Workshop PHP
Part 2

Liviu Bălan

```php
<?php

$cars = array('Volvo', "BMW", "Toyota");
var_dump($cars);
echo $cars[0] . "\n";

```

Terminal: Local

```
[liviu.balan@gen-docker75100-all-dev workshop]$ php 21-01-array.php
array(3) {
  [0]=>
  string(5) "Volvo"
  [1]=>
  string(3) "BMW"
  [2]=>
  string(6) "Toyota"
}
Volvo
[liviu.balan@gen-docker75100-all-dev workshop]$
```

```php
<?php

$cars = array('Volvo', "BMW", "Toyota");
$cars[3] = 'Dacia';
$cars[] = 'Trabant';
var_dump($cars);

```

Terminal: Local

```
[liviu.balan@gen-docker75100-all-dev workshop]$ php 21-02-array_indexed.php
array(5) {
  [0]=>
  string(5) "Volvo"
  [1]=>
  string(3) "BMW"
  [2]=>
  string(6) "Toyota"
  [3]=>
  string(5) "Dacia"
  [4]=>
  string(7) "Trabant"
}
[liviu.balan@gen-docker75100-all-dev workshop]$
```

```php
<?php

$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);

for ($x = 0; $x < $arrlength; $x++) {
    echo $cars[$x] . "\n";
}

```

Terminal: Local

```
[liviu.balan@gen-docker75100-all-dev workshop]$ php 21-03-array_indexed_for.php
Volvo
BMW
Toyota
[liviu.balan@gen-docker75100-all-dev workshop]$
```

```php
<?php

$age = array('Peter' => 35, 'Ben' => 37, 'Joe' => 43);
var_dump($age);
echo $age['Peter'] . "\n";

```

Terminal: Local

```
[liviu.balan@gen-docker75100-all-dev workshop]$ php 21-04-array_associative.php
array(3) {
  ["Peter"]=>
  int(35)
  ["Ben"]=>
  int(37)
  ["Joe"]=>
  int(43)
}
35
[liviu.balan@gen-docker75100-all-dev workshop]$
```

```php
<?php

$age = array('Peter' => 35, 'Ben' => 37, 'Joe' => 43);
foreach ($age as $key => $value) {
    echo 'Key=' . $key . ', Value=' . $value . "\n";
}

```

Terminal: Local

```
[liviu.balan@gen-docker75100-all-dev workshop]$ php 21-05-array_associative_foreach.php
Key=Peter, Value=35
Key=Ben, Value=37
Key=Joe, Value=43
[liviu.balan@gen-docker75100-all-dev workshop]$
```

```php
<?php

$cars = array(
    array('Volvo', 'Sweden'),
    array('BMW', 'Germany'),
);

var_dump($cars);
var_dump($cars[1][1]);

```

Terminal: Local

```
[liviu.balan@gen-docker75100-all-dev workshop]$ php 21-06-array_multidimensional.php
array(2) {
  [0]=>
  array(2) {
    [0]=>
    string(5) "Volvo"
    [1]=>
    string(6) "Sweden"
  }
  [1]=>
  array(2) {
    [0]=>
    string(3) "BMW"
    [1]=>
    string(7) "Germany"
  }
}
string(7) "Germany"
[liviu.balan@gen-docker75100-all-dev workshop]$
```

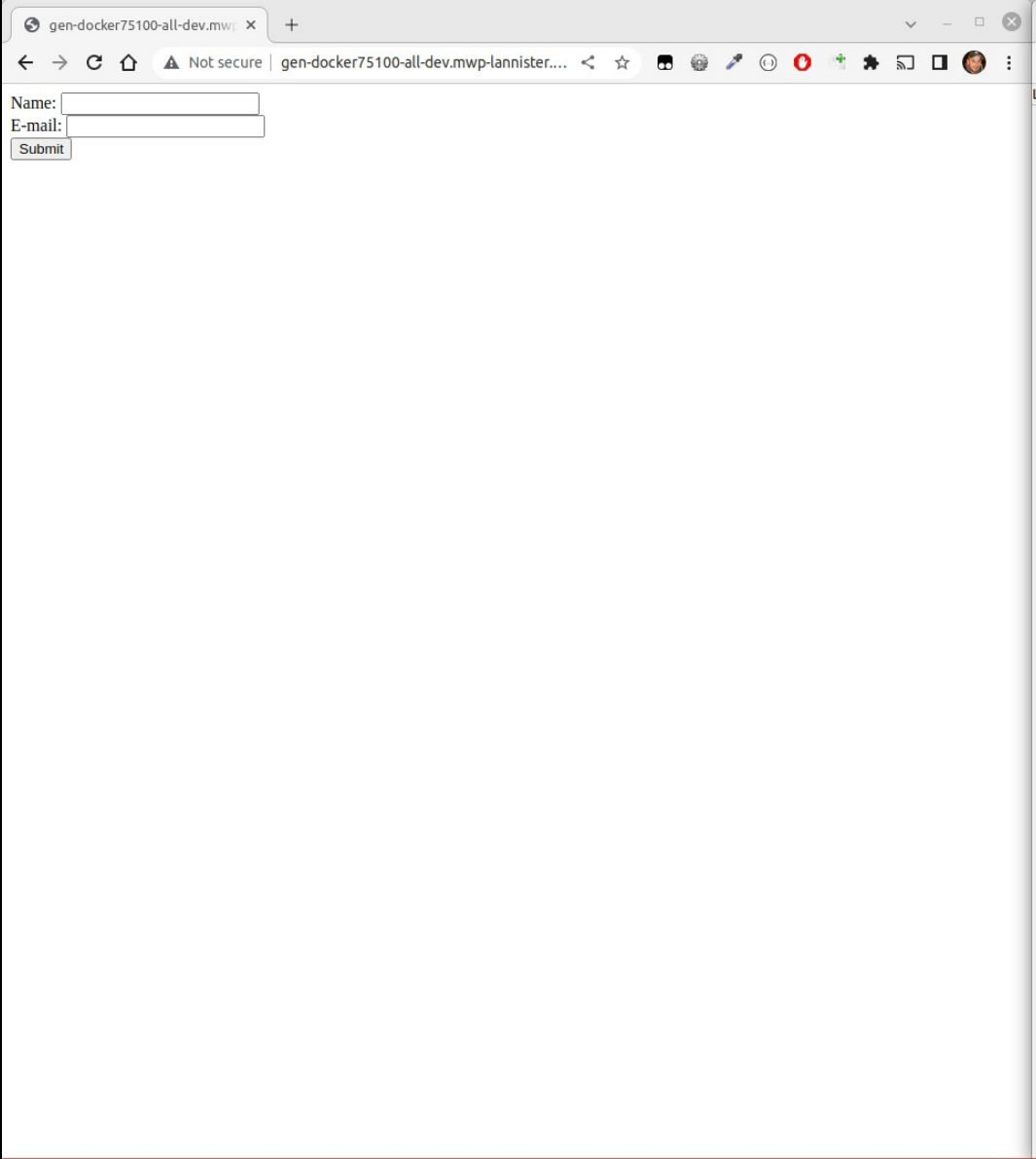
```php
<?php

$cars = array(
    array('Volvo', 'Sweden'),
    array('BMW', 'Germany'),
);

for ($i = 0; $i < count($cars); $i++) {
    for ($j = 0; $j < count($cars[$i]); $j++) {
        echo $cars[$i][$j] . ', ';
    }
}
echo "\n";

