Лекції 7-9. ООП в PHP, огляд PHP-фреймворків, безпека.

Тема 7. Класи в PHP. Створення та клонування об’єктів. Успадкування. Перевизначення методів. Інтерфейси. Обробка виняткових ситуацій. Трейти.

Об’єктно-орієнтований PHP

- Об'єктно-орієнтований код працює повільніше, ніж процедурний код, але полегшує моделювання та керування складними структурами даних.
- PHP підтримує об'єктно-орієнтоване програмування з версії 3.0, і з тих пір ця об'єктна модель була значно розширена і реформована.

Декларування класів та створення об'єктів

```
<?php
class ExampleClass
{
// class code
}
<?php
$exampleObject = new ExampleClass();
$anotherObject = new ExampleClass;
```

Синтаксис та обмеження для успадкування

Синтаксис та обмеження для успадкування наведені в Таблиці 4.

Таблиця 4. Синтаксис та обмеження для успадкування

Concept	Syntax	Limitation
Inherit from a class	class A extends A_Parent	Class may have only one parent
Interface inheritance	Interface A extends B, C	Interface can inherit multiple interfaces
Inherit from an abstract class	Interface A extends B, C	Interface can inherit multiple interfaces
Implement interface	class A implements A_ Interface	Class can implement multiple interfaces
Trait	class Foo { use A_trait; }	Class can use multiple traits

Приклад. Оголошення класу товару (файл commodity.php)

```
<?php
class Commodity {
     public $name; // Назва
     public $category; // Категорія
     public $price; // Ціна
     public $availability; // Наявність
     function __construct($name, $category, $price = null,
$availability = False){
              echo 'Запущено конструктор...<br/>';
              $this->name = $name;
              $this->category = $category;
              $this->price = $price;
              $this->vailability = $availability;
```

```
    }
function __destruct() {
        echo 'Запущено деструктор...<br/>';
    }
function getPrice() {
        return (is_null($this->price) ?'N/A':$this->price);
    }
function setPrice($new_price){
        $this->price = $new_price;
    }
}
?>
```

Створення об'єктів

```
<?php
require_once 'commodity.php';
$obj = new Commodity('Розробка на PHP 5', 'book');
echo $obj->name.'-'.$obj->getPrice().'<br/>';
$obj->setPrice(230.0);
echo $obj->name.'-'.$obj->getPrice().'<br/>';
?>
```

Клонування об'єктів

```
<?php
require_once 'commodity.php';
$obj1= new Commodity('Розробка на PHP 5','book');
$obj1->price = 150.0;
$obj2 = $obj1; // Копіювання вказівників
$obj2->price = 230.0;
echo $obj1->price.'<br/>'; // 230.0
echo $obj2->price.'<br/>'; // 230.0
?>
```

Для клонування: $obj2 = clone $obj1;

Конструктор копіювання можна перевантажити, додавши в клас свою функцію __clone(). (Ця функція без аргументів, але може звертатися до вхідного об'єкта через $this та до копії через $that)

Простори імен

- Починаючи з PHP 5.3, крім класів доступний ще один спосіб організації проекту - простір імен.
- Він дозволяє організувати код у вигляді віртуальної ієрархії, що нагадує файлову систему.
- Як файли з однаковими іменами ізольовані, перебуваючи в різних каталогах, так класи, функції і константи PHP можуть бути ізольовані по різних просторах імен.
- Це дозволяє уникати конфліктів зі сторонніми бібліотеками, а також полегшує пошук і завантаження файлів.

Повне визначення імен в просторі імен

Якщо ви працюєте в просторі імен, то інтерпретатор припускає, що імена відносяться до поточного простору імен.

```
<?php
namespace MyApp\Helpers;
class Formatters
{
public static function asCurrency($val) {
// statement
}
}
```

Якщо ми хочемо використати цей клас з іншого простору імен, нам потрібно надати повний шлях до класу, як у цьому прикладі:

```
<?php
namespace MyApp\Lib;
echo MyApp\Helpers\Formatters::asCurrency(10);
```

Крім того, ви можете використовувати оператор use для імпорту простору імен, щоб вам не довелося постійно використовувати довгий формат:

```
<?php
namespace MyApp\Lib;
use MyApp\Helpers\Formatters;
echo Formatters::asCurrency(10);
```

Автозавантаження класів

Класи повинні бути визначені перед їх використанням, але ви можете використовувати автозавантаження для завантаження класів, коли це необхідно.

Стандарт кодування PSR-4 визначає, де PHP шукатиме клас.

Автозавантаження на PHP виконується функцією spl_autoload_register ().

```
<?php
function my_autoloader($class) {
include 'classes/' . $class . '.class.php';
}
spl_autoload_register('my_autoloader');
```

Успадкування

```
<?php
require_once 'commodity.php';
class Book extends Commodity {
    public $authors;
    function __construct($name, $authors, $category,
                         $price = null,
                         $availability = False) {
        parent::__construct($name, $category, price,
                            $availability)
        $this->authors = $authors;
    }
    function getAuthors() {
        return $this->authors;
    }
}
?>
```

Статичні члени класу

```
<?php
   class Visitor
   {
      private static $visitors = 0;
      function __construct()
      {
            self::$visitors++;
      }
     static function getVisitors()
     {
            return self::$visitors;
     }
    }
$visits = new Visitor();
    echo Visitor::getVisitors()."<br />";

$visits2 = new Visitor();
    echo Visitor::getVisitors()."<br />";
?>
```

The results are as follows:

1

2

Перевизначення методів

```
<?php
  $object = new Son;
  $object->test();
  $object->test2();
  class Dad
  {
    function test()
    {
      echo " [Class Dad] I am your Father<br> ";
    }
  }
class Son extends Dad
 {
    function test()
    {
      echo " [Class Son] I am Luke<br> ";
    }
    function test2()
    {
      parent::test();
    }
  }
?>
```

Magic (__*) Methods. Методи доступу __get() і __set()

Методи __get() і __set() викликаються при звертанні до недоступних властивостей класу. Наприклад, якщо ми б вирішили зберігати всі властивості товару в масиві, то ці методи нам би знадобились.

```
<?php
class Commodity {
    private $properties;
    function __set($name, $value) {
       echo “Задання нової властивості $name = $value";
        $this->properties[$name]=$value;
    }
    function __get($name) {
        echo “Читання значення властивості", $name;
        return $this->properties[$name];
    }
}
?>
... $book = new Book(...);
$book->weight = 100;...$weight=$book->weight;
```

Фінальні методи.

Метод, при визначенні якого використано ключове слово final, не можна перевантажувати в класах-спадкоємцях. Якщо final використано при визначенні самого класу, то породження від нього інших класів стає неможливим.

Методи __toString() та __call()

Метод __toString() викликається інтерпретатором PHP, якщо йому необхідно перетворити об'єкт в рядок (тобто при кожному виклику функцій echo() та print()). Перевантаживши його, ви можете вказати, як об'єкт повинен виглядати в рядковому вигляді.

Метод __call() викликається, якщо викликаний метод в класі не визначено.

Інші спеціальні методи (magic methods)

__sleep(), __wakeup() – потрібні при пакуванні/розпакуванні

__autoload() – описує, як підключаються файли з визначенням класів

```
class Car{
 public function __call($name, $arguments){
        echo "hello world!";
    }
}
$car = new Car();
$car->hello();
```

Метод hello() не визначений. Тому викликається метод __call().

Розглянемо ще один приклад:

```
class Car
{
    protected $_color;
    protected $_model;
    public function __call($name, $arguments)
    {
        $first = isset($arguments[0]) ? $arguments[0] : null;
        switch ($name)
        {
               case "getColor":
```

```
                return $this->_color;
               case "setColor":
                $this->_color = $first;
                return $this;
            case "getModel":
                return $this->_model;
            case "setModel":
                $this->_model = $first;
                return $this;
        }
    }
}
$car = new Car();
$car->setColor("blue")->setModel("b-class");
echo $car->getModel();
```

Абстрактні методи та класи

```
<?php
abstract class Commodity {
    ...
    abstract function printProperties();
}
?>
```

Інтерфейси

```
interface Int1 {
    function funct1();
    function funct2();
    . . .
}
class MyClass implements Int1, Int2 {
    public function funct1() {
        echo "Виклик метода 1";
    }
    public function funct2() {
        echo "Виклик метода 2";
    }
}
```

Розглянемо приклад.

```
class Person
{
  private $firstname;
  private $lastname;
  public function getFullName()
  {
      $fullname = $this->firstname." ".$this->lastname;
      return $fullname;
  }
  public function setFullName($aFirstname, $aLastname)
  {
      $this->firstname = $aFirstname;
      $this->lastname = $aLastname;
  }
}
```

```
interface JobCodes
{
     const PAYROLL = "01";
     const MANAGER = "02";
     const RETAIL  = "03";
}

interface StandardFunctions
{
     public function getJobTitle($aJobCode);
     public function showFullName();
}
class Employee extends Person
     implements JobCodes, StandardFunctions
{
     private $employeeId;
     private $jobcode;
     public function __construct($aFirstname, $aLastname,
                         $aEmpId, $aJobcode)
     {
          $this->setFullName($aFirstname, $aLastname);
          $this->employeeId = $aEmpId;
          $this->jobcode = $aJobcode;
     }
     function getJobTitle($aJobCode) {   }
     function showFullName() { }
}
```

Вбудовані інтерфейси

```
interface Iterator {
 public function rewind(); // Returns the iterator the beginning
 public function next();  // Get to the next member
 public function key(); // Get the key of the current object.
 public function current(); //Get the value of the current object
 public function valid(); // Is the current index valid?
}
```

Будь-який клас, який реалізує інтерфейс Iterator, може використовуватись в циклах, до них буде застосовано ітератор.

