Language Injections (mix-ins) in PhpStorm

In almost every project, there are situations when we want to work with small islands of different languages embedded in our source code. For example, pieces of JavaScript that are contained within a PHP string. Or portions of HTML in a JavaScript variable. Or database queries inside a string. There are plenty of examples! PhpStorm makes it really easy to work with them by treating these fragments as the language they contain, providing syntax highlighting and code completion for that fragment. Let's see how this works.

1. Recognizing a String as Language Injection

In many situations, the editor will automatically recognize a string as a language injection. In some cases, we may have to inject them manually.

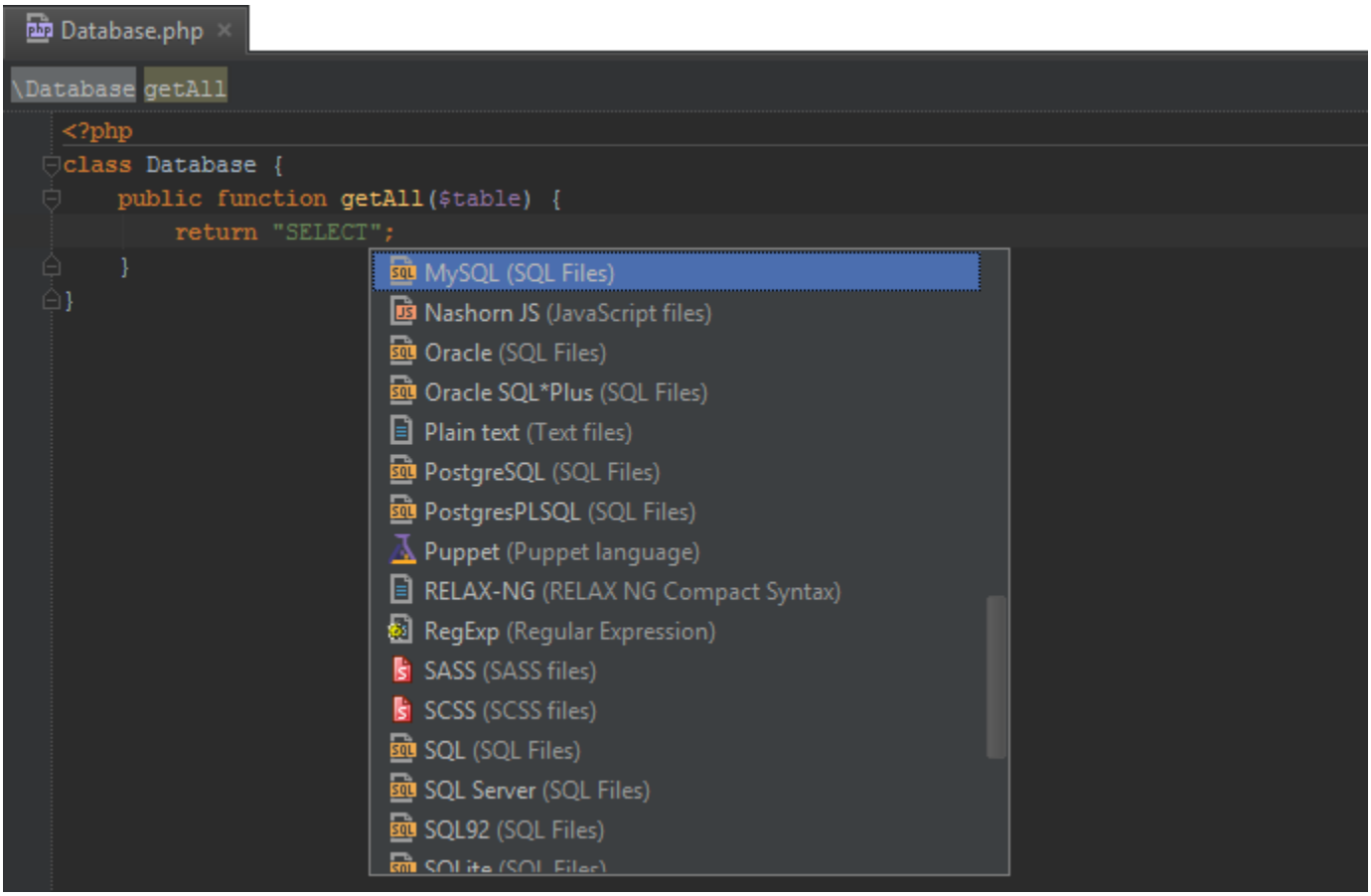
1.1. Automatically Injecting a Language Reference

For many languages like HTML, JavaScript, PHP, XML, SQL and so on, the IDE will recognize language injections automatically. It can be a string defined using double quotes, single quotes or HEREDOC. PhpStorm will even recognize concatenated strings. Once a language fragment has been detected, the editor will highlight it, like the JavaScript string literal we're creating here:

1.2. Manually Injecting a Language Reference

For situations where the IDE can not automatically determine the language we're injecting in a string, we can tell it which

language we're intending to use. Using Alt+Enter on the string contents, we can invoke the Inject Language/Reference intentio n and select the language contained in the string.



