PHP Code Refactorings in PhpStorm

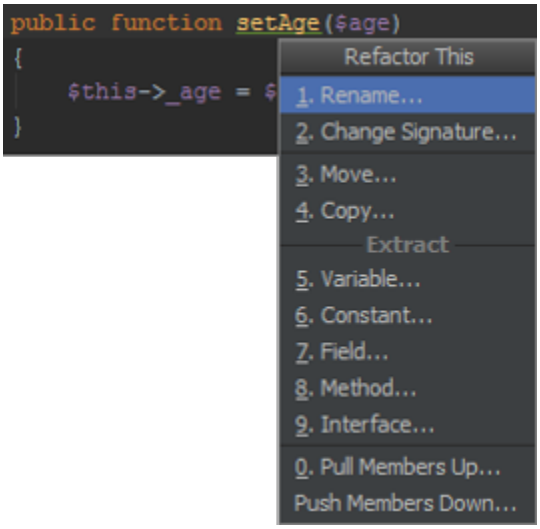
As Martin Fowler says: "Refactoring is a controlled technique for improving the design of an existing code base." Refactorings can be a simple rename of a variable or extracting an interface from a class. PhpStorm provides many refactorings for the various languages it supports and carries them out automatically, making sure existing code is updated and will not break. In this tutorial, we will look at the available refactorings for PHP code.

While this tutorial covers refactorings for PHP, the IDE also supports refactoring other languages like JavaScript, HTML and so forth. Give refactoring a try in these languages as well; the Refactor This is available in the editor for them, too.

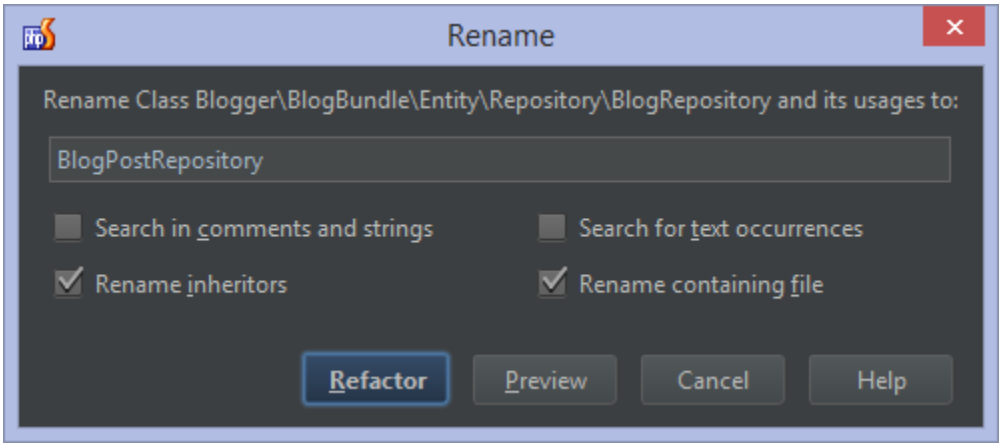
Refactoring in PhpStorm

Before we dive in to all available refactorings, let's see what a typical refactoring looks like. They all do different things, but there are some things they have in common. Let's try renaming a class in our codebase!

To do this, we can use the Refactor | Rename... context menu on any file or symbol, or place the cursor on it and use the Ctrl+F6 keyboard shortcut to invoke the rename refactoring immediately. Note that we could also use the Refactor This action. This last one is very convenient to invoke refactorings as we only need to remember one keyboard shortcut, Ctrl+Shift+Alt+T (Ctrl-T on Mac OS X), to show a pop-up with the different refactorings that we can do.



Whichever route we take, PhpStorm will show us the following dialog:

