

# PHP 5.4

Die wichtigsten Neuerungen im Überblick

 @ManuelKiessling

# SHORT ARRAY SYNTAX

```
$a = array( 'a', 'b', 'c' );
```

```
$a = [ 'a', 'b', 'c' ];
```

```
$a = [ 'one' => 1, 'two' => 2 ];
```

# FUNCTION ARRAY DEREFERENCING

```
function foo() {  
    return array(1, 2, 3);  
}
```

```
echo foo[2]; // prints 3
```

# CLASS MEMBER ACCESS ON INSTANTIATION

```
class Foo {  
    public function bar() {  
        echo 'foobar';  
    }  
}
```

```
(new Foo) ->bar(); // foobar
```

# TRAITS

“

*The best way to understand what traits are and how to use them is to look at them for what they essentially are: language assisted copy and paste.*

*If you can copy and paste the code from one class to another then you have a candidate for a trait.”*

# TRAITS

Unterschiedlich Klassen implementieren unterschiedliches Verhalten – aber es gibt einige Aspekte, die in praktisch allen Klassen eines Systems implementiert werden sollen.

	User	Item
Verhalten:	login() buy() pay()	order() ship() changeColor()
Aspekt:	logging	logging

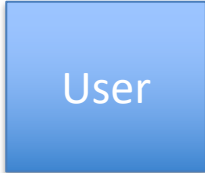

# TRAITS

```
class User {  
    public function login() {  
        //...  
        $this->logger->log('login', $user->name, time());  
    }  
}
```

```
class Item {  
    public function ship() {  
        //...  
        $this->logger->log('shipped', $item->id, time());  
    }  
}
```

# TRAITS

Unterschiedlich Klassen implementieren unterschiedliches Verhalten – aber es gibt einige Aspekte, die in praktisch allen Klassen eines Systems implementiert werden sollen.

		
Verhalten:	login() buy() pay()	order() ship() changeColor()
Aspekt:	logging	logging



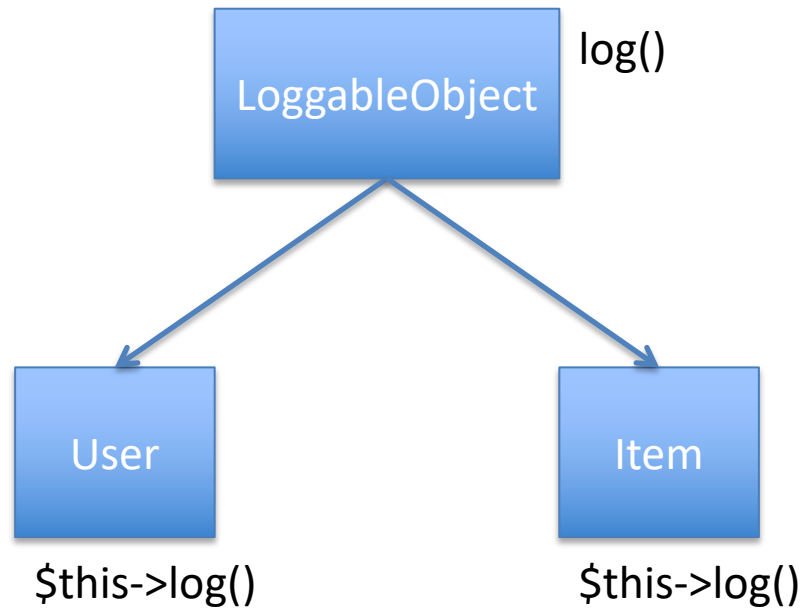
# TRAITS

Was, wenn das *logger* Objekt ab sofort den Zeitstempel als ersten Parameter erwartet?

Wie verhindern wir, dass wir in allen Klassen, die den Logger verwenden, den Code anpassen müssen?

