages)
        debugger_new_not_on_server (3 usages)
            C:\MY_PROJECTS\PHP_PROJECTS\debugger_new_not_on_server (3 usages)
                extract_function_refactoring.php (1 usage)
                    getRoomPoints (1 usage)
                        (25: 19) $roomPoints = multiplication($roomPrice,$_factor);
                extract_method_refactoring.php (1 usage)
                    ExtractMethodRefactoring (1 usage)
                        getRoomPoints (1 usage)
                            (21: 23) $roomPoints = multiplication($roomPrice,$_factor);
                find_usages.php (1 usage)
                    find_usages (1 usage)
                        factor (1 usage)
                            (17: 25) $factor = multiplication($i, $factor);
3: Find
```

Then specify the search scope and view the detected occurrences in and Find tool window.

Actually, there are several commands that help you find out where a certain symbol is used: you can jump from one usage to another in the current file (`Ctrl+F7`), view usages in the current file color-coded (`Ctrl+Shirt+F7`), or across a whole project in a popup list (`Ctrl+Alt+F7`).
If you want to jump from a symbol to its declaration, just middle-click its name, or press `Ctrl+B`.

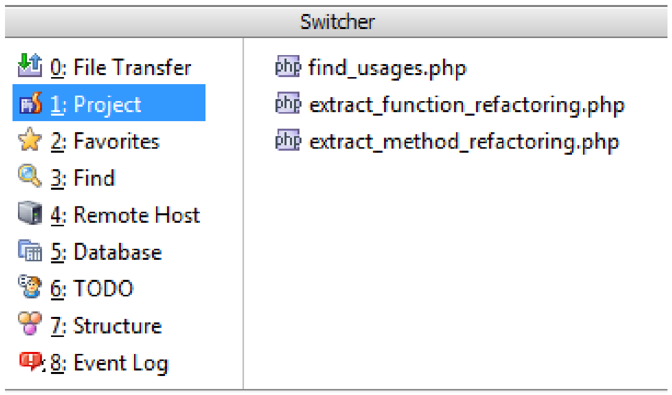
To quickly find an element by name and open it in the editor, use navigation pop-up: press `Shift+Shift` (for Search Everywhere), `Ctrl+N` (for a class), `Ctrl+Shift+N` (for a file), or `Ctrl+Shift+Alt+N` (for a symbol), and start typing the name you are looking for. The list of matching names shrinks as you type, but this is just one of the handy facilities: you can use the asterisk wildcard, all caps for CamelHumps, or spaces for snake_case names, slashes for nested folders and so on, and so forth.


Enter class name: ☐ Include non-project classes (Ctrl+N)
new
newClass .../
newClass2 .../
newPhpClass .../

Quick search is the easiest way to find a file: with the Project tool window having the focus, you just start typing, and see how the matching nodes are highlighted in the tree view. Actually, this method of finding files or symbols works in any tool window.

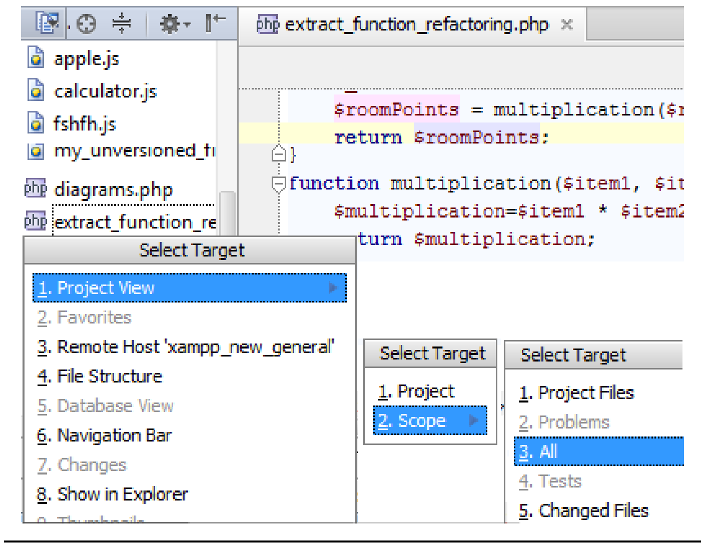
IDE Components

Ways to navigate across the IDE are numerous, and we'll briefly outline just some of them. Let's start with the switcher: press `Ctrl+Tab` to show the switcher, which is a list of the PhpStorm's tool windows and open files, and then, keeping the `Ctrl` key pressed, use Tabor arrow keys to scroll to the component you want to go to.



