

# Kaj ti bo SPL, če imaš namespace?



10. jul. 2010 14:48

---

spletna\_postaja\programerji\GajCapuder

gaj.capuder@spletna-postaja.com

# Jaz

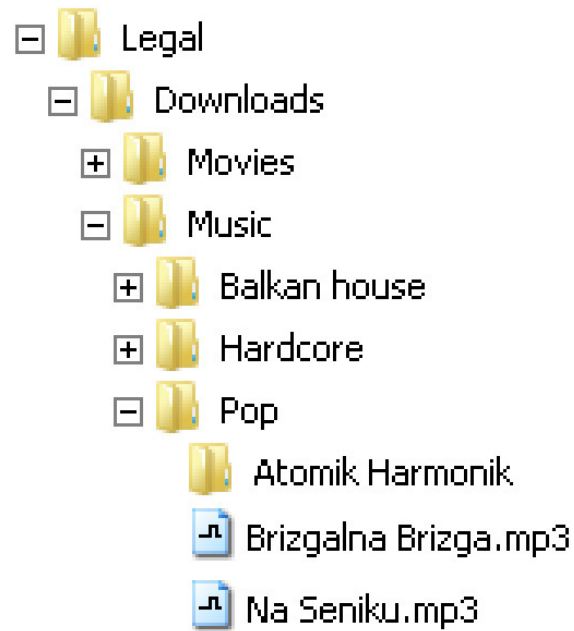
- Direktor razvoja v podjetju spletnaPOSTAJA
- Razvijalec Mišmasterja, Mišbossa, Miš...
- Avtor skoraj ~~±000~~ **1286** bolj ali manj konkretnih včasih ciničnih in nikoli preveč žaljivih php-si.com sporočil, skrivajoč pod psevdonimom "ace"
- Zend certificirani spletni programer
- ~~445.252~~ **595.434** vrstic kode v dobrih ~~6~~ **8** letih
- ~~24~~ **24** let, škorpijon



## Ti

- PHP 5 uporabnik
- I ❤️ OOP
- I ❤️ koda v prezentacijah

# Namespaces



## Namespaces - Wiki

*"For many programming languages, a namespace is a context for identifiers. In an operating system, an example of namespace is a directory. It contains items which must have unique names."*



## Namespaces - Problematika

- Podvajanje imen z internimi/eksternimi komponentami (SoapClient, Datetime, ...) => **Fatal error**
- Možnost okrajšave (alias) dolgih imen razredov/funkcij/konstant (Zend\_Http\_Client\_Adapter\_Exception)
- Združevanje povezanih razredov/funkcij/konstant

## Namespaces - Definicija

```
01 <?php /* framework/utils/DateTime.php */
02 namespace framework\utils;
03 class DateTime {
04
05 }
06
07 namespace framework\common;
08 class DateTime {
09
10 }
```

## Namespaces - Uporaba

```
1 <?php
2 include('framework/utils/DateTime.php');
3
4 new framework\utils\DateTime; // object(framework\utils\DateTime)
5 new DateTime; // object(DateTime)
```

## Namespaces - Uporaba v namespace-u

```
1 <?php
2 namespace calendar;
3 include('framework/utils/DateTime.php');
4
5 new framework\utils\DateTime; // 'calendar\framework\utils\DateTime'
  not found
6 new \framework\utils\DateTime; // object(framework\utils\DateTime)
7 new DateTime; // 'calendar\DateTime' not found
8 new \DateTime; // object(DateTime)
```

## Namespaces - Uvoz

```
01 <?php
02 namespace calendar;
03 use framework\utils\DateTime;
04 use \DateTime as InternalDateTime;
05 use framework\utils as Utils;
06 include('framework/utils/DateTime.php');
07
08 new DateTime; // object(framework\utils\DateTime)
09 new InternalDateTime; // object(DateTime)
10 new Utils\DateTime; // object(framework\utils\DateTime)
```