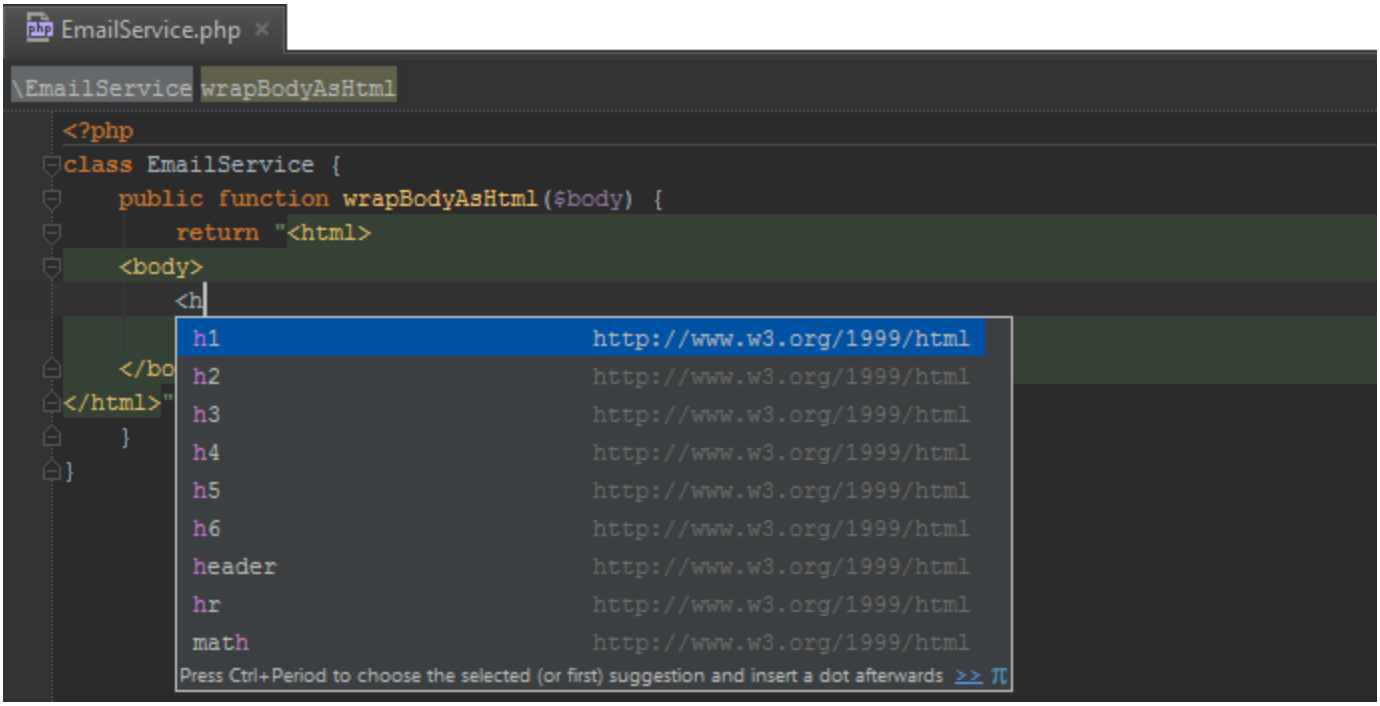
When manually injecting language references, it is possible that PhpStorm will only "remember" the injected language for a limited period of time. This period will depend on the language, context and the modifications that we make in other parts of our source code. If this happens, we can easily inject a language reference again (if the IDE doesn't recognize the language injection automatically).

2. Editing a Language Injection

Fragments of different languages can be edited inline, within the string literal, or in a separate editor tab.