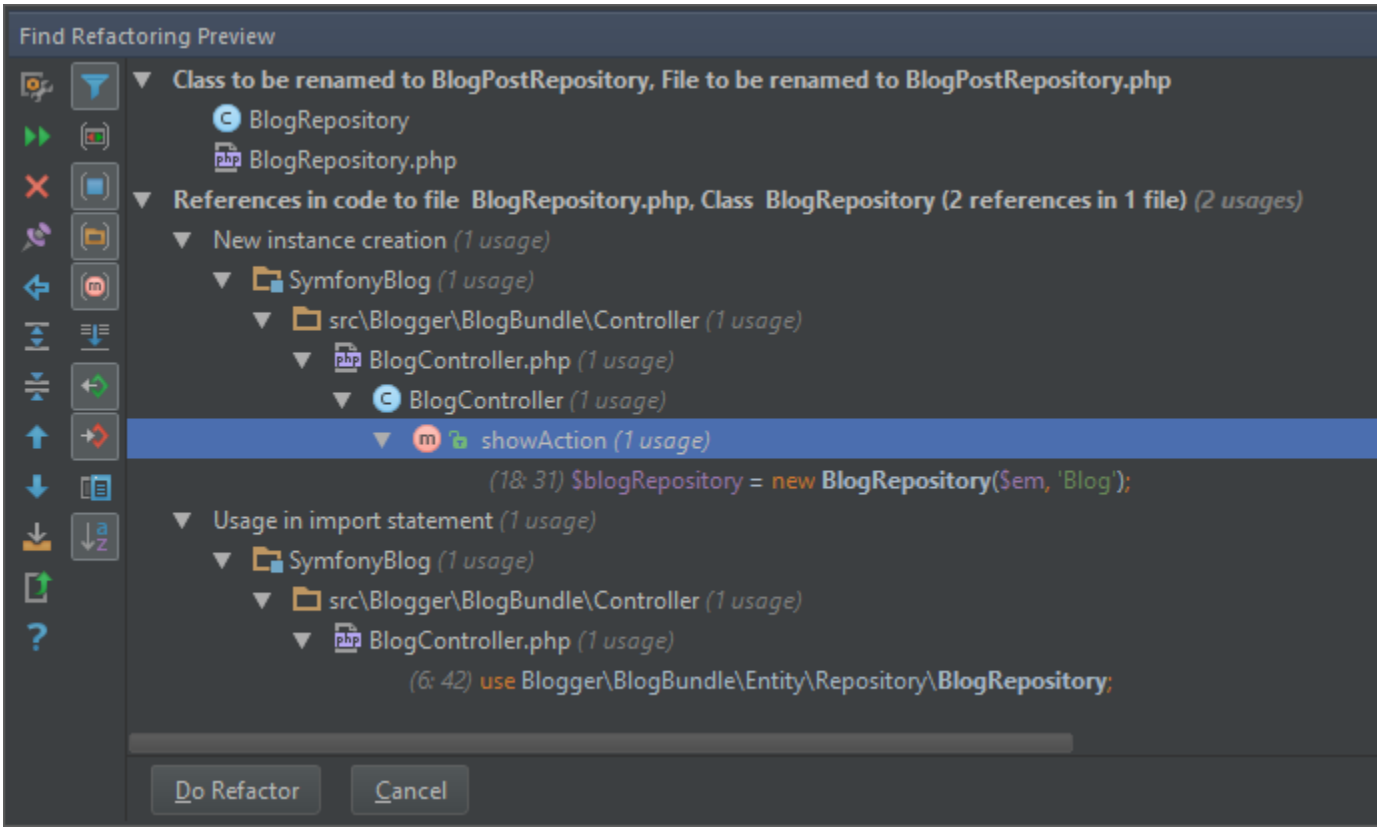


For every refactoring, we will get a different dialog in which we can provide the options for it. In this case, we have to provide the new name for the class we're about to rename. Some of the options in this dialog will be available for other refactorings as well: we can (optionally) search in comments and strings and do some sort of find-replace in there, we can search for text occurrences and so on.

When invoking a refactoring in PhpStorm, the IDE will:

- Perform the refactoring
- Track down and correct the affected code references automatically
- Warn about occurrences it cannot update automatically

If you are unsure about the outcome, consider clicking the Preview button before carrying out the refactoring. It will open the Find Refactoring Preview Tool Window and show us all of the actions the refactoring will perform.



In this case, it will:

- Rename the containing file
- Update one of the places where our code uses new and instantiates the class we're renaming
- Update one of the namespace import statements

We can filter and search in this tool window, and optionally select one of the occurrences and use Delete to have PhpStorm ignore that one whenever we perform the refactoring. This may come in handy when we're also searching and replacing in comments and strings. Usually we will not exclude actual code refactorings.

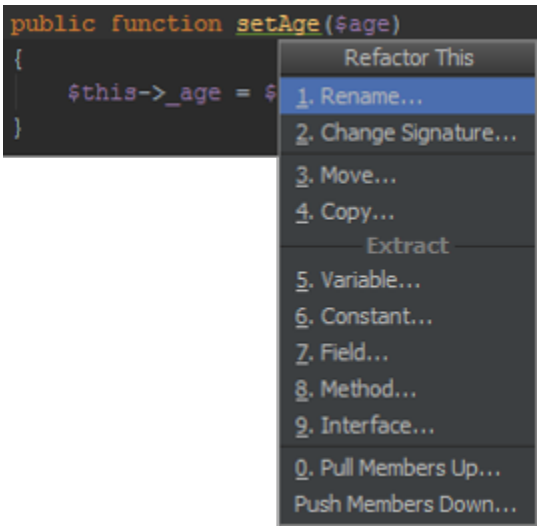
Clicking Do Refactor will perform the refactoring and update our project's codebase.

Ever wondered why we can not just rename or move a file in the IDE and be done with it? The reason is that it considers this as a refactoring: it changes the way the project is structured and potentially affects the source code and conventions of the programming language or framework being used. To make sure these conventions are met in the best way possible, the IDE requires us to invoke a rename (or move) refactoring to safely perform this action.

The Refactor This Action

While most refactorings in PhpStorm have their own shortcuts, we may not know all of them by heart. We may also be unfamiliar with the various refactorings available for a given file or symbol. And that's where the Refactor This action comes in handy!

In the Project View, Structure Tool Window, Editor or a UML Class Diagram, we can place the cursor on any file or symbol and use the Refactor | Refactor This context menu or press Ctrl+Shift+Alt+T (Ctrl-T on Mac OS X). This will display a pop-up with the different refactorings that we could carry out.