## Namespaces - Funkcije

```
1 <?php
2 namespace framework\utils;
3
4 file_get_contents('file.txt'); // failed to open stream
```

```
01 <?php
02 namespace framework\utils;
03
04 function file_get_contents() {
05     echo "owned";
06 }
07
08 file_get_contents('file.txt'); // owned
09 \file_get_contents('file.txt'); // failed to open stream
10 \framework\utils\file_get_contents('file.txt'); // owned
```

## Namespaces - Konstante

```
1 <?php
2 namespace framework\utils;
3
4 echo PHP_VERSION; // 5.3.2
```

```
1 <?php
2 namespace framework\utils;
3
4 define('PHP_VERSION', '6'); // PHP_VERSION already defined
5 const PHP_VERSION = '6';
6 echo PHP_VERSION; // 6
7 echo \PHP_VERSION; // 5.3.2
```

## Namespaces - Dinamika

```
01 <?php
02 namespace framework\utils;
03 class Files {
04
05 }
06
07 $className = 'Files';
08 new $className; // 'Files' not found
09
10 $className = 'framework\utils\Files';
11 new $className; // object(framework\utils\Files)
12
13 $className = __NAMESPACE__ . '\Files';
14 new $className; // object(framework\utils\Files)
15
16 new namespace\Files; // object(framework\utils\Files)
```

## Namespaces - Autoload!

```
1 <?php /* autoload.php */
2 spl_autoload_register(function($class){
3     $fileName = str_replace('\\', DIRECTORY_SEPARATOR, $class);
4     $path = __DIR__ . DIRECTORY_SEPARATOR . $fileName . '.php';
5     if (is_readable($path)) {
6         return include($path);
7     }
8 })
```

```
1 <?php
2 namespace calendar;
3 use framework\utils\DateTime;
4 include('../autoload.php');
5
6 new DateTime; // framework/utils/DateTime.php
7 new \framework\utils\Files; // framework/utils/Files.php
```

## Namespaces - Migracija



---

spletna\_postaja\programerji\GajCapuder

gaj.capuder@spletna-postaja.com

# Namespaces - Migracija 1

## Prej

```
1 <?php /* Zend/Http/Client/Adapter/Exception.php */
2 class Zend_Http_Client_Adapter_Exception {
3
4 }
```

## Potem

```
1 <?php /* Zend/Http/Client/Adapter/Exception.php */
2 namespace Zend\Http\Client\Adapter;
3 class Exception {
4
5 }
```

```
1 <?php
2 use Zend\Http\Client\Adapter\Exception;
3 new Exception;
```

## Namespaces - Migracija 2

### Prej

```
1 <?php /* project/plugins/news/news.php */
2 class News {
3
4 }
```

### Potem

```
1 <?php /* project/plugins/news/news.php */
2 namespace project\plugins\news;
3 class News {
4
5 }
```

```
1 <?php
2 use project\plugins\news\News;
3 new News;
```

## Namespaces - Bottom line

- Boljša organizacija kode
- Nekonfliktna koda
- Pregled nad odvisnostmi (dependencies)
- ~~Zend\_Http\_Client\_Adapter\_Exception~~ **Exception**
- include()?
- Dokumentacija [php.net/namespaces](http://php.net/namespaces)



## SPL - Manj pomembne informacije

- 12. 2. 2004
- PHP ~~5.0~~ **5.3**
- Paket 66 razredov (class) in vmesnikov (interface) + 13 funkcij
- Zend engine integracija
- [php.net/~helly/php/ext/spl/](http://php.net/~helly/php/ext/spl/)

## SPL - All inclusive

- Iterator-ji
- Informacije o datotečnem sistemu
- Objektni array
- Podatkovne strukture
- Organizacija exception-ov

