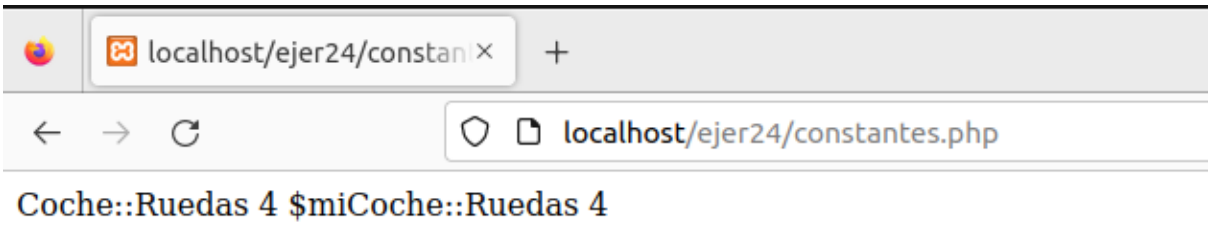
Laura Orts Ramon

Ejercicio 24 PHP

Constantes

```php
<?php class Coche
{
   const RUEDAS = 4;
}
// Obtener el valor mediante el nombre de la clase:
echo "Coche::Ruedas " . Coche::RUEDAS . "\n";
// Obtener el valor mediante el objeto:
$miCoche = new Coche();
echo "\$miCoche::Ruedas ". $miCoche::RUEDAS . "\n";
?>
```

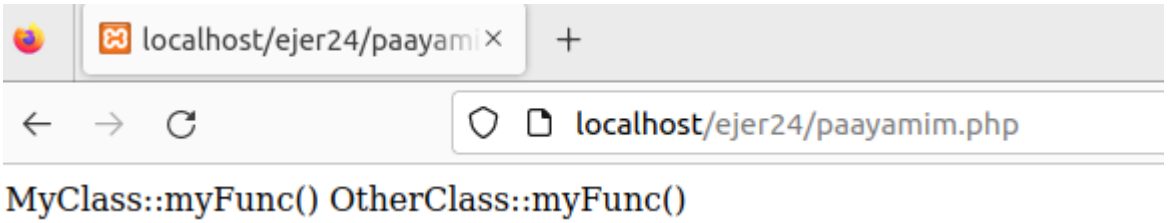
Resultado:



Se puede comprobar que se puede acceder a la propiedad RUEDAS tanto de la clase como de una instancia de la misma.

Paamayim Nekudotayim (::)A

```php
<?php
class MyClass
{
protected function myFunc() {
echo "MyClass::myFunc()\n";
}
}
class OtherClass extends MyClass
{
// Sobrescritura de definición parent
public function myFunc()
{
// Pero todavía se puede llamar a la función parent
parent::myFunc();
echo "OtherClass::myFunc()\n";
}
}
$class = new OtherClass();
$class->myFunc();
/*
La salida por pantalla será:
MyClass::myFunc()
OtherClass::myFunc()
*/
?>
```

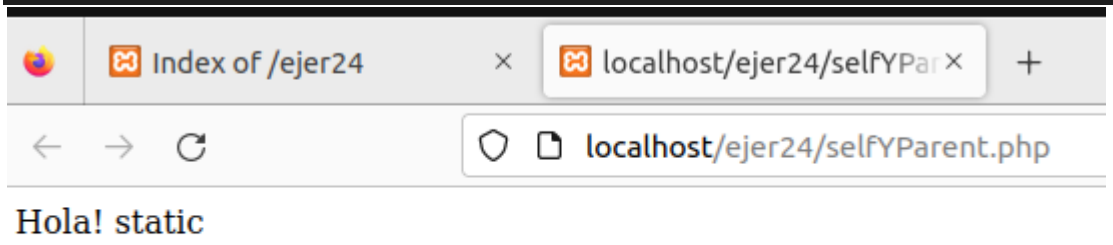


Se comprueba que se puede llamar a la función del parent incluso en el método que está sobreescribiendo dicha función con el operador resolución '::'.

Laura Orts Ramon

Self y Parent

```php
<?php
class MiClase {
const CONSTANTE = 'Hola! ';
}
class Clase2 extends MiClase
{
public static $variable = 'static';
public static function miFuncion() {
echo parent::CONSTANTE;
echo self::$variable;
}
}
Clase2::miFuncion();
/* La salida por pantalla será:
Hola! static*/
?>
```



Se comprueba que se puede llamar a parámetros del parent utilizando la palabra reservada ‘parent’ y el operador resolución ‘::’.