```

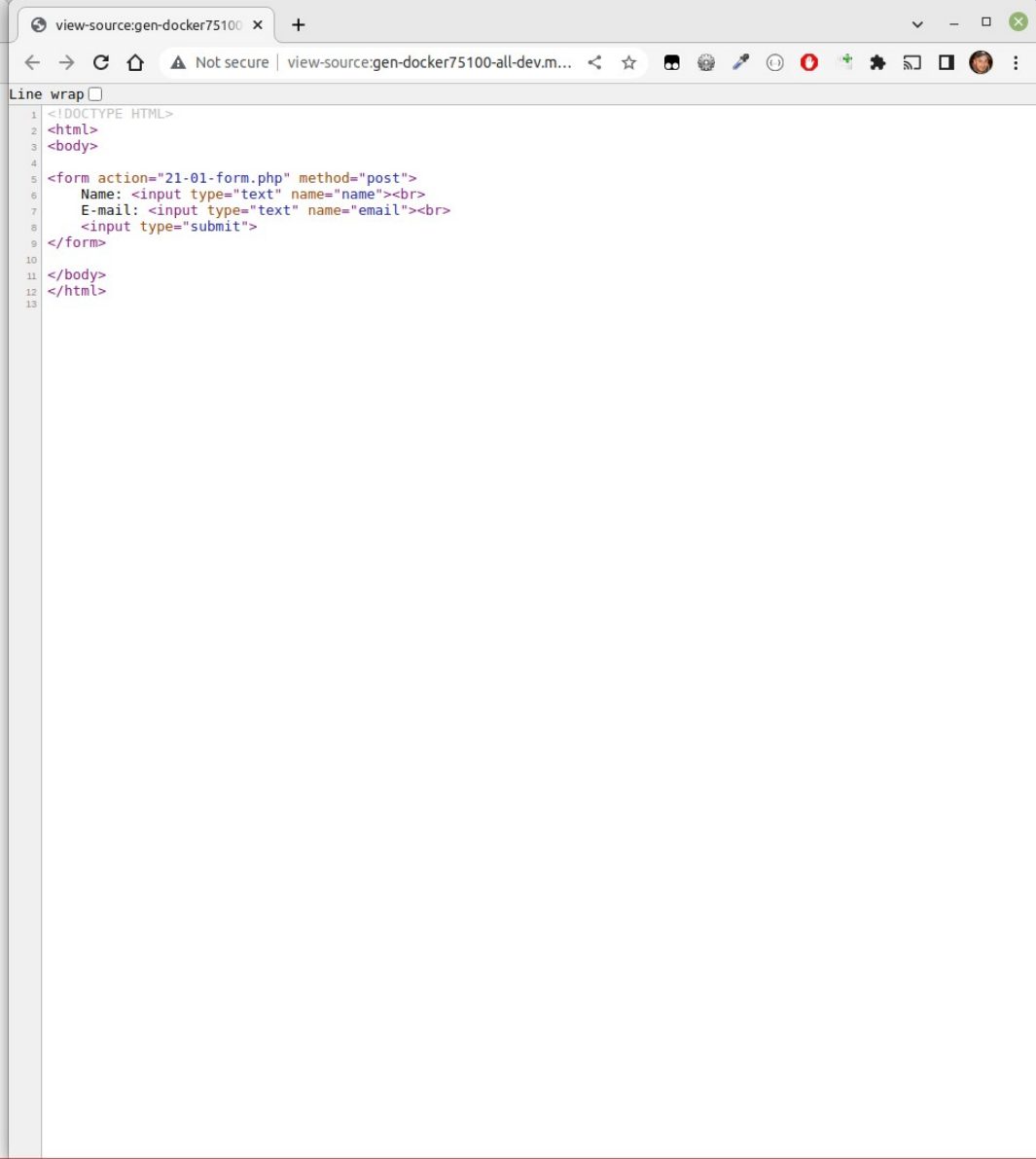
Terminal: Local

```
[liviu.balan@gen-docker75100-all-dev workshop]$ php 21-07-for.php
Volvo, Sweden, BMW, Germany,
[liviu.balan@gen-docker75100-all-dev workshop]$
```

```php
<?php

// See: https://www.php.net/manual/en/ref.array.php

```

```html
<!DOCTYPE HTML>
<html>
<body>

<form action="22-01-form.php" method="post">
    Name: <input type="text" name="name"><br>
    E-mail: <input type="text" name="email"><br>
    <input type="submit">
</form>

</body>
</html>

```


gen-docker75100-all-dev.mwp
Not secure
gen-docker75100-all-dev.mwp-lannister....
Name:
E-mail:
Submit


view-source:gen-docker75100
Not secure
view-source:gen-docker75100-all-dev.m...
Line wrap
<!DOCTYPE HTML>
<html>
<body>
<form action="21-01-form.php" method="post">
Name: <input type="text" name="name">
E-mail: <input type="text" name="email">
<input type="submit">
</form>
</body>
</html>

```
<html>
<body>

<p>
<?php
var_dump($_POST);
if (!empty($_POST)) {
    echo '<p>Name: ' . $_POST['name'] . '</p>';
}
?>
</p>
<form action="22-02-form.php" method="post">
    Name: <input type="text" name="name"><br>
    E-mail: <input type="text" name="email"><br>
    <input type="submit">
</form>

</body>
</html>

```

array(2) { ["name"]=> string(11) "Liviu Balan" ["email"]=> string(19) "liviu.balan@emag.ro" }

Name: Liviu Balan

Name:
E-mail:
Submit

GET

- Information displayed in the URL
- Send information limits (~2000 characters)
- It is possible to bookmark the page
- Not recommended for sending passwords or other sensitive information

POST

- Information invisible to others (embedded within the body of the HTTP request)
- No limits on the amount of information to send
- It is not possible to bookmark the page
- Supports advanced functionality such as support for multi-part binary input while uploading files to server

Attacks

- https://www.acunetix.com/blog/articles/injection-attacks/

What Are Injection Attacks

Ian Muscat | April 18, 2019

Injection attacks refer to a broad class of attack vectors. In an injection attack, an attacker supplies untrusted input to a program. This input gets processed by an interpreter as part of a command or query. In turn, this alters the execution of that program.

Injections are amongst the oldest and most dangerous attacks aimed at web applications. They can lead to data theft, data loss, loss of data integrity, denial of service, as well as full system compromise. The primary reason for injection vulnerabilities is usually insufficient user input validation.

This attack type is considered a major problem in web security. It is listed as the number one web application security risk in the OWASP Top 10 – and for a good reason. Injection attacks, particularly SQL Injections (SQLi attacks) and Cross-site Scripting (XSS), are not only very dangerous but also widespread, especially in legacy applications.

What makes injection vulnerabilities particularly scary is that the attack surface is enormous (especially for XSS and SQL Injection vulnerabilities). Furthermore, injection attacks are a very well understood vulnerability class. This means that there are many freely available and reliable tools that allow even inexperienced attackers to abuse these vulnerabilities automatically.

Types of Injection Attacks

SQL injection (SQLi) and Cross-site Scripting (XSS) are the most common injection attacks but they are not the only ones. The following is a list of common injection attack types.