# SPL - Nepregleden abecedni seznam

AppendIterator	FindFile	RecursiveDualIterator
ArrayAccess	InfiniteIterator	RecursiveFilterIterator
ArrayIterator	IniGroups	RecursiveIterator
ArrayObject	InvalidArgumentException	RecursiveIteratorIterator
BadFunctionCallException	Iterator	RecursiveRegexIterator
BadMethodCallException	IteratorAggregate	RecursiveTreeIterator
CachingIterator	IteratorIterator	RegexFindFile
CachingRecursiveIterator	KeyFilter	RegexIterator
CallbackFilterIterator	LengthException	RuntimeException
Countable	LimitIterator	SearchIterator
DbalArray	LogicException	SeekableIterator
DbalReader	NoRewindIterator	Serializable
DirectoryFilterDots	OuterIterator	SimpleXMLIterator
DirectoryGraphIterator	OutOfBoundsException	SplFileInfo
DirectoryIterator	OutOfRangeException	SplFileObject
DirectoryTree	OverflowException	SplObjectStorage
DirectoryTreeIterator	ParentIterator	SplObserver
DomainException	RangeException	SplSubject
DualIterator	RecursiveArrayIterator	SubClasses
EmptyIterator	RecursiveCachingIterator	Traversable
Exception	RecursiveCompareDualIterator	UnderflowException
FilterIterator	RecursiveDirectoryIterator	UnexpectedValueException

# SPL - Too much!

AppendIterator	FindFile	RecursiveDualIterator
ArrayAccess	InfiniteIterator	RecursiveFilterIterator
ArrayIterator	IniGroups	RecursiveIterator
ArrayObject	InvalidArgumentException	RecursiveIteratorIterator
BadFunctionCallException	Iterator	RecursiveRegexIterator
BadMethodCallException	IteratorAggregate	RecursiveTreeIterator
CachingIterator	IteratorIterator	RegexFindFile
CachingRecursiveIterator	KeyFilter	RegexIterator
CallbackFilterIterator	LengthException	RuntimeException
Countable	LimitIterator	SearchIterator
Dbarray	LogicException	SeekableIterator
Dbareader	NoRewindIterator	Serializable
DirectoryFilterDots	OuterIterator	SimpleXMLIterator
DirectoryGraphIterator	OutOfBoundsException	SplFileInfo
DirectoryIterator	OutOfRangeException	SplFileObject
DirectoryTree	OverflowException	SplObjectStorage
DirectoryTreeIterator	ParentIterator	SplObserver
DomainException	RangeException	SplSubject
DualIterator	RecursiveArrayIterator	SubClasses
EmptyIterator	RecursiveCachingIterator	Traversable
Exception	RecursiveCompareDualIterator	UnderflowException
FilterIterator	RecursiveDirectoryIterator	UnexpectedValueException

## SPL - Iterators

- Algoritmi za listanje/filtriranje/operiranje z seti podatkov
- Poljubne strukture podatkov (DB rezultati, menijska struktura, paginacija, datumi, ..)
- Datotečni sistem

```
1 Iterator extends Traversable {
2     abstract public mixed current()
3     abstract public scalar key ()
4     abstract public void next ()
5     abstract public void rewind ()
6     abstract public boolean valid ()
7 }
```

## SPL - ArrayIterator

- PHP 5 omogoča listanje čez vse vidne property-je
- Iterator omogoča nadzorovano/poljubno listanje

```
1 <?php
2 $data = array('a', 'b', 'c', 'd', 'e');
3 $arrayIterator = new ArrayIterator($data);
4 while($arrayIterator->valid()) {
5     echo $arrayIterator->current() . " "; // a b c d e
6     $arrayIterator->next();
7 }
8 $arrayIterator->rewind();
```

```
1 <?php
2 $data = array('a', 'b', 'c', 'd', 'e');
3 $arrayIterator = new ArrayIterator($data);
4 foreach ($arrayIterator as $current) {
5     echo $current . " "; // a b c d e
6 }
```

## SPL - LimitIterator

- array\_slice()
- Paginacija

```
1 <?php
2 $data = array('a', 'b', 'c', 'd', 'e');
3 $arrayIterator = new ArrayIterator($data);
4 $limitIterator = new LimitIterator($arrayIterator, 1, 2);
5 foreach ($limitIterator as $value) {
6     echo $value . " "; // b c
7 }
```

# SPL - InfiniteIterator

- Kombinatorika

```
1 <?php
2 $data = array('a', 'b', 'c', 'd', 'e');
3 $arrayIterator = new ArrayIterator($data);
4 $limitIterator = new LimitIterator($arrayIterator, 1, 2);
5 $infiniteIterator = new InfiniteIterator($limitIterator);
6 foreach ($infiniteIterator as $value) {
7     echo $value . " "; // b c b c b c b c b c b c ...
8 }
```

```
1 <?php
2 $data = array('a', 'b', 'c', 'd', 'e');
3 $arrayIterator = new ArrayIterator($data);
4 $infiniteIterator = new InfiniteIterator($arrayIterator);
5 $limitIterator = new LimitIterator($infiniteIterator, 0, 10);
6 foreach ($limitIterator as $value) {
7     echo $value . " "; // a b c d e a b c d e
8 }
```

## SPL - RecursiveArrayIterator

- Hierarhije
- Menijska struktura

```
01 <?php
02 $data = array(
03     'a',
04     'b' => array(1, 2, 3),
05     'c',
06     'd' => array(4, 5),
07     'e'
08 );
09 $iterator = new RecursiveArrayIterator($data);
10 $recursiveIterator = new RecursiveIteratorIterator($iterator);
11 foreach ($recursiveIterator as $value) {
12     echo $recursiveIterator->getDepth() . "-" . $value . " ";
13     // 0-a 1-1 1-2 1-3 0-c 1-4 1-5 0-e
14 }
```

## Bolijo oči?



## SPL - DirectoryIterator

- Rahlo kompleksno na prvi pogled?

```
01 <?php
02 $dir = "/home";
03 if (is_dir($dir)) {
04     if ($dh = opendir($dir)) {
05         while (($file = readdir($dh)) !== false) {
06             echo $file . "\n";
07         }
08         closedir($dh);
09     }
10 }
```

## SPL - DirectoryIterator

- Preglednejše
- Krajše
- Varnejše
- Catchable (UnexpectedValueException)

```
1 <?php
2 $dir = "/home";
3 foreach (new DirectoryIterator($dir) as $file) {
4     echo $file . "\n";
5 }
```

## SPL - SplFileInfo

- `filemtime()`, `filesize()`, ...

```
01 <?php
02 $dir = '/home/project/';
03 foreach (new DirectoryIterator($dir) as $file) {
04     if ($file->isDot() || $file->isDir()) continue;
05     echo $file->getFilename(); // index.php
06     echo $file->getPath(); // /home/project/
07     echo $file->getRealPath(); // /home/project/index.php
08     echo $file->getSize(); // 27216
09     echo $file->getMTime(); // 1278616413
10     echo $file->isReadable(); // 1
11     echo $file->isWritable(); // 1
12     echo $file->getOwner(); // 100
13 }
```

# SPL - RecursiveDirectoryIterator

- Podirektojska struktura

```
1 <?php
2 $dir = "/home";
3 $iterator = new RecursiveDirectoryIterator($dir);
4 foreach (new RecursiveIteratorIterator($iterator) as $file) {
5     echo $file . "\n";
```

## SPL - RecursiveDirectoryIterator on steroids

- Limitirano regex rekurzivno iskanje datotek

```
1 <?php
2 $dir = "/home";
3 $iterator = new RecursiveDirectoryIterator($dir);
4 $recursiveIterator = new RecursiveIteratorIterator($iterator);
5 $regex = new RegexIterator($recursiveIterator, '#^.+\.php$#i');
6 $limitIterator = new LimitIterator($regex, 0, 10);
7 foreach ($limitIterator as $file) {
8     echo $file . "\n"; // Prvih 10 *.php
9 }
```

## SPL - ArrayObject

Slika, ki predstavlja objekt, ki je hkrati array.

## SPL - ArrayObject

- Hitrejši/alternativen dostop do podatkov objekta
- Template engines

```
01 ArrayObject {
02     /* IteratorAggregate */
03     public function getIterator();
04     /* ArrayAccess */
05     public function offsetGet($name);
06     public function offsetSet($name, $value);
07     public function offsetExists($name);
08     public function offsetUnset($name);
09     /* Countable */
10     public function count();
11     /* Serializable */
12     public function serialize();
13     public function unserialize();
14     ...
15 }
```

## SPL - ArrayObject

```
01 <?php
02 $array = array(
03     'ena' => 1,
04     'dva' => 2,
05     'tri' => 3,
06 );
07 $object = new ArrayObject($array);
08 $array['ena'] = 4; // $object->offsetSet('ena', 4);
09 $object->asort();
10 foreach ($object as $key => $value) {
11     echo "$key=>$value "; // dva=>2 tri=>3 ena=>4
12 }
13 echo key($object); // dva
14 echo $object->getIterator()->key(); // dva
15 echo count($object); // 3
```

## SPL - Primer - Config

```
01 <?php
02 class Config extends ArrayObject {
03     public function __construct($array) {
04         foreach ($array as $name => $value) {
05             if (is_array($value)) {
06                 $value = new static($value);
07             }
08             $this[$name] = $value;
09         }
10     }
11     public function __get($name) {
12         return $this[$name];
13     }
14 }
```

## SPL - Primer - Config

```
01 <?php
02 include('Config.php');
03
04 $data = array(
05     'db' => array(
06         'user' => 'root',
07         'pass' => '****',
08         'name' => 'baza',
09         'host' => 'localhost',
10     )
11 );
12 $config = new Config($data);
13
14 echo $config->db->user; // root
15 echo $config['db']['user']; // root
```

```
1 {*template.tpl*}
2
3 {$config->db->name}
4 {$config.db.name}
```

## SPL - Primer - Registry

```
01 <?php
02 class Registry implements ArrayAccess {
03     protected $objects = array();
04     public function setObject($name, $object) {
05         $this->objects[$name] = $object;
06     }
07     public function getObject($name) {
08         return $this->objects[$name];
09     }
10     public function offsetGet($name) {
11         return $this->getObject($name);
12     }
13     public function offsetSet($name, $value){
14         $this->setObject($name, $value);
15     }
16     public function offsetExists($name){
17         return isset($this->objects[$name]);
18     }
19     public function offsetUnset($name){
20         $this->setObject($name, null);
21     }
22 }
```

## SPL - Primer - Registry

```
01 <?php
02 include('Registry.php');
03 include('Config.php');
04
05 $data = array(
06     'db' => array(
07         'user' => 'root',
08         'pass' => '****',
09         'name' => 'baza',
10         'host' => 'localhost',
11     )
12 );
13 $registry = new Registry;
14 $registry->setObject('config', new Config($data));
15 echo $registry['config']['db']['user'];
```

```
1 {*template.tpl*}
2
3 {$registry.config.db.user}
```

## SPL - There is more

- LogicException(s), RuntimeException(s)
- SplQueue, SplStack, SplHeap, SplObjectStorage
- SPL funkcije
  - spl\_autoload\_\*
  - spl\_class\_\*
  - spl\_iterator\_\*

## SPL - Bottom line

- Preproste rešitve pogostih problemov
- Fleksibilna sintaksa
- Konsistentnost
- Dokumentacija?

## The End



- Pritožbe: [gaj.capuder@spletna-postaja.com](mailto:gaj.capuder@spletna-postaja.com)
- Vprašanja: ?
- Koda: [gaj.si/phpkonferenca/2010/](http://gaj.si/phpkonferenca/2010/) (pritisni "tab")

# Help

- [php.net/namespaces](http://php.net/namespaces)
- [www.phparch.com/2010/03/29/namespaces-in-php/](http://www.phparch.com/2010/03/29/namespaces-in-php/)
- [www.sitepoint.com/blogs/2009/07/13/php-53-namespaces-basics/](http://www.sitepoint.com/blogs/2009/07/13/php-53-namespaces-basics/)
  
- [php.net/~helly/php/ext/spl](http://php.net/~helly/php/ext/spl)
- [www.phpro.org/tutorials/Introduction-to-SPL.html](http://www.phpro.org/tutorials/Introduction-to-SPL.html)
- [devzone.zend.com/article/2565](http://devzone.zend.com/article/2565)
  
- [seld.be/notes/introducing-slippy-html-presentations](http://seld.be/notes/introducing-slippy-html-presentations)