Mark Occurrences (PDT)

Writer	Date	Comment	Approved
Shalom Gibly	7/4/2008	1. Dev2QA	
		2. Target Version – PDT 1.1	

TOC

1. Introduction

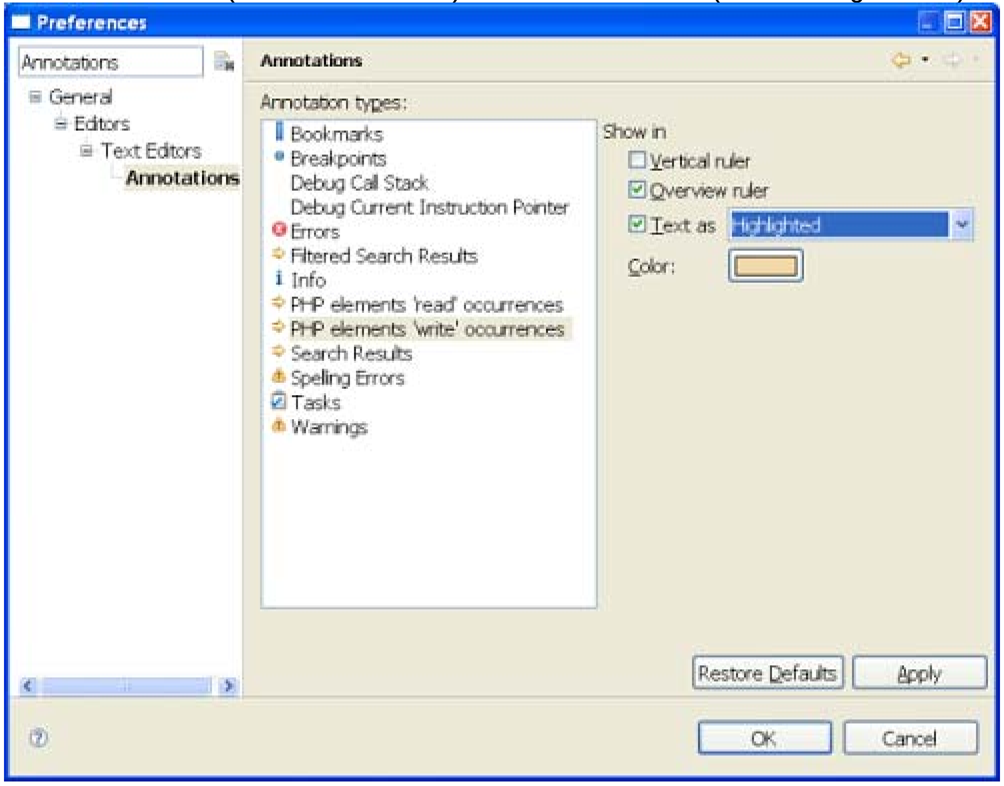
1.1 Requirement Rationale

Mark Occurrences" is a very simple and self-evident feature. When working in a *php* editor, turn on "Mark Occurrences" in the preference or press Alt+Shift+O. Then press on a variable, method, or type to see where it is referenced. What you will see are the highlighted usages of that variable, method or type.

The "Mark Occurrences" button can be toggled on and off. While it is on, clicking on a variable will highlight its usages. Click on a different variable and the highlights will change. The basic idea is very simple.

1.2 New Terminology

- Occurrence / Location – a location is a range in the document that should be marked as occurrence
- Occurrence type – can be either "Write Occurrence" (for occurrences that are in a write mode – will be colored as orange) or "Read Occurrence" (for occurrences that are in a read mode – will be colored as gray)
- Occurrence annotation – can be either 'write annotation' for write-occurrence, or 'read annotation' for read occurrence.
  The Annotations are defined in the Annotations preferences page. The page controls the way the annotations appear in the editor area, the annotations bar (editor's left side) and the ruler bar (editor's right side).



