# Разработка многостраничного сайта на PHP

ТЕМА: РАСШИРЕННЫЕ МЕТОДЫ
ЗАНЯТИЕ № 4 - ПРАКТИКА

# Тема занятия – Расширенные методы

Цель занятия –

Создать условия для формирования умений использования расширенных методов.

# Актуализация

На прошлом занятии мы изучили расширенные методы в PHP, такие как:

1) PSR-1,
2) PSR-2,
3) PSR-4,
4) PRS-6,
5) PSR-12,
6) CLI...

Теперь научимся применять эти теоретические навыки на практике.

# Содержание :

# Задача № 1

Напишите программу, которая будет запрашивать у пользователя число, а затем вывести на экран факториал этого числа.

Используйте PSR-4 для структурирования кода.

# Решение

```php
<?php

namespace MyApp;

class Factorials {
    public function getFactorial($n)
    {
        $factorial = 1;
        for ($i = 1; $i <= $n; $i++) {
            $factorial *= $i;
        }
        return $factorial;
    }
}

require __DIR__ . '/vendor/autoload.php';

use MyApp\Factorial;

$number = readline("Enter a number: ");
$factorial = new Factorials();
echo "Factorial of $number is: " . $factorial->getFactorial($number) . "\n";

?>
```

# Задача № 2

Напишите программу, которая будет приниматься на вход, разбирать ее на слова и выводить каждое слово на новый сценарий. Используйте PSR-2 для форматирования кода.

```php
<?php

namespace MyApp;

class WordSplitters
{
    public function splitWords($string)
    {
        $words = explode(' ', $string);
        return $words;
    }
}

require __DIR__ . '/vendor/autoload.php';

use MyApp\WordSplitter;

$string = readline("Enter a string: ");
$wordSplitter = new WordSplitters();
$words = $wordSplitter->splitWords($string);
foreach ($words as $word) {
    echo $word . "\n";
}
?>
```

# Задача № 3

Напишите программу, которая будет принимать на два счета и выводить на экран их сумму, разность, произведение и частное. Используйте PSR-12 для форматирования кода.

# Решение

```php
namespace MyApp;
class Calculators{
    public function add($a, $b)
    {
        return $a + $b;
    }

    public function subtract($a, $b)
    {
        return $a - $b;
    }

    public function multiply($a, $b)
    {
        return $a * $b;
    }

    public function divide($a, $b)
    {
        return $a / $b;
    }
}
```

```php
require __DIR__ . '/vendor/autoload.php';

use MyApp\Calculator;

$a = readline("Enter first number: ");
$b = readline("Enter second number: ");

$calculator = new Calculators();

echo "Sum: " . $calculator->add($a, $b) . "\n";
echo "Difference: " . $calculator->subtract($a, $b) . "\n";
echo "Product: " . $calculator->multiply($a, $b) . "\n";
echo "Quotient: " . $calculator->divide($a, $b) . "\n";
?>
```

# Задача № 4

Необходимо написать класс, который будет получать данные с API и кэшировать их на заданный период времени.

Для этого будем использовать интерфейс PSR-6 и использовать его в виде класса FilesystemCache.

# Решение

```php
<?php
use Symfony\Component\Cache\Adapter\FilesystemAdapter;
use Psr\Cache\CacheItemPoolInterface;

class ApiDataCache
{
    private $cache;
    private $apiUrl;

    public function __construct(CacheItemPoolInterface $cache, $apiUrl)
    {
        $this->cache = $cache;
        $this->apiUrl = $apiUrl;
    }

    public function getData($cacheKey, $cacheTTL = 3600)
    {
        $cacheItem = $this->cache->getItem($cacheKey);

        if ($cacheItem->isHit()) {
            return $cacheItem->get();
        }

        $data = $this->fetchDataFromApi($this->apiUrl);

        $cacheItem->set($data);
        $cacheItem->expiresAfter($cacheTTL);

        $this->cache->save($cacheItem);

        return $data;
    }

    private function fetchDataFromApi($apiUrl)
    {
        // ...
        // Some code to fetch data from the API using CURL or some other library
        // ...

        return $data;
    }
}
?>
```

# Задача № 5

Написать программу, которая будет генерировать случайную задачу заданной загрузки. Длину строки необходимо получать в качестве аргумента командной строки. Строка может состоять из латинского алфавита и цифры.

# Решение

```php
<?php
// Получаем длину строки из аргументов командной строки
$length = isset($argv[1]) ? (int) $argv[1] : 10;

// Массив возможных символов для генерации строки
$characters = array_merge(range('a', 'z'), range('A', 'Z'), range(0, 9));

// Генерируем случайную строку заданной длины
$string = '';
for ($i = 0; $i < $length; $i++) {
    $randomIndex = array_rand($characters);
    $string .= $characters[$randomIndex];
}

echo $string;
?>
```

```
php generate_string.php 12

// U1xL8lq3TjvO
```

# Задача № 6

Написана программа, которая будет получать на два входа массива и сливать их в один, удаляя дубликаты. Массивы необходимо получить в качестве аргументов командной строки, каждый элемент массива разделен запятой.

