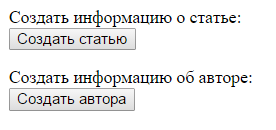
**Практическое занятие 9. Создание и использование классов в php**

**Задача 1.** В зависимости от желания пользователя, автоматически сгенерировать формы для описания статей или личностей (авторов статей).

Необходимо решить эту задачу, используя объектно-ориентированный подход.

В отдельных файлах создаются два класса – статья и личность. Класс «статья» имеет следующие поля; заголовок, автор и описание, «личность» имеет поля имя, фамилию и e-mail.

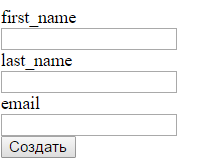
Сначала отображается форма, где пользователь выбирает, что он хочет создать, – описание статьи или человека (точнее, это будут две формы – для описания статьи и для описания человека).



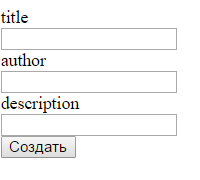
В файле для обработки этих форм создаем экземпляры выбранного класса и отображаем их.

В результате получим следующее:

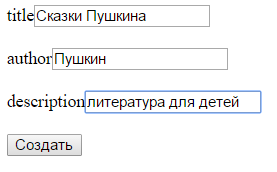
При создании описания личности:



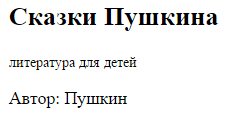
При создании описания статьи:



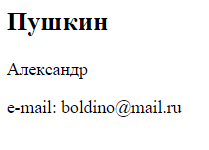
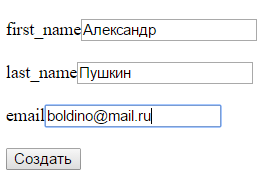
После ввода данных в форму о статье



отобразить данные формы на странице:



Для информации об авторе:



**Задача на повышенную оценку.**

Реализовать один из предложенных классов и применить их.

1. **Класс Date**

Необходимо реализовать класс для работы с датой. Пусть этот класс параметром конструктора принимает дату в формате год-месяц-день и имеет следующие методы:

<?php

class Date

{

public function \_\_construct($date = null)

{

// если дата не передана - пусть берется текущая

}

public function getDay()

{

// возвращает день

}

public function getMonth($lang = null)

{

// возвращает месяц

// переменная $lang может принимать значение ru или en

// если эта не пуста - пусть месяц будет словом на заданном языке

}

public function getYear()

{

// возвращает год

}

public function getWeekDay($lang = null)

{

// возвращает день недели

// переменная $lang может принимать значение ru или en

// если эта не пуста - пусть день будет словом на заданном языке

}

public function addDay($value)

{

// добавляет значение $value к дню

}

public function subDay($value)

{

// отнимает значение $value от дня

}

public function addMonth($value)

{

// добавляет значение $value к месяцу

}

public function subMonth($value)

{

// отнимает значение $value от месяца

}

public function addYear($value)

{

// добавляет значение $value к году

}

public function subYear($value)

{

// отнимает значение $value от года

}

public function format($format)

{

// выведет дату в указанном формате

// формат пусть будет такой же, как в функции date

}

public function \_\_toString()

{

// выведет дату в формате 'год-месяц-день'

}

}

?>

Давайте посмотрим, как мы будем пользоваться нашим классом:

<?php

$date = new Date('2025-12-31');

echo $date->getYear(); // выведет '2025'

echo $date->getMonth(); // выведет '12'

echo $date->getDay(); // выведет '31'

echo $date->getWeekDay(); // выведет '3'

echo $date->getWeekDay('ru'); // выведет 'среда'

echo $date->getWeekDay('en'); // выведет 'wednesday'

?>

<?php

echo (new Date('2025-12-31'))->addYear(1); // '2026-12-31'

echo (new Date('2025-12-31'))->addDay(1); // '2026-01-01'

echo (new Date('2025-12-31'))->subDay(3)->addYear(1); // '2026-12-28'

?>

1. **Класс Interval**

Необходимо реализовать класс, который будет находить разницу между двумя датами. Пусть конструктор этого класса параметрами принимает две даты, представляющие объекты класса Date, созданного нами в предыдущем уроке, и находит разницу между датами в днях, месяцах и годах:

<?php

class Interval

{

public function \_\_construct(Date $date1, Date $date2)

{

}

public function toDays()

{

// вернет разницу в днях

}

public function toMonths()

{

// вернет разницу в месяцах

}

public function toYears()

{

// вернет разницу в годах

}

}

?>

Давайте посмотрим, как мы будем пользоваться нашим классом:

<?php

$date1 = new Date('2025-12-31');

$date2 = new Date('2026-11-28');

$interval = new Interval($date1, $date2);

echo $interval->toDays(); // выведет разницу в днях

echo $interval->toMonths(); // выведет разницу в месяцах

echo $interval->toYears(); // выведет разницу в годах

?>

1. **Класс Validator**

Необходимо реализовать класс для валидации строк:

<?php

class Validator

{

public function isEmail($str)

{

// проверяет строку на то, что она корректный email

}

public function isDomain($str)

{

// проверяет строку на то, что она корректное имя домена

}

public function inRange($num, $from, $to)

{

// проверяет число на то, что оно входит в диапазон

}

public function inLength($str, $from, $to)

{

// проверяет строку на то, что ее длина входит в диапазон

}

}

?>

1. **Класс CookieShell**

Необходимо реализовать класс-оболочку для работы с куки. Как вы должны уже знать, работа с куками в PHP реализована не очень однородным образом: чтобы установить куку мы используем одну функцию, а чтобы прочитать - другую.

