Создание экземпляра класса (объекта)

Введение

Классы

Класс — это шаблон, по которому создаются объекты.

Невозможно создать объект на лету, как это происходит с массивами. **Объект создаётся только на основе своего описания — класса**. Этим реализация объектов в PHP отличается от JavaScript. В JS объектам не нужны классы, они могут быть созданы и модифицированы, когда и как угодно.

Объекты

Объекты в PHP — это **тип данных**, который позволяет хранить в переменной набор из свойств и их значений, а также встроенные функции - методы. Объекты похожи по своей структуре на ассоциативные массивы, но имеют свои особенности:

- Объекты могут содержать отдельные значения, каждое под своим ключом. Эти значения называются **свойствами** объекта.

- Объекты могут иметь внутри себя функции — их называют **методами** объекта.
- Значения свойств объекта могут быть любого типа: **число**, **строка**, **массив**, другой **объект**.
- В отличие от массива, объекты не позволяют добавлять в себя новые значения (точнее – не должны позволять это делать).
- Чтобы создать новый объект, необходимо сначала создать его описание — **класс**.

Работа с объектами в PHP включает в себя создание объектов, присваивание объектов переменным, копирование и клонирование объектов, а также проверку типа объекта.

Рассмотрим эти аспекты более подробно.

- Создание объекта.
- Присваивание объектов переменным.
- Копирование и клонирование объектов.
- Проверка типа объекта.
- Уничтожение объекта.

Особенности объектов и их отличия от массивов

PHP-объекты похожи на массивы, но со своими особенностями. Объекты могут содержать отдельные значения, каждое под своим ключом. Эти значения называются **свойствами** объекта.

PHP-объекты могут иметь внутри себя функции — их называют **методами** объекта. Методы могут обращаться к любым свойствам объекта, читать и записывать туда данные.

> ***Важно.*** Методы объекта могут изменять **свойства** и **состояния** объекта.

Значение свойства объекта может быть любого типа: число, строка, массив, другой объект. Но, в отличие от массива, объекты не позволяют добавлять в себя новые значения.

> ***Важно.*** Объект всегда имеет конечное число своих свойств и методов.

Менять значения существующих свойств можно, а удалять и заменять их — нельзя. Что в корне отличается от поведения массива, ведь там добавлять и удалять значения можно в любое время.

> ***Примечание.*** Если быть более точным – программист может менять количество свойств объекта. Но **парадигма** объектного программирования не рекомендует писать код таким образом.

Самая большая особенность объектов — то, как они **создаются**. Если массив создаётся либо пустым, либо сразу с набором значений, то объекты устроены иначе. Объекты не существуют сами по себе. Чтобы создать новый объект, придётся вначале создать его описание — класс. Класс описывает то, из чего состоит объект.

В предыдущей теме мы познакомились с одним из способов создания экземпляра класса (объекта). Но создавать объекты можно не только с помощью ключевого слова **new**.

Создание объекта

Объекты класса могут быть инициализированы следующими способами:

- Оператор **new.**
- **Преобразование** в объект.
- Специальные **функции**:
    - **json_decode**(),
    - **mysqli_fetch_object**().

Объект с помощью оператора new

Для создания объекта из класса в PHP может использоваться ключевое слово (оператор) **new**. Новый объект будет создан, если только конструктор объекта во время ошибки не выбрасывает исключение. Класс рекомендуют определять перед тем, как создавать экземпляр класса; иногда это обязательное требование.

Для того чтобы создать **объект** из класса **Person** используем следующий **синтаксис**:

```
// создание экземпляра класса (объекта)

$person = new Person();
```

> ***Замечание.*** Круглые скобки после названия класса **разрешается опускать**, если нет аргументов, которые требуется передать в конструктор класса (про конструкторы класса немного позже).

Пример альтернативного создания экземпляра класса:

```
// создание экземпляра класса (объекта), не имеющего параметров

$person = new Person;
```

example_1.

```
<?php

    // определение класса Person
    class Person {
        // свойство класса
        public $id_personnel = "значение 1";
        // константа класса
        public const MYSQLI = "значение 2";
        // методы класса
```

```
        // функция поиска педагога
        public function getPersonnel() {
            // PHP код
        }
    }
    // создание экземпляра класса (объекта)
    $person = new Person();
    // вывод объекта
    echo "<pre>";
    print_r($person);
    echo "</pre>";
?>
```

PHP создаст новый экземпляр класса **и в том случае**, если с ключевым словом **new** указать **переменную**, которая содержит **строку** (string) с **названием класса**. Чтобы создать экземпляр класса, который определен в пространстве имён, требуется указать абсолютное имя класса.

```
// создание экземпляра класса (объекта) через переменную
$className = 'Person';
$person = new $className();
```

example_2.

```
<?php
    // определение класса Person
    class Person {
        // свойство класса
        public $id_personnel = "значение 1";
    }
```

```php
    // создание экземпляра класса (объекта) ) через переменную
    $className = 'Person';
    $person = new $className();
    // вывод объекта
    echo "<pre>";
    print_r($person);
    echo "</pre>";
?>
```

PHP поддерживает ключевое слово **new** с **произвольными выражениями**. Это позволяет создавать более сложные экземпляры, если выражение создаёт **строку** (string). В качестве выражения может быть использована функция.