If you select a file in one of the IDE components, and want to view it in another one (the editor, Project view, Navigation bar, or a changelist), then use Select Target(`Alt+F1`).

And finally, don't forget that pressing `Esc` will bring you back to the editor, wherever you are!



```
    $roomPoints = multiplication($r
    return $roomPoints;
}
function multiplication($item1, $it
    $multiplication=$item1 * $item2
      turn $multiplication;
```

Run and Debug Your Application

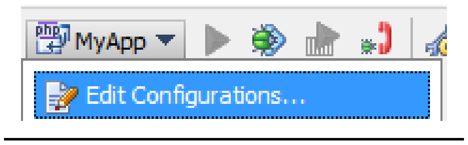
The easiest way to debug your web application is initiating a debugging session from the browser and working in the zero-configuration debugging mode, provided that you have created the corresponding debugger cookies and enabled control over debugging through them. We recommend starting debugging using this method. To connect to the running session, click the Start Listen PHP Debug Connections button on the toolbar. Refer to the tutorial for more information.

No preliminary steps are required if you are going to run and debug an application directly on a remote host. The only thing you need is register access to this host in PhpStorm to enable synchronization.

If you are going to run and debug an application on your computer, you need to configure local PHP Development Environment.

With PhpStorm, you can run entire PHP applications as well as particular classes or files. To run a class or a file, open it in the Editor or select it in the Project view,
then choose Run <file name> on the context menu of the selection o just press `Ctrl+Shift+F10` .

To launch an entire application you need a special profile, or a run/debug configuration, which represents a set of run/debug start-up properties. Run configurations are created in theRun/Debug Configurations dialog.

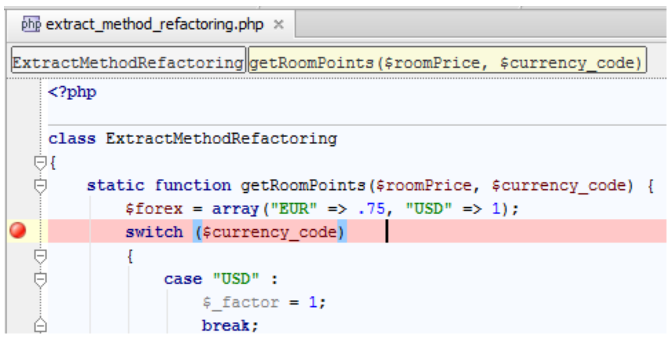


Depending on how where you want your application to run and where you want to view its results, choose the configuration type:

- PHP Web Application to view application output in a browser.

- PHP Script to view the application output in the Run tool window.
- Built-in Web server

Your application or script runs into a run-time error? To find out its origin, you will have to do some debugging. It starts with placing breakpoints (just click the left gutter of the line where you want the breakpoint to appear), at which program execution will be suspended, so you can explore program data:


extract_method_refactoring.php
ExtractMethodRefactoring
getRoomPoints($roomPrice, $currency_code)

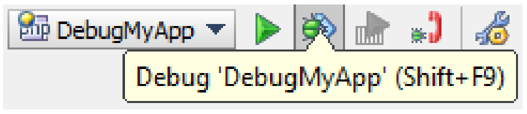
```
<?php

class ExtractMethodRefactoring
{
    static function getRoomPoints($roomPrice, $currency_code) {
        $forex = array("EUR" => .75, "USD" => 1);
        switch ($currency_code)
        {
            case "USD" :
                $_factor = 1;
                break;
```

PhpStorm supports integration with Xdebugand Zend Debugger. Before you start a debugging session, you need to download, install and configure one of these tools.

To initiate a debugging session from PhpStorm, you will need a debug configuration:

- PHP Web Application for entire applications.
- PHP HTTP Request for separate pages, for example, when you need to “come” to a specific page with certain data.

To start debugging, click the Debug button on the toolbar or pressing `Shift+F9`.



