

# Language Injections (mix-ins) in PhpStorm



## Redirection Notice

This page will redirect to <https://www.jetbrains.com/help/phpstorm/using-language-injections.html> in about 2 seconds.

[Tweet](#)

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
  - 1.1. Automatically Injecting a Language Reference
  - 1.2. Manually Injecting a Language Reference
- 2. Editing a Language Injection
  - 2.1. Editing a Language Injection within a String Literal
  - 2.2. Editing a Language Injection in a Separate Editor Tab
- (optional) 3. Working with Databases and SQL Language Injections

## 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:

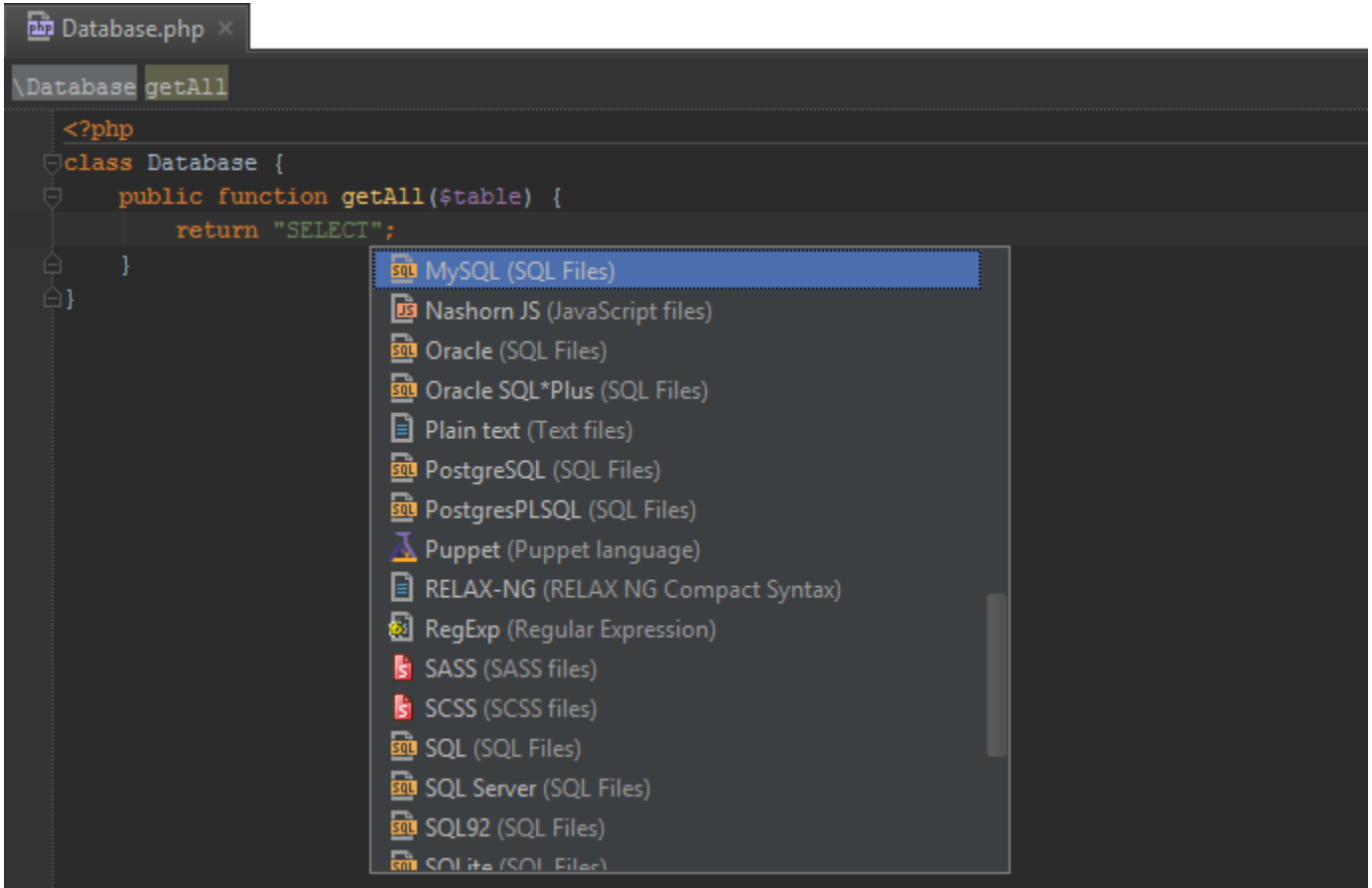
```
helloworld.html x
<!DOCTYPE html>
<html>
  <head lang="en">
    <meta charset="UTF-8">
    <title></title>


    <script>
      var helloWorld = function(name) {
        return '<p>Hello, ' + name + '</p>!';
      };
    </script>
  </head>
  <body>
</body>
</html>
```

