ClickBlocks\Utils\InfoClass

General information

Inheritance	no
Child classes	no
Interfaces	ArrayAccess
Source	Framework/utils/infoclass.php

InfoClass is designed for getting information about internal and user defined classes and interfaces. You can also use **InfoClass** to change code structure of classes and interfaces. For example, you can add new properties/methods, change access modifiers of existent classes, add PHP-doc comments to them and so on. **InfoClass** can be used as an auxiliary class for building your own code generator of classes.

Public non-static properties

tab

```
public integer $tab = 2
```

Number of spaces of indentation.

Public non-static methods

__construct()

```
public void __construct(mixed $class)
```

$class	mixed	class name or class object.

Constructor. Extracts information about the given class.

getInfo()

```
public array getInfo()
```

Returns full information about the class. Example, let's we have the following class:

```
namespace MyNamespace\SpaceA;

interface ITestA {}

class TestA implements ITestA
{
  const A = 1;

  public $foo = 'one';

  protected static $lock = 'locked';

  protected function &access(&$var = null, array $data)
  {
    $x = false;
    return $x;
```

```
  }
}
```

We can get information about this class as follows:

```
$info = new InfoClass('MyNamespace\SpaceA\TestA');
print_r($info->getInfo());
```

The above example will output:

```
Array
(
    [name] => MyNamespace\SpaceA\TestA
    [shortName] => TestA
    [namespace] => MyNamespace\SpaceA
    [comment] => /**
 * My test class.
 */
    [file] => C:\sites\smartypassword\classes.php
    [startLine] => 7
    [endLine] => 23
    [isAbstract] =>
    [isFinal] =>
    [isInstantiable] => 1
    [isInterface] =>
    [isInternal] =>
    [isIterateable] =>
    [isUserDefined] => 1
    [isCloneable] => 1
    [isTrait] =>
    [inNamespace] => 1
    [extension] =>
    [parentName] =>
    [parentShortName] =>
    [parentNamespace] =>
    [interfaces] => Array
        (
            [MyNamespace\SpaceA\ITestA] => Array
                (
                    [name] => MyNamespace\SpaceA\ITestA
                    [shortName] => ITestA
                    [namespace] => MyNamespace\SpaceA
                )

        )

    [constants] => Array
        (
            [A] => 1
        )

    [properties] => Array
        (
            [foo] => Array
                (
                    [isDefault] => 1
                    [isPrivate] =>
                    [isProtected] =>
                    [isPublic] => 1
                    [isStatic] =>
                    [defaultValue] => one
                    [comment] =>
                )

            [lock] => Array
                (
```

```
                [isDefault] => 1
                [isPrivate] =>
                [isProtected] => 1
                [isPublic] =>
                [isStatic] => 1
                [defaultValue] => locked
                [comment] =>
            )

    )

[methods] => Array
    (
        [access] => Array
            (
                [isAbstract] =>
                [isConstructor] =>
                [isDestructor] =>
                [isFinal] =>
                [isPrivate] =>
                [isProtected] => 1
                [isPublic] =>
                [isStatic] =>
                [returnsReference] => 1
                [comment] =>
                [startLine] => 18
                [endLine] => 22
                [numberOfParameters] => 2
                [numberOfRequiredParameters] => 2
                [arguments] => Array
                    (
                        [0] => Array
                            (
                                [name] => var
                                [class] =>
                                [isDefaultValueAvailable] => 1
                                [isArray] =>
                                [isOptional] =>
                                [isPassedByReference] => 1
                                [allowsNull] => 1
                                [defaultValue] =>
                                [canBePassedByValue] =>
                            )

                        [1] => Array
                            (
                                [name] => data
                                [class] =>
                                [isDefaultValueAvailable] =>
                                [isArray] => 1
                                [isOptional] =>
                                [isPassedByReference] =>
                                [allowsNull] =>
                                [defaultValue] =>
                                [canBePassedByValue] => 1
                            )

                    )

                [code] =>
$x = false;
return $x;

            )

    )
```

```
)
```

getCodeConstant()

```
public string getCodeConstant(string $constant)
```

$constant	string	the constant name.