Annotations

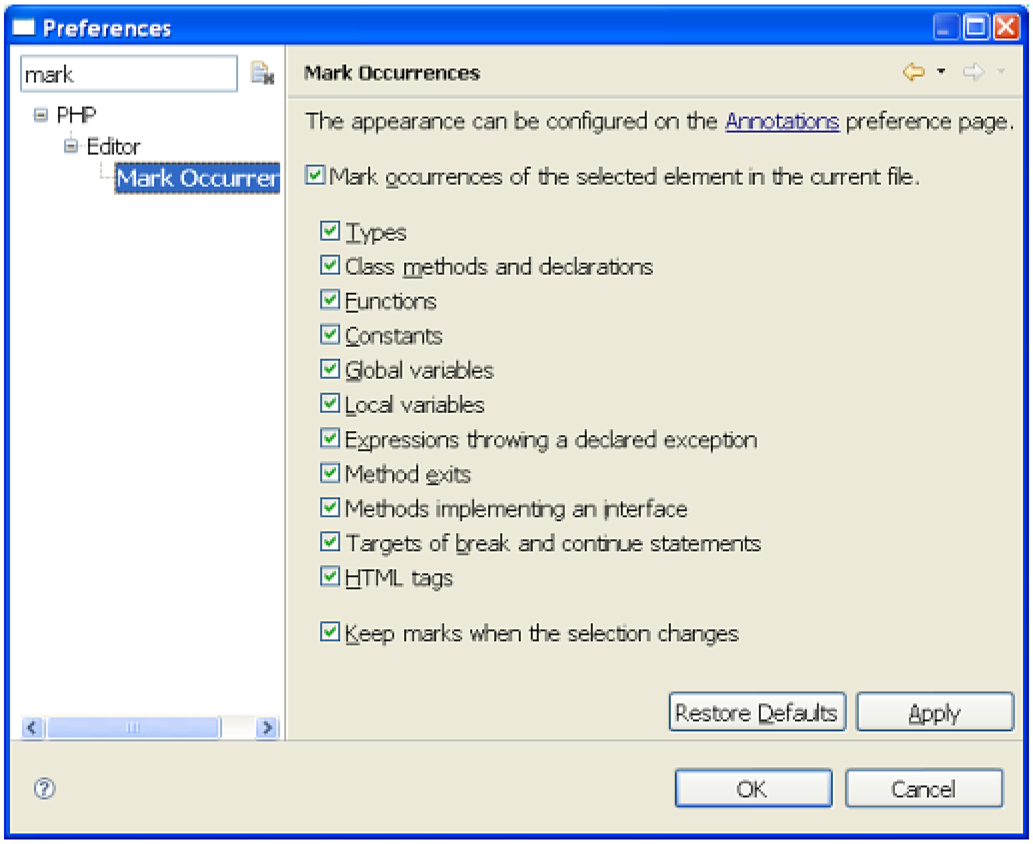
Annotation types:
- Bookmarks
- Breakpoints
- Debug Call Stack
- Debug Current Instruction Pointer
- Errors
- Filtered Search Results
- Info
- PHP elements 'read' occurrences
- PHP elements 'write' occurrences
- Search Results
- Spelling Errors
- Tasks
- Warnings

Show in
- [ ] Vertical ruler
- [x] Overview ruler
- [x] Text as Highlighted

Color:

2. Detailed Description

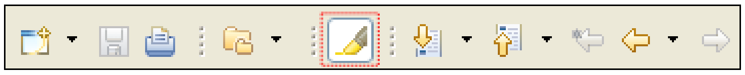
2.1 Enable Mark Occurrence (PHP → Editor → Mark Occurrences)