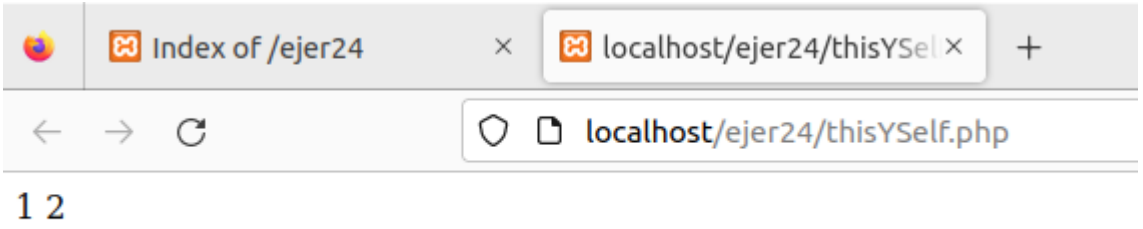
Laura Orts Ramon

Diferencia entre $this y self::

```php
<?php
class funcion {
private $valor_no_estático = 1;
private static $valor_estático = 2;
function __construct() {
echo $this->valor_no_estático . ' ' . self::$valor_estático;
}
}

$class = new funcion();

?>
```



Podemos ver aquí que con $this accedemos al valor no estático y con self accedemos al valor estático.

Laura Orts Ramon

```php
<?php
error_reporting(E_ALL | E_STRICT);
class Metodosestáticos
{
   private static $atributoestático = 'Soy un atributo estático';
   public static $atrestáticoPublico = 'Soy estático pero publico';
   private $atributoNoestático = 'No soy estático';
   public static function metodoestático()
   {
       echo '<br />' . self::$atributoestático . ' accedido por
self::';
   }
   public static function metodoestáticoThis()
   {
       self::metodoNoestático();
       $this->metodoNoestático();
   }
   public static function accederAtributoNoestático()
   {
       echo '<br />' . $this->atributoNoestático;
   }
   public function metodoNoestático()
   {
       echo '<br />' . self::$atributoestático;
       echo '<br />' . $this->atributoestático;
   }
}


?>
```

Pruebas

```php
echo Metodosestáticos::$atributoestático;
```

Obtenemos el error **Cannot access private property Metodosestáticos::$atributoestático** debido a que $atributoestático es private.

Laura Orts Ramon

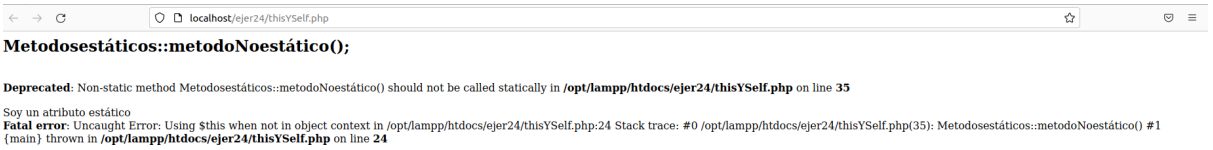
```
echo Metodosestáticos::$atrestáticoPublico;
```

Pintamos el $atrestáticoPublico sin problema, dado que es estático y público.


```
Metodosestáticos::metodoestático();
```

Accedemos al atributo estático privado perfectamente desde self.


```
Metodosestáticos::metodoNoestático();
```



Al ser un método instanciado no se puede acceder desde la clase, solo desde una instancia de la misma. Por esto mismo nos da error la segunda parte, al intentar acceder a $this.

```
Metodosestáticos::metodoestáticoThis();
```

Si intentamos llamar a obtendremos los siguientes errores:

El primero de todos es un warning debido a que no debemos llamar a un método no estático desde una llamada a la clase.

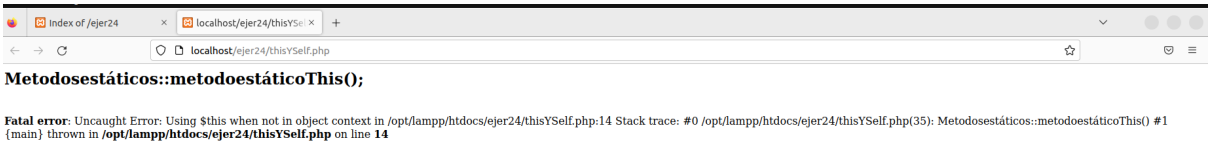
El error que obtenemos es debido a que estamos llamando a un atributo estático con $this desde la clase sin instanciar.

Si cambiamos el metodoestáticoThis() a

```
    public static function metodoestáticoThis()
    {
        $this::metodoNoestático();
        $this->metodoNoestático();
    }
```

Laura Orts Ramon

Obtenemos el siguiente error



Debido a que no hemos instanciado la clase, y por tanto no podemos usar $this.

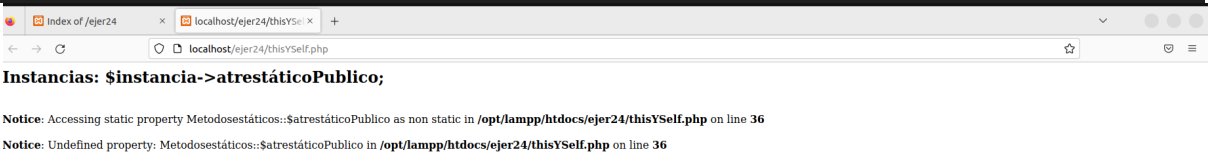
Utilizando una instancia de clase

Si utilizamos una instancia de clase tendremos acceso a $this y al operador flecha.

```
$instancia->atributoestático;
```

Obtenemos un error ya que el atributo es privado.

```
$instancia->atrestáticoPublico;
```

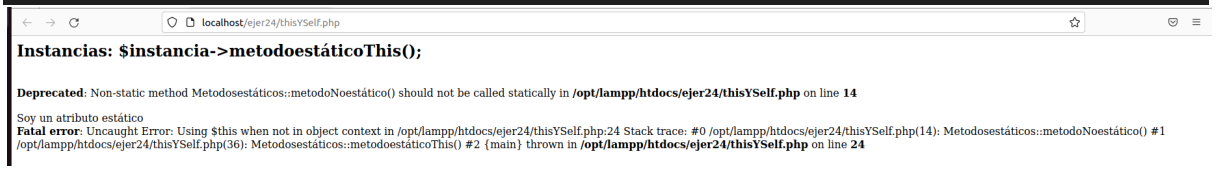


Nos da error al acceder también ya que desde una instancia no se puede acceder a atributos de clase, incluso si estos son públicos.

```
$instancia->metodoestático();
```

No hay ningún problema ya que el método es estático y el atributo es accedido mediante self.

```
$instancia->metodoestáticoThis();
```



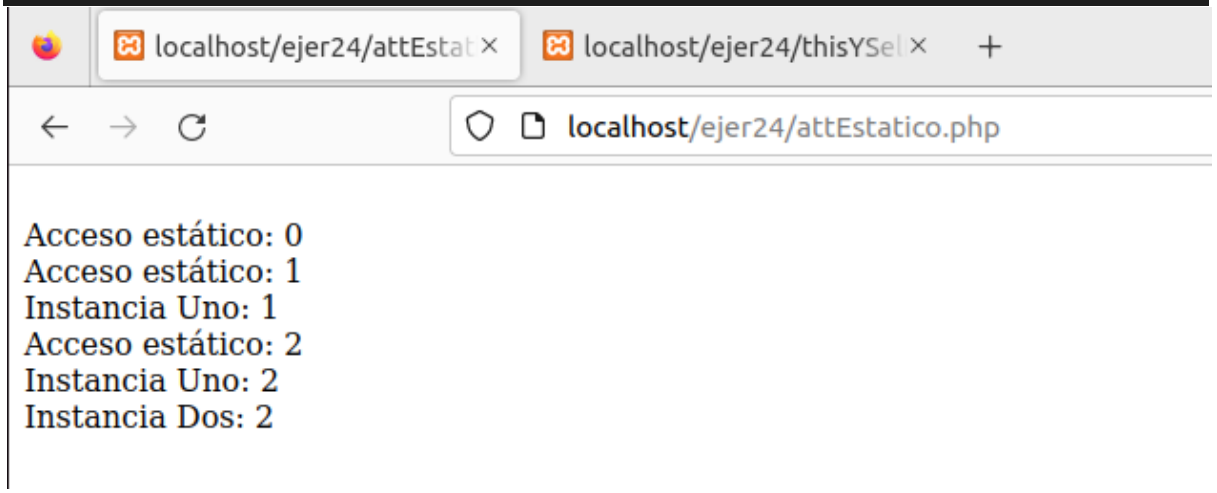
Aquí PHP nos dice que un método no estático no se debería llamar estáticamente, pero igualmente realiza la acción.

Además el siguiente error se nos da por usar $this es un contexto fuera de objeto, es decir, en un método estático.

Laura Orts Ramon

Como se comporta un Atributo estático

```php
<?php
class estático
{
private static $cantidadInstancias = 0;
public function __construct()
{
self::$cantidadInstancias++;
}
public static function getInstancias()
{
return self::$cantidadInstancias;
}
}
echo '<br/>Acceso estático: ' . estático::getInstancias(); // (0)
$instancia_01 = new estático();
echo '<br/>Acceso estático: ' . estático::getInstancias(); // (1)
echo '<br/>Instancia Uno: ' . $instancia_01->getInstancias(); // (2)
$instancia_02 = new estático();
echo '<br/>Acceso estático: ' . estático::getInstancias(); // (3)
echo '<br/>Instancia Uno: ' . $instancia_01->getInstancias(); // (4)
echo '<br/>Instancia Dos: ' . $instancia_02->getInstancias(); // (5)
?>
```



Como podemos observar en el primer acceso estático hay 0 instancias creadas.
En el segundo acceso estático, después de crear la instancia nos dice que hay 1 instancia creada.
Si le pedimos el dato a la instancia 1 nos da el mismo número.

Laura Orts Ramon

Ahora si creamos una instancia nueva y pedimos a la clase el número de instancias obtenemos 2, igual que si se lo pedimos a la instancia 1 (que ha sido actualizada automáticamente) y a la instancia 2.