Injection attack	Description	Potential impact
Code injection	The attacker injects application code written in the	Full system

```php
<?php

// See: https://www.w3schools.com/html/html_form_elements.asp

```

PHP OOP

- https://www.php.net/manual/en/language.oop5.php

Change language: English

Submit a Pull Request Report a Bug

Classes and Objects

Table of Contents

User Contributed Notes

+ add a note

OOP

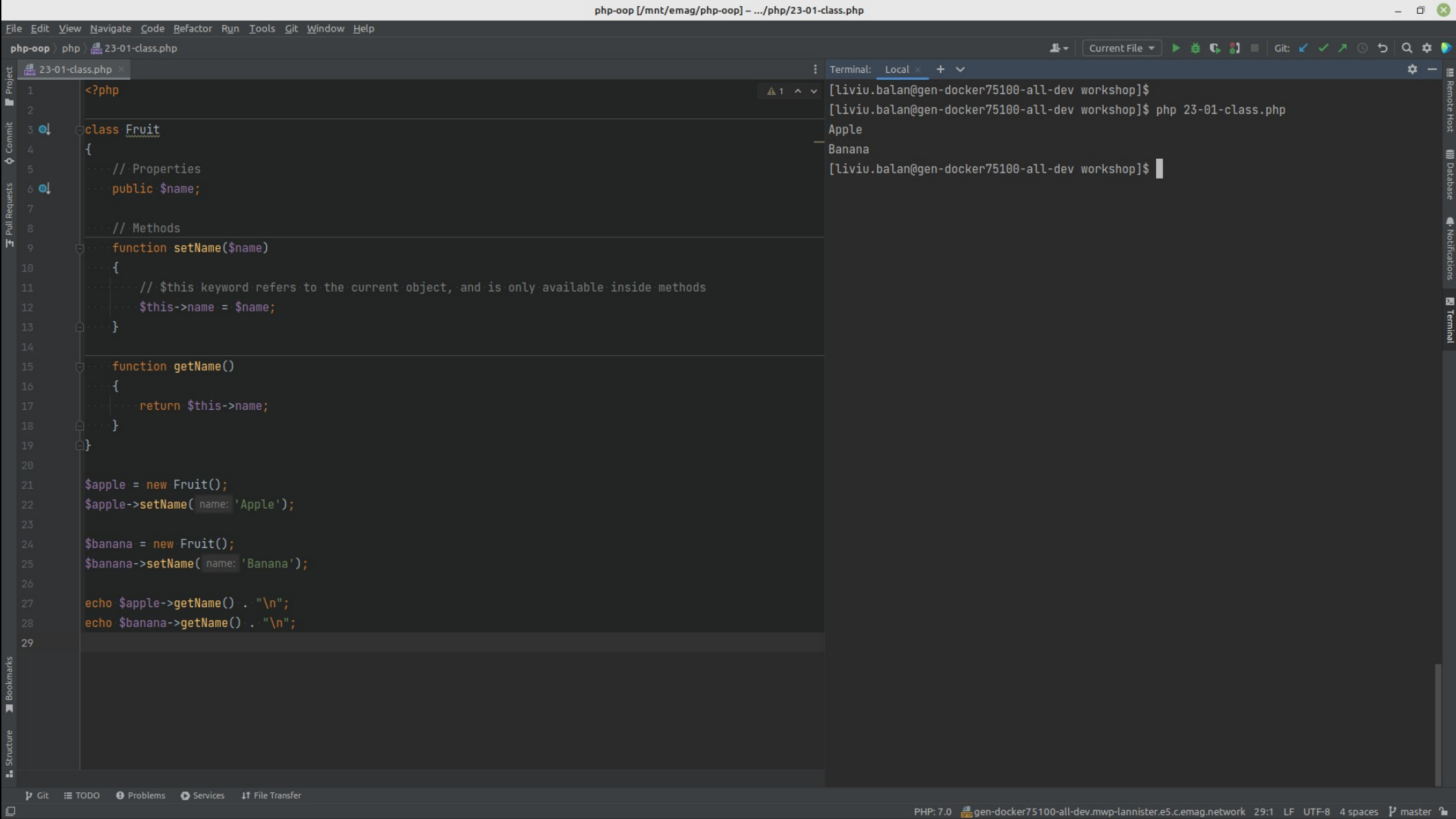
- Provides a clear structure for the programs
- DRY "Don't Repeat Yourself"
- Makes the code easier to maintain, modify and debug
- Create full reusable applications with less code and shorter development time