Using the up/down keys and Enter (or the numeric identifier for each entry in the pop-up), we can invoke the refactoring.

Available Refactorings

There are many refactorings available in the IDE. In this section, let's go over them and see what actions they perform. For every refactoring, we'll add a link to the PhpStorm Web Help, which contains a full description of all available options for a given refactoring.

All refactorings are available from the Refactor context menu or via their respective keyboard shortcuts.

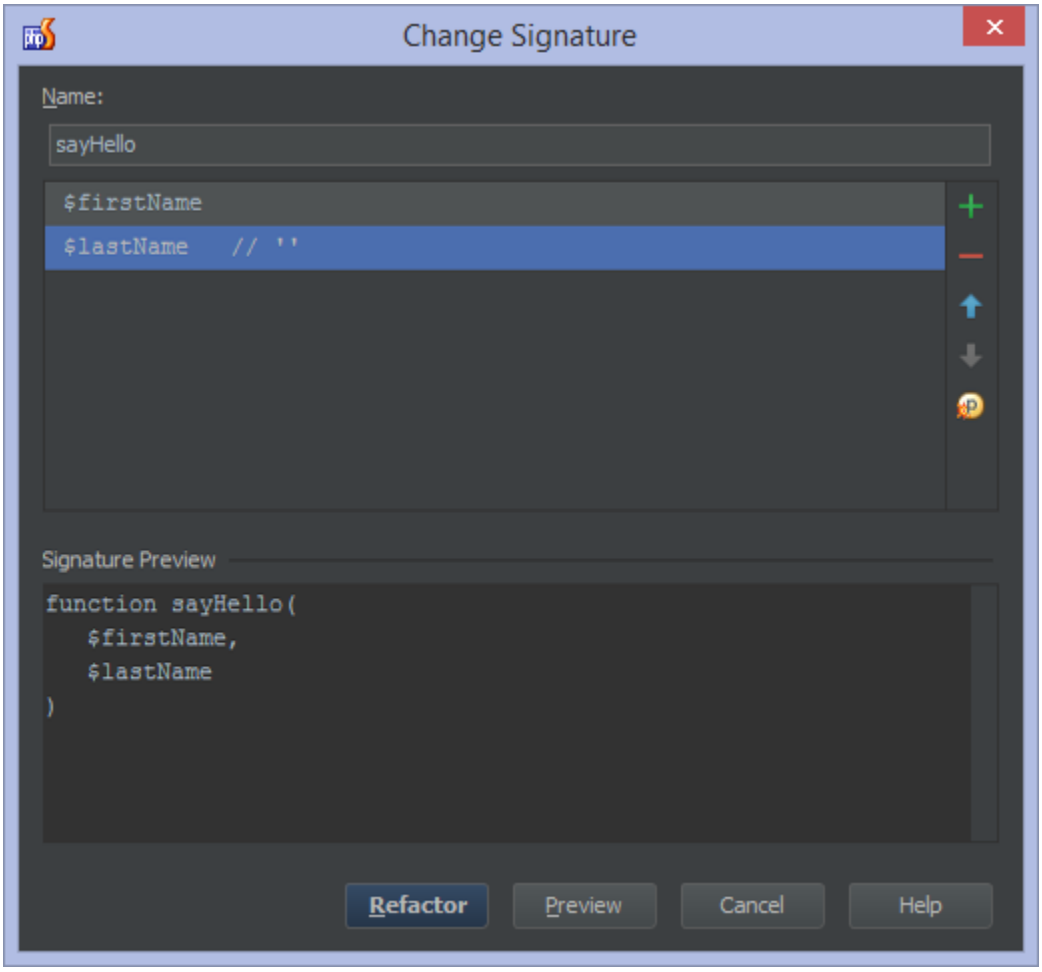
Check out the Hands-on PhpStorm workshop materials to download a sample project in which all of these refactorings can be tried.

Change Signature Refactoring

When we want to change the name of a function or add/remove parameters for it, we can use the Change Signature refactorin g by pressing Ctrl+F6 (or CMD-F6 on Mac OS X). It can:

- Change the function name
- Add new parameters and remove the existing ones
- Assign default values to the parameters
- Reorder parameters
- Change parameter names

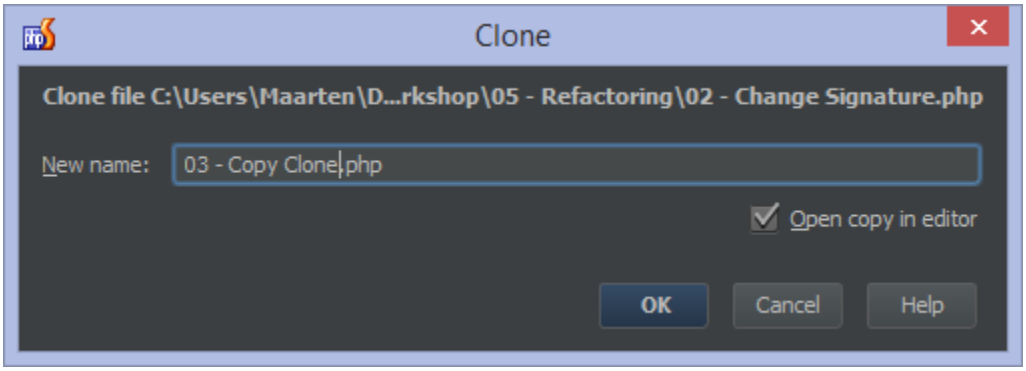
- Propagate new parameters through the function call hierarchy



Check Web Help for full details.