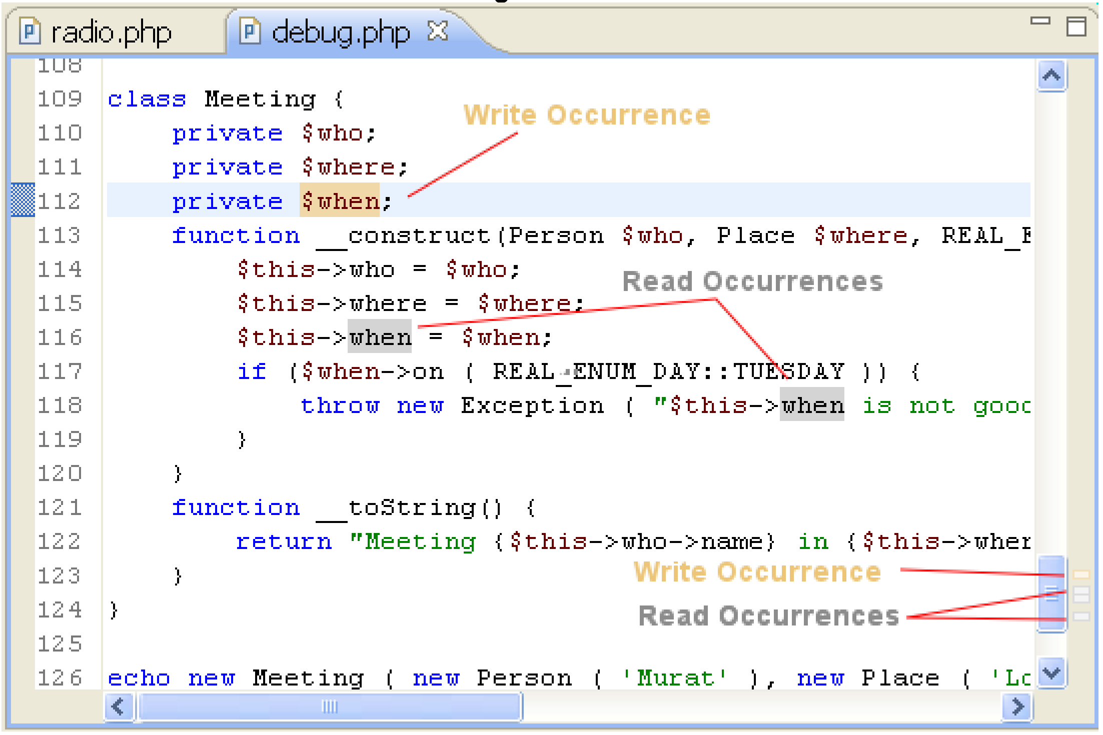
The appearance can be configured on the Annotations preference page.

- [✓] Mark occurrences of the selected element in the current file.
- [✓] Types
- [✓] Class methods and declarations
- [✓] Functions
- [✓] Constants
- [✓] Global variables
- [✓] Local variables
- [✓] Expressions throwing a declared exception
- [✓] Method exits
- [✓] Methods implementing an interface
- [✓] Targets of break and continue statements
- [✓] HTML tags
- [✓] Keep marks when the selection changes

The Enablement of the 'Mark Occurrences' can also be done through the application's tool-bar.
A dedicated toggle button was added. Pressing it should enable or disable the marking of the occurrences.
Pressing Alt+Shift+O will provide the same functionality.



2.2 Basic UI: Annotations and background of tokens



```
class Meeting {
    private $who;
    private $where;
    private $when;
    function __construct(Person $who, Place $where, REAL_E
        $this->who = $who;
        $this->where = $where;
        $this->when = $when;
        if ($when->on ( REAL_ENUM_DAY::TUESDAY )) {
            throw new Exception ( "$this->when is not gooc
        }
    }
    function __toString() {
        return "Meeting {$this->who->name} in {$this->wher
    }
}

echo new Meeting ( new Person ( 'Murat' ), new Place ( 'Lc
```

- When hovering with the mouse over the right ruler bar annotations an appropriate text should appear in a tool-tip:

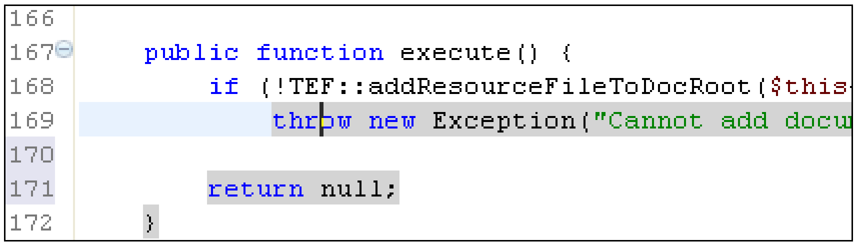
  1. When hovering over a ‘Write Occurrence’ annotation – Display ‘Write occurrence of XX’.
  2. When hovering over a ‘Read Occurrence’ annotation – Display ‘Occurrence of XX’.
  3. When hovering over an ‘Exit Point’ annotation – Display ‘Exit point of XX’.
  4. When hovering over ‘Break’ / ‘Continue’ scope annotation, e.g. *foreach*, *while*, *switch* etc. – Display ‘Target of XX’.

2.3 List of occurrences finders (e.g. the occurrences we mark):

2.3.1 Method Exit

Mark exit points (throws / return / end of flow) of a method.
Notes:

- *throws* / *return* statements are marked
- if the end of the function body is reachable we mark the enclosing curly

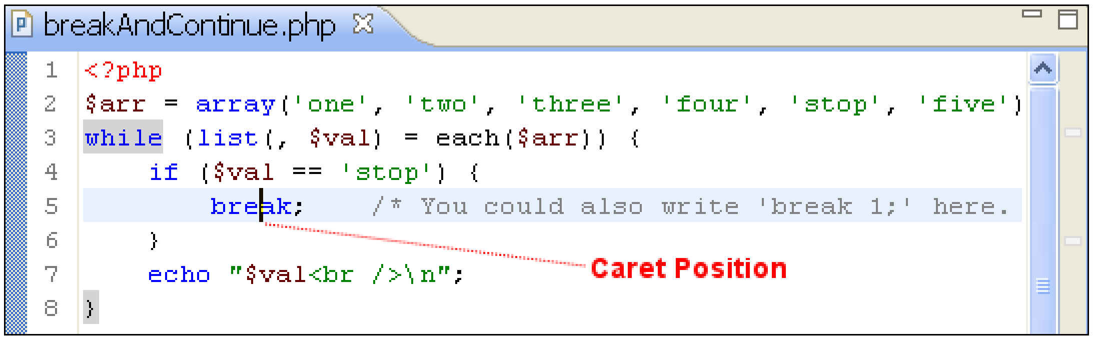


```
    public function execute() {
        if (!TEF::addResourceFileToDocRoot($this
            throw new Exception("Cannot add docu

        return null;
    }
```

2.3.2 Break / Continue

Mark scope (*for*, *foreach*, *while*, *do-while* or *switch* structure) of a break / continue statements.
Notes:

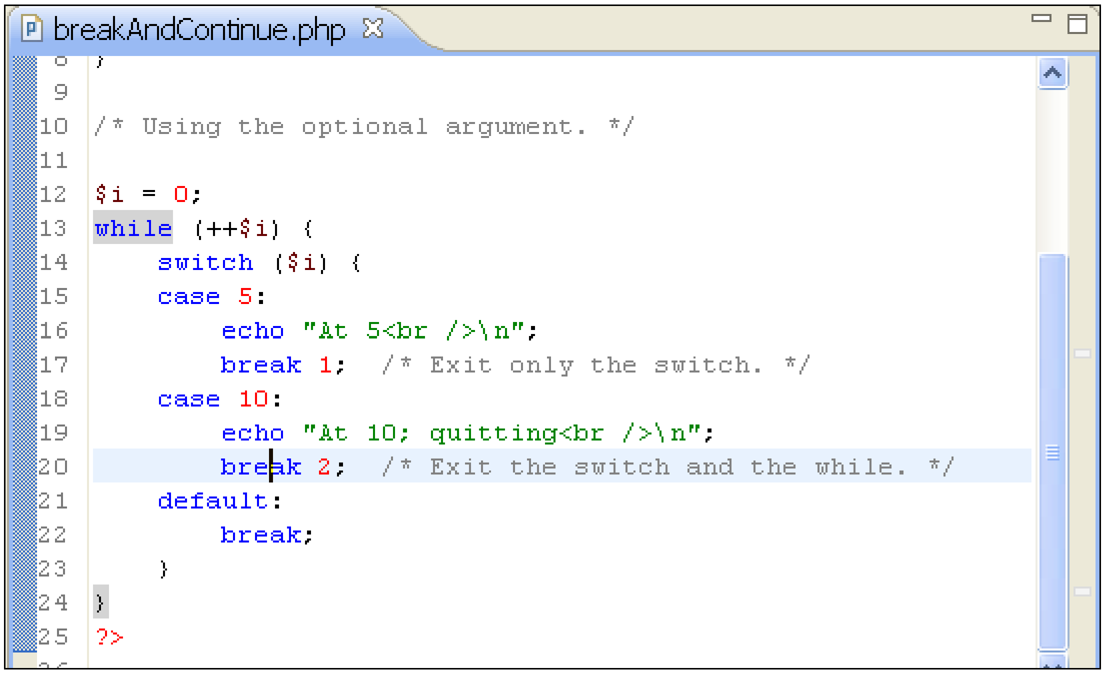
- *break/ continue* accepts an optional numeric argument which tells it how many nested enclosing structures are to be broken out of.
- In case the code contains this argument, our implementation marks the scope that matches the level.



breakAndContinue.php

```
<?php
$arr = array('one', 'two', 'three', 'four', 'stop', 'five')
while (list(, $val) = each($arr)) {
    if ($val == 'stop') {
        break;    /* You could also write 'break 1;' here.
    }
    echo "$val<br />\n";
}
```

Break 1 level – Mark the 'while' scope.


breakAndContinue.php

```
/* Using the optional argument. */

$i = 0;
while (++$i) {
    switch ($i) {
    case 5:
        echo "At 5<br />\n";
        break 1;  /* Exit only the switch. */
    case 10:
        echo "At 10; quitting<br />\n";
        break 2;  /* Exit the switch and the while. */
    default:
        break;
    }
}
?>
```

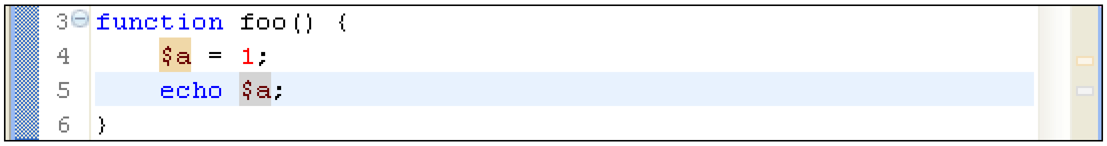
Break 2nd level – Skip the 'Switch' scope and mark the 'while' scope.

- When the scope does not have a body that is wrapped in a curly bracket, only the mark scope (while, foreach, for, etc.) is marked.