# TRAITS

## Variante 1: Gemeinsame Vererbung



# TRAITS

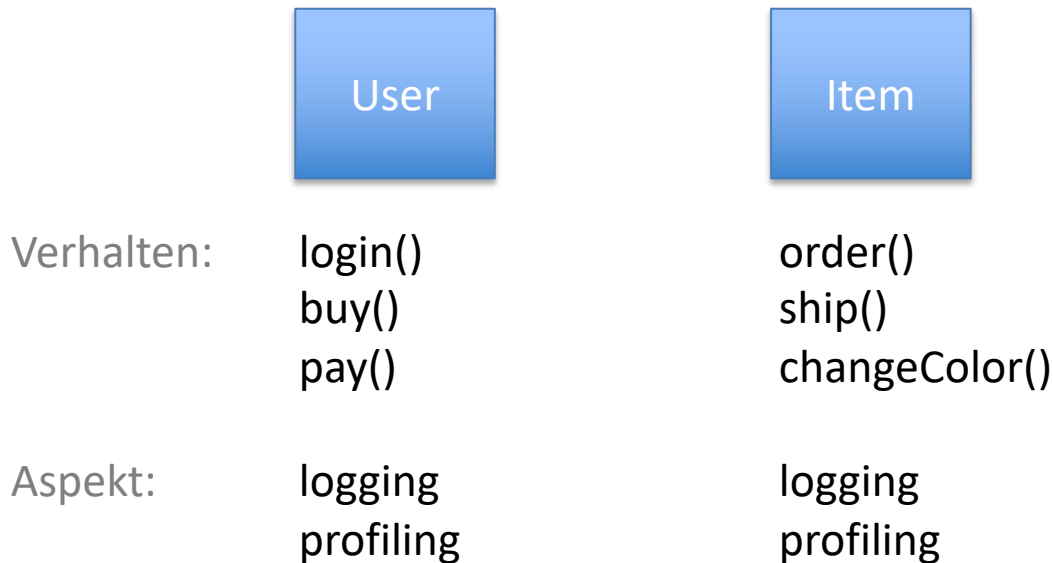
```
class LoggableObject {  
    public function log(event, identifier, timestamp) {  
        $this->logger->log(event, identifier, timestamp);  
    }  
}
```

```
class User extends LoggableObject {  
    public function login() {  
        //...  
        $this->log('login', $user->name, time());  
    }  
}
```

```
Class Item extends LoggableObject {  
    public function ship() {  
        //...  
        $this->log('login', $item->id, time());  
    }  
}
```

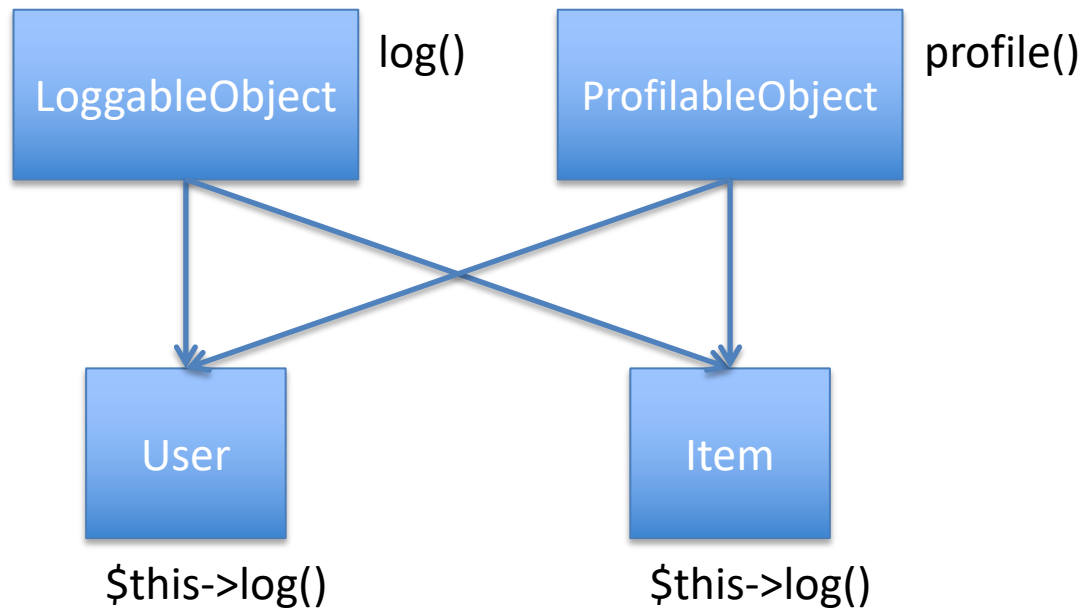
# TRAITS

Was, wenn wir einen zweiten Aspekt haben?