Copy/Clone Refactoring

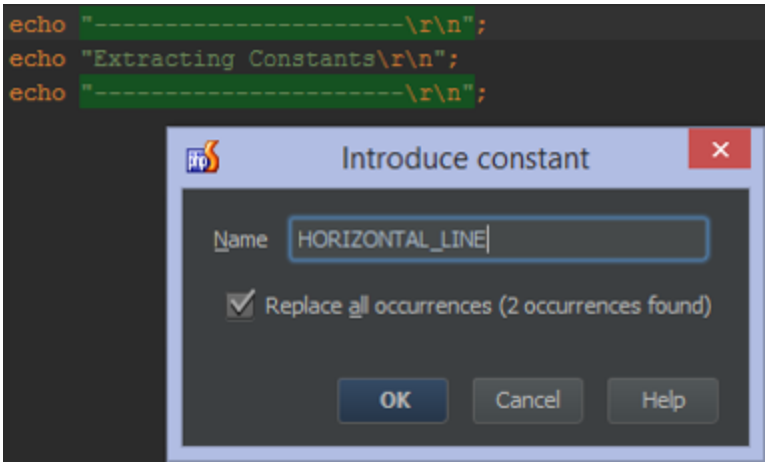
Using the Copy/Clone Refactoring, we can copy a class, file or directory to another directory, or clone it within the same directory. We can invoke it by selecting a file and using the keyboard shortcuts (F5 to copy, Shift+F5 to clone) or by dragging the file and dropping it in a folder whilst we hold the Ctrl key down.



Check Web Help for full details.

Extract Constant Refactoring

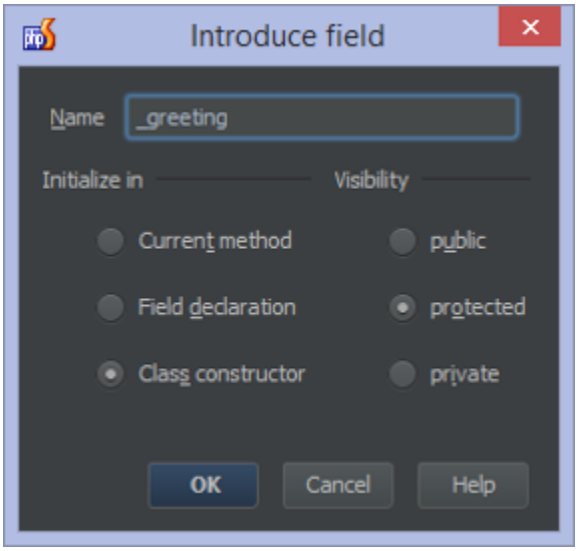
Using the Extract Constant Refactoring, we can select a value in the editor and extract it into a constant (use Ctrl+Alt+C or Alt-CMD-C on Mac OS X) to make our code cleaner and more maintainable.



Check Web Help for full details.