При этом выражения берут в круглые скобки.

example_3.

```php
<?php
    // определение классов
    class NoteBook {};
    class PC {};
    // определение функции
    function getClass () {
        // ...
        return 'NoteBook';
    }
    // создание объектов
    $note = new (getClass());
    $pc = new ('P' . 'C');
```

```
    // вывод объектов
    echo "<pre>";
    print_r($note);
    print_r($pc);
    echo "</pre>";
?>
```

Преобразование в объект

Для преобразования в объект может быть использовано ключевое слово – **object**.

Если **object** преобразовывается в **object**, объект не изменится.

Если значение **другого типа** преобразовывается в **object**, создаётся новый экземпляр встроенного класса **stdClass**. Если значение было **null**, новый экземпляр будет пустым.

Массивы преобразуются в **object** с именами полей, названными согласно **ключам массива** и **соответствующими им значениям**.

example_4.

```
<?php
    // ассоциативный PHP-массив
    $personnelARR = array (
        'id_personnel' => '2',
        'surname' => 'Трост',
        'name' => 'Дмитрий',
        'patronymic' => 'Юрьевич',
        'post' => 'Программист',
        'category' => 'Первая',
```

```
        'level_edu' => 'Высшее профессиональное',
        'rating' => '4.56',
        'actual' => '1',
        'hidden' => '0',
        'note' => ''
    );
    // обращение к элементам массива
    echo '<h2>' . $personnelARR['name'] . ' ' .
    $personnelARR['surname'] . '</h2>';
    // вывод структуры
    echo "<pre>";
    print_r($personnelARR);
    echo "</pre>";
    // преобразуем массив в объект
    $personOBJ = (object) $personnelARR;
    // обращение к свойствам объекта
    echo '<h2>' . $personOBJ->name . ' ' . $personOBJ->surname .
    '</h2>';
    // вывод структуры
    echo "<pre>";
    print_r($personOBJ);
    echo "</pre>";
?>
```

Объект с помощью функций PHP

Некоторые **функции** PHP создают экземпляры класса **stdClass**, например, функции:

- **json_decode**()
- **mysqli_fetch_object**().

Рассмотрим пример декодирования строки **JSON-формата** в PHP-объект.

example_5.

```php
<?php
    // ассоциативный PHP-массив
    $personnel = array (
        'id_personnel' => '2',
        'surname' => 'Трост',
        'name' => 'Дмитрий',
        'patronymic' => 'Юрьевич',
        'post' => 'Программист',
        'category' => 'Первая',
        'level_edu' => 'Высшее профессиональное',
        'rating' => '4.56',
        'actual' => '1',
        'hidden' => '0',
        'note' => ''
    );
    // кодируем массив в JSON-формат
    $arr = json_encode($personnel);
    // декодируем JSON как PHP-объект
    $persOBJ = json_decode($arr, 0);
    // вывод структуры
    echo "<pre>";
    print_r($persOBJ);
```

```
    echo "</pre>";
    // декодируем JSON как PHP-массив
    $persARR = json_decode($arr, 1);
    // вывод структуры
    echo "<pre>";
    print_r($persARR);
    echo "</pre>";
?>
```

А теперь выполним запрос к базе данных и сохраним результирующий набор в виде **объекта**.

example_6.

```
<?php
    // константы для подключения
    define('HOST', 'localhost');
    define('USER', 'root');
    define('PWD', '');
    define('DB', 'db_distance');
    // подключение к серверу СУБД
    $mysqli = new mysqli(HOST, USER, PWD, DB);
    // запрос, получение набора записей
    $row = $mysqli
        -> query('SELECT * FROM `personnel` WHERE `id_personnel`
        = 2')
        -> fetch_object();
    // вывод записи из таблицы в виде объекта
    echo "<pre>";
```

```
    print_r($row);
    echo "</pre>";
?>
```

Проверка типа объекта

Тип PHP объекта можно проверить с помощью функции **instanceof**.