Приклад.

```
<?php
class iter implements iterator {
    private $items;
    private $index = 0;
    function __construct(array $items) {
        $this->items = $items;
    }
    function rewind() {
        $this->index = 0;
    }
    function current() {
        return ($this->items[$this->index]);
    }
```

```
    function key() {
        return ($this->index);
    }
function next() {
        $this->index++;
        if (isset($this->items[$this->index])) {
            return ($this->items[$this->index]);
        } else {
            return (NULL);
        }
    }
function valid() {
        return (isset($this->items[$this->index]));
    }
}
$x = new iter(range('A', 'D'));
foreach ($x as $key => $val) {
    print "key=$key\t value=$val\n";
}
Результат
key=0   value=A
key=1   value=B
key=2   value=C
key=3   value=D
```

Константи класу

```
class MyClass {
    const CONSTANT = 'value';
}
```

Звертання до констант класу

```
echo MyClass::CONSTANT; // Виводе "value"
```

Статичні властивості та методи PHP нагадують C++.

Порівняння об'єктів

При використанні операції порівняння (==) об'єкти рівні, якщо вони є екземпляри одного класу, мають однаковий набір атрибутів та однакові значення цих атрибутів.

При використанні операції перевірки ідентичності(===) об'єкти вважаються ідентичними, якщо вони посилаються на один і той же екземпляр одного класу.

Анонімні класи

PHP 7 ввів анонімні класи, які дозволяють вам визначити клас на льоту і створити з нього об'єкт. Ось простий приклад використання анонімного класу:

```
<?php
$object = new class('argument') {
public function __construct(string $message) {
echo $message;
}
```

```
};
```

Рефлексія

- В PHP функції API рефлексії дозволяють вам перевіряти елементи PHP під час виконання та отримувати інформацію про них.
- API Reflection був введений в PHP 5.0, а за замовчуванням був включений в PHP 5.3.
- Одне з найпоширеніших місць, де використовується відображення, - це unit-тестування.
- Один з прикладів того, де полегшує відображення, це тестування приватної властивості в класі. Ви можете використовувати рефлексію, щоб зробити доступною приватну властивість.

```
<?php
$reflectionObject = new ReflectionClass('Exception');
print_r($reflectionObject->getMethods());
```

Обробка виняткових ситуацій (виключень)

PHP 5 додає парадигму обробки виключень, вводячи структуру try/throw/catch. Вам залишається тільки створити клас, який успадковує клас винятків Exception:

```
class Exception {
/* Properties */
protected string $message ;
protected int $code ;
protected string $file ;
protected int $line ;
/* Methods */
public __construct ([ string $message = "" [, int $code = 0
                                    [, Exception $previous = NULL
]]] )
final public string getMessage ( void )
final public Exception getPrevious ( void )
final public int getCode ( void )
final public string getFile ( void )
final public int getLine ( void )
final public array getTrace ( void )
final public string getTraceAsString ( void )
public string __toString ( void )
final private void __clone ( void )
}
```

Приклад 1.

```
<?php
     try{
     @$fp = fopen("file.txt","w");
     if(!$fp)throw new Exception("Неможливо відкрити файл");
     // Запис даних
     . . .
     fclose($fp);
     }
     catch(Exception $ext){
          echo "Помилка в рядку ", $ext->getLine();
```

```
        echo $ext->getMessage();
    }
?>
```

Приклад 2.

```
<?php
class NonNumericException extends Exception {
    private $value;
    private $msg = "Error: the value %s is not numeric!\n";
    function __construct($value) {
        $this->value = $value;
    }
    public function info() {
        printf($this->msg, $this->value);
    }
}
try {
    $a = "my string";
    if (!is_numeric($argv[1])) {
        throw new NonNumericException($argv[1]);
    }
if (!is_numeric($argv[2])) {
        throw new NonNumericException($argv[2]);
    }
    if ($argv[2] == 0) {
        throw new Exception("Illegal division by zero.\n");
    }
    printf("Result: %f\n", $argv[1] / $argv[2]);
}
catch(NonNumericException $exc) {
    $exc->info();
    exit(-1);
}
catch(Exception $exc) {
    print "Exception:\n";
    $code = $exc->getCode();
    if (!empty($code)) {
        printf("Erorr code:%d\n", $code);
    }
    print $exc->getMessage() . "\n";
    exit(-1);
}
print "Variable a=$a\n";
?>
```

Результат

```
./script3.1.php 4 2
Result: 2.000000
Variable a=my string
./script3.1.php 4 A
Error: the value A is not numeric!
./script3.1.php 4 0
Exception:
Illegal division by zero.
```

Standard PHP Library (SPL)

Стандартна бібліотека PHP - це сукупність класів та інтерфейсів, які є рецептами для вирішення загальних проблем програмування. Вона доступна і скомпільована в PHP з версії 5.0.0.

Класи діляться на категорії.

Основні категорії: Data Structures, Iterators, Exceptions, File Handling, ArrayObject, SplObserver and SplSubject.

Трейти (Traits)

Трейти були введені в PHP 5.4.0 і призначені для полегшення деяких обмежень одиночного успадкування.

Трейт містить сукупність методів і властивостей, подібно до класу, але не може створити екземпляр самостійно.

Замість цього трейт входить до класу, і клас може потім використовувати його методи та властивості, як якщо б вони були оголошені в самому класі.

Ми використовуємо ключове слово trait, щоб оголосити трейт; Щоб включити його в клас, ми використовуємо ключове слово use. Клас може використовувати кілька трейтів.

Оголошення та використання трейтів

```
<?php
trait Singleton
{
     private static $instance;
     public static function getInstance() {
          if (!(self::$instance instanceof self)) {
               self::$instance = new self;
          }
          return self::$instance;
     }
}
class UsingTraitExample
{
     use Singleton;
}
$object = UsingTraitExample::getInstance();
var_dump($object instanceof UsingTraitExample); // true
```

Тема 8. Шаблон MVC. Огляд сучасних фреймворків. Використання Composer. Фреймворки Yii , Laravel, CodeIgniter, Zend Framework. Стандарти PSR.

Код PHP та HTML-шаблон сторінки

Є декілька підходів до розділення шаблону сторінки та коду сценарію [18].

1-й варіант. “Вкраплення” HTML в код (це взагалі без відділення – самий гірший варіант)

2-й варіант. Вставка коду в шаблон

```
<html> <body><h2>Останні новини:</h2>
<? $f=fopen("../news.txt) ?>
<? for($i=...){ ?>
     <li><?=$i?>-та новина:<?=fgets($f,1024)?>
<?}?>
</body></html>
```

Це трохи краще.

3-й варіант. Model-View-Controller

Модель. Представляє предметну область системи, її вміст. Включає БД системи та код, що з нею працює.

Представлення. Це шаблон, що визначає зовнішній вигляд.

Контролер. Код бізнес-логіки, приймає дані від користувача, пов'язує Модель та Шаблон (Дивись Рис.42).

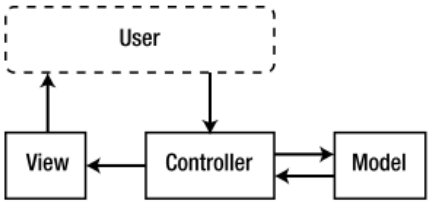


Рис.42 Шаблон Модель-Представлення-Контролер

Огляд сучасних PHP-фреймворків

Фреймворк (англ. Framework, каркас, платформа, структура) – інфраструктура програмних рішень, що полегшує розробку складних систем.

Найпопулярнішими PHP-фреймворками станом на 2018 рік вважаються Laravel, Symfony, Yii, CakePHP, Zend Framework, CodeIgniter.

Що надають фреймворки?

- Каркас застосунку на основі ООП та шаблону MVC.
- Реалізують нові стандарти PSR.
- Використовують Composer.
- Вбудовані інструменти тестування.
- Можливість швидкої розробки .
- Забезпечують добре організований, зрозумілий код, який легко модифікувати.
- Високу безпеку вашого сайту.
- Сучасні веб-розробницькі практики .

Стандарти PSR

PHP Standards Recommendations

Стандарт PSR-1. Основний стандарт кодування.

- Використовувати тільки теги <?php і <?= .
- Тільки UTF-8 без BOM.
- Класам НЕОБХІДНО давати імена в стилі **StudlyCaps .**
- Константам класів НЕОБХІДНО давати імена у верхньому регістрі з символом підкреслення як роздільник: **MAIN_PRODUCT.**

- Методам НЕОБХІДНО давати імена в стилі **camelCase .**

PSR-2. Стиль кодування

- Для оформлення відступів ПОВИННІ використовуватися чотири пробіли (але не знак табуляції).
- Довжина рядка < 120.
- Після визначення простору імен (**namespace**) і після блоку імпорту просторів імен (**use**) ПОВИНЕН бути один порожній рядок.
- Відкриваюча фігурна дужка у визначенні класу ПОВИННА розташовуватися на новому рядку.
- Видимість НЕОБХІДНО оголошувати для всіх властивостей і методів (public, private ...).
- В кінці файлу не закривати PHP **?>.**

Composer

Composer - це пакетний менеджер рівня додатків для мови програмування PHP, який надає засоби з управління залежностями в PHP-додатку.

- Dependency Manager for PHP.
- https://getcomposer.org/
- Аналог npm для Node.js, NuGet для Visual Studio, Bundler для Ruby.
- З'явився 1 березня 2012 р.
- Composer працює через інтерфейс командного рядка і встановлює залежності (наприклад бібліотеки) для додатків.
- Має офіційний репозиторій пакетів https://packagist.org/

Установка Composer під Windows та XAMPP

- Скачуємо сам Composer
  - https://getcomposer.org/Composer-Setup.exe
- Інсталлюємо
  - Під час інсталяції треба буде вказати шлях до php.exe (в xampp це як правило: C:\xampp\php\php.exe)
  - Потрібно включити php_openssl.dll (розкоментувати extension = php_openssl.dll). Йдемо в C:\xampp\php\php.ini і включаємо, що потрібно.

Тема 9. Безпека створюваних web-застосунків. Файл .htaccess. MOD_REWRITE.

Безпека створюваних web-застосунків.

Безпека є головною проблемою для веб-програм. Навіть такі великі організації, як Організація Об'єднаних Націй, були зламані з використанням дуже простих недоліків безпеки [5].

Безпека в середовищі Інтернет включає в себе багато різних аспектів, таких як система захисту Web-сервера, безпеку баз даних, перевірка даних, що вводяться користувачами, прийоми і методи шифрування та інше.

Configuration

Бажано використовувати останні стабільні релізи PHP.

Переконайтеся, що ви зберегли свою операційну систему. Регулярно застосовуйте оновлення безпеки та стежте за новою інформацією про безпеку.

Errors and Warnings

Вам слід настроїти PHP, щоб приховати попередження та помилки під час їх виникнення. Помилки та попередження можуть дати хакеру ключ до внутрішніх функцій вашого коду, таких як назви каталогів та те, які бібліотеки ви використовуєте. Така інформація може допомогти йому використовувати вразливі місця у вашому стекі.

PHP as an Apache Module

Якщо PHP працює як модуль Apache, він буде запускатися з використанням того самого користувача, що і сервер Apache. Це означає, що він матиме ті самі дозволи та доступ, що і користувач Apache.

Краще налаштувати користувача для Apache, а не запустити його як "nobody". Користувач Apache повинен мати обмежений доступ до файлової системи, і не повинен перебувати в списку sudoers.

Ви повинні використовувати налаштування PHP open_basedir, щоб обмежити доступ до каталогів PHP. Ви можете протиставити його налаштуванню doc_root, що впливає на те, файли з яких каталогів PHP буде обслуговувати.

Session Security

Дві зони уваги, які вам слід знати, - це "викрадення сеансу" (session hijacking) та "фіксація сеансу" (“session fixation).

Викрадення сеансу

HTTP - це протокол без збереження стану, і веб-сервер, як очікується, одночасно обслуговуватиме кількох різних відвідувачів.

Сервер повинен мати можливість розрізняти клієнтів і робить це, призначаючи кожному клієнту ідентифікатор сеансу. Ідентифікатор сеансу можна отримати, викликаючи session_id(). Він створюється після виклику функції session_start().

Коли клієнт робить наступні запити на сервер, їм надають ідентифікатор сеансу, який дозволяє серверу пов'язувати запит із сеансом.

Клієнти можуть надавати сеанс як за допомогою файлів cookie, так і з параметром URL-адреси.

Файли cookie краще, але вони не завжди доступні. Якщо PHP не може використовувати файл cookie, він автоматично і прозоро використовуватиме

URL-адресу, якщо ви не встановите параметр session.use_only_cookies у файлі php.ini.

Очевидним, що якщо ви зможете вказати на сервер чийсь ідентифікатор сеансу, то ви зможете маскуватися як цей користувач.

На Рис.43 показано сценарій, в якому злодійний користувач Боб здатний перехопити повідомлення Аліси на сервер. Bad Bob читає запит і витягує ідентифікатор сеансу (який міститься в заголовках файлів cookie для запиту HTTP). Тоді він може представити цей ідентифікатор сеансу на сервер, який тепер не може відрізнити його від Аліси.

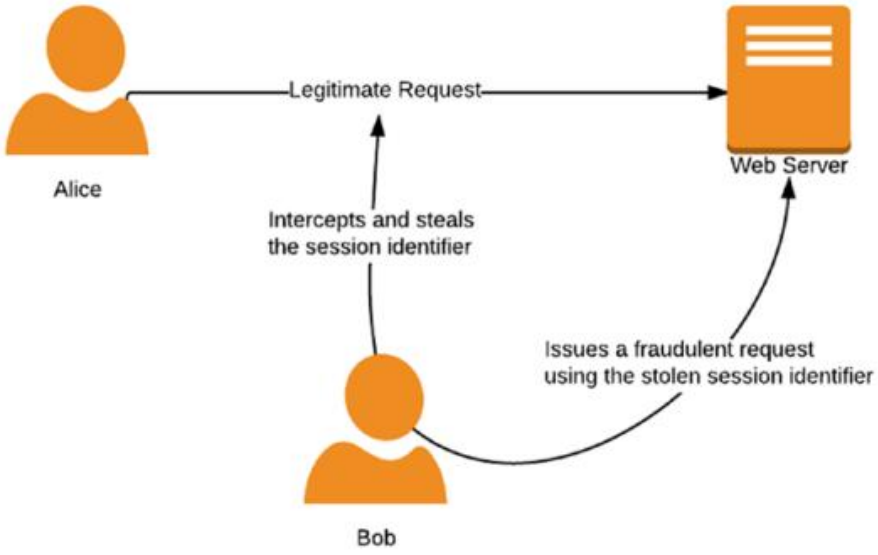


Рис.43. Викрадення сеансу

Отримання ідентифікатора сеансу іншого користувача може виконуватися кількома способами.

Якщо ідентифікатор сеансу відповідає передбачуваному шаблону, то зловмисник може спробувати визначити, що це буде для користувача. PHP використовує дуже випадковий спосіб генерування ідентифікатора сесії, тому вам не потрібно турбуватися про це.

Перевіряючи мережевий трафік між клієнтом і сервером, зловмисник може прочитати ідентифікатор сеансу. Ви можете встановити session.cookie_secure = On, щоб cookie-сесії були доступні лише через HTTPS, щоб пом'якшити це. HTTPS також буде шифрувати запитувану URL-адресу, і тому, якщо ідентифікатор сеансу буде переданий як параметр у запиті, він буде зашифрований.

Атаки проти клієнта, такі як атака XSS або троянець, запущені на комп'ютері, також можуть виявити ідентифікатор сеансу. Це може бути частково пом'якшене шляхом встановлення директиви session.cookie_httponly.

Session Fixation

Закріплення сесії використовує слабкість у веб-додатку. Деякі програми не генерують новий ідентифікатор сеансу для користувача під час автентифікації. Замість цього вони дозволяють використовувати існуючий ідентифікатор сеансу.

Напад відбувається, коли противник створює сеанс на веб-сервері. Він знає ідентифікатор сеансу для цього сеансу. Тоді він вводять в оману користувача, який використовує цей сеанс, і автентифікує себе. Після цього атакуючий може використовувати відомий ідентифікатор сеансу, і має права користувача автентифікації.

Існує декілька способів встановити ідентифікатор сеансу, і фактичний метод буде залежати від того, як програма приймає ідентифікатор.

Найпростіший спосіб зробити це було б передати ідентифікатор сесії в URL, як тут http://example.org/index.php?PHPSESSID=1234.

Найкращий спосіб зменшити ризик зафіксування сесії - це викликати функцію session_regenerate_id() кожного разу, коли змінюється рівень привілеїв, наприклад, після входу в систему.

Ви можете встановити session.use_strict_mode=On у вашому файлі конфігурації. Цей параметр змусить PHP використовувати тільки сайти, які ідентифікують сеанс. Він відхилить ідентифікатор сеансу, який буде надано користувачем. Це пом'якшить спроби маніпулювати файлом cookie.

Налаштування session.use_cookies=On та session.use_only_cookies=On перешкодить PHP приймати ідентифікатор сеансу з URL-адреси.

XSS-ін'єкція (Міжсайтовий скриптинг)

XSS-ін'єкція – це атака, яка дозволяє зловмиснику вставляти в HTML-код сайту вставки шкідливого HTML-коду, використовуючи скрипти JavaScript.

Специфіка подібних атак полягає в тому, що для атаки на сервер в якості засобу атаки використовується авторизований на цьому сервері клієнт.

XSS розшифровується як Cross-Site Scripting (CSS вже використано, тому XSS).

Атака ефективна, оскільки клієнт вважає, що код походить з веб-сайту, якому він довіряє. Код може мати доступ до ідентифікаторів сеансу, файлів cookie, даних зберігання HTML та іншої інформації, пов'язаної з сайтом.

Існує декілька поширених типів атак XSS: stored, reflected та DOM.

У збереженій (stored) атаці XSS зловмисник може отримати вхід у місце збережігання на сервері. Прикладом можуть бути коментарі користувача, які відображаються на сайті і зберігаються в базі даних. Коли сайт виводить список коментарів користувачів іншому відвідувачеві, вони отримають шкідливий код.

У відображеній (reflected) атаці XSS зловмисник може скористатися веб-сайтом, щоб вивести щось безпосередньо. Найпоширенішою формою цієї атаки є помилка заповнення форми, яка попередньо заповнює поля введення за допомогою раніше наданих полів або виводить помилкове значення поля. Надіславши відвідувача на створену URL-адресу, яка містить шкідливий код, як повідомлення про помилку (наприклад), зловмисник може змусити клієнта виконати його в контексті надійного сайту.

Атака DOM це така, яка цілком залежить від сторінки. Шкідливий код читається з елемента на сторінці, і викликає код, що виконується в самій сторінці.

Крім того, атаки XSS можна класифікувати як атаки на стороні сервера або на стороні клієнта. Атака на стороні сервера - це така, коли сам сервер видає шкідливий код. Клієнтська XSS виникає, коли ненадійні дані, надані користувачем, використовуються для оновлення DOM з небезпечним викликом JavaScript.

Пом'якшення нападів XSS

Найважливіше правило, яке слід дотримуватися - ніколи не надсилати неекрановані вихідні дані для клієнта. Завжди фільтруйте дані та видаляйте шкідливі теги, перш ніж надсилати їх клієнту.

Запам'ятайте цю мантру: “Filter input, escape output”.

Для цього є три корисні функції: htmlspecialchars(), htmlentities() і strip_tags().

Найбезпечнішим способом екранування виведення перед відображенням є використання filter_var ($ string, FILTER_SANITIZE_STRING).

CSRF-атака. (Cross Site Request Forgery)

Атака CSRF (“Підробка міжсайтових запитів”) експлуатує довіру, яку веб-сайт має до клієнта. У цих атаках зловмисник нав'язує клієнту виконання команди на веб-сайті, який довіряє цьому клієнту.

Найбільш поширеною формою є відправлення запиту POST до форми введення.

Уявіть, що Аліса ввійшла на веб-сайт свого банку, який має форму, яка дозволяє їй переказати гроші на інший рахунок. Чак знає кінцеву точку цієї форми та ті поля вхідних даних, які вона має. Він якимось чином намагається змусити веб-браузер Аліси відправляти запит POST на цю форму, наказуючи банку переказати гроші на його рахунок. Банк довіряє веб-браузеру Alice, оскільки він має дійсний сеанс і виконує запит.

Існує багато способів, як Чак обманює веб-браузер Аліси, зокрема, використання фреймів і JavaScript.

Щоб пом'якшити ці запити, потрібно створити унікальний і дуже випадковий токен, який ви зберігаєте в сеансі Аліси. Коли ви видаєте форму, ви включаєте цей токен, так що, коли Аліса подає форму, вона також подає токен. Перш ніж обробляти форму, ви перевіряєте, чи поданий токен відповідає токену, що зберігається в її сеансі.

Чак не може дізнатися, який токен на сеансі Аліси, і тому не зможе його включити в його POST. Ваш код відхилить запит, яким він хоче обдурити Алісу, оскільки він не має дійсного токену.

Часто банки часто вимагають, щоб особа повторно автентифікувала під час виконання чутливої операції, і часто це вимагає двофакторної аутентифікації в рамках цього процесу.

SQL Injection

SQL-ін'єкція є найпоширенішою формою атаки в Інтернеті, і одна з найпростіших для захисту. SQL-ін'єкція відбувається, коли зловмисник може вставляти шкідливі команди в SQL-запит для виконання базою даних.

Багато установок баз даних дозволяють базі даних записувати файли на диск. Ця функція дозволяє хакерам створювати задні двері, використовуючи базу даних для запису скриптів PHP у каталозі, в якому веб-сервер обслуговує його.

Це означає, що ефект SQL ін'єкції не обмежується тим, що ваша база даних буде під загрозою, але може призвести до того, що зловмисник зможе виконувати довільний код у вашій базі даних.

Проблема з ін'єкцією SQL витікає з того факту, що в операторі SQL містяться дані та синтаксис. Забезпечуючи включення користувацьких даних в синтаксис функції, ми створюємо можливість того, що шкідливі дані можуть перешкоджати синтаксису.

Prepared Statements (підготовлені вирази)

Найефективнішим способом початку пом'якшення SQL-ін'єкції на мові PHP є використання виключно підготовлених виразів для взаємодії з вашою базою даних. Це допоможе виключити більшість атак на SQL-ін'єкцію, однак для цього не достатньо для надійності.

Підготовлені вирази настільки важливі, що драйвер PDO буде емулювати їх, якщо основний драйвер не підтримує їх.

Підготовлені вирази виконуються в три етапи:

1. Налаштуйте вираз з заповнювачами для даних.

2. Прив'яжіть фактичні дані до виразу.

3. Виконайте підготовлений вираз.

Можна пов'язати нові дані з виразом, який ви вже виконали, а потім запустити його знову за допомогою нового виразу. Двигун бази даних не повинен знову аналізувати SQL, що дає покращення продуктивності, крім переваг безпеки.

Цей код наводить приклад того, як підготувати, прив'язати та виконати вираз:

```
<?php
$stmt = $dbh->prepare("SELECT * FROM REGISTRY where name = ?");
$stmt->bindParam(':name', $_GET['name'], PDO::PARAM_STR, 12);
$stmt->execute();
```

Зверніть увагу, що функція PDO::prepare() повертає об'єкт типу PDOStatement.

Ми використовуємо GET-змінну безпосередньо, тому нам не потрібно екранувати її, оскільки вона пов'язана як змінна з PDOStatement::bindParam() і не може змінити синтаксис SQL, який буде запускатися.

Інші драйвери баз даних на PHP також підтримують підготовлені вирази. Ось приклад з посібника для MySQL:

```
     /* Prepared statement, stage 1: prepare */
     if (!($stmt = $mysqli->prepare("INSERT INTO test(id) VALUES
(?)"))) {
     echo "Prepare failed: (" . $mysqli->errno . ") " . $mysqli-
>error;
     }
     /* Prepared statement, stage 2: bind and execute */
     $id = 1;
     if (!$stmt->bind_param("i", $id)) {
     echo "Binding parameters failed: (" . $stmt->errno . ") " .
$stmt->error;
     }
     if (!$stmt->execute()) {
     echo "Execute failed: (" . $stmt->errno . ") " . $stmt->error;
     }
```

Escaping (екранування)

Менш ефективним способом пом'якшення SQL-ін'єкції є уникнення спеціальних символів, перш ніж надсилати їх до бази даних. Цей спосіб більш схильний до помилок, ніж використання підготовлених виразів.

Якщо ви збираєтеся спробувати уникнути спеціальних символів, ви повинні використовувати певну функцію бази даних (наприклад, mysqli_real_escape_string()) або PDO::quote(), але не загальні функції, такі як addslashes().

Remote Code Injection

Впровадження віддаленого коду - це атака, коли зловмисник може отримати сервер для включення та виконання свого коду.

Functions That Evaluate Strings as Code

Деякі функції, такі як eval(), exec() та system(), чутливі до віддаленого використання коду. Якщо ви виконуєте змінну, яка включає користувацький ввід, вони зможуть вводити команди за допомогою escape символів.

Ви можете пом'якшити це, використовуючи escapeshellargs(), щоб виключити аргументи, передані до команди оболонки. Функція escapeshellcmd() сама екранує команди оболонки.

Функція assert() використовується для того, щоб переконатися, що певна умова є істинною і виконувати певні дії, якщо це не так. Це корисно для налагодження, але ви повинні вимкнути її для production. Ви можете використовувати функцію assert_options(), щоб налаштувати та вимикати assert().

Гра з include та require

Функції include() і require() дозволяють включати файли, вказані URL-адресою, якщо ввімкнено параметр PHP конфігурації allow_url_include.

Найпоширенішою подією є випадки, коли люди використовують змінну GET у URL-адресі, щоб визначити який-небудь динамічний вміст для включення. Це дуже аматорська помилка.

Наприклад, сайт може мати таку URL-адресу, як http://example.com/index.php?sidebar=welcome, а потім динамічно додавати файл welcome.php на бічну панель.

Зловмисник міг надати URL-адресу замість рядка "welcome" і мати свій власний код на сервері з тими ж рівнями привілеїв, що і користувач веб-сервера.

Щоб протистояти виникненню такої проблеми, ви можете вимкнути enable_url_fopen, використати basename() до вказаної вами змінної, щоб шляхи були включені лише з білого списку.

```
<?php
$page = $_GET['page'];
$allowedPages = array('adverts','contacts','information');
if ( in_array($page, $allowedPages) ) {
include basename($page . '.html');
}
```

Email Injection

Користувачам можливо ввести шістнадцяткові контрольні символи, які дозволяють їм змінювати текст повідомлення або список одержувачів.

Наприклад, якщо ваша форма дозволяє людині вводити свою електронну адресу як поле "від" для електронного листа, наступна рядок призведе до того, що додаткові одержувачі будуть включені як cc і сліпими одержувачами копії копії повідомлення:

sender@example.com%0ACc:target@email.com%0ABcc:anotherperson@emailexample
.
com,stranger@shouldhavefiltered.com

Зловмисник також може надати власне тіло і навіть змінити тип MIME відправленого повідомлення. Це означає, що ваша форма може бути використана спамерами для відправлення пошти.

Ви можете захиститись від цього декілька способів.

Переконайтеся, що ви правильно фільтруєте вхід, який ви використовуєте при надсиланні пошти. Функція filter_var() надає ряд прапорців, які ви можете використовувати, щоб переконатися, що ваші вхідні дані відповідають потрібному шаблону.

```
<?php
$from = $_POST["sender"];
$from = filter_var($from, FILTER_SANITIZE_EMAIL);
// send the email
```

Ви також можете встановити та використовувати Suhosin PHP розширення. Воно забезпечує директиву suhosin.mail.protect, яка буде захищати це.

Файл .HTACCESS (від англ. Hypertext access)

.HTACCESS – це файл додаткової конфігурації веб-сервера Apache.

Дозволяє задавати велику кількість додаткових параметрів і дозволів для роботи веб-сервера в окремих каталогах.

Файл .htaccess можна розмістити в довільному каталозі. Директиви цього файлу діють на всі файли поточного та вкладених каталогів.

Використання:

- Авторизація, аутентифікація.
- Зміна URL-адрес (довгих, складних на короткі).
- Заборона/Відкриття доступу для певних IP-адрес.

MOD_REWRITE. Директиви складного перенаправлення

MOD_REWRITE – це модуль в складі Apache. Цей модуль дозволяє «переписувати» URL сторінки «на льоту».

За спеціальними правилами запит на URL з клієнтської програми буде розбитий на частини, проаналізований та перероблений в інший URL перед виконанням запиту.

Зазвичай це використовують для конвертування динамічного URL з параметрами у статичний з іменем файла (створення "дружніх" адрес).

Наприклад, запит до сайту новин:

```
www.новини.in.ua/search.php?день=12&місяць=квітень&рік=2006
```

перетвориться на:

```
www.новини.in.ua/search-12-april-2006.html
```

Як використовувати MOD_REWRITE

Створити директиви і розмістити їх в файлі .htaccess

- Директива RewriteEngine - вмикає / вимикає механізм mod_rewrite
- Директива RewriteRule - описує правило зміни адреси URL
- Директива RewriteCond - визначає умову, при якій відбувається перетворення.
- RewriteCond - визначає умову для якого-небудь правила.
- Перед директивою RewriteRule знаходиться нуль або декілька директив RewriteCond

Приклад. Перенаправляємо запити на сторінку dummy.html на сайт Google, використовуючи перенаправлення 301

```
RewriteEngine on
RewriteRule ^dummy\.html$ http://www.google.com/ [R=301]
```

Як працює RewriteRule

Узагальнений синтаксис директиви має вигляд:

```
RewriteRule Pattern Substitution [Optional Flags]
```

Pattern - регулярний вираз шаблону. Якщо URL відповідає шаблону, то правило виконується.

Substitution - новий URL, який буде використовуватися замість адреси, що відповідає шаблону.

[Optional Flags] - один або кілька прапорів, які визначають поведінку правила.

Приклад. Запобігаємо використанню посилань на зображення.

```
RewriteEngine on
RewriteCond %{HTTP_REFERER} !^$
RewriteCond   %{HTTP_REFERER}   !^http://(www\.)?example\.com/.*$
[NC]
RewriteRule .+\.(gif|jpg|png)$ - [F]
```

Набір правил у файлі .htaccess говорить:

- якщо змінна HTTP_REFERER містить значення,
- і воно не починається на http://example.com/ або http://www.example.com/,
- і запитуваний URL містить ім'я файлу зображення, то треба відмовити запитом з помилкою "403 Forbidden“ (NC – чутливість до регістру символів)

Приклад. Направлення всіх запитів одному скрипту для обробки.

Для цього потрібно додати у файл .htaccess наступний вміст:

```
RewriteEngine   on
RewriteCond   %{REQUEST_FILENAME}  !-f
RewriteCond   %{REQUEST_FILENAME}  !-d
RewriteRule   ^(.*)$  index.php  [L, QSA]
```

Скрипт index.php буде брати URL з $_SERVER['REQUEST_URI']

Приклад. Заборона завантаження файлів зображень з вашого сайту.

Треба записати в кінці файлу .htaccess:

```
Options +FollowSymlinks
RewriteEngine On
RewriteCond %{HTTP_REFERER} !^$
RewriteCond %{HTTP_REFERER} !^http://(www.)?vash_site.com/ [nc]
RewriteRule                                          .*.(gif|jpg|png)$
http://vash_site.com/img/stop_stealing.gif[nc]
```

Де stop_stealing.gif – зображення, яке буде з'являтися при спробі завантажити gif | jpg | png.

Лекції 10-12. Мова JavaScript на боці клієнта і сервера та інші мови web-програмування

Тема 10. Технологія Node.js

Що таке Node.js ?

Node.js – платформа з відкритим кодом для написання серверної частини веб-застосунків на мові JavaScript.

Node.js характеризується такими властивостями:

- асинхронна однопотокова модель виконання запитів;
- неблокуюче введення/виведення;
- система модулів CommonJS (специфікація для модулів);
- движок JavaScript Google V8;

- для управління модулями використовується пакетний менеджер npm (node package manager).

Платформа Node.js призначена для створення додатків реального часу, що оброблюють великі обсяги даних (DIRTy-додатків).

Автор технології – Ryan Dahl (2009).

По суті не придумав нічого нового:

“node.js is a set of bindings to the V8 JavaScript VM” (Ryan Dahl)

- V8 – Google JavaScript Engine, що використовується в браузері Chrome.
- Краща технологія 2012 року по версії InfoWorld.
- Сайт nodejs.org

Ryan Dahl запропонував новий спосіб мислення про те, як треба будувати програмне забезпечення web-системи.

Головні ідеї:

- Виконання серверного коду в одному потоці.
- Асинхронне виконання операцій введення-виведення.
- Серверний JavaScript.

Це було і до Node.js, але саме Ryan Dahl все це об’єднав при розробці Node.js.

Неблоковане введення/виведення

Traditional I/O

```
var result = db.query(“select x from table_Y”);
doSomethingWith(result); // wait for result!
doSomethingWithOutResult(); // execution is bloked!
```

Non-bloking I/O

```
var result = db.query(“select x from table_Y”, function(result){
     doSomethingWith(result); // wait for result!
});
doSomethingWithOutResult (); // executes without any delay!
```

Платформа чи середовище?

- Node - це платформа, а не середовище розробки (framework) JavaScript-додатків.
- Поширеною помилкою є ототожнення Node з середовищами типу Rails або Django. Насправді Node є набагато більш низькорівневим інструментом.

Одне з популярних середовищ Node-розробки – це Express (http://expressjs.com/)

Ключові частини Node.js

Архітектура Node.js показана на Рис. 44.

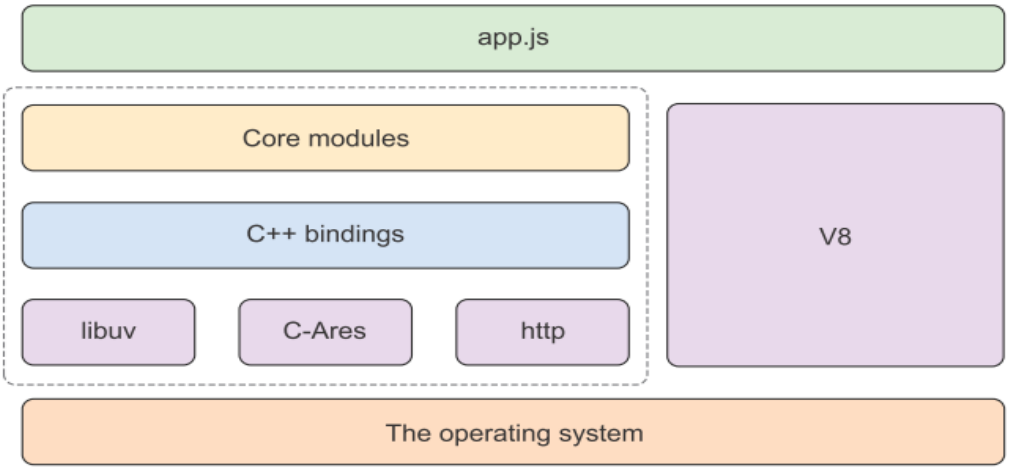


Рис.44. Ключові частини Node.js

libuv – binary library, which provides a fast run loop and non-blocking I/O for networking and the file system.

http – binary library for HTTP .

Трошки практики

Багато речей можна реалізувати на Node.js в декілька рядків коду. Реалізація HTTP-сервера:

```
var http = require ( 'http' );
http.CreateServer( function (req, res){
     res.writeHead( 200, { 'Content-Type': 'text/plain'});
     res.End('Hello World\n');
}).listen(process.env.PORT, process.env.IP);
console.log('Server running!');
```

Проблеми архітектур більшості web застосунків

Робота з базою даних:

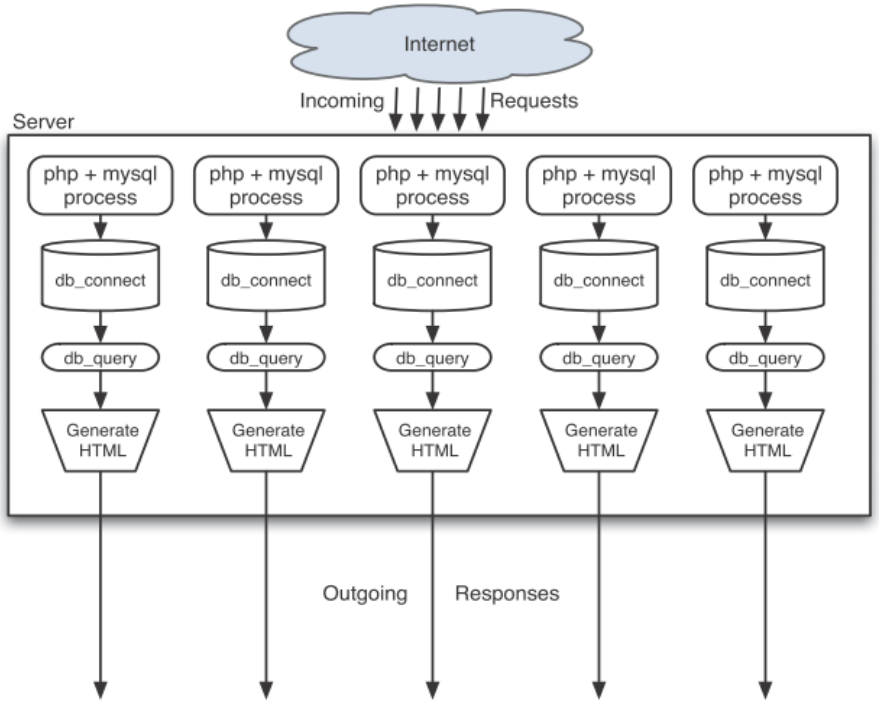
```
result = query('select * from T')
```

Що відбувається під час виконання цього коду?

Створюється окремий поток.

Серверний код чекає виконання операції.

Якщо запитів багато (Рис.45), то сервер може бути заблокованим.



.1 Traditional blocking IO web servers

Рис.45. Традиційний блокований Input/Output на сервері

Затрати на операції в сучасних комп'ютерах:
L1: 3 такти, L2: 14 тактів, RAM: 250 тактів,
Диск: 41 000 000, Мережа: 240 000 000 тактів.

Виділення потоку на обробку запиту клієнта
Перемикання між контекстами потоків затратне.
При масивній конкуренції ви не можете виділяти поток для кожного запиту. Таким чином всі запити, що надходять, будуть чекати.

Як працює Node.js в якості сервера?

Обробка запитів в Node.js ілюструют Рис.46 та Рис.47.

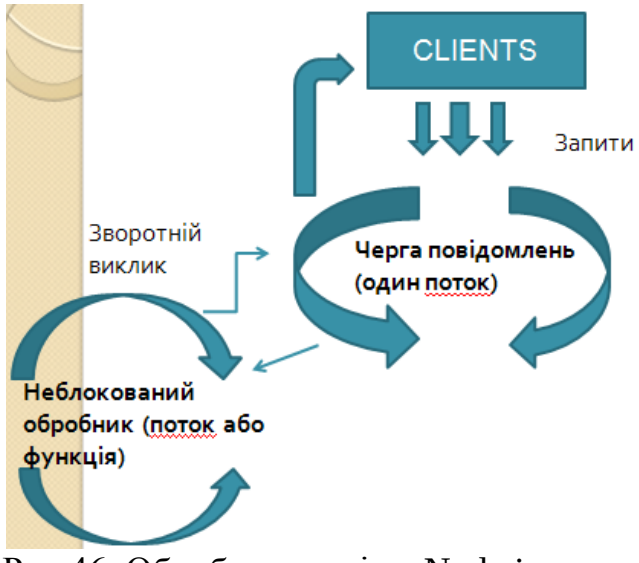


Рис.46. Обробка запитів в Node.js

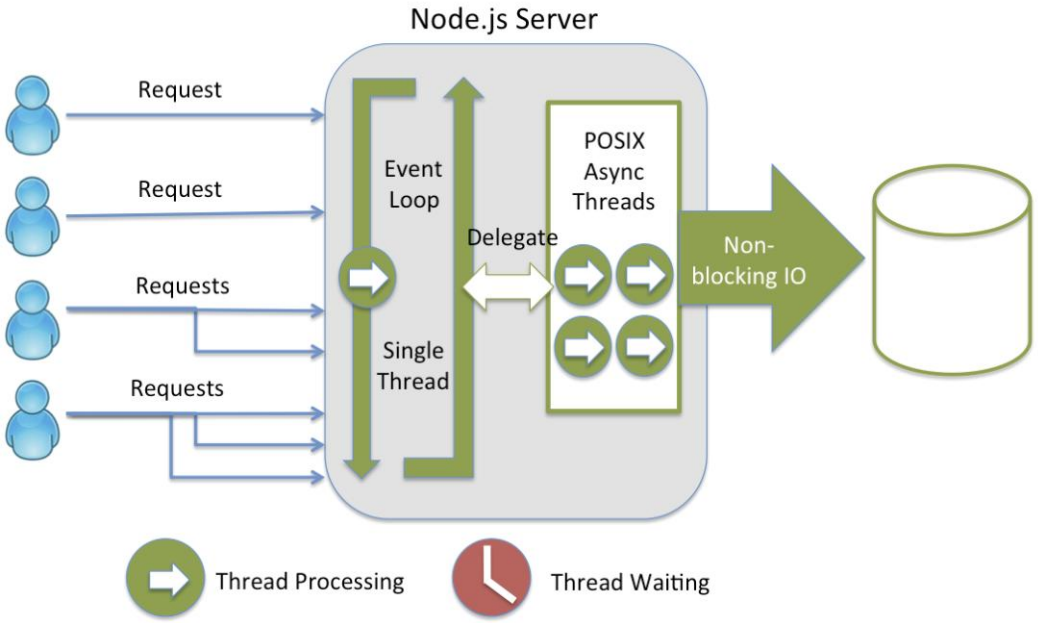


Рис.47. Обробка запитів на сервері Node.js

Порівняльна оцінка швидкодії Apache- та NGINX-серверів (http://mng.bz/eaZT сайт WebFaction) показана на Рис.48.

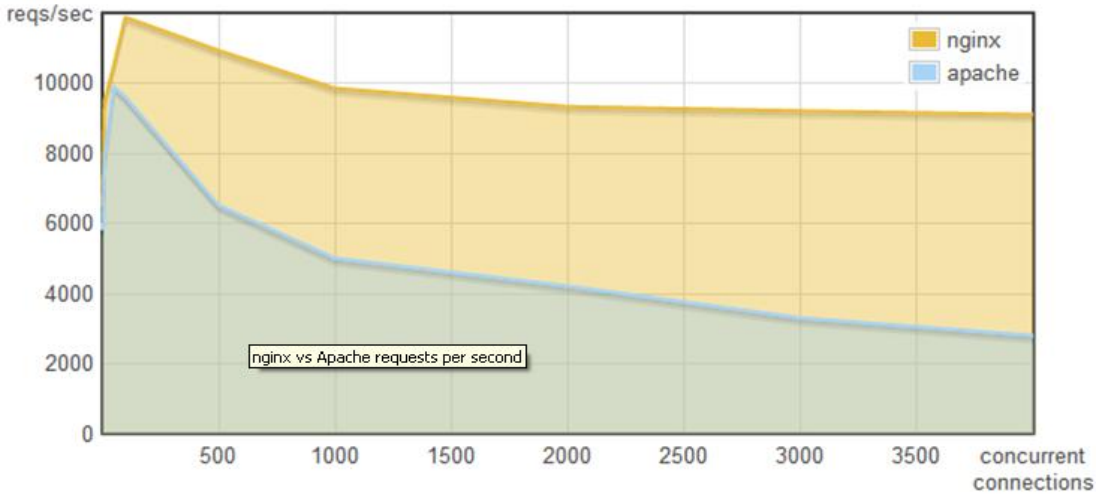


Рис.48. Швидкодії Apache- та NGINX-серверів

По осі X – кількість з'єднань, по осі Y – кількість запитів, що обробляються за секунду.

Перевага NGINX (як і Node.js) в тому, що замість багатопоточного підходу використовується асинхронність.

DIRTy-застосунки

Назва DIRTy походить від абревіатури DIRT (data-intensive real-time), що відноситься до застосунків реального часу.

Хорошим прикладом створеного в Node DIRTy-застосунку є Browserling (browserling.com). Ця програма дозволяє у вікні одного браузера відкривати і використовувати інші браузери.

Додатковий проект Testling (testling.com).

Що ж там всередині? Модулі.

Розробка в Node.js основана на модулях (модулі – це JavaScript-файли).

Саме модулі дозволяють застосунку спілкуватися з зовнішнім світом та структурувати ваш код.

Існують три джерела модулів:

- Вбудовані в Node (core modules).
- Ваші файли (власні модулі).
- Глобальний репозиторій.

Вбудовані модулі (core modules)

Забезпечують серверну інфраструктуру.

Модуль підключається так:

```
var fs = require ( 'fs' );
```

Приклади модулів:

fs (http://nodejs.org/api/fs.html) - file system module.

http (http://nodejs.org/api/http.html) – HTTP.

crypto (http://nodejs.org/api/crypto.html).

os - операційна система.

Глобальний репозиторій

Це екосистема Node Package Manager (NPM).

https://www.npmjs.org > 400 000 модулів.

Версії Node.js

Версії Node.js показані на Рис.49 (https://github.com/nodejs/Release).

Release	Status	Codename	Initial Release	Active LTS Start	Maintenance LTS Start	End-of-life
v0.10.x	**End-of-Life**	-	2013-03-11	-	2015-10-01	2016-10-31
v0.12.x	**End-of-Life**	-	2015-02-06	-	2016-04-01	2016-12-31
4.x	**Maintenance LTS**	Argon	2015-09-08	2015-10-01	2017-04-01	2018-04-30
5.x	**End-of-Life**		2015-10-29			2016-06-30
6.x	**Active LTS**	Boron	2016-04-26	2016-10-18	2018-04-30	April 2019
7.x	**End-of-Life**		2016-10-25			2017-06-30
8.x	**Active LTS**	Carbon	2017-05-30	2017-10-31	April 2019	December 2019[1]
9.x	**Current Release**		2017-10-01			June 2018
10.x	**Pending**	Pending	Apr 2018	October 2018	April 2020	April 2021

Рис.49. Версії Node.js

Огляд версій

Node.js v4.0.0 – Aug 2015. Движок v8 v4.5

Підтримка ES6 стандарту:

- Блочна область видимості
- Класи
- Типізовані масиви
- Генератори
- Promises
- Symbols
- Рядкові шаблони
- Колекції (Map, Set, і так далі)

Node.js v6.0.0 – Apr 2016. Движок v8 v5.0

- Зміни в API
- Поліпшили підтримку EcmaScript 6
- Конструктор об'єкта Buffer змінив свою поведінку
- Об'єкт EventEmitter отримав два нових методи prependListener і prependOnceListener
- Методи fs.realpath і fs.realpathSync тепер використовує оновлену логіку libuv
- Відмова від підтримки WinXP і Vista
- Поліпшили продуктивність, надійність і безпеку системи

Установка Node

http://nodejs.org/#download

На березень 2018:
Downloads: Latest LTS Version: 8.9.4 (includes npm 5.6.0).

Після завантаження натисніть INSTALL.

Після завершення установки ви отримаєте можливість запускати Node і npm з командного рядка Windows (режим REPL – Read-Eval-Print-Loop).

Відкрийте термінал (cmd.exe), наберіть "node" і натисніть Enter.

Робота в режимі терміналу показана на Рис.50.

Рис.50. Робота з Node.js в режимі терміналу

Багато прикладів для Node запускають через термінал.

Можна обійтися стандартним терміналом (cmd.exe).

Можна використати Far Manager 3.0.

Додаткові переваги можна отримати, використовуючи ConEmu (http://bit.ly/Con-Emu/) або Console2 (http://bit.ly/Console_2/).

Запуск скриптів

Приклад 1. Файл hello.js : console.log(“Hello, World!”);
Запускаємо: node hello.js

Приклад 2. HTTP-сервер. Файл server.js

```
var http = require("http");
http.createServer(function(request, response) {
   response.writeHead(200, {"Content-Type": "text/plain"});
   response.write("Hello World");
   response.end();
 }).listen(8888);
```

Запускаємо сервер: node server.js
Запускаємо браузер, http://localhost:8888/
Отримаємо : Hello World

Реалізація потоків даних

Платформа Node підходить для програмування потоків даних та організації їх передачі. Завдяки передачі даних фрагмент за фрагментом розробник отримує можливість обробляти дані по мірі їх накопичення, а не чекати, поки будуть передані всі дані, а потім виконувати які-небудь дії.

```
var stream = fs.createReadStream('./resource.json')
stream.on('data', function (chunk) {
console.log(chunk)
})
stream.on('end', function () {
console.log('finished')
})
```

Подія data викликається після появи нового фрагмента даних, а подія end - після завантаження всіх фрагментів коду.

Записувані потоки даних

У Node також підтримуються записувані потоки даних, що дозволяють записувати фрагменти даних.

Один з подібних потоків - об'єкт відповіді (res), генерований у відповідь на запит до HTTP-сервера.

```
var http = require('http');
var fs = require('fs');
http.createServer(function (req, res) {
res.writeHead(200, {'Content-Type': 'image/png'});
     // Передача по каналах з зчитуваного потоку в записуваний
fs.createReadStream('./image.png').pipe(res);
}).listen(3000);
console.log('Server running at http://localhost:3000/');
```

Розробка власних модулів

Модуль logger.js:

```
exports.info = function ( msg ) {
    console.log( new Date() + ': ' + msg);
}
exports.error = function ( msg) {
    console.error( msg);
}
```

Головний файл app.js:

```
var  logger = require( './logger');
logger.info( 'Hello world!');
```

Запуск скрипта:

```
node  app.js
```

Модуль Point.js

```
function Point ( x, y) {
    this.x = x;
    this.y = y;
}
Point.prototype.print = function () {
    console.log( this.x + ', ' + this.y);
}
module.exports = Point;
```

Головний файл app.js

```
var Point = require( './Point');
var  pt = new Point( 50, 60);
pt.print();
```

Читання файлу в асинхронному режимі

```
var http = require('http'); // Файл server_readFile.js
var fs = require('fs');
http.createServer(function(req, res){
    fs.readFile('hello.js', 'utf8', function(err, data){
        res.writeHead(200,                {'Content-Type':
'text/plain'});
        if(err)
    res.write('Could not find or open file for reading\n');
        else
            res.write(data);
        res.end();
    });
}).listen(8124,  function(){  console.log('bound  to  port
8124');});
console.log('Server running on 8124/');
```

Маршрутизація (routing)

```
var http = require('http');
http.createServer(function(req,res){
var      path      =      req.url.replace(/\/?(?:\?.*)?$/,
'').toLowerCase();
switch(path) {
    case '':
        res.writeHead(200, { 'Content-Type': 'text/plain'
});
```

```
                res.end('Homepage');
                break;
          case '/about':
                res.writeHead(200, { 'Content-Type': 'text/plain'
});
                res.end('About');
                break;
     default:
          res.writeHead(404, { 'Content-Type': 'text/plain' });
          res.end('Not Found');
          break;
     }
     }).listen(3000);
     console.log('Server started on localhost:3000; press Ctrl-C to
terminate....');
```

Спробуйте вказати URL:
http://localhost:3000
http://localhost:3000/about
http://localhost:3000/foo

Відображення статичних ресурсів

Нехай в теці public знаходяться файли home.html, about.html, notfound.html та в теці public/img файл logo.jpg. Розглянемо, як сервер Node може їх відобразити.

```
     // Файл server_StaticResources.js
     var http = require('http'),
     fs = require('fs');
     function serveStaticFile(res, path, contentType,
responseCode) {
          if(!responseCode) responseCode = 200;
          fs.readFile(__dirname + path, function(err,data) {
                if(err) {
                     res.writeHead(500, { 'Content-Type':
'text/plain' });
                     res.end('500 - Internal Error');
                } else {
     res.writeHead(responseCode,
                          { 'Content-Type': contentType });
                     res.end(data);
                }
          });
     }
     http.createServer(function(req,res){
                var path = req.url.replace(/\/?(?:\?.*)?$/, '')
                     .toLowerCase();
          switch(path) {
                case '':
                     serveStaticFile(res, '/public/home.html',
                'text/html');
                     break;
     case '/about':
```

```
                    serveStaticFile(res, '/public/about.html',
                    'text/html');
                    break;
               case '/img/logo.jpg':
                    serveStaticFile(res, '/public/img/logo.jpg',
                         'image/jpeg');
                    break;
               default:
                    serveStaticFile(res, '/public/404.html',
'text/html',                         404);
                    break;
          }
     }).listen(3000);
     console.log('Server started on localhost:3000; press Ctrl-C
to terminate....');
```

Використання менеджера пакетів NPN

Приклад. Відправка SMS через сервіс Twilio

```
npn  install  twilio
```

Файл app.js:

```
     var  twilio = require('twilio');
     var  client = twilio();
     client.sendMessage({
          to: '+37……..',
          from: '+37…..',
          body: 'Hello world'
     });
```

Запуск:

```
node  app.js
```

Примітка. Треба мати account на www.twilio.com

Файл package.json

- Є невід'ємною частиною NPM екосистеми і має особливе значення для NPM.
- Дуже важливо, щоб він був правильно налаштований, коли ви хочете поділитися своїм модулем зі світом, або якщо ви споживаєте модулі від інших людей.
- Щоб створити файл package.json в поточній папці, просто запустіть код: npm init
- Буде задано декілька питань (ім'я модуля, версія, автор, . . .)

Ви отримаєте файл package.json на зразок такого:

```
{
     "name": "ExpressCourseDemo",
     "version": "0.0.0",
     "description": "This demo app for the Express course",
     "author":                "Carlos                Souza
<carloshrosouza@gmail.com.com>",
     "dependencies": {
          "body-parser": "^1.8.1",
          "debug": "^0.8.1",
```

```
            "express": "^4.9.5",
            "lodash": "^2.4.1"
            },
        "scripts": {    "start": "node app.js"  }
}
```

Saving Dependencies

Коли ви встановлюєте якийсь модуль, тобто запускаєте npm install, у вас є додатковий прапорець командного рядка (--save), що говорить NPM записати в файл package.json інформацію про те, що ви встановили, як показано в лістингу:

```
npm install underscore --save
```

В файлі package.json з'явиться запис

```
"dependencies": {
    "underscore": "^1.6.0"
  }
```

Refresh the node_modules Folder

Усі додаткові модулі, які ви додаєте до вашого застосунку з допомогою npm, попадають в каталог node_modules.

Щоб оновити папку node_modules з вашого файлу package.json, ви можете запустити таку команду:

```
npm install
```

Висновок. Ще одна перевага використання package.json є те, що ви можете не зберігати папку node_modules, так як ви завжди можете отримати її копію з npmjs.org простою командою npm install

Створення нового Node-проекту

- Створити каталог проекту.
- Перейти в нього.
- Запустити npm init для створення файлу package.json (файл маніфесту проекта).
- За допомогою npm підключити до проекту необхідні модулі.
- Створити головний JavaScript файл app.js (та, можливо, інші файли).
- Виконати проект:
  node app.js

Вправа 1.

- Створити каталоги NPM1, NPM2 та зайти в NPM1
- npm install underscore
- Створити файл app.js
  ```
  var arreg = require('underscore');
  arreg.each(
  ```

```
[1,2,3,4,5], function(num){
     console.log("Number : "+num);
}
);
```

- Запустити застосунок: node app.js.
- Зайти в каталог NPM2.
- npm init (для створення файлу package.json).
- На запит name та description ввести npm2, на інші запити натискати Enter.
- На останній запит (Is this ok?) відповісти yes.
- Переглянути файл package.json.
- npm install underscore –save
- Переглянути файл package.json, впевнившись, що в ньому з'явилось підключення бібліотеки underscore.
- Скопіюємо файл app.js з NPM1 в NPM2.
- Відкриємо в редакторі файл package.json та додамо в секцію "dependencies" рядок "express": "^4.0".
- Подамо команду npm install (в проект буде доданий фреймворк express.js).
- npm install body-parser –save
- Впевнюємося, що до проекту підключився модуль body-parser, призначений для розбору запиту POST.

Вправа 2

- Створіть каталог ROUTE_EXPRESS, а в ньому – каталог BASIC і війдіть в нього.
- npm init .
- На запит name ввести server, на запит entry point ввести server.js, на інші запити натискати Enter а в кінці – y.
- npm install express –save
- Створіть каталог public в каталозі BASIC.
- Створіть в каталозі BASIC файл server.js з наступним вмістом:

```
var express = require('express');
     var app = express();
     app.get(
                '/', function(req,res){
               res.sendFile(__dirname                    +
'/public/index.html');
                }
          )
     app.listen(9000);
     console.log("Server initialised…");
```

- Створіть в каталозі public файл index.html з наступним вмістом:
<h1> This page is testing EXPRESS</h1>
- Повертаємось в каталог BASIC та запускаємо сервер: node server.js

- В браузері набираємо адресу localhost:9000/ (повинна активізуватися сторінка index.html).
- Створюємо в каталозі public файл contacts.html з наступним вмістом:
  <h1> Contacts in Node.js</h1>
- Редагуємо файл server.js, додавши після app.get(. . .) крапку з комою (;) та додавши наступне:

```
app.get(
                  '/contacts', function(req,res){
                 res.sendFile(__dirname +
'/public/contacts.html');
                  }
          )
```

- Запускаємо сервер: node server.js
- В браузері набираємо адресу localhost:9000/contacts (повинна активізуватися сторінка contacts.html).
- В каталозі ROUTE_EXPRESS створюємо каталог FULL та переходимо в нього.
- npm init
- На запит name ввести full, на інші запити натискати Enter а в кінці – y (буде сформовано файл package.json).
- npm install express --save
- npm install body-parser --save
- Створіть каталог public в каталозі FULL.
- Створюємо в каталозі public файл index.html з наступним вмістом:

```
<form name="form1" action="/login" method="post">
<label>User:</label><input type="text" name="user"
id="user">
<br>
 <label>Password:</label><input type="password"
name="password" id="password">
<br> ><input type="submit">
</form>
```

- Створіть в каталозі FULL файл server.js з наступним вмістом:

```
var express = require('express');
var bodyParser= require('body-parser');
var app = express();
app.use(bodyParser.urlencoded({extended : false}));
app.get(
                  '/', function(req,res){
                 res.sendFile(__dirname                          +
'/public/index.html');
                  }
          )
app.post(
                  '/login', function(req,res){
                 var user_name=req.body.user;
```

```
                    var password=req.body.password;
                    console.log("The     user's     name     is:
"+user_name+" And password is: "+password);
                    res.end("ending");
                     }
              )
        app.listen(9000);
        console.log("Server initialised…");
```

- Запускаємо сервер: node server.js (з каталогу FULL).
- В браузері набираємо адресу localhost:9000/ (повинна активізуватися сторінка index.html з формою).
- Вводимо логін, пароль і натискаємо Submit.

В браузері з'явиться "ending", а, повернувшись до консолі, бачимо інформацію про введений логін і пароль.

Автоматичний рестарт

При внесенні змін в ваш проект часто повторюються команди зупинки сервера та його повторного запуску. Це можна автоматизувати за допомогою пакета nodemon. Інсталюйте його глобально:

```
npm install -g nodemon
```

Застосовуйте nodemon для запуску вашого сервера:

```
nodemon server.js
```

Тема 11. Огляд фреймворків для Node. Основи Express.js

Фреймворки призначені для створення каркасу застосунку, прискорення розробки. Для Node існують декілька фреймворків (Express.js, Hapi.js, Koa.js, Meteor.js, Total.js, Compound.js, Geddy.js, Sails.js, Adonis.js), самим популярним з яких є Express.js.

Основи Express

Що таке Express.js?

- Express.js – це веб-фреймворк на базі модуля HTTP ядра Node.js (і компонентів фреймворку Connect). Express – був надбудовою над Connect.
- Компоненти Connect називають middleware.
- Вони є наріжним каменем філософії фреймворку, який є "конфігурація важливіше конвенції" (configuration over convention).
- Іншими словами, розробники можуть вільно вибрати ті бібліотеки , що ім потрібні для конкретного проекту ("нічого зайвого").

- Такий підхід забезпечує гнучкість і здатність тонко налаштовувати свої проекти (компонентний підхід замість монолітного).
- Express 4.0 (2014) вже не залежить від Connect.

Express.js бере на себе:

- Розбір тіла запиту HTTP.
- Розбір cookies.
- Управління сеансами.
- Організація маршрутизації запитів (зв'язки URL з методами HTTP).
- Визначення правильних заголовків відповіді, засноване на типах даних.
- Обробка помилок.
- Витяг параметрів URL.

Express.js :

- Використовує структури Node.js.
- Використовує шаблонізатори Jade/Pug, EJS , Mustache, Handlebars.
- Доступ до шаблону MVC
  - Model - - the Data
  - View - - the Template
  - Controller - - the JavaScript
- Створює (генерує) структуру каталогів проекту завдяки Scaffolding.

Інсталяція Express.js

Інсталяція останньої стабільної версії:

```
npm install express
```

Використовуйте @ для інсталяції специфічної версії:

```
npm install express@3.3.5
```

Формується каталог
node_modules / express

Інсталяція глобально: `npm install -g express`
Інсталяція в даний проект: `npm install --save express`
(якщо є готовий файл package.json)

"Hello World" на Express

Файл app.js:

```
var express = require("express");
var app = express();
app.get("/", function(request, response) {
        response.send("Hello, world!");
});
app.listen(3000, function() {
```

```
   console.log("Express app started on port 3000.");
});
```
Запускаємо з консолі: `node app.js`
Запускаємо браузер: http://localhost:3000/
Отримаємо: Hello, world!

Що таке cURL?

- Це утиліта командного рядка, що дозволяю взаємодіяти з різними видами серверів за різними протоколами з синтаксисом URL.
- cURL надає можливість гнучкого формування запиту із заданням таких параметрів, як cookie, user_agent, referrer і будь-яких інших заголовків.
- За допомогою cURL можна, наприклад, отримати вміст html-сторінки з командного рядка, не використовуючи для цього браузер:
  - curl http://localhost:3000/
  - Отримаємо (для попереднього прикладу): “Hello, world!”
- Сайт: http://curl.haxx.se/download.html

Відповідь у вигляді JSON

Функція send () конвертує Objects та Arrays в JSON.
```
Сервер (фрагмент):
app.get("/blocks", function(request, response) {
     var blocks = ['Fixed','Movable','Rotating'];
        response.send(blocks);
});
```
Запускаємо з консолі: `node app.js`
```
З іншої консолі:
$ curl –i http://localhost:3000/blocks
Отримаємо:
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8

["Fixed","Movable","Rotating"]
```
Ми тут використали прапорець `–i` щоб вивести заголовки відповіді.

Відповідь у вигляді HTML

Функція send () передає рядки у вигляді HTML.
```
Сервер (фрагмент):
app.get("/blocks", function(request, response) {
     var blocks = '<ul><li>Fixed</li><li>Movable</li></ul>';
        response.send(blocks);
});
```
Запускаємо з консолі: `node app.js`
```
З іншої консолі:
```

```
$ curl -i http://localhost:3000/blocks
Отримаємо:
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/html; charset=utf-8

<ul><li>Fixed</li><li>Movable</li></ul>
```

Перенаправлення на відносний шлях

```
Сервер (фрагмент):
app.get("/blocks", function(request, response) {
        response.redirect('/parts');
});
```

Запускаємо з консолі: `node app.js`

```
З іншої консолі:
$ curl -i http://localhost:3000/blocks
Отримаємо:
HTTP/1.1 302 Moved Temporarity
X-Powered-By: Express
Location: /parts
Content-Type: text/plain; charset=utf-8

Moved Temporarity. Redirecting to /parts
```

Приклад. Відсилання файлів методом sendFile().

Створимо два файли і директорію як показано на Рис.51.

Рис.51. Фрагмент файлової системи

//Файл index.html

```
<DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Building Blocks</title>
</head>
<body>
    <h1>Blocks</h1>
</body>
</html>
```

//Файл app.js

```
var express = require('express');
var app = express();

app.get('/', function(request, response){
    response.sendFile(__dirname + '/public/index.html');
```

```
});
app.listen(3000);
```

Запускаємо з консолі: `node app.js`
Запускаємо браузер: `http://localhost:3000/`
Отримаємо результат, показаний на Рис.52.

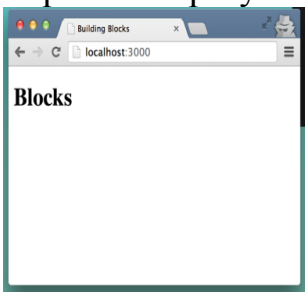


Рис.52. Результат запуску додатку

Middleware (програмне забезпечення проміжної ланки)

Зазвичай, терміном middleware позначали програмне забезпечення (на боці сервера), що об'єднує разом шар бізнес-логіки з сервером бази даних.

В нашому випадку для Node.js під терміном middleware слід розуміти колекцію модулів, які дозволяють інтегрувати в наш застосунок якусь функціональність на боці сервера або клієнта.

У випадку з Express до цієї категорії відноситься частина ланцюжка додатків, кожен з яких виконує певну функцію, пов'язану із запитом HTTP, - обробляє його чи обробляє запит для наступних проміжних додатків. Список програмного забезпечення проміжної ланки, що працює з Express, досить великий (http://expressjs.com/en/resources/middleware.html).

Стек обробки запитів в «чистому» Node.js та Express показаний на Рис.53 та Рис.54

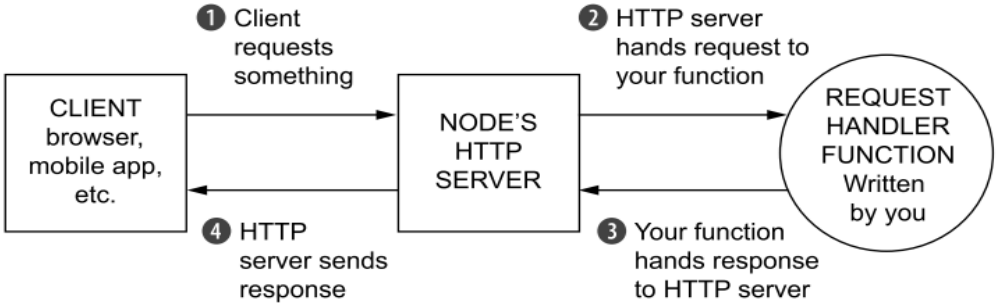


Рис.53. Потік обробки запитів в Node.js

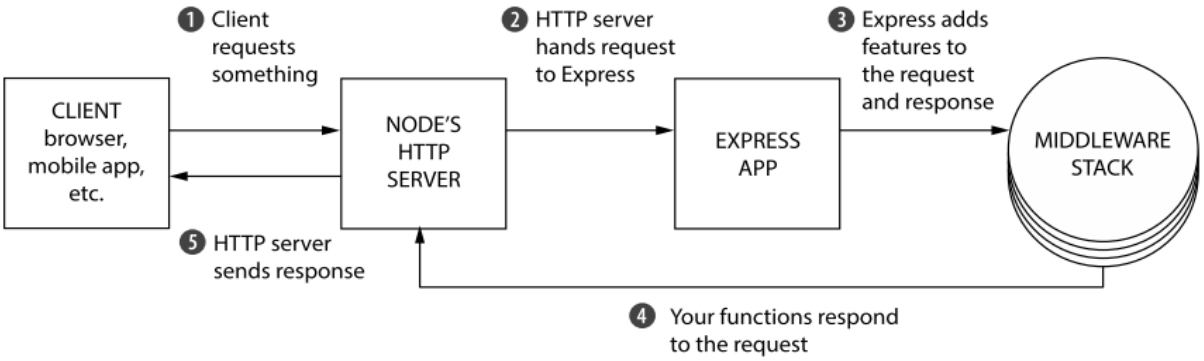


Рис.54. Потік обробки запитів в Express

Монтування middleware

Метод app.use додає middleware до стека застосунку.

Приклад. Монтування static middleware.

Для надання статичних файлів, наприклад, зображень, файлів CSS і JavaScript в Express використовується middleware express.static. Для того щоб почати безпосереднє надання файлів, необхідно передати ім'я каталогу, в якому знаходяться статичні ресурси, в middleware express.static.

Модифікуємо трохи наш попередній приклад.

```
//Файл app.js
var express = require('express');
var app = express();

app.use(express.static('public'));

app.listen(3000);
```

Тепер запустимо застосунок і побачимо, що результат буде точно таким, як на Рис.50.

Зверніть увагу, що static middleware обслуговую все, що знаходиться в папці public. Наприклад, якщо в папку public додати файл blocks.jpg і в файл index.html після 8-го рядка додати рядок <img src='blocks.jpg'> і перезапустити застосунок, то ми матимемо результат, показаний на Рис.55.