Extract Field Refactoring

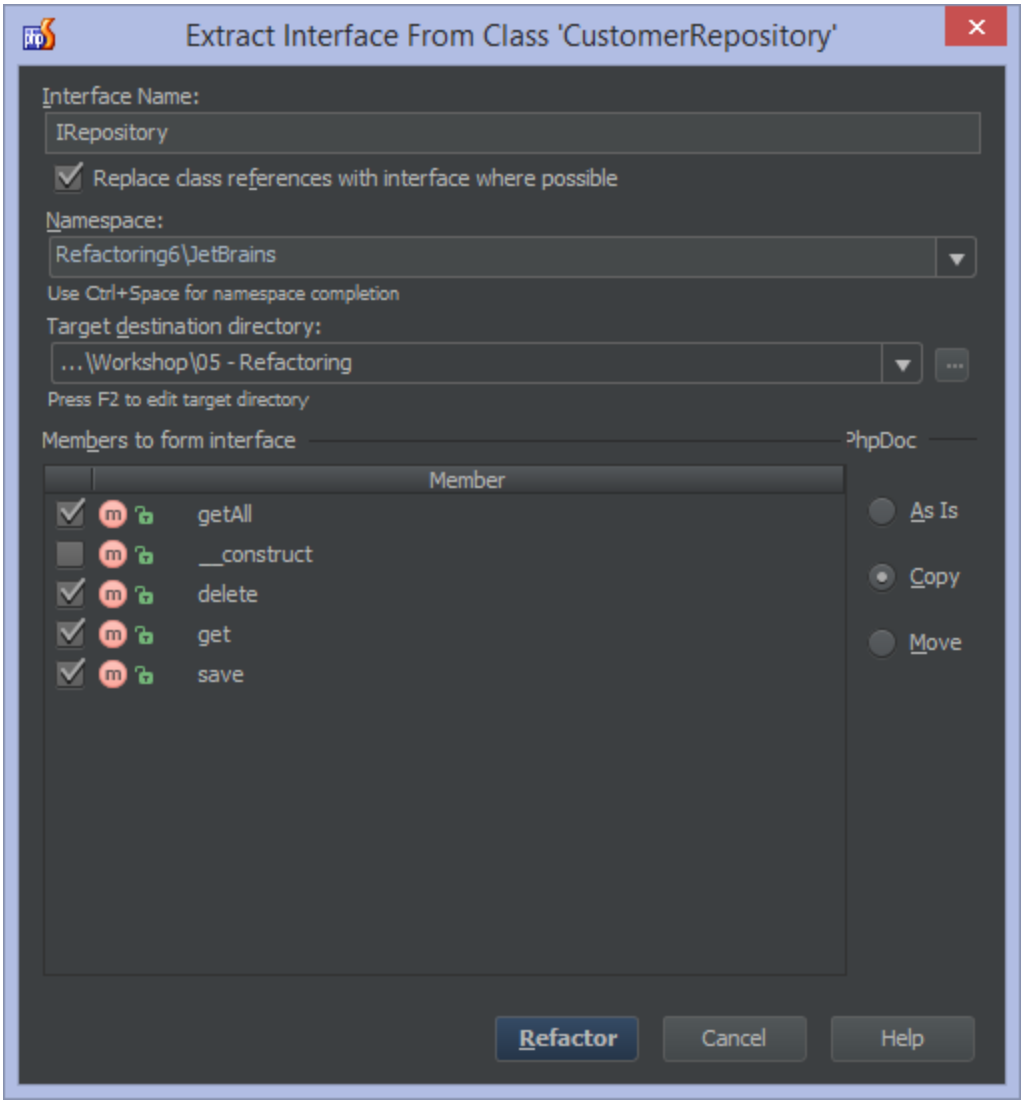
Similar to the #Extract Constant Refactoring, we can select a value in the editor and extract it into a field by pressing Ctrl+Alt+F (or Alt-CMD-F on Mac OS X). We can let PhpStorm initialize the field with the selected value in the class constructor, in the field declaration or in the method where we selected the value.



Check Web Help for full details.

Extract Interface Refactoring

The Extract Interface refactoring allows us to quickly create a new interface based on a selected interface or class. We can provide a name for our interface, specify its namespace and target directory, pick the members (fields, functions) we want to extract, and select whether we want to copy PHPDoc blocks.