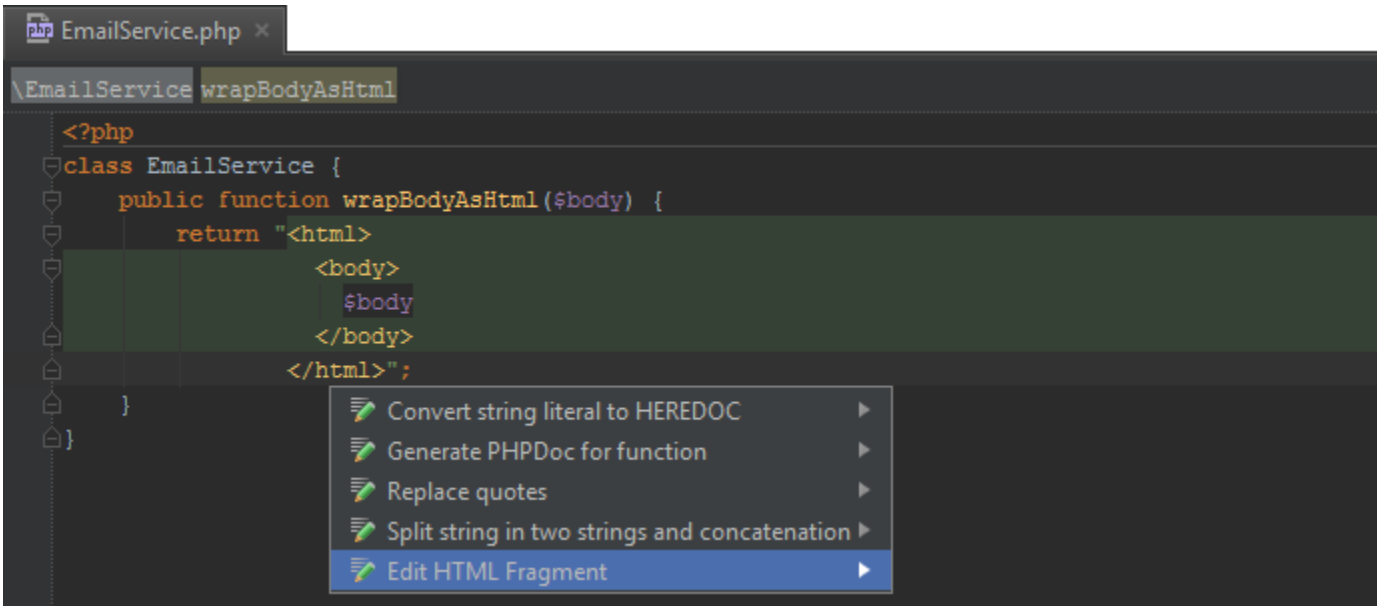
2.1. Editing a Language Injection within a String Literal

Once the editor knows about the language injection, we can start working with it. Syntax and error highlighting and coding assistance are extended to this string.