And last but not the least, you can also debug JavaScript in Google Chrome and Firefox right from PhpStorm, which is delivered with a built-in debugger. Create debug configuration of the type JavaScript Debug and click the Debug button on the toolbar.

See also:
- Zero-configuration Web Application Debugging with PhpStorm
- Configuring PHP Development Environment
- Running PHP Applications
- Debugging PHP Applications
- Running and Debugging JavaScript

Test Your Application

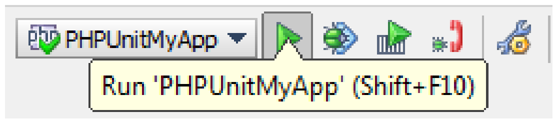
PhpStorm supports unit testing of PHP applications through integration with the PHPUnit tool. Before you start unit testing:

1. Install and configure this tool depending on the operating system you use and your system settings.

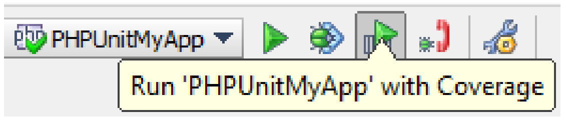
2. Integrate the tools with PhpStorm on the PHP page of the Settings dialog.

To generate a stub of a test class, select the PHP class to create unit tests for and choose Generate PHPUnit Test.

Appoint the tests to run in a run configuration of the type Run/Debug Configuration: PHPUnit or Run/Debug Configuration. PHPUnit on Server, depending on whether you want to run tests locally or on the server. To run the tests, select the relevant configuration and click the Run button on the toolbar, or just press `Shift+F10`.



To start monitoring code coverage, select the relevant configuration and click the Run with coverage button on the toolbar.



See also: Testing PHP Applications
Monitoring Code Coverage for PHP Applications

Customize Everything!

Look at the main toolbar — there is a Settings button. Clicking this button opens the Settingsdialog box, where you can change your project structure, set up version control, and tweak your working environment to make the development process a real pleasure. Some of the settings pertain to a particular project — for example, project structure, version control configuration, or file colors. Others — such as the Editor settings, keymaps, or live templates — pertain to your whole working environment, and thus can be configured even without an open project.

Editor

The whole bunch of pages under the Editor node (File | Settings | Appearance & Behavior) helps you tune every aspect of the editor's behavior. Note that PhpStorm comes with the pre-defined color scheme, but if you want to make up something very personalized, you are welcome to do it.

Coding style

PhpStorm contains a set of bundled predefined coding styles. Applying them helps your code meet the Drupal, Zend, PEAR, PSR1/PSR2, Symfony2, and other language-specific coding standards. To appoint a predefined coding style to a language, click the link Set From in the upper-right corner and choose the language or framework to copy code style from in the drop-down list, that appears.

Coding style is defined in the dedicated language page under the Code Style node of the Settingsdialog box.

Appearance

This is the page where you can select “look and feel” of your PhpStorm installation (File | Settings |Appearance & Behavior | Appearance). Just click the Theme drop-down, and select the scheme you like better. You don’t need to close the Settings dialog to observe the results of your experiments: click Apply, see what happens, and close the dialog when you are quite happy with what you’ve got.

File Colors

You project might contain several sites, each one with its own set of files with the same names. When they are opened in the editor, it’s rather confusing... PhpStorm helps make them different by painting their editor tabs (File | Settings | Appearance & Behavior | File Colors). You have to break down your project into smaller chunks — scopes (for example, a scope per site), and select a color for each one.

Keymap

The set of keyboard shortcuts you work with is one of your most intimate habits — your fingers “remember” certain combinations of keys, and changing this habit is rather painful. PhpStorm supplies you with a great default keymap making your coding really productive and convenient, however, you can change it to another one (File | Settings | Appearance & Behavior | Keymap). There are some pre-defined keymaps, moreover, you can create your own keymap on the base of an existing one.

See also: PhpStorm Default Keymap

That’s it Folks!

Here we have given a very concise overview of some vital PhpStorm facilities, just to give you a quick start. There are numerous important features that make developer’s life nice and easy, and the source code nice and clean. Try now these primary steps, and then dig deeper. Enjoy!

We realize you may still have questions. We welcome you to ask them on PhpStorm Discussion Forum.