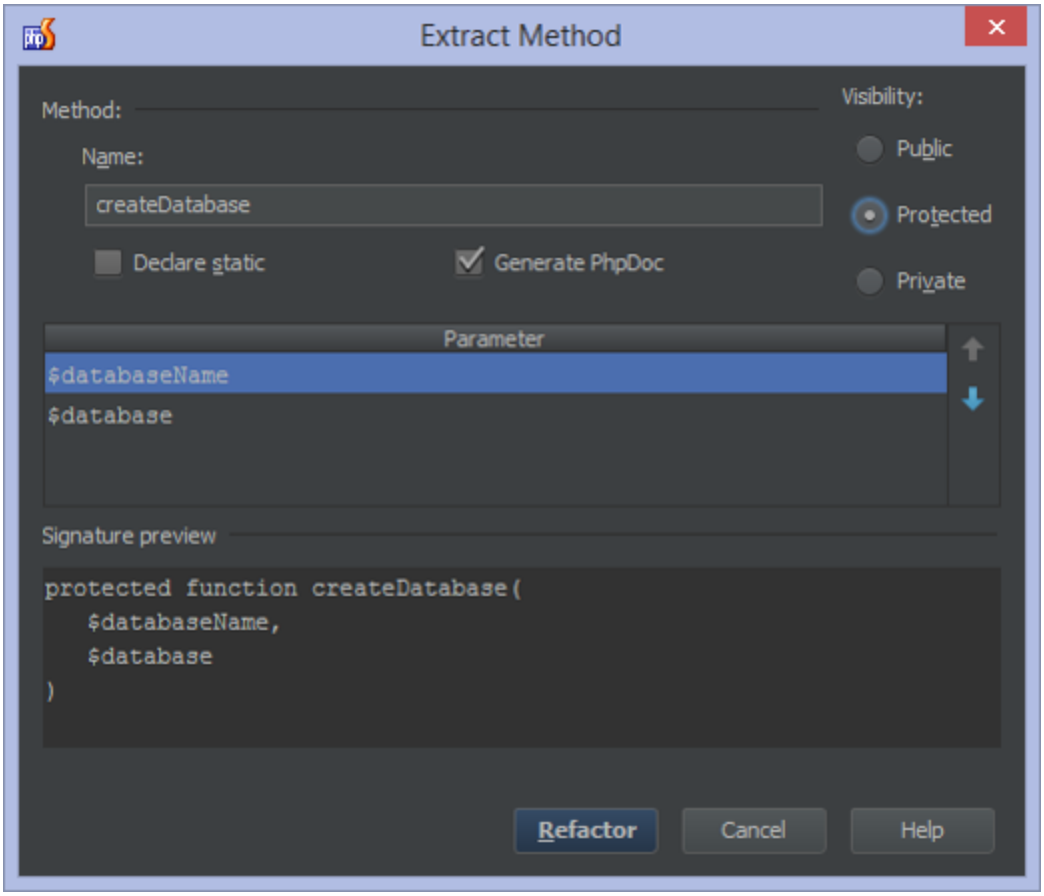


No keyboard shortcut is assigned for this refactoring. Use Refactor This or the Refactor | Extract Interface context menu.

Check Web Help for full details.

Extract Method Refactoring

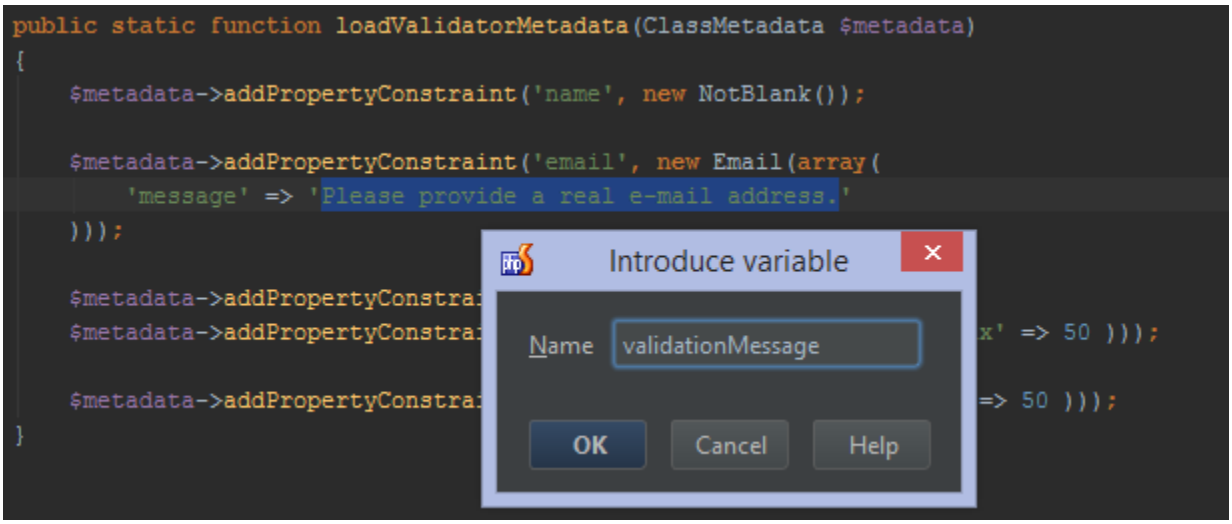
Whenever we want to extract a piece of code into a separate function, we can make use of the Extract Method Refactoring by pressing Ctrl+Alt+M (or Alt+CMD+M on MacOS X). It will extract the selected block of code into a function, detecting parameters and return values. We can choose the visibility of the function (public, private, or protected), the name of the function, how we want the result to be returned (using return or by reference), and optionally generate PHPDoc blocks.



Check Web Help for full details.

Extract Variable Refactoring

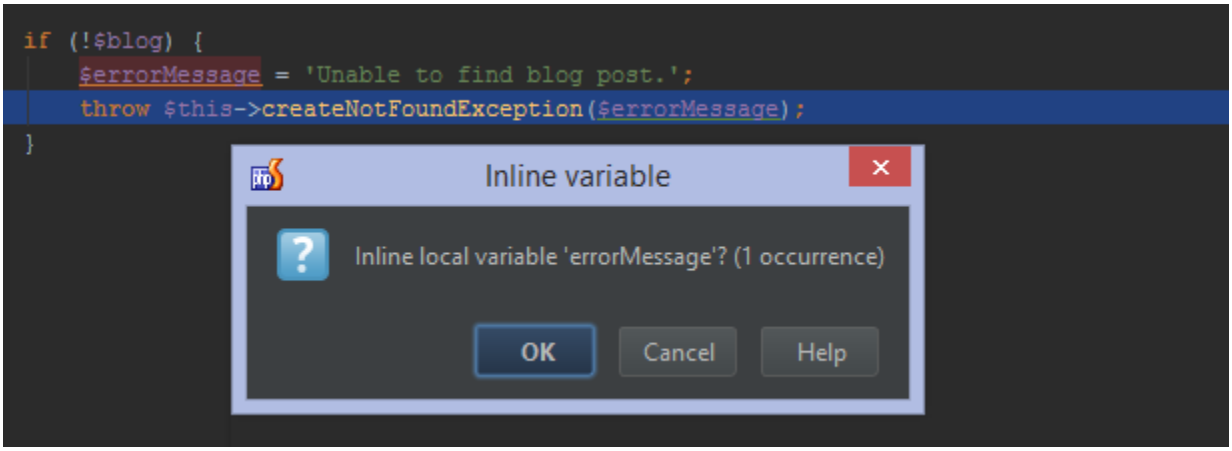
We can extract a selected expression into a variable by pressing Ctrl+Alt+V (Alt-CMD-V on Mac OS X). The original expression will be replaced with the new variable.