Функция позволяет проверить, является ли объект **экземпляром** определенного класса или его **предков**.

Пример:

```
<?php
    // определение классов
    class Car {};
    class Note {};
    // создание экземпляра класса (объекта)
    $note = new Note();
    if ($note instanceof Note) {
        echo "Это объект класса Note";
    }
```

Копирование и клонирование объектов

Копирование объектов

Объекты могут быть присвоены переменным. Это позволяет работать с объектом, используя переменную, а также передавать объекты в функции и методы.

Пример:

```
// копируем объект в переменную

$newobj = $obj;
```

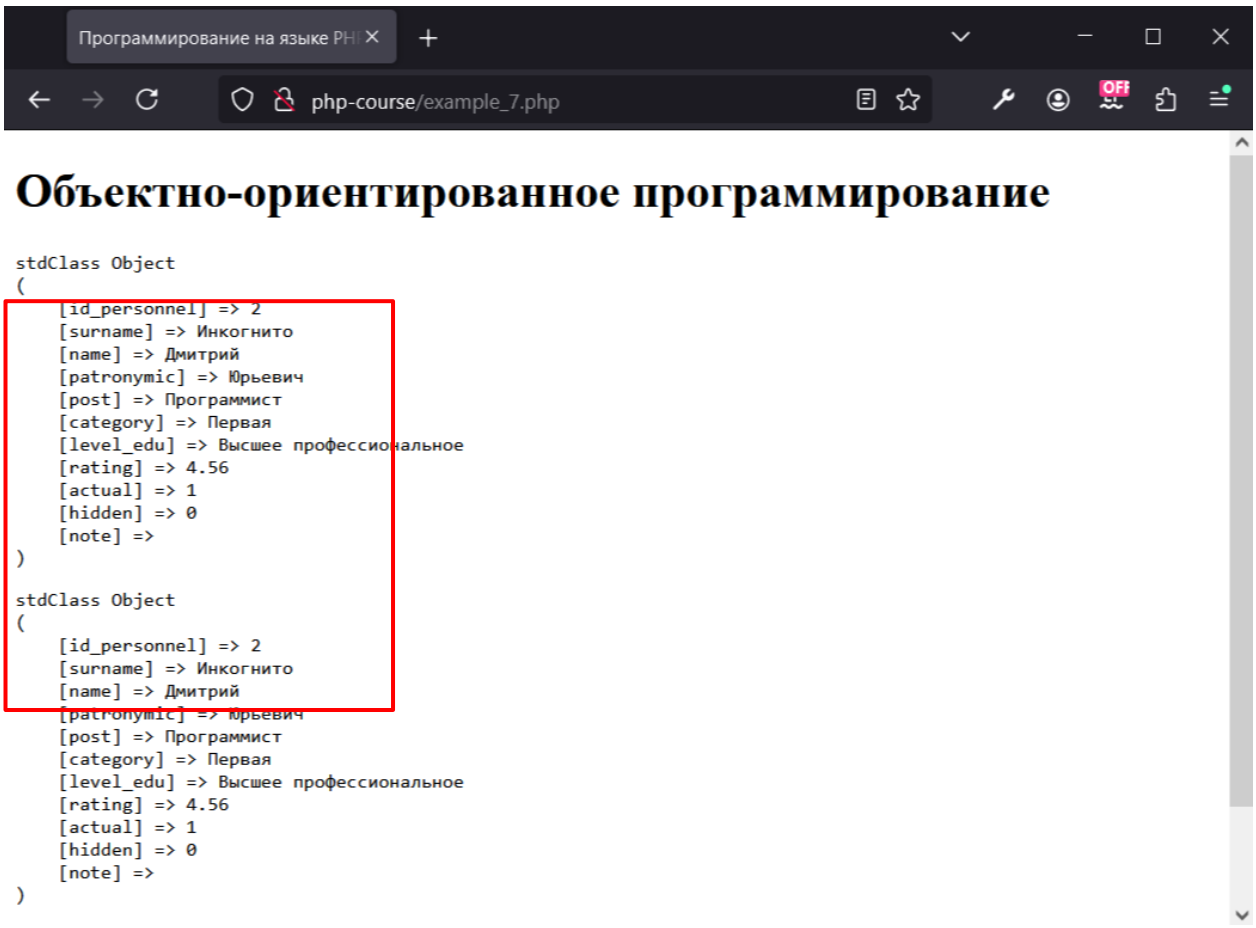
Копирование объектов в PHP может быть поверхностным или глубоким. При **поверхностном** копировании **создается новая ссылка на существующий объект**, и изменения в одном объекте отражаются в другом.