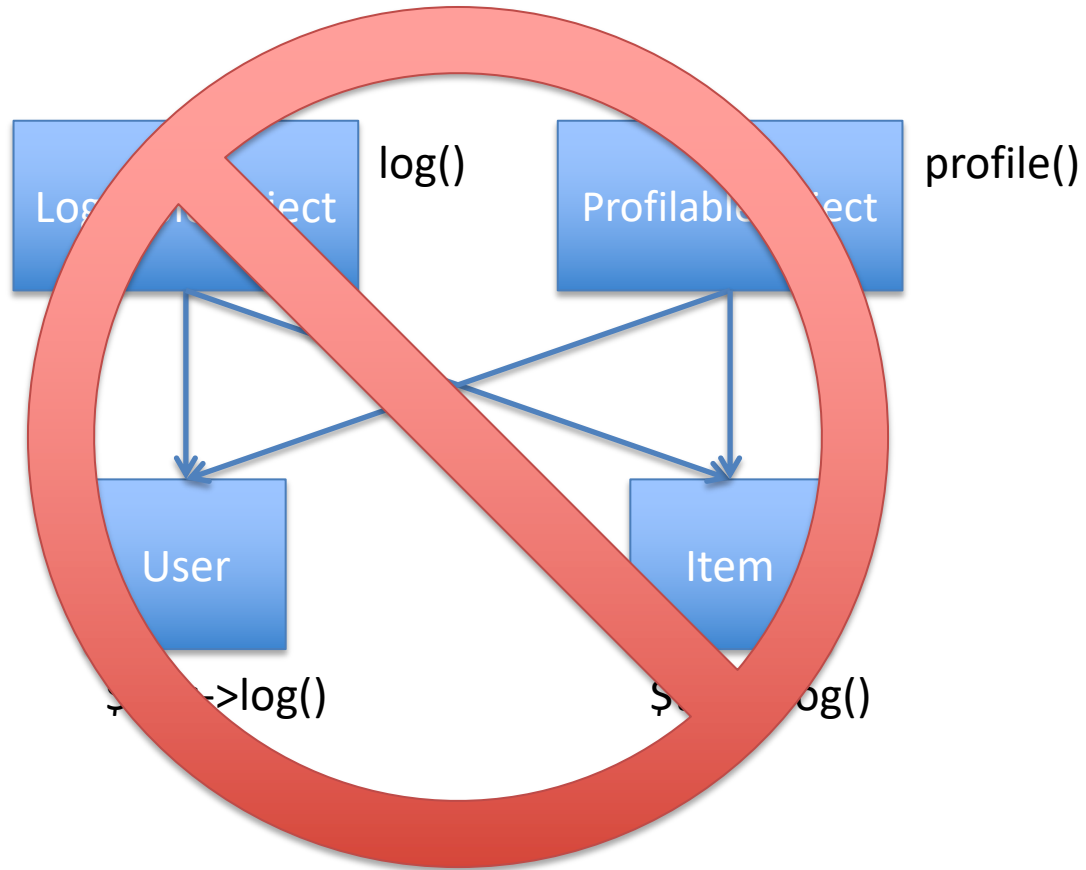
# TRAITS

## Variante 1: Mehrfachvererbung



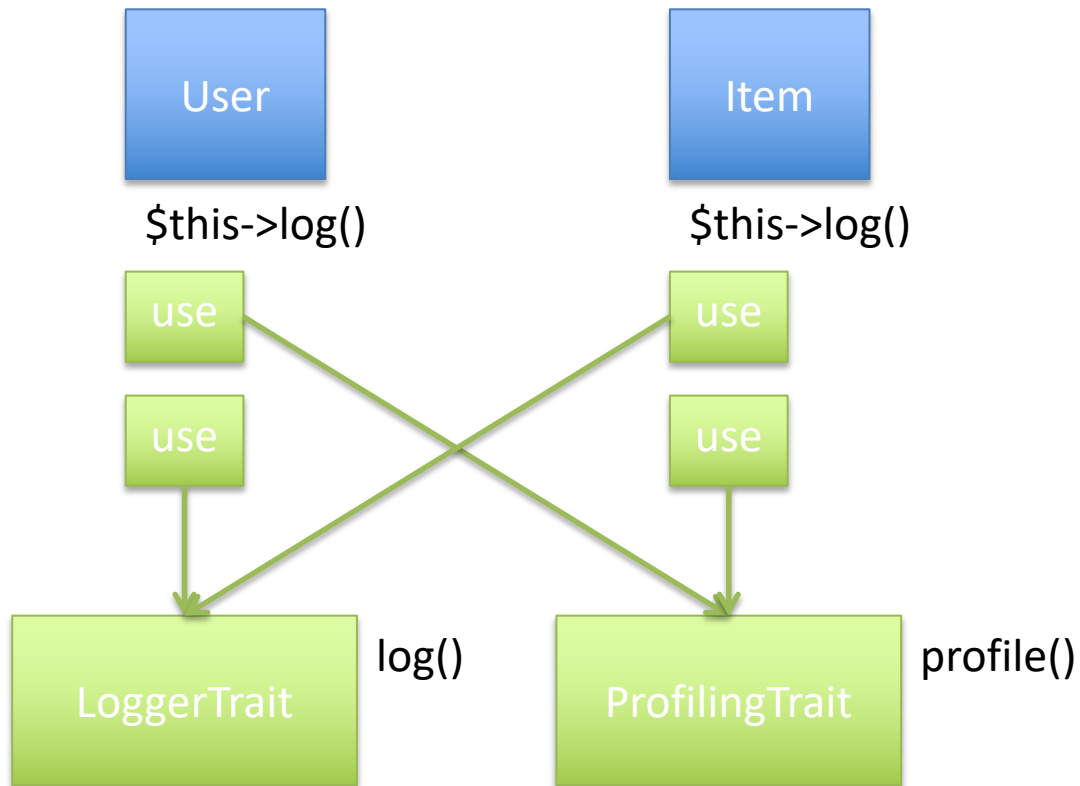
# TRAITS

## Variante 1: Mehrfachvererbung



# TRAITS

## Variante 2: Traits



# TRAITS

```
trait Logger {  
    public function log(event, identifier, timestamp) {...}  
}
```

```
trait Profiler {  
    public function profile() {...}  
}
```

```
class User {  
    use Logger, Profiler;  
    public function login() {  
        $this->log('login', $user->name, time());  
        $this->profile(...);  
    }  
}
```

```
Class Item {  
    use Logger, Profiler;  
    public function ship() {  
        $this->log('login', $item->id, time());  
        $this->profile(...);  
    }  
}
```



# Danke. Fragen?

 @ManuelKiessling