Check Web Help for full details.

Inline Refactoring

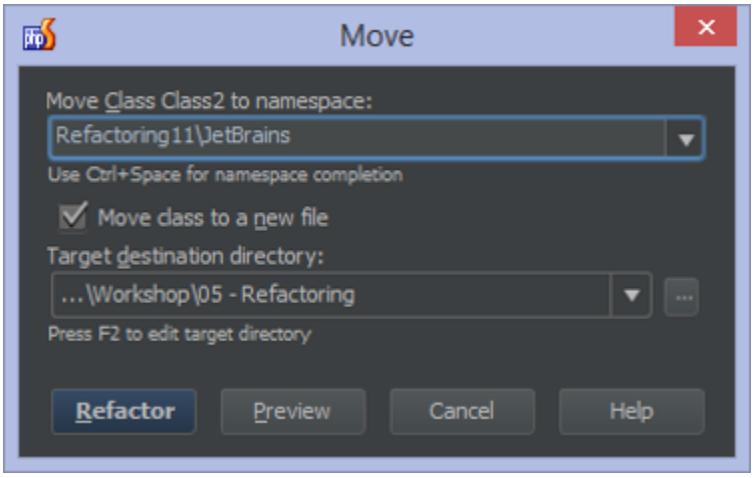
The Inline Refactoring Replace is the opposite of Extract Variable / Extract Method refactorings: it replaces redundant variables or functions with the full expression. We can select a variable and press Ctrl+Alt+N (or Alt-CMD-N on Mac OS X) to invoke it.



Check Web Help for full details.

Move Refactoring

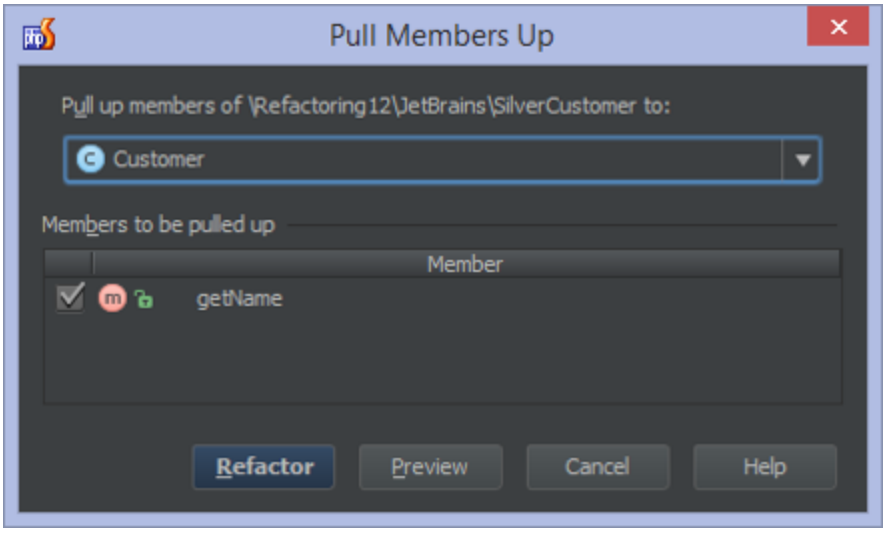
With the Move Refactoring (F6), we can change the location of a file, directory, class or static member. We can select the namespace and file to where we want to move the item.



Check Web Help for full details.

Pull Members Up / Push Members Down Refactoring

Imagine having the following class hierarchy: Customer extends BaseCustomer which in turn extends Person. With the Pull Members Up / Push Members Down Refactoring, we can move any member in any class of that hierarchy to the subclass or to the super class. For example, we could move a getDisplayName() function from Customer to Person.

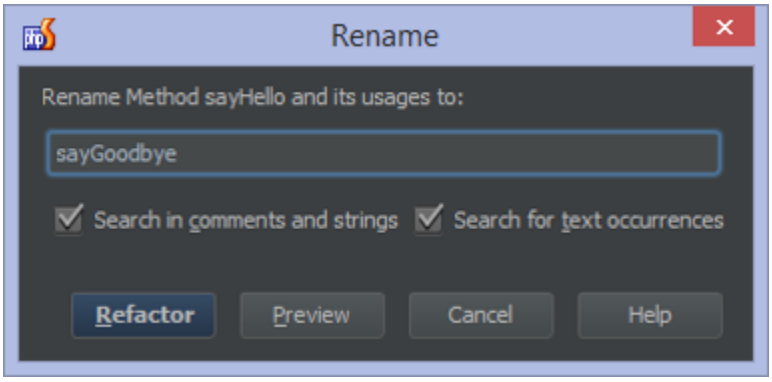


No keyboard shortcut is assigned for this refactoring. Use Refactor This or the Refactor | Pull Members Up or Refactor | Push Members Down context menu.