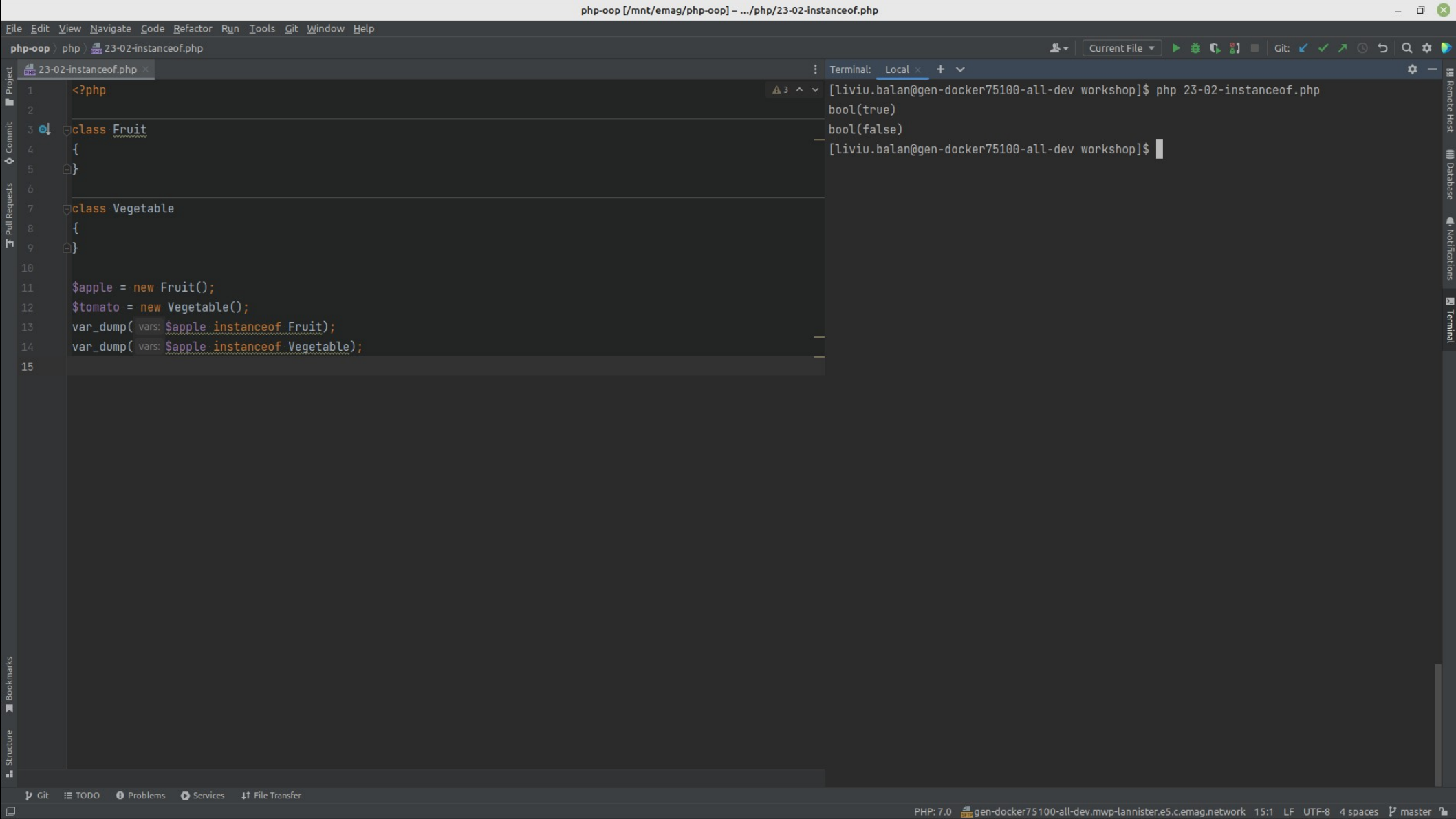
Classes and Objects

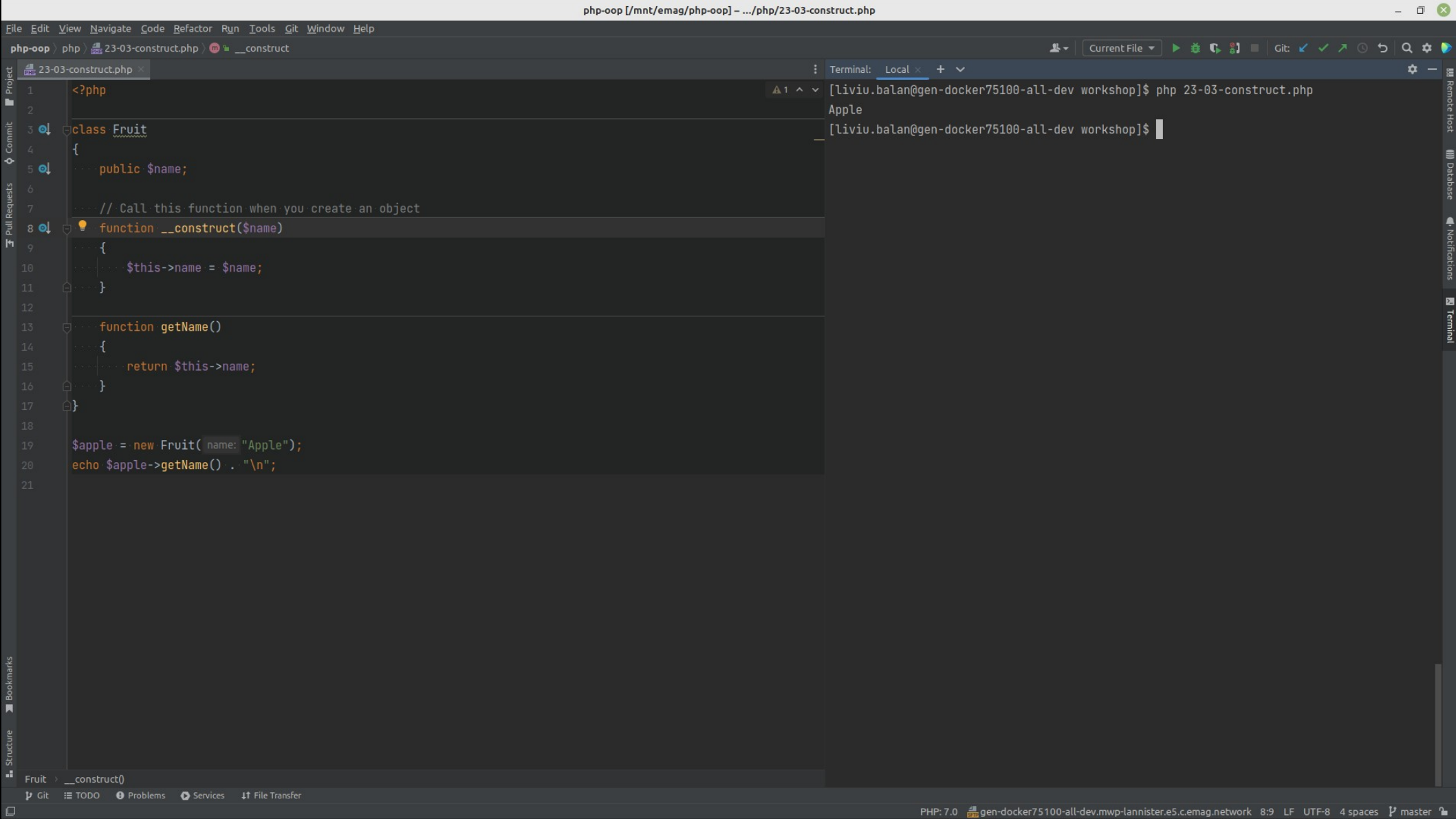
- Class is a template for objects
- Object is an instance of a class

Classes and Objects

class	objects
Fruit	apple
	banana
	mango


php-oop [/mnt/emag/php-oop] – .../php/23-01-class.php
File Edit View Navigate Code Refactor Run Tools Git Window Help
php-oop php 23-01-class.php
Current File
Git:
23-01-class.php
<?php
class Fruit
{
// Properties
public $name;
// Methods
function setName($name)
{
// $this keyword refers to the current object, and is only available inside methods
$this->name = $name;
}
function getName()
{
return $this->name;
}
}
$apple = new Fruit();
$apple->setName(name: 'Apple');
$banana = new Fruit();
$banana->setName(name: 'Banana');
echo $apple->getName() . "\n";
echo $banana->getName() . "\n";
Terminal: Local
[liviu.balan@gen-docker75100-all-dev workshop]$
[liviu.balan@gen-docker75100-all-dev workshop]$ php 23-01-class.php
Apple
Banana
[liviu.balan@gen-docker75100-all-dev workshop]$
Project
Commit
Pull Requests
Bookmarks
Structure
Remote Host
Database
Notifications
Terminal
Git
TODO
Problems
Services
File Transfer
PHP: 7.0
gen-docker75100-all-dev.mwp-lannister.e5.c.emag.network
29:1
LF
UTF-8
4 spaces
master


php-oop [/mnt/emag/php-oop] – .../php/23-02-instanceof.php
File Edit View Navigate Code Refactor Run Tools Git Window Help
php-oop php 23-02-instanceof.php
Current File
Git:
23-02-instanceof.php
<?php
class Fruit
{
}
class Vegetable
{
}
$apple = new Fruit();
$tomato = new Vegetable();
var_dump(vars: $apple instanceof Fruit);
var_dump(vars: $apple instanceof Vegetable);
Terminal: Local
[liviu.balan@gen-docker75100-all-dev workshop]$ php 23-02-instanceof.php
bool(true)
bool(false)
[liviu.balan@gen-docker75100-all-dev workshop]$
Project
Commit
Pull Requests
Bookmarks
Structure
Remote Host
Database
Notifications
Terminal
Git
TODO
Problems
Services
File Transfer
PHP: 7.0
gen-docker75100-all-dev.mwp-lannister.e5.c.emag.network
15:1
LF
UTF-8
4 spaces
master


php-oop [/mnt/emag/php-oop] – .../php/23-03-construct.php
File Edit View Navigate Code Refactor Run Tools Git Window Help
php-oop › php › 23-03-construct.php › __construct
Current File
Git:
23-03-construct.php
<?php
class Fruit
{
public $name;
// Call this function when you create an object
function __construct($name)
{
$this->name = $name;
}
function getName()
{
return $this->name;
}
}
$apple = new Fruit(name: "Apple");
echo $apple->getName() . "\n";
Terminal: Local
[liviu.balan@gen-docker75100-all-dev workshop]$ php 23-03-construct.php
Apple
[liviu.balan@gen-docker75100-all-dev workshop]$
Fruit › __construct()
Git TODO Problems Services File Transfer
PHP: 7.0 gen-docker75100-all-dev.mwp-lannister.e5.c.emag.network 8:9 LF UTF-8 4 spaces master