> ***К сведению***. При копировании ссылки на объект изменения в одном объекте **отразятся на состоянии** другого.

example_7.

```
<?php

    // ассоциативный PHP-массив

    $personnel = array (

        'id_personnel' => '2',

        'surname' => 'Трост',

        'name' => 'Дмитрий',

        'patronymic' => 'Юрьевич',

        'post' => 'Программист',

        'category' => 'Первая',

        'level_edu' => 'Высшее профессиональное',

        'rating' => '4.56',

        'actual' => '1',

        'hidden' => '0',

        'note' => ''

    );
```

```php
    // преобразуем массив в объект
    $obj = (object) $personnel;
    // копируем объект в переменную
    $newobj = $obj;
    // меняем свойство объекта
    $newobj->surname = "Инкогнито";
    // здесь Инкогнито - логично
    echo "<pre>";
    print_r($newobj);
    echo "</pre>";
    // но и здесь Инкогнито тоже
    echo "<pre>";
    print_r($obj);
    echo "</pre>";
?>
```



При **глубоком копировании** создается полная независимая копия объекта.

> ***Важно***. Для создания независимой копии объекта используем клонирование.

Клонирование объектов

Для создания глубокой копии объекта можно использовать метод **clone**.

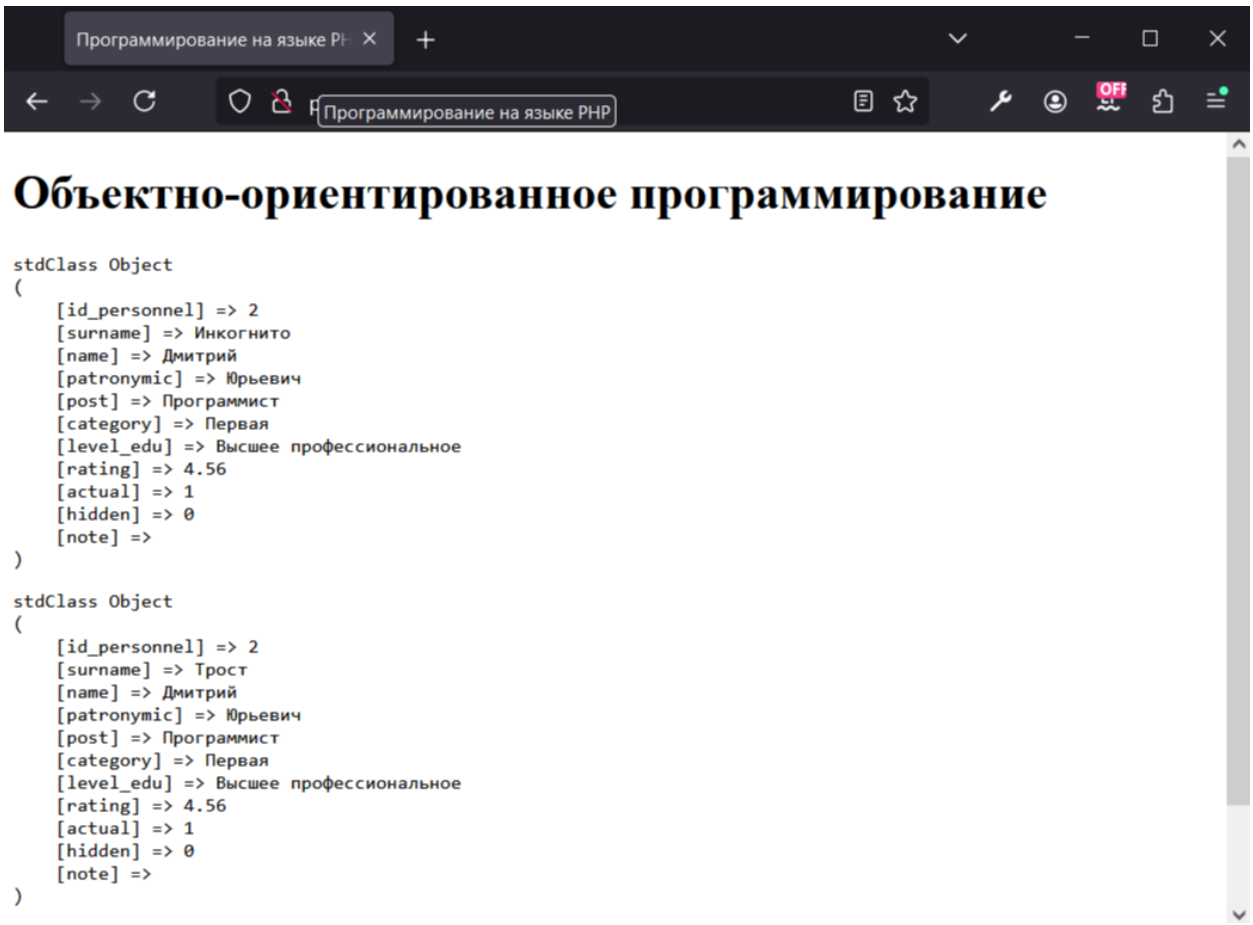
Пример:

```
// клонируем объект

$newobj = clone $obj;
```