Check Web Help for full details.

Rename Refactoring

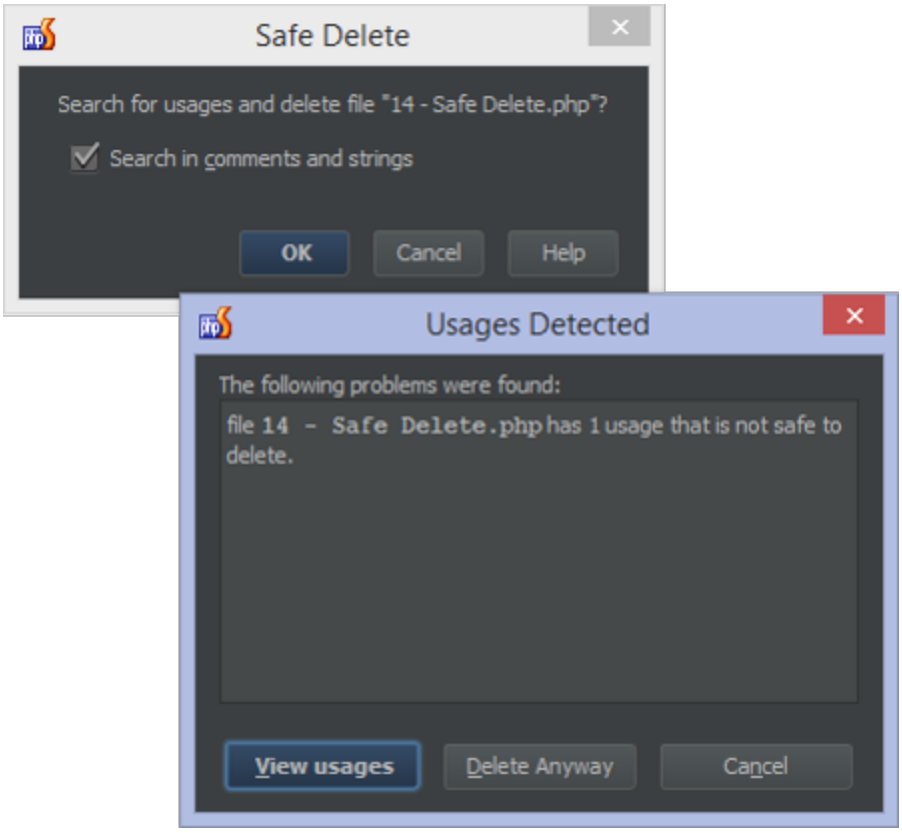
Using the Rename Refactoring (Shift+F6), we can rename symbols, automatically correcting all references in our codebase. We can rename Classes, Methods, Fields, Functions, Variables, Parameters, CSS Color Values, Files and Directories.



Check Web Help for full details.

Safe Delete Refactoring

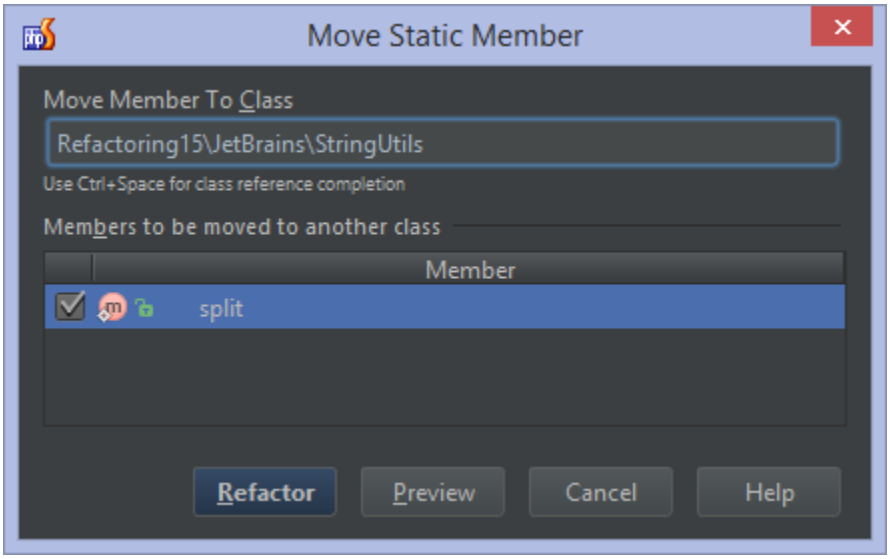
Not sure if a symbol or file is being used elsewhere but still want to delete it? Simply deleting it may break our application, but using the Safe Delete Refactoring (Alt+Delete or CMD-Delete on Mac OS X) we will get notified when references to it still exist. This allows us to either immediately delete the file or symbol when they are no longer used, but will give a warning if it is still in use.



Check Web Help for full details.

Move Static Member Refactoring

With the new Move Static Member Refactoring (F6), we can move static fields and methods to another type. For example, when classes contain static methods that are nothing but utility methods, we can move them into a separate type.



Check Web Help for full details.

Tweet