Необходимо сделать свой набор методов для работы с куками, оформив этот набор в виде следующего класса:

class CookieShell

{

public function set($name, $value, $time)

{

// устанавливает куки

// $time задает время в сек, через сколько кука умрет

}

public function get($name)

{

// получает куки

}

public function del($name)

{

// удаляет куки

}

public function exists($name)

{

// проверяет наличие куки

}

}

Давайте посмотрим примеры того, как должен работать наш класс:

<?php

$csh = new CookieShell;

$csh->set('test', '123', 3600 \* 24);

echo $csh->get('test'); // выведет 123

?>

<?php

$csh = new CookieShell;

$csh->set('test', '123', 3600 \* 24);

$csh->del('test');

echo $csh->get('test'); // выведет null

?>

Работа с куками на чистом PHP кошмарна. Если вы установите куку и попытаетесь ее сразу прочитать через $\_COOKIE, то ничего не получится:

<?php

setcookie('name', 'value', time() + 3600);

var\_dump(**$\_COOKIE**['name']); // выведет null

?>

Дело в том, что установленная кука будет доступна только после обновления страницы! Но чаще всего она нам нужна сразу, после установки.

Для этого можно использовать слежующий прием:

<?php

setcookie('name', 'value', time() + 3600);

**$\_COOKIE**['name'] = 'value'; // принудительно запишем в массив

var\_dump(**$\_COOKIE**['name']); // выведет 'value'

?>

Аналогичным образом рекомендуется поступать и при удалении куки.

**Задача** С помощью созданного класса реализуйте счетчик обновления страницы, работающий на куках.

1. **Класс SessionShell**

Работа с сессиями в PHP реализована гораздо лучше работы с куками. Тем не менее, очень часто делают оболочку и над сессиями.

Необходимо реализовать такую оболочку и для единообразия набор методов этой оболочки по возможности сделать таким же, как и набор методов оболочки для работы с куками.

Вот заготовка такого класса:

<?php

class SessionShell

{

// Удобно стартуем сессию в конструкторе класса:

public function \_\_construct()

{

if (!isset(**$\_SESSION**)) {

session\_start();

}

}

public function set($name, $value)

{

// устанавливает переменную сессии

}

public function get($name)

{

// получает переменную сессии

}

public function del($name)

{

// удаляет переменную сессии

}

public function exists($name)

{

// проверяет переменную сессии

}

public function destroy()

{

// разрушает сессию

}

}

?>

1. **Класс FileManipulator**

Необходимо создать класс для работы с файлами:

<?php

class FileManipulator

{

public function create($filePath)

{

// создает файл

}

public function delete($filePath)

{

// удаляет файл

}

public function copy($filePath, $copyPath)

{

// копирует файл

}

public function rename($filePath, $newName)

{

// переименовывает файл

// вторым параметром принимает новое имя файла (только имя, не путь)

}

public function replace($filePath, $newPath)

{

// перемещает файл

}

public function weigh($filePath)

{

// узнает размер файла

}

}

?>

1. **Класс File**

Необходимо реализовать класс для работы с файлами. Пусть этот класс реализует следующий интерфейс:

<?php

interface iFile

{

public function \_\_construct($filePath);

public function getPath(); // путь к файлу

public function getDir(); // папка файла

public function getName(); // имя файла

public function getExt(); // расширение файла

public function getSize(); // размер файла

public function getText(); // получает текст файла

public function setText($text); // устанавливает текст файла

public function appendText($text); // добавляет текст в конец файла

public function copy($copyPath); // копирует файл

public function delete(); // удаляет файл

public function rename($newName); // переименовывает файл

public function replace($newPath); // перемещает файл

}

?>

1. **Класс DatabaseShell**

Необходимо реализовать оболочку для работы с базой данных. Такая оболочка позволит нам не писать сырые SQL запросы, а делать их средствами PHP. Зачастую это занимает гораздо меньше кода и проще в обслуживании.

Здесь следует сказать, что подобные оболочки для работы с базой зачастую занимают более 1000 строк. Мы не будем делать такого монстра, а просто реализуем некоторый упрощенный вариант, чтобы вы поняли принцип. Например, вот такой:

<?php

class DatabaseShell

{

private $link;

public function \_\_construct($host, $user, $password, $database)

{

$this->link = mysqli\_connect($host, $user, $password, $database);

mysqli\_query($this->link, "SET NAMES 'utf8'");

}

public function save($table, $data)

{

// сохраняет запись в базу

}

public function del($table, $id)

{

// удаляет запись по ее id

}

public function delAll($table, $ids)

{

// удаляет записи по их id

}

public function get($table, $id)

{

// получает одну запись по ее id

}

public function getAll($table, $ids)

{

// получает массив записей по их id

}

public function selectAll($table, $condition)

{

// получает массив записей по условию

}

}

?>

Пользоваться таким классом можно следующим образом. Создаем объект для работы:

<?php

$db = new DatabaseShell('localhost', 'root', 'root', '', 'test');

?>

Сохраним нового юзера:

<?php

$db->save('users', ['name' => 'user1', 'age' => '23']);

?>

Удалим юзера с id равным 3:

<?php

$db->del('users', 3);

?>

Удалим юзеров с заданными id:

<?php

$db->delAll('users', [1, 2, 3, 5]);

?>

Получим юзера с id равным 3:

<?php

$user = $db->get('users', 3);

?>

Получим юзеров с заданными id:

<?php

$users = $db->getAll('users', [1, 2, 3, 5]);

?>

Получим юзеров по условию:

<?php

$users = $db->getAll('users', 'where id >= 3');

?>