23-04-construct.php

```php
<?php

class Fruit
{
    public $name;
    public $color;

    function __construct($name, $color)
    {
        $this->name = $name;
        $this->color = $color;
    }

    function getName()
    {
        return $this->name;
    }

    function getColor()
    {
        return $this->color;
    }
}

$apple = new Fruit( name: "Apple", color: "red");
echo $apple->getName() . "\n";
echo $apple->getColor() . "\n";
```

Terminal: Local

```
[liviu.balan@gen-docker75100-all-dev workshop]$ php 23-04-construct.php
Apple
red
[liviu.balan@gen-docker75100-all-dev workshop]$
```

23-05-destruct.php

```php
<?php

class Fruit
{
    public $name;

    function __construct($name)
    {
        $this->name = $name;
    }

    function __destruct()
    {
        echo "The fruit is {$this->name}.\n";
    }
}

$apple = new Fruit( name: "Apple");

```

Terminal: Local

```
[liviu.balan@gen-docker75100-all-dev workshop]$ php 23-05-destruct.php
The fruit is Apple.
[liviu.balan@gen-docker75100-all-dev workshop]$
```

```php
<?php

// See: https://www.php.net/manual/en/language.oop5.magic.php

```

Change language: English
Submit a Pull Request Report a Bug

Magic Methods

Magic methods are special methods which override PHP's default's action when certain actions are performed on an object.

Caution All methods names starting with __ are reserved by PHP. Therefore, it is not recommended to use such method names unless overriding PHP's behavior.

The following method names are considered magical: __construct(), __destruct(), __call(), __callStatic(), __get(), __set(), __isset(), __unset(), __sleep(), __wakeup(), __serialize(), __unserialize(), __toString(), __invoke(), __set_state(), __clone(), and __debugInfo().

Warning All magic methods, with the exception of __construct(), __destruct(), and __clone(), *must* be declared as `public`, otherwise an **`E_WARNING`** is emitted. Prior to PHP 8.0.0, no diagnostic was emitted for the magic methods __sleep(), __wakeup(), __serialize(), __unserialize(), and __set_state().

Warning If type declarations are used in the definition of a magic method, they must be identical to the signature described in this document. Otherwise, a fatal error is emitted. Prior to PHP 8.0.0, no diagnostic was emitted. However, __construct() and __destruct() must not declare a return type; otherwise a fatal error is emitted.

__sleep() and __wakeup()

```
public __sleep(): array
```

```
public __wakeup(): void
```

serialize() checks if the class has a function with the magic name __sleep(). If so, that function is executed prior to any serialization. It can clean up the object and is supposed to return an array with the names of all variables of that object that should be serialized. If the method

Access Modifiers

- **public** - accessed from everywhere. Default
- **protected** - accessed within the class and by classes derived from that class
- **private** - can ONLY be accessed within the class

23-07-access_modifiers_property.php

```php
<?php


class Fruit
{
    public $name;
    protected $color;
    private $weight;
}

$mango = new Fruit();
$mango->id = 1; // OK
$mango->name = 'Mango'; // OK
//$mango->color = 'Yellow'; // ERROR
//$mango->weight = '300'; // ERROR

```

Terminal: Local

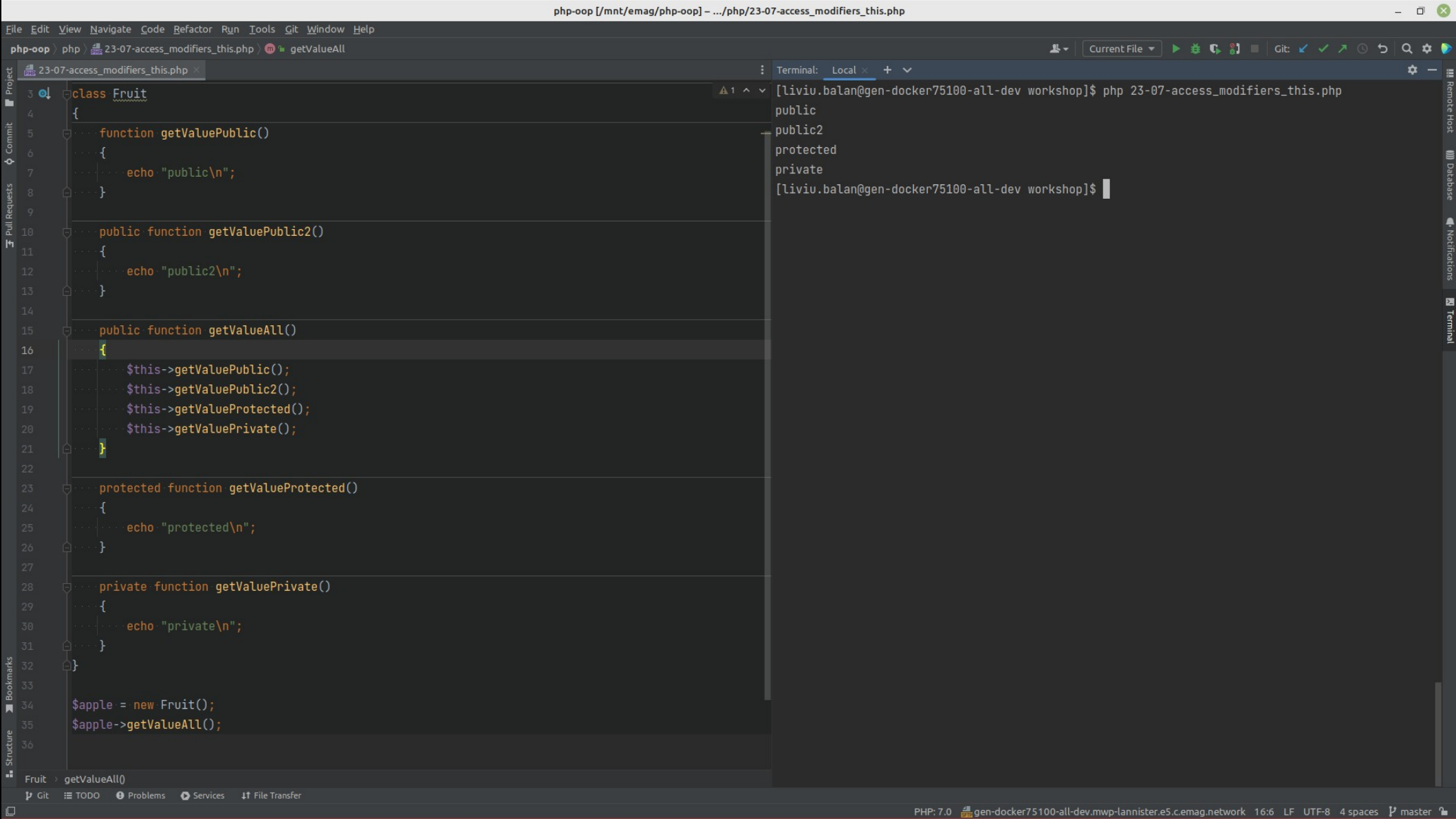
```
[liviu.balan@gen-docker75100-all-dev workshop]$ php 23-07-access_modifiers_property.php
[liviu.balan@gen-docker75100-all-dev workshop]$
```