2.3.3 Occurrences of other PHP elements

Mark instances of the following instances and usages:

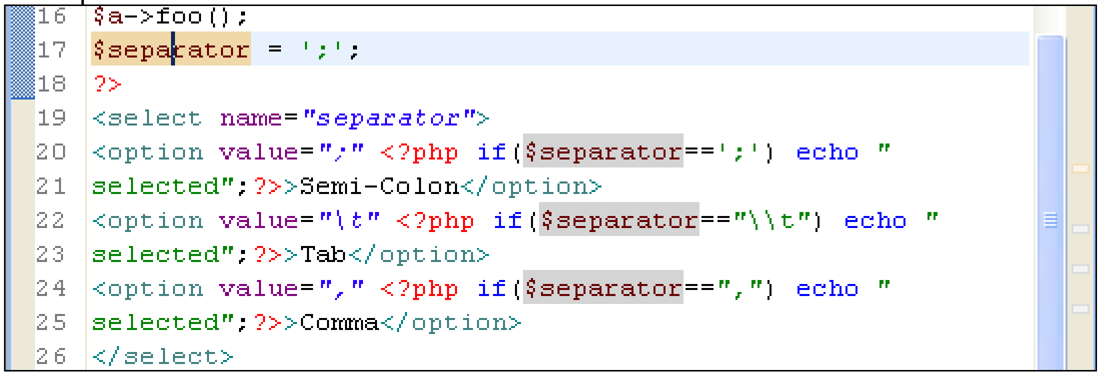
- **Local variable**



```
function foo() {
    $a = 1;
    echo $a;
}
```

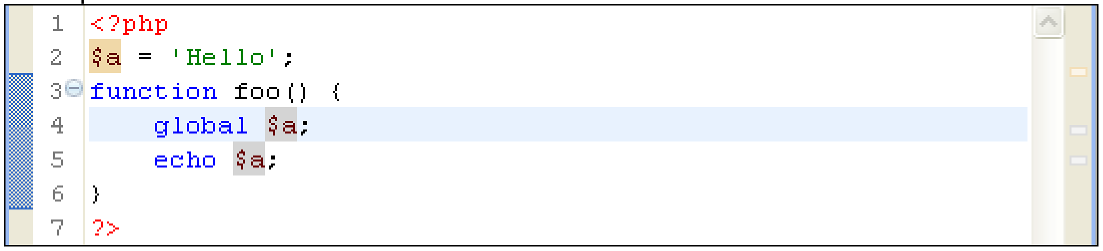
- **Global variable**

Example 1:



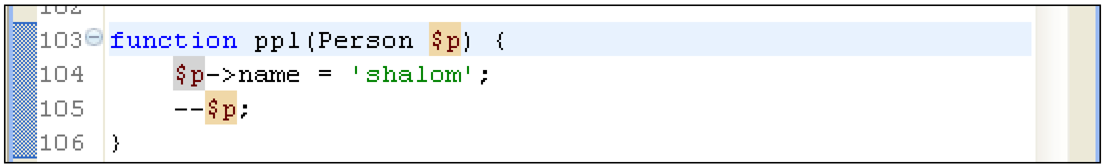
```
$a->foo();
$separator = ';';
?>
<select name="separator">
<option value=";" <?php if($separator==';') echo "
selected";?>>Semi-Colon</option>
<option value="\t" <?php if($separator=="\\t") echo "
selected";?>>Tab</option>
<option value="," <?php if($separator==",") echo "
selected";?>>Comma</option>
</select>
```

Example 2:



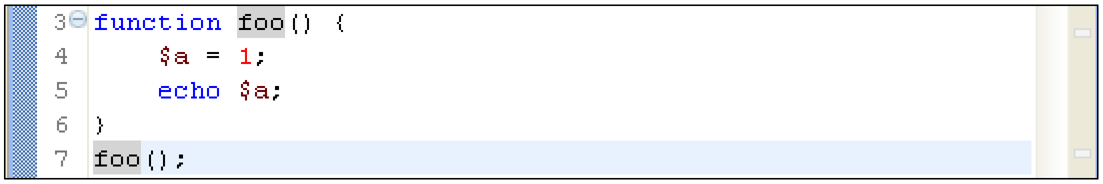
```php
<?php
$a = 'Hello';
function foo() {
    global $a;
    echo $a;
}
?>
```