Returns code definition of the constant. If the constant doesn't exist the method returns FALSE. Example:

```
$info = new InfoClass('MyNamespace\SpaceA\TestA');
echo $info->getCodeConstant('A');
```

The execution result of the above example will be as follows:

```
const A = 1;
```

getCodeProperty()

```
public string getCodeProperty(string $property)
```

$property	string	the property name.

Returns code definition of the property. If the property doesn't exist the method returns FALSE. Example:

```
$info = new InfoClass('MyNamespace\SpaceA\TestA');
echo $info->getCodeProperty('foo');
```

The above example will output:

```
public $foo = 'one';
```

getCodeMethod()

```
public string getCodeMethod(string $method)
```

$method	string	the method name.

Returns code definition of the class method. If such class method doesn't exist the method will return FALSE. Example:

```
$info = new InfoClass('MyNamespace\SpaceA\TestA');
echo $info->getCodeMethod('access');
```

The execution result of the above script will be as follows:

```
  protected function &access(&$var = null, array $data)
  {
    $x = false;
    return $x;
  }
```

getCodeClass()

```
public string getCodeClass()
```

Returns code definition of the class.

```
$info = new InfoClass('MyNamespace\SpaceA\TestA');
echo $info->getCodeClass();
```

The above example will output:

```
/**
 * My test class.
 */
class TestA implements ITestA
{
  const A = 1;

  public $foo = 'one';

  protected static $lock = 'locked';

  protected function &access(&$var = null, array $data)
  {
    $x = false;
    return $x;
  }
}
```

save()

```
public boolean|integer save(string $file = null)
```

$file	string	the full path to the file for saving class definition.

Saves code definition of the class to file where it is defined or to new file. The method returns FALSE if impossible to rewrite code and the number of bytes that were written to the file otherwise. Example:

```
$info = new Utils\InfoClass('MyNamespace\SpaceA\TestA');
$info->save('myclass.php');
```

offsetSet()

```
public void offsetSet(mixed $var, mixed $value)
```

$var	mixed	the parameter name.
$value	mixed	value of the parameter.

Sets new parameter of the class information. This method is one of the four methods of **ArrayAccess** interface. Example:

```
$info = new InfoClass('MyNamespace\SpaceA\TestA');
$info['properties']['foo']['isStatic'] = 1;
$info['properties']['foo']['comment'] = '/**' . PHP_EOL . '   * Changed property' . PHP_EOL . '   */';
echo $info->getCodeProperty('foo');
```

The above example will output:

```
  /**
   * Changed property
   */
  public static $foo = 'one';
```

offsetExists()

```
public boolean offsetExists(mixed $var)
```

$var	mixed	the parameter name.

Checks whether the class parameter exists. This method is one of the four methods of **ArrayAccess** interface. Example:

```
$info = new InfoClass('MyNamespace\SpaceA\TestA');
echo (int)isset($info['properties']['someprop']); // will output 0
```

offsetUnset()

```
public void offsetUnset(mixed $var)
```

$var	mixed	the parameter name.

Removes the requested class parameter. This method is one of the four methods of **ArrayAccess** interface. Example:

```
$info = new InfoClass('MyNamespace\SpaceA\TestA');
echo (int)isset($info['properties']['foo']); // the output is 1
unset($info['properties']['foo']);           // removes property 'foo'
echo (int)isset($info['properties']['foo']); // the output is 0
```

offsetGet()

```
public mixed &offsetGet(mixed $var)
```

$var	mixed	the parameter name.

Returns value of the class parameter. This method is one of the four methods of **ArrayAccess** interface. Example:

```
$info = new InfoClass('MyNamespace\SpaceA\TestA');
print_r($info['properties']['foo']);
```

The above example will output:

```
Array
(
    [isDefault] => 1
    [isPrivate] =>
    [isProtected] =>
    [isPublic] => 1
    [isStatic] =>
    [defaultValue] => one
    [comment] =>
)
```

Protected non-static properties

info

```
protected array $info = []
```

Information about the class.

Protected non-static methods

extraction()

```
protected void extraction(mixed $class)
```

$class	mixed	class name or class object.

Extracts full information about the class and place into **$info** property.