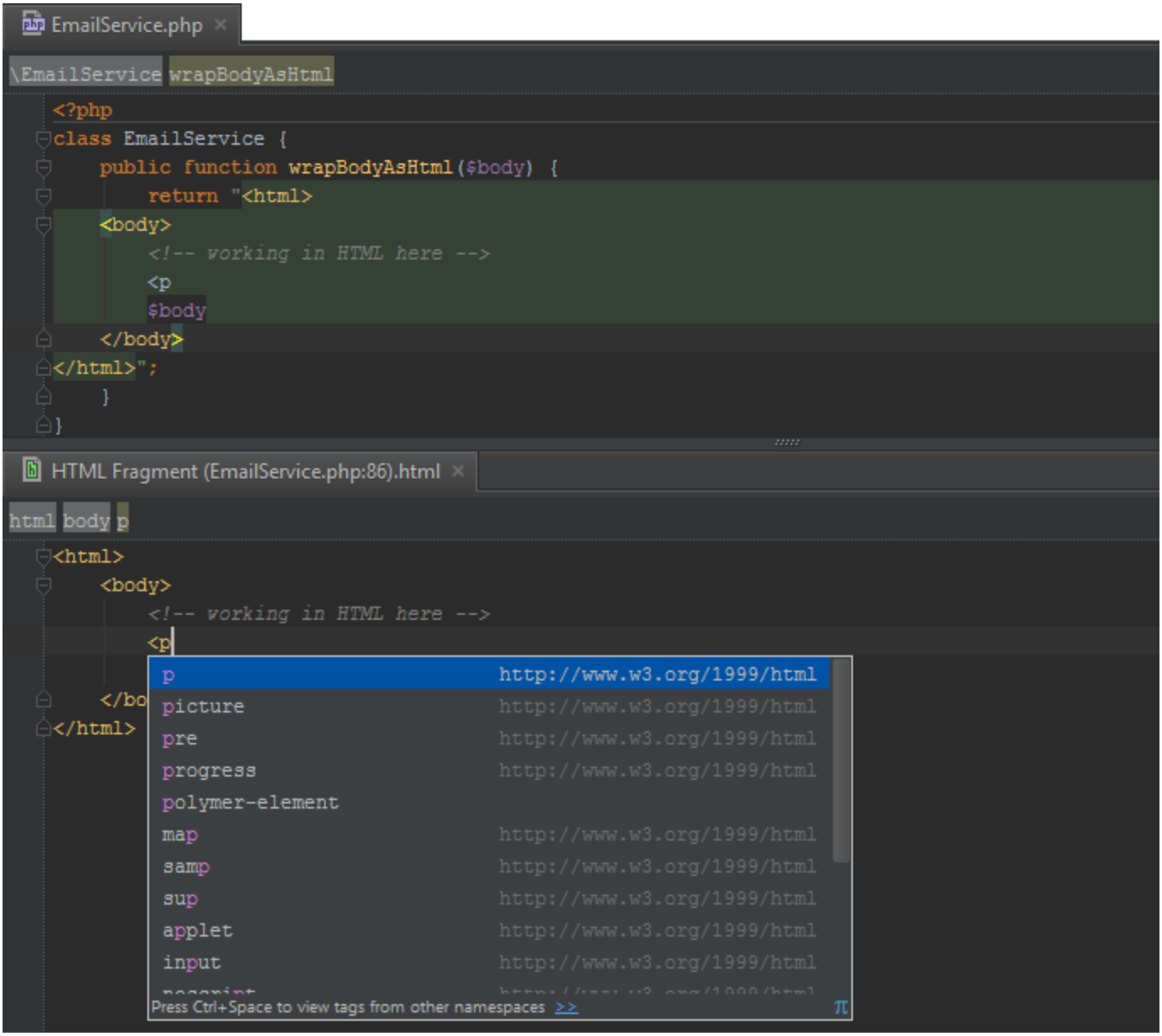


2.2. Editing a Language Injection in a Separate Editor Tab

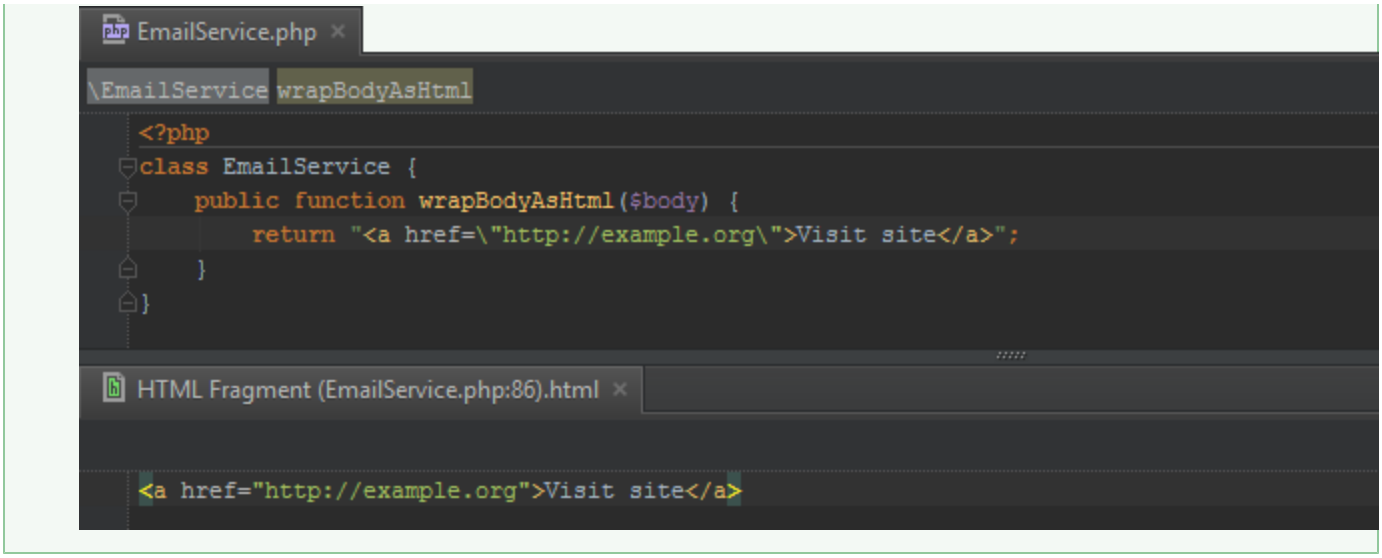
We can also use the Edit <language> Fragment intention to open a separate editor tab in which we can work with the fragment.