# Решение

```php
<?php
// Получаем массивы из аргументов командной строки
$args = $_SERVER['argv'];
$firstArray = explode(',', $args[1]);
$secondArray = explode(',', $args[2]);

// Объединяем два массива в один и удаляем дубликаты
$mergedArray = array_unique(array_merge($firstArray, $secondArray));

// Выводим результат
echo implode(',', $mergedArray);
?>
```

# Задача № 7

Напишите программу, которая будет считать содержимое файла и выводить количество вхождений каждого слова в этом файле. Имя файла необходимо получить в качестве аргумента командной строки. Слова в файле могут быть разделены всеми символами, кроме букв и цифр.

# Решение

```php
<?php
use Psr\SimpleCache\CacheInterface;
use Symfony\Component\Cache\Simple\FilesystemCache;

function countWords($filename, CacheInterface $cache)
{
    // Проверяем, существует ли файл
    if (!file_exists($filename)) {
        throw new Exception("File not found");
    }

    // Читаем содержимое файла в строку
    $content = file_get_contents($filename);

    // Разбиваем строку на слова и удаляем пустые значения
    $words = preg_split('/[^a-z0-9]+/i', $content, -1, PREG_SPLIT_NO_EMPTY);

    // Проверяем, есть ли кэшированный результат
    $cacheKey = md5($filename);
    $cachedResult = $cache->get($cacheKey);
    if ($cachedResult) {
        return $cachedResult;
    }

    // Подсчитываем количество вхождений каждого слова
    $result = array_count_values($words);

    // Кэшируем результат на 5 минут
    $cache->set($cacheKey, $result, 300);

    return $result;
}

// Используем кэш в виде файловой системы
$cache = new FilesystemCache();

// Получаем имя файла из аргументов командной строки
$filename = $argv[1];

// Считаем количество вхождений каждого слова в файле
try {
    $wordCount = countWords($filename, $cache);
    foreach ($wordCount as $word => $count) {
        echo "$word: $count\n";
    }
} catch (Exception $e) {
    echo "Error: " . $e->getMessage() . "\n";
}
?>
```

# Задача № 8

Написать программу, которая будет принимать на вход массив чисел и находить среднее арифметическое всех элементов массива. Массив необходимо получить в качестве аргумента командной строки, каждый элемент массива разделить запятой.

# Решение

```php
<?php
// Получаем массив из аргументов командной строки, кроме имени скрипта
$array = array_slice($argv, 1);

// Функция для преобразования строковых чисел в числа
$convert_to_int = function($value) {
    return (int)$value;
};

// Преобразуем каждый элемент массива в число
$array = array_map($convert_to_int, $array);

// Функция для подсчета суммы элементов массива
$sum = function($carry, $item) {
    return $carry + $item;
};

// Находим сумму элементов массива
$sum_of_array = array_reduce($array, $sum);

// Находим количество элементов в массиве
$count = count($array);

// Находим среднее арифметическое
$average = $sum_of_array / $count;

// Выводим результат
echo "Среднее арифметическое: $average\n";
?>
```

```
php average.php 1,2,3,4,5
//Среднее арифметическое: 3
```

# Задача № 9

Написать программу, которая будет получать входной документ и выводить ее на экран, заменяя все гласные буквы на символ "!" и все происходит на символе "?". Строку нужно получать в качестве аргумента командной строки.

# Решение

```php
<?php
// Получаем входную строку из аргументов командной строки
$input = $argv[1];

// Заменяем гласные буквы на символ "!"
$vowels = array('a', 'e', 'i', 'o', 'u', 'y');
$output = str_replace($vowels, '!', $input);

// Заменяем пробелы на символ "?"
$output = str_replace(' ', '?', $output);

// Выводим результат
echo $output;
?>

php my_program.php "Hello World"
//H!ll? W!rld
```

# Задача № 10

Написана программа, которая будет получать входной сигнал и проверку, является ли она палиндромом (читается одинаково слева направо и справа налево). Строку нужно получать в качестве аргумента командной строки.

# Решение

```php
<?php
$input = $argv[1]; // получаем входную строку из аргумента командной строки
$reversed = strrev($input); // переворачиваем строку задом наперед
if ($input === $reversed) { // сравниваем исходную строку с перевернутой версией
    echo "Строка является палиндромом";
} else {
    echo "Строка не является палиндромом";
}
?>
```

```
php palindrome.php "шалаш"
// Строка является палиндромом


php palindrome.php "привет"
// Строка не является палиндромом
```

# Рефлексия

Сегодня были решены задачи по теме: «Расширенные методы в PHP»

Ответьте на несколько вопросов:

1. Какая задача была самая интересная?
2. Какая задача показалась наиболее сложной?
3. Какую задачу вы поняли?


Спасибо
за
внимание