Запись на курсы по HTML, CSS, JavaScript, PHP, фреймворкам и CMS, **а также**: помощь в поиске работы и заказов, стажировка на реальных проектах→

урок 47 из 107

Bepcтка JavaScript PHP NodeJs Vue React Laravel WordPress AJAX Парсинг

Бесплатные курсы по React для новичков. Начало 4-го ноября→ Конкурс CSS картинок. Тема: Хэллоуин. Призовой фонд: 100\$. Подробности→

Разрешение конфликтов в трейтах

Так как один класс может использовать несколько трейтов, то нас может поджидать проблема, возникающая тогда, когда два трейта имеют одноименные методы.

В этом случае РНР выдаст фатальную ошибку. Чтобы поправить ситуацию, нужно будет разрешить конфликт имен явным образом. Как это делается - посмотрим на практике.

Пусть у нас есть два трейта с одинаковым методом **method**:

```
1
    <?php
2
      trait Trait1
3
4
        private function method()
5
6
           return 1;
7
8
      }
9
10
      trait Trait2
11
12
        private function method()
13
14
           return 2;
15
16
   · ;>
17
```

Пусть у нас также есть класс **Test**, использующий оба наших трейта. Если просто подключить оба трейта к нашему классу, то PHP выдаст ошибку, так как у трейтов есть совпадающий методы:

```
1 <?php
2 // Данный код выдаст ошибку!
3 class Test
4 {
5 use Trait1, Trait2; // подключаем трейты
6 }
7 ?>
```

Давайте разрешим (в данном контексте это слово значит *разрулим*) конфликт имен наших трейтов. Для этого существует специальный оператор **insteadof** (переводится *вместо чего-то*).

Давайте с помощью этого оператора скажем следующее: *использовать метод method трейта Trait1* вместо такого же метода трейта *Trait2*:

```
<?php
1
2
      class Test
3
4
        use Trait1, Trait2 {
5
           Trait1::method insteadof Trait2;
6
7
      }
8
9
      new Test;
10
   · ;>
```

Как вы видите, синтаксис тут следующий: вначале имя трейта, потом два двоеточия, потом имя метода, потом наш оператор insteadof и имя второго трейта.



Давайте проверим:

```
1
   <?php
2
      class Test
3
        use Trait1, Trait2 {
4
5
          Trait1::method insteadof Trait2;
6
7
8
        public function __construct()
9
10
          echo $this->method(); // выведет 1, тк
            это метод первого трейта
11
12
      }
13
14
      new Test;
15 | ?>
```

Итак, в нашем классе мы сказали, что если используется метод method, то следует брать его из первого трейта.

Можно и наоборот - взять метод второго трейта:

```
<?php
1
2
      class Test
3
4
        use Trait1, Trait2 {
          Trait2::method insteadof Trait1;
5
        }
6
7
8
        public function __construct()
9
          echo $this->method(); // выведет 2, тк
10
            это метод второго трейта
11
12
      }
13
14
      new Test;
15 | ?>
```

В любом случае, если мы указываем использовать метод одного трейта, то метод второго трейта оказывается недоступным.

Можно использовать и метод второго трейта, переименовав его через ключевое слово \mathbf{as} , вот так:

```
1
   <?php
2
     class Test
3
4
        use Trait1, Trait2 {
          Trait1::method insteadof Trait2; // берем мет
5
            од из первого трейта
6
          Trait2::method as method2; // метод второго тре
            йта будет доступен как method2
8
        public function __construct()
9
10
          echo $this->method() + $this->method2();
11
            // выведет 3
12
13
     }
14
15
     new Test;
16 | ?>
```

При желании можно переименовать и метод первого трейта:



```
<?php
1
     class Test
2
3
       use Trait1, Trait2 {
4
5
          Trait1::method insteadof Trait2;
         Trait1::method as method1; // метод первого тре
6
            йта будет доступен как method1
7
         Trait2::method as method2; // метод второго тре
            йта будет доступен как method2
8
9
       public function __construct()
10
11
          echo $this->method1() + $this->method2();
12
            // выведет 3
13
14
     }
15
16
     new Test;
17 | ?>
```

Использовать ключевое слово as без определения главного метода через insteadof нельзя, это выдаст ошибку:

```
1
   <?php
2
      // Данный класс выдаст ошибку:
      class Test
3
4
      {
5
        use Trait1, Trait2 {
          Trait1::method as method1;
6
          Trait2::method as method2;
7
8
        }
9
        public function __construct()
10
11
          echo $this->method1() + $this->method2();
12
13
14
15
16
      new Test;
17 | ?>
```

Задача 47.1

Сделайте 3 трейта с названиями **Trait1**, **Trait2** и **Trait3**. Пусть в первом трейте будет метод **method**, возвращающий 1, во втором трейте - одноименный метод, возвращающий 2, а в третьем трейте - одноименный метод, возвращающий 3.

Задача 47.2

Сделайте класс **Test**, использующий все три созданных нами трейта. Сделайте в этом классе метод **getSum**, возвращающий сумму результатов методов подключенных трейтов.