- **Parameter**



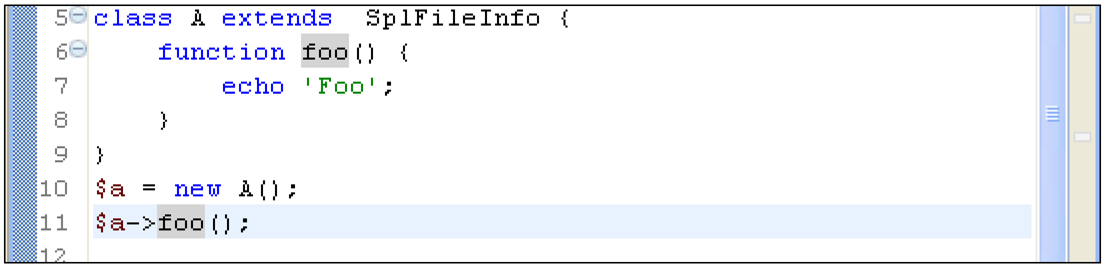
```php
function ppl(Person $p) {
    $p->name = 'shalom';
    --$p;
}
```

- **Function**



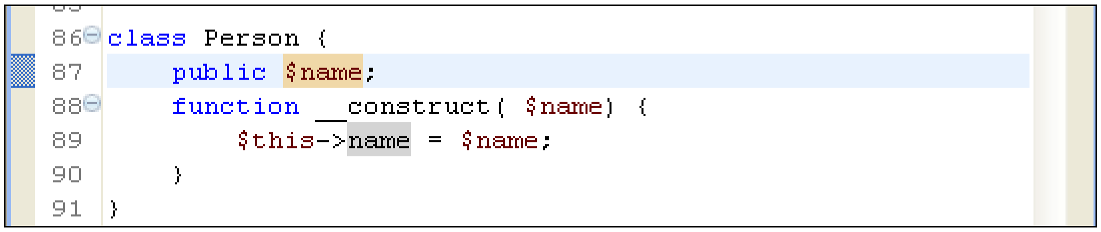
```php
function foo() {
    $a = 1;
    echo $a;
}
foo();
```

- **Method**



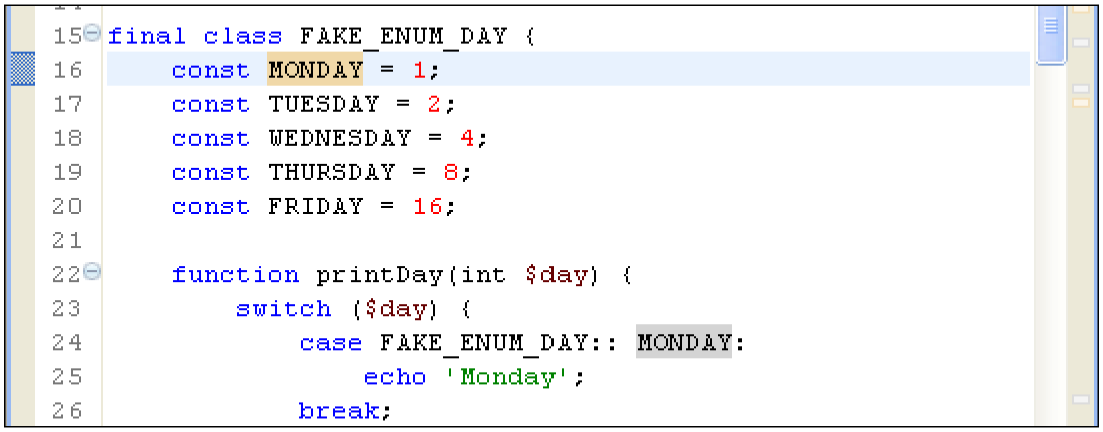
```php
class A extends  SplFileInfo {
    function foo() {
        echo 'Foo';
    }
}
$a = new A();
$a->foo();
```

