Search

Keyboard Shortcuts

?
: This help

j
: Next menu item

k
: Previous menu item

g p
: Previous man page

g n
: Next man page

G
: Scroll to bottom

g g
: Scroll to top

g h
: Goto homepage

g s
: Goto search
(current page)

/
: Focus search box

Методы перечислений »
« Основы перечислений

- Руководство по PHP
- Справочник языка
- Перечисления

Change language: Russian

Типизированные перечисления

По умолчанию у вариантов перечислений нет скалярного эквивалента. Это обычные одноэлементные объекты. Однако существуют случаи, когда вариантам перечислений нужно обращаться к базе данных или аналогичному хранилищу данных, поэтому полезно иметь встроенный скалярный (и, следовательно, тривиально сериализуемый) эквивалент, определённый внутренне.

Чтобы определить скалярный эквивалент для перечислений, пользуются следующим синтаксисом:

```
<?php

enum Suit: string
{
case Hearts = 'H';
case Diamonds = 'D';
case Clubs = 'C';
case Spades = 'S';
}
?>
```

Вариант со скалярным эквивалентом называется типизированным (Backed Case), поскольку он «поддержан» более простым значением. Перечисление, у которого все варианты типизированные, называется «типизированным перечислением» (Backed Enum). Типизированное перечисление может содержать только типизированные варианты. Чистое перечисление может содержать только чистые варианты.

Типизированное перечисление может поддерживаться типами `int` или `string` и такое перечисление поддерживает только один тип за раз (то есть нельзя объединять `int|string`). Если перечисление помечено как имеющее скалярный эквивалент, тогда все варианты должны иметь определённый явно уникальный скалярный эквивалент. Не существует автоматически генерируемых скалярных эквивалентов (например, последовательных целых чисел). Типизированные варианты должны быть уникальными; двум вариантам типизированного перечисления не может принадлежать один и тот же скалярный эквивалент. Однако константа может относиться к варианту, фактически создавая псевдоним. Смотрите «Константы перечислений».

Эквивалентные значения должны быть строками или строковыми выражениями. Константы и постоянные выражения не поддерживаются. То есть `1 + 1` разрешено, а `1 + SOME_CONST` — нет.

У типизированных вариантов есть дополнительное доступное только для чтения свойство `value` — это значение, заданное в определении варианта.

```
<?php

print Suit::Clubs->value;
// Выведет "C"
?>
```

Чтобы свойство `value` оставалось доступным только для чтения, было запрещено назначать переменную в качестве ссылки на неё. То есть следующий код выдаст ошибку:

```
<?php

$suit = Suit::Clubs;
$ref = &$suit->value;
// Error: Cannot acquire reference to property Suit::$value
?>
```

Типизированные перечисления реализуют внутренний интерфейс BackedEnum, который даёт два дополнительных метода:

- `from(int|string): self` возьмёт скаляр и вернёт вариант перечисления, которому он принадлежит. Если вариант, который соответствует варианту перечисления, не найден, метод выбросит исключение ValueError. Это в основном полезно тогда, когда входной скаляр надёжен, а отсутствие значения перечисления надо рассматривать как ошибку, останавливающую приложение.
- `tryFrom(int|string): ?self` возьмёт скаляр и вернёт вариант перечисления, которому он принадлежит. Если

вариант, который соответствует варианту перечисления, не найден, метод вернёт `null`. Это в основном полезно тогда, когда входной скаляр ненадёжен и вызывающая функция хочет реализовать свою обработку ошибок или логику значения по умолчанию.

Методы `from()` и `tryFrom()` следуют стандартным правилам слабой/строгой типизации. В режиме слабой типизации допустима передача целого числа или строки, система преобразует значение и найдёт вариант, который ему соответствует. Передача числа с плавающей точкой также будет работать с принудительным преобразованием. В режиме строгой типизации передача целого числа в метод `from()` в перечислении со строковой типизацией (или наоборот) в любом случае приведёт к исключению TypeError, как и передача числа с плавающей точкой. Все остальные типы параметров выбросят исключение TypeError в обоих режимах.

```
<?php

$record = get_stuff_from_database($id);
print $record['suit'];

$suit = Suit::from($record['suit']);
// Недопустимые данные выбросят исключение ValueError: "X" is not a valid scalar value for enum "Suit"
print $suit->value;

$suit = Suit::tryFrom('A') ?? Suit::Spades;
// Недопустимые данные возвращают значение null, поэтому вместо этого будет использовано Suit::Spades.
print $suit->value;
?>
```

Ручное определение метода `from()` или `tryFrom()` в типизированных перечислениях приведёт к фатальной ошибке.

User Contributed Notes

There are no user contributed notes for this page.