23-07-access_modifiers_method.php

```php
<?php

class Fruit
{
    function getValuePublic()
    {
        echo "public\n";
    }

    public function getValuePublic2()
    {
        echo "public2\n";
    }

    protected function getValueProtected()
    {
        echo "protected\n";
    }

    private function getValuePrivate()
    {
        echo "private\n";
    }
}

$apple = new Fruit();
$apple->getValuePublic();
$apple->getValuePublic2();
//$apple->getValueProtected(); // Error
//$apple->getValuePrivate(); // Error
```

Terminal: Local

```
[liviu.balan@gen-docker75100-all-dev workshop]$ php 23-07-access_modifiers_method.php
public
public2
[liviu.balan@gen-docker75100-all-dev workshop]$
```


php-oop [/mnt/emag/php-oop] – .../php/23-07-access_modifiers_this.php
File Edit View Navigate Code Refactor Run Tools Git Window Help
php-oop › php › 23-07-access_modifiers_this.php › getValueAll
Current File
Git:
23-07-access_modifiers_this.php
class Fruit
{
function getValuePublic()
{
echo "public\n";
}
public function getValuePublic2()
{
echo "public2\n";
}
public function getValueAll()
{
$this->getValuePublic();
$this->getValuePublic2();
$this->getValueProtected();
$this->getValuePrivate();
}
protected function getValueProtected()
{
echo "protected\n";
}
private function getValuePrivate()
{
echo "private\n";
}
}
$apple = new Fruit();
$apple->getValueAll();
Terminal: Local
[liviu.balan@gen-docker75100-all-dev workshop]$ php 23-07-access_modifiers_this.php
public
public2
protected
private
[liviu.balan@gen-docker75100-all-dev workshop]$
Fruit › getValueAll()
Git TODO Problems Services File Transfer
PHP: 7.0 gen-docker75100-all-dev.mwp-lannister.e5.c.emag.network 16:6 LF UTF-8 4 spaces master

PSR-12

- https://groups.google.com/g/php-fig/c/DOeWi4lJQsA?pli=1
- https://www.php-fig.org/psr/psr-12/
-

PSR-12: Order of constants, methods and properties definitions

2320 views

Subscribe

Кирилл Фрейман
to PHP Framework Interoperability Group

Jun 4, 2017, 4:19:38 PM

Hi everyone!

In PHP de-facto exists some best practice of ordering methods and properties.
It seems to me that this order is most common:

```
Constants
    public const
    protected const
    private const
Properties
    public static properties
    public properties
    protected static properties
    protected properties
    private static properties
    private properties
Methods
    magic methods
    public static methods
    public methods
    protected static methods
    protected methods
    private static methods
    private methods
```

Should we standartize it?

Alessandro Lai
I agree that a sequence of constants-properties-methods is ok and expected. But apart of requiring

Jun 5, 2017, 10:14:13 AM

Andreas Möller
Should we standartize it? How about enforcing on a per-project-basis with https://github.com/

Jun 5, 2017, 2:06:02 PM

Alexander Makarov
Putting publicly visible properties before private ones is something new to me. On Sunday, June 4,

Jun 8, 2017, 11:58:29 AM

Stefano Torresi
I don't think this detail is worth standardising, ordering is very situational and can bear a lot

Jun 8, 2017, 12:42:56 PM

PSR-12: Extended Coding Style

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Overview

This specification extends, expands and replaces PSR-2, the coding style guide and requires adherence to PSR-1, the basic coding standard.

Like PSR-2, the intent of this specification is to reduce cognitive friction when scanning code from different authors. It does so by enumerating a shared set of rules and expectations about how to format PHP code. This PSR seeks to provide a set way that coding style tools can implement, projects can declare adherence to and developers can easily relate to between different projects. When various authors collaborate across multiple projects, it helps to have one set of guidelines to be used among all those projects. Thus, the benefit of this guide is not in the rules themselves but the sharing of those rules.

PSR-2 was accepted in 2012 and since then a number of changes have been made to PHP which has implications for coding style guidelines. Whilst PSR-2 is very comprehensive of PHP functionality that existed at the time of writing, new functionality is very open to interpretation. This PSR, therefore, seeks to clarify the content of PSR-2 in a more modern context with new functionality available, and make the errata to PSR-2 binding.

Previous language versions

Throughout this document, any instructions MAY be ignored if they do not exist in versions of PHP supported by your project.

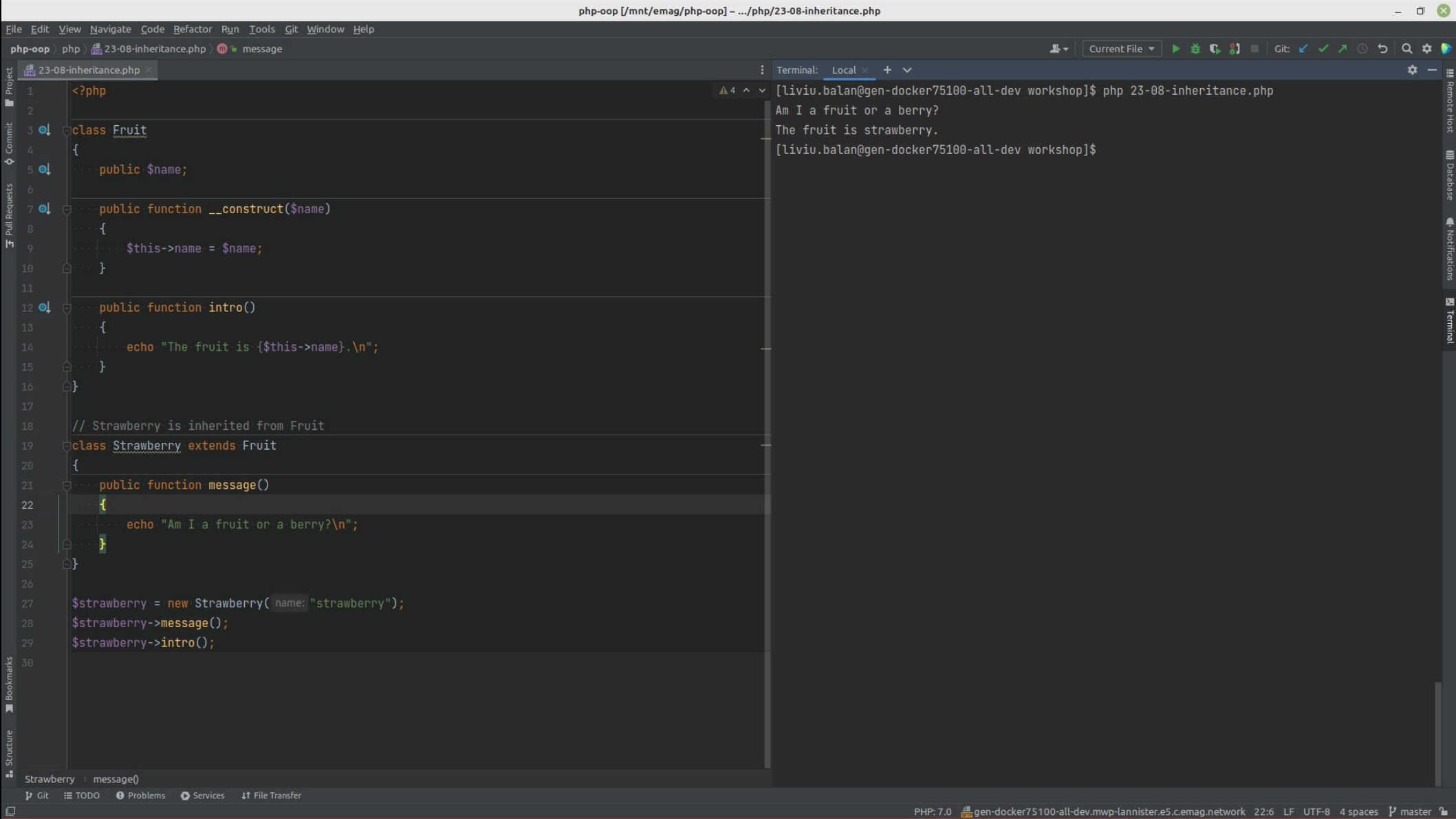
Additional Info:

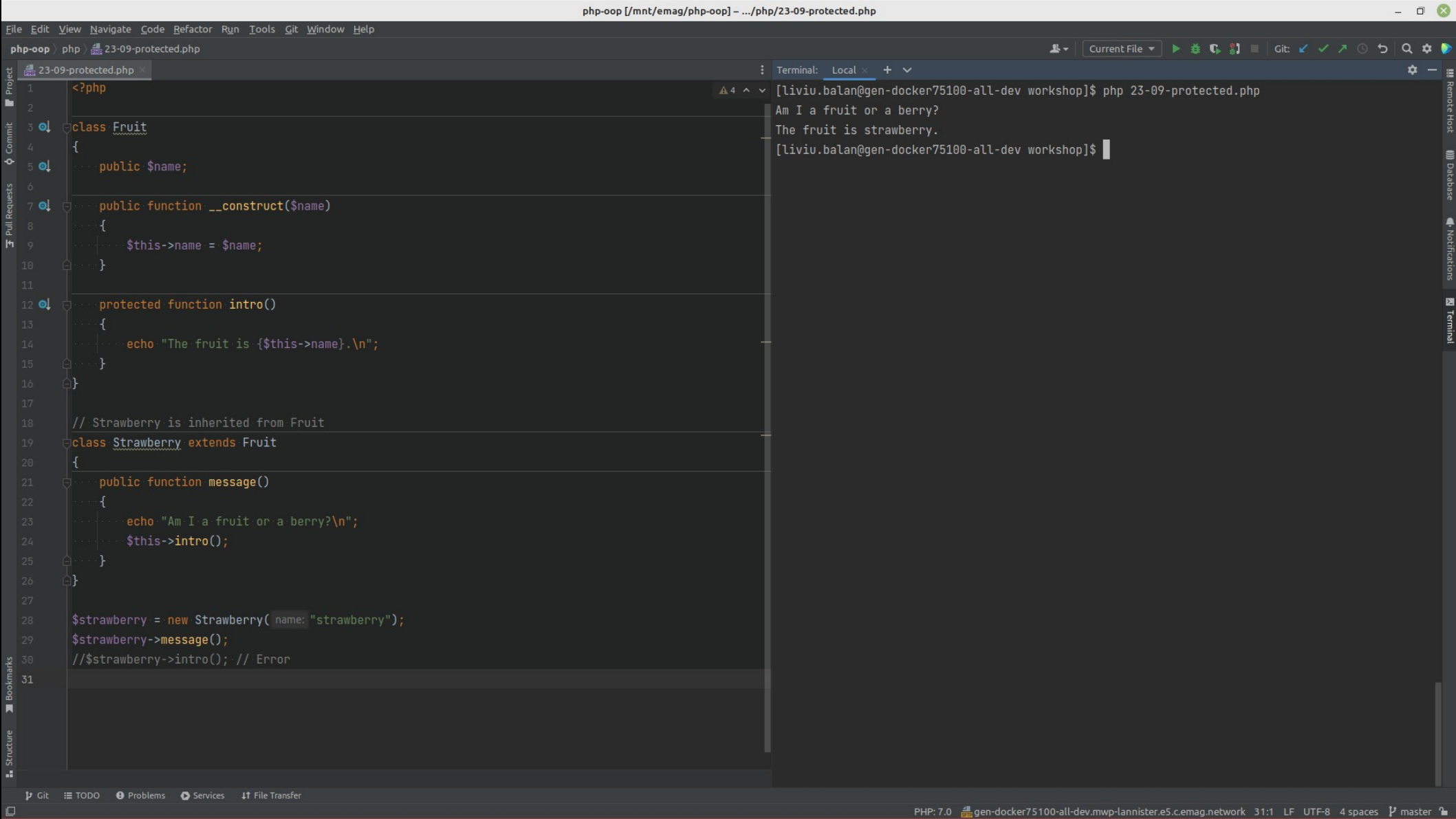
PSR-12: Extended Coding Style

PSR-12 Meta Document

Inheritance

- Inheritance in OOP = When a class derives from another class
- An inherited class is defined by using the **extends** keyword


php-oop [/mnt/emag/php-oop] – .../php/23-08-inheritance.php
File Edit View Navigate Code Refactor Run Tools Git Window Help
php-oop › php › 23-08-inheritance.php › message
Current File
Git:
23-08-inheritance.php
<?php
class Fruit
{
public $name;
public function __construct($name)
{
$this->name = $name;
}
public function intro()
{
echo "The fruit is {$this->name}.\n";
}
}
// Strawberry is inherited from Fruit
class Strawberry extends Fruit
{
public function message()
{
echo "Am I a fruit or a berry?\n";
}
}
$strawberry = new Strawberry(name: "strawberry");
$strawberry->message();
$strawberry->intro();
Terminal: Local
[liviu.balan@gen-docker75100-all-dev workshop]$ php 23-08-inheritance.php
Am I a fruit or a berry?
The fruit is strawberry.
[liviu.balan@gen-docker75100-all-dev workshop]$
Strawberry › message()
Git TODO Problems Services File Transfer
PHP: 7.0 gen-docker75100-all-dev.mwp-lannister.e5.c.emag.network 22:6 LF UTF-8 4 spaces master