### 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 intention 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.

```
<?php
class EmailService {
    public function wrapBodyAsHtml($body) {
        return "<html>
        <body>
        <h
    </body>
</html>"
    }
}
```

h1	http://www.w3.org/1999/html
h2	http://www.w3.org/1999/html
h3	http://www.w3.org/1999/html
h4	http://www.w3.org/1999/html
h5	http://www.w3.org/1999/html
h6	http://www.w3.org/1999/html
header	http://www.w3.org/1999/html
hr	http://www.w3.org/1999/html
math	http://www.w3.org/1999/html

Press Ctrl+Period to choose the selected (or first) suggestion and insert a dot afterwards >> π

## 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.

```
<?php
class EmailService {
    public function wrapBodyAsHtml($body) {
        return "<html>
        <body>
        $body
        </body>
        </html>";
    }
}
```

- Convert string literal to HEREDOC
- Generate PHPDoc for function
- Replace quotes
- Split string in two strings and concatenation
- Edit HTML 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.

The screenshot shows an IDE with two tabs. The top tab, 'EmailService.php', contains a PHP class definition for 'EmailService' with a 'wrapBodyAsHtml' method. The method returns an HTML string with a body containing a comment and a paragraph tag. The bottom tab, 'HTML Fragment (EmailService.php:86).html', shows the rendered HTML output. A dropdown menu is open over the paragraph tag, listing various HTML elements and their corresponding W3C URLs. The 'p' element is selected.

```
<?php
class EmailService {
    public function wrapBodyAsHtml($body) {
        return "<html>
        <body>
            <!-- working in HTML here -->
            <p>
                $body
            </p>
        </body>
    </html>";
    }
}
```

```
<html>
  <body>
    <!-- working in HTML here -->
    <p>
      p http://www.w3.org/1999/html
      picture http://www.w3.org/1999/html
      pre http://www.w3.org/1999/html
      progress http://www.w3.org/1999/html
      polymer-element
      map http://www.w3.org/1999/html
      samp http://www.w3.org/1999/html
      sup http://www.w3.org/1999/html
      applet http://www.w3.org/1999/html
      input http://www.w3.org/1999/html
      ...
    </p>
  </body>
</html>
```

- ✔ 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.

```
EmailService.php x
\EmailService wrapBodyAsHtml
<?php
class EmailService {
    public function wrapBodyAsHtml($body) {
        return "<a href='\"http://example.org\">Visit site</a>";
    }
}

HTML Fragment (EmailService.php:86).html x
<a href='\"http://example.org\">Visit site</a>
```

### (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.

```
test.php x
<?php
$query = "SELECT c FROM person";
    city_id (people.person) INT
    city (people) People
    city (people) People (Production)
    country (people) People
    country (people) People (Production)
    CAST (ANY:any AS ((SIGNED | UNSIGNED) [INTEGER]) | da... TypeOf(2)
    CEIL (X:real) real
    CEILING (X:real) real
    CHAR (char:int* USING ref:reference/CHARSET) string
    CHAR (char:int*) string
    CHAR_LENGTH (string) int
    Dot, space and some other keys will also close this lookup and be inserted into editor >> π
```

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.

```
<?php
$query = "SELECT city_id FROM person";
```

✔ 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.

```
public function getAll($filter)
{
    global $wpdb;

    // Build query
    $query = "SELECT * FROM $wpdb->posts";
}
```

```
SELECT * FROM wp_posts
```

Tweet