- **Field**



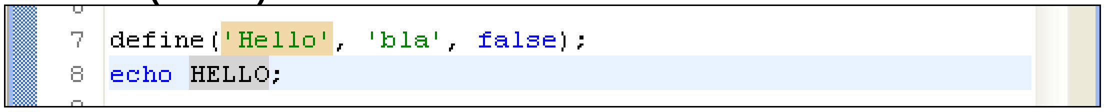
```php
class Person {
    public $name;
    function __construct( $name) {
        $this->name = $name;
    }
}
```

- **Class constants**



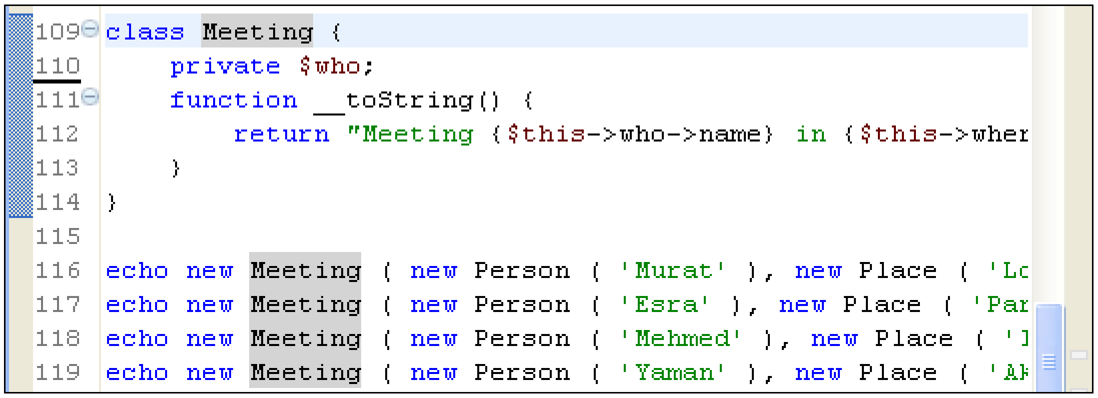
```
final class FAKE_ENUM_DAY {
    const MONDAY = 1;
    const TUESDAY = 2;
    const WEDNESDAY = 4;
    const THURSDAY = 8;
    const FRIDAY = 16;

    function printDay(int $day) {
        switch ($day) {
            case FAKE_ENUM_DAY:: MONDAY:
                echo 'Monday';
            break;
```

- **Constant (define)** – See note below



```
define('Hello', 'bla', false);
echo HELLO;
```

- **Class / Interface**



```
class Meeting {
    private $who;
    function __toString() {
        return "Meeting {$this->who->name} in {$this->wher
    }
}

echo new Meeting ( new Person ( 'Murat' ), new Place ( 'Lc
echo new Meeting ( new Person ( 'Esra' ), new Place ( 'Par
echo new Meeting ( new Person ( 'Mehmed' ), new Place ( 'I
echo new Meeting ( new Person ( 'Yaman' ), new Place ( 'Ak
```

Note: Constants that are defined using the 'define' keyword are case sensitive by default, but can also be defined as case insensitive.
In case we recognize the 'define' line in the code, we identify the case sensitivity and mark the occurrences of the constant according to the case. In case we recognized a constant that was probably defined in an included file, we treat it as **case-insensitive** just to make sure that all of its possible occurrences are marked.

2.3.4 HTML Tags

Mark starting and ending nodes of the current tag.
When the curette is placed over an HTML element tag, the occurrence of the closing or opening tag should be marked as well.

3. Known Bugs

- Due to a WTP bug (#136923), the highlighting of HTML tags occurrences inside the document is malfunctioned.
  Although you will see the annotations on the right ruler bar, you will not see the highlighted HTML tag in the document unless you set the annotation to present itself as a 'Squiggly line' under the General->Editors->Text Editors->Annotations.

Shalom Gibly