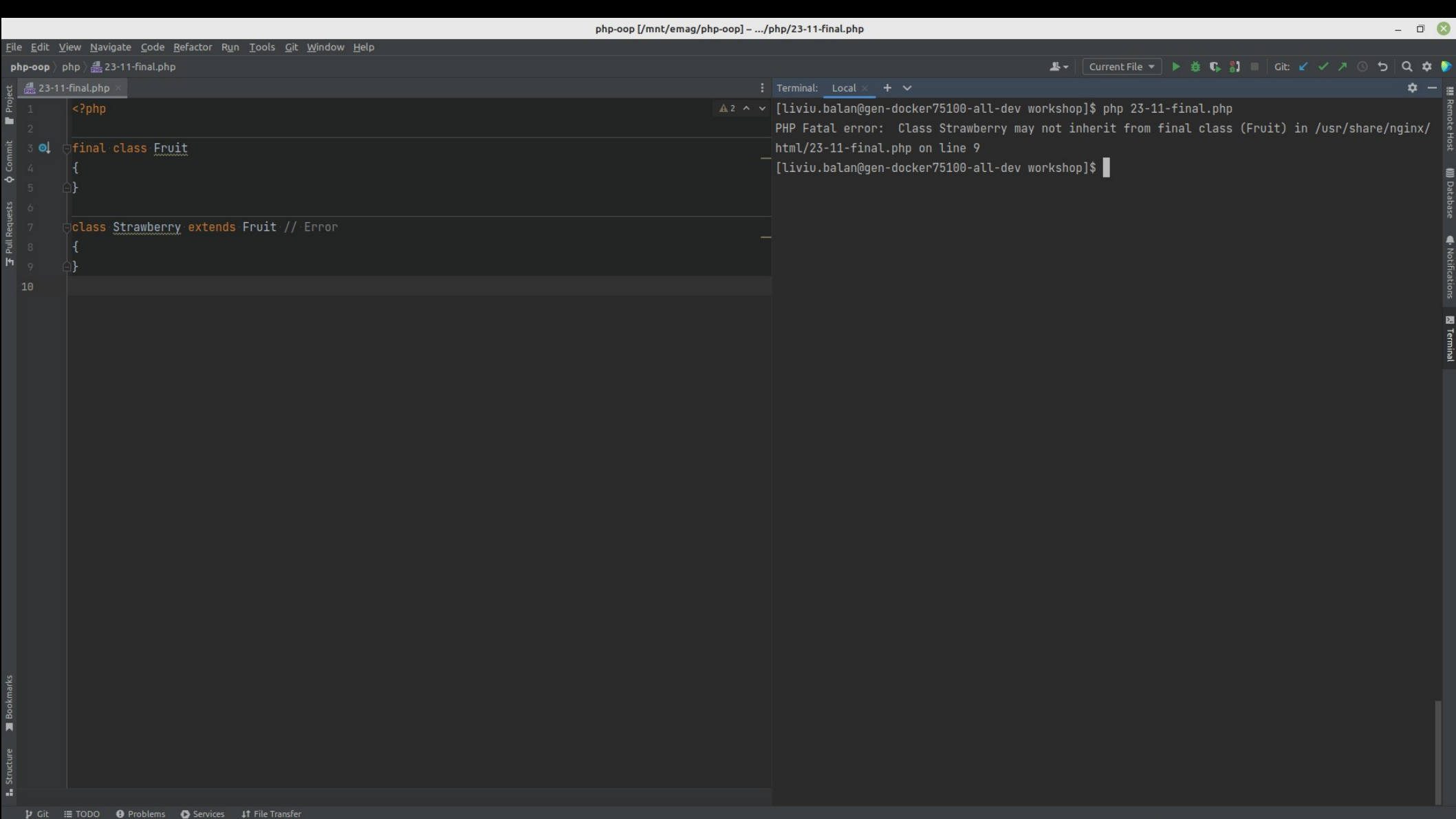

php-oop [/mnt/emag/php-oop] – .../php/23-09-protected.php
File Edit View Navigate Code Refactor Run Tools Git Window Help
php-oop php 23-09-protected.php
Current File
Git:
23-09-protected.php
<?php
class Fruit
{
public $name;
public function __construct($name)
{
$this->name = $name;
}
protected function intro()
{
echo "The fruit is {$this->name}.\n";
}
}
// Strawberry is inherited from Fruit
class Strawberry extends Fruit
{
public function message()
{
echo "Am I a fruit or a berry?\n";
$this->intro();
}
}
$strawberry = new Strawberry(name: "strawberry");
$strawberry->message();
//$strawberry->intro(); // Error
Terminal: Local
[liviu.balan@gen-docker75100-all-dev workshop]$ php 23-09-protected.php
Am I a fruit or a berry?
The fruit is strawberry.
[liviu.balan@gen-docker75100-all-dev workshop]$
Project
Commit
Pull Requests
Bookmarks
Structure
Remote Host
Database
Notifications
Terminal
Git
TODO
Problems
Services
File Transfer
PHP: 7.0
gen-docker75100-all-dev.mwp-lannister.e5.c.emag.network
31:1
LF
UTF-8
4 spaces
master

23-10-override.php

```php
<?php

class Fruit
{
    public $name;

    public function __construct($name) {
        $this->name = $name;
    }

    public function intro() {
        echo "The fruit is {$this->name}.\n";
    }
}

// Strawberry is inherited from Fruit
class Strawberry extends Fruit
{
    public $color;

    public function __construct($name, $color) {
        $this->name = $name;
        $this->color = $color;
    }

    public function intro() {
        echo "The fruit is {$this->name} and {$this->color}.\n";
    }
}

$strawberry = new Strawberry( name: "strawberry",  color: "red");
$strawberry->intro();
```

Terminal: Local

```
[liviu.balan@gen-docker75100-all-dev workshop]$ php 23-10-override.php
The fruit is strawberry and red.
[liviu.balan@gen-docker75100-all-dev workshop]$
```


php-oop [/mnt/emag/php-oop] – .../php/23-11-final.php
File Edit View Navigate Code Refactor Run Tools Git Window Help
php-oop php 23-11-final.php
Current File
Git:
23-11-final.php
<?php
final class Fruit
{
}
class Strawberry extends Fruit // Error
{
}
Terminal: Local
[liviu.balan@gen-docker75100-all-dev workshop]$ php 23-11-final.php
PHP Fatal error: Class Strawberry may not inherit from final class (Fruit) in /usr/share/nginx/html/23-11-final.php on line 9
[liviu.balan@gen-docker75100-all-dev workshop]$
Project
Commit
Pull Requests
Bookmarks
Structure
Remote Host
Database
Notifications
Terminal
Git
TODO
Problems
Services
File Transfer

Variables Scope

- Local
- Global
- Static

23-12-variables_scope.php

```php
<?php

$type = 'apple'; // global scope
class Fruit
{
    public function getGlobal() {
        // echo $type; // Error
        echo $GLOBALS['type'] . "\n";
    }

    public function getLocal() {
        $type = 'banana';
        var_dump($type);
    }

    public function getLocal2() {
        // var_dump($type); // Error
    }

    public function getStatic() {
        static $x = 0; // Preserve previous value
        echo $x . "\n";
        $x++;
    }
}

$fruit = new Fruit();
$fruit->getGlobal();
$fruit->getLocal();
//$fruit->getLocal2(); // Error
$fruit->getStatic();
$fruit->getStatic();

```

Terminal: Local

```
[liviu.balan@gen-docker75100-all-dev workshop]$ php 23-12-variables_scope.php
apple
string(6) "banana"
0
1
[liviu.balan@gen-docker75100-all-dev workshop]$
```

Namespaces

- Allow for better organization by grouping classes that work together to perform a task
- They allow the same name to be used for more than one class

```php
<?php

/*...*/

namespace Symfony\Component\HttpFoundation;

/**
 * Request stack that controls the lifecycle of requests.
 *
 * @author Benjamin Eberlei <kontakt@beberlei.de>
 */
class RequestStack
{
    /**
     * @var Request[]
     */
    private $requests = [];

    /**
     * Pushes a Request on the stack.
     *
     * This method should generally not be called directly as the stack
     * management should be taken care of by the application itself.
     */
    public function push(Request $request)
    {
        $this->requests[] = $request;
    }

    /**
     * Pops the current request from the stack.
     *
     * This operation lets the current request go out of scope.
     *
     * This method should generally not be called directly as the stack
```

Q&A

Practice