example_8.

```php
<?php
    // ассоциативный PHP-массив
    $personnel = array (
        'id_personnel' => '2',
        'surname' => 'Трост',
        'name' => 'Дмитрий',
        'patronymic' => 'Юрьевич',
        'post' => 'Программист',
        'category' => 'Первая',
        'level_edu' => 'Высшее профессиональное',
        'rating' => '4.56',
        'actual' => '1',
        'hidden' => '0',
        'note' => ''
    );
    // преобразуем массив в объект
    $obj = (object) $personnel;
    // клонируем объект
    $newobj = clone $obj;
    // меняем свойство объекта
    $newobj->surname = "Инкогнито";
    // здесь Инкогнито
    echo "<pre>";
    print_r($newobj);
    echo "</pre>";
    // исходный объект изменения не коснулись
```

```
    echo "<pre>";
    print_r($obj);
    echo "</pre>";
?>
```



Уничтожение объекта

В PHP объекты уничтожаются автоматически, когда на них больше нет **активных ссылок.** Однако, мы также можем явно уничтожить объект с помощью функции **unset**().

В демонстрационном фрагменте кода **example_9** рассмотрим пример явного **уничтожения** объекта.

example_9.

```php
<?php
    // ассоциативный PHP-массив
    $personnel = array (
        'id_personnel' => '2',
        'surname' => 'Трост',
        'name' => 'Дмитрий',
        'patronymic' => 'Юрьевич',
        'post' => 'Программист',
        'category' => 'Первая',
        'level_edu' => 'Высшее профессиональное',
        'rating' => '4.56',
        'actual' => '1',
        'hidden' => '0',
        'note' => ''
    );
    // преобразуем массив в объект
    $obj = (object) $personnel;
    // копируем объект в переменную
    $newobj = $obj;
    echo "<pre>";
    print_r($newobj);
    echo "</pre>";
    unset($obj);
    unset($newobj);
    echo "<pre>";
    print_r($newobj);
```

```
    echo "</pre>";
?>
```

Практика применения

В заключительном разделе рассмотрим примеры преобразования ассоциативного массива и результирующего набора в **массивы объектов**.

Демонстрационный пример **example_10**. Перебираем строки ассоциативного массива и сохраняем в массив объектов.

example_10. **team.php**

```
<?php
    $team = array(
        array('id_team' => '1','name' => 'Aerosmith','alias' =>
        'aerosmith','country' => 'США','content' => '','date' =>
        '1970','style' => 'хард-рок','path' =>
        'assets/teams/aerosmith.jpg','note' => NULL),
        // ...
    );
?>
```

example_10. **example_10.php**

```
<?php
    // подключаем двумерный массив
    include "team.php";
    // преобразуем двумерный ассоциативный массив в массив
    объектов
```

```
    foreach ($team as $arr) {
        $arrOBJ[] = (object) $arr;
    }
    // ассоциативный массив
    echo "<pre>";
    var_dump($team);
    echo "<hr>";
    // массив объектов
    var_dump($arrOBJ);
    echo "</pre>";
?>
```

Демонстрационный пример **example_11.** Выбираем из базы данных набор записей и сохраняем в массив объектов.

example_11.

```
<?php
    // константы подключения
    define('HOST', 'localhost');
    define('USER', 'root');
    define('PWD', '');
    define('DB', 'db_distance');
    // подключение к серверу СУБД
    $mysqli = new mysqli(HOST, USER, PWD, DB);
    // выполнение запроса, извлечение набора записей
    $rows = $mysqli
        -> query('SELECT * FROM `personnel` WHERE `id_personnel`
        IN (2,3,4,5)')
```

```
        -> fetch_all(MYSQLI_ASSOC);
    $arrOBJ = Array();
    // преобразуем двумерный ассоциативный массив в массив
    объектов
    foreach ($rows as $item) {
        array_push($arrOBJ, (object)$item);
    }
    unset($arrOBJ[3], $arrOBJ[2]);
    // ассоциативный массив
    echo "<pre>";
    var_dump($rows);
    echo "<hr>";
    // массив объектов
    print_r($arrOBJ);
    echo "</pre>";
?>
```