In this separate editor tab, we can work with the source code in the corresponding language. PhpStorm will update the original string literal with the modifications we make.

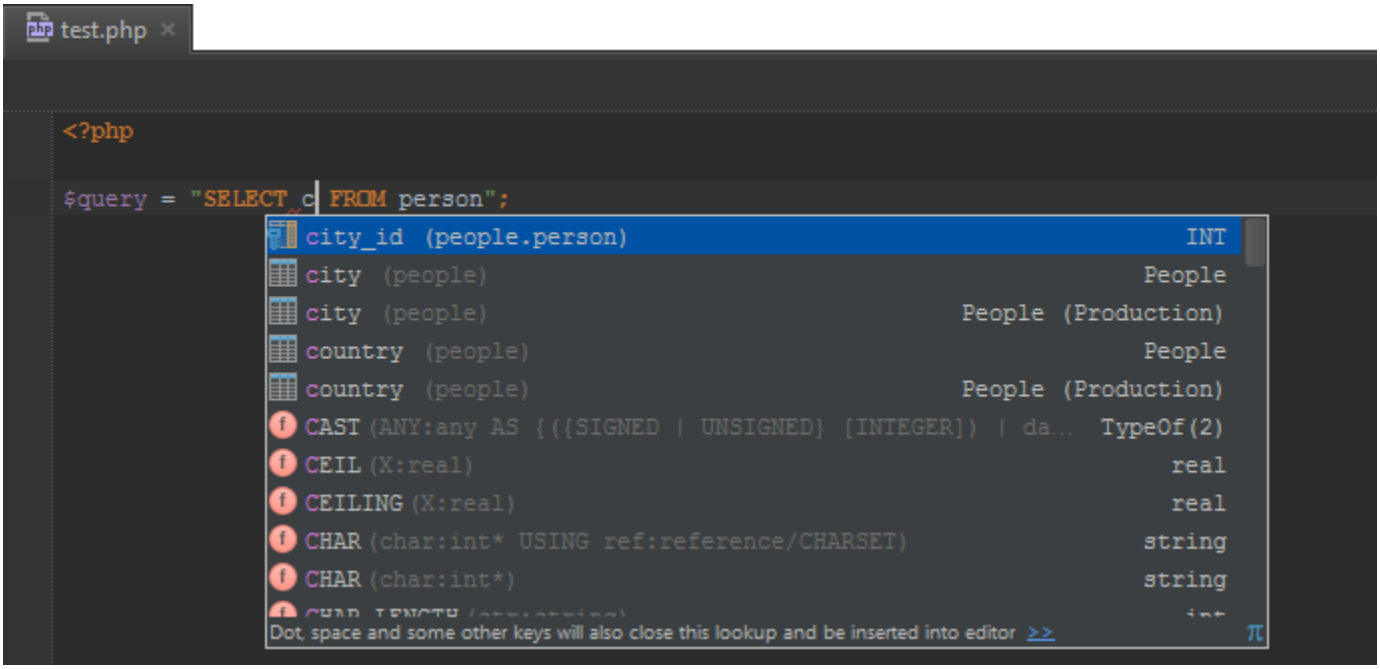


A big advantage of working with separate editor tabs for language injections is that they will do escaping for us. For example, as we add a hyperlink in a string that used double quotes, the double quotes of our hyperlink will be escaped with backslashes in the original PHP document.

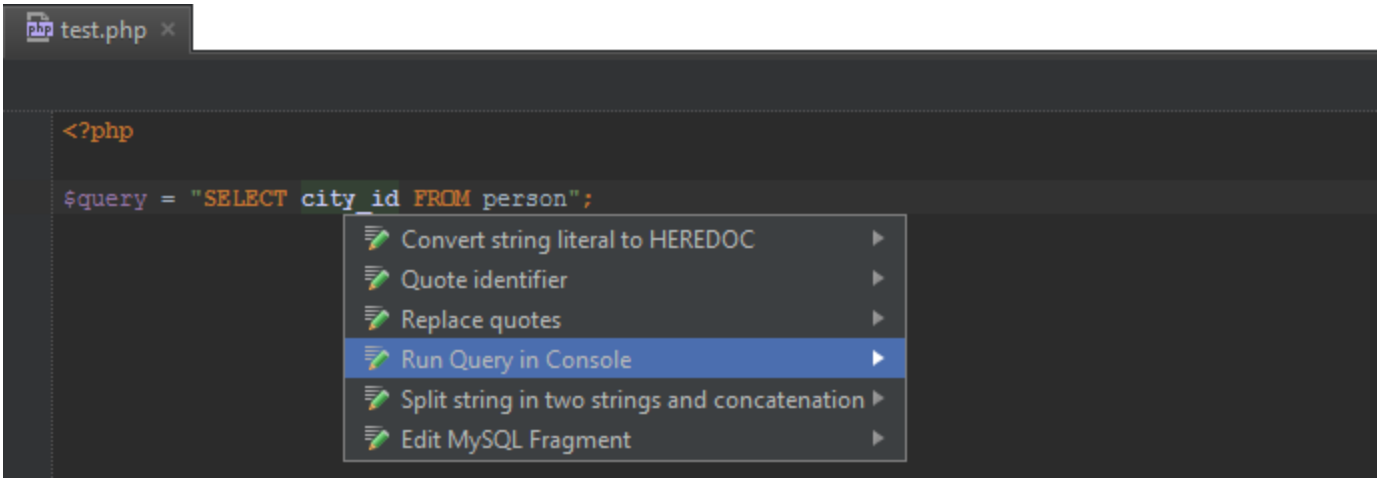


(optional) 3. Working with Databases and SQL Language Injections

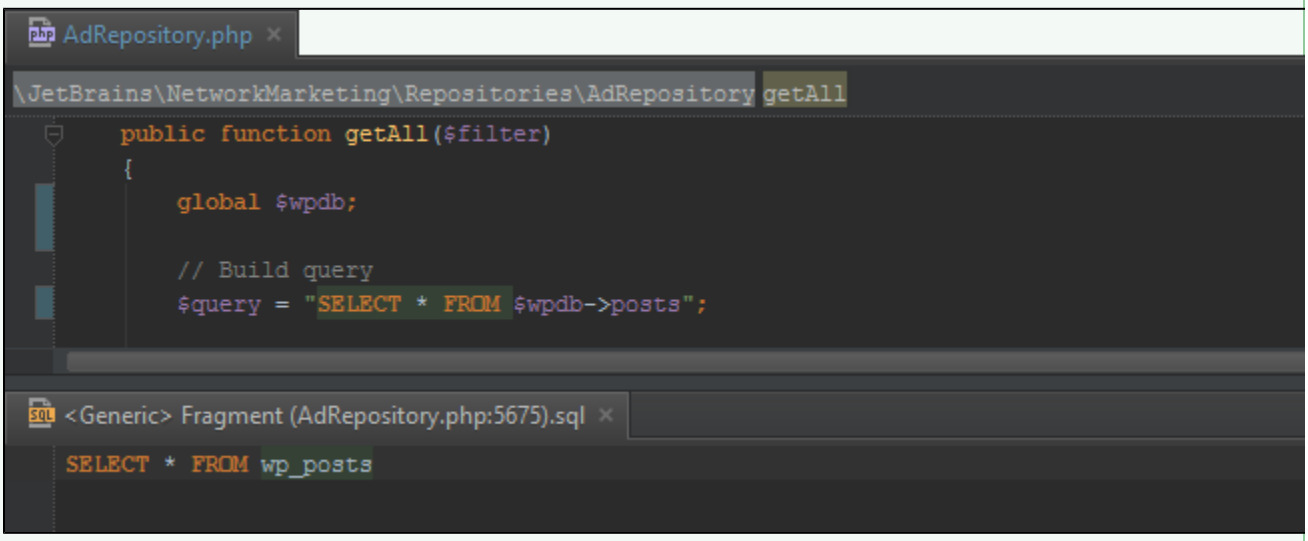
A very nice thing about language injections is how they behave when a database is connected to the project we are working in (see Databases and SQL Editor in PhpStorm). Not only will PhpStorm recognize SQL syntax and provide syntax highlighting and code completion on the injected language itself, but it will also know about tables and columns.



We can edit these queries in a separate editor by using the Edit MySQL Fragment intention. Since a database connection is configured, PhpStorm also lets us run this query immediately in the database console.



Some plugins, like the WordPress plugin, extend language injections. For example, when using the $wpdb->wp_posts variable (which references the wp_posts table in the WordPress database), the language injection editor tab will know about that variable and make the injected SQL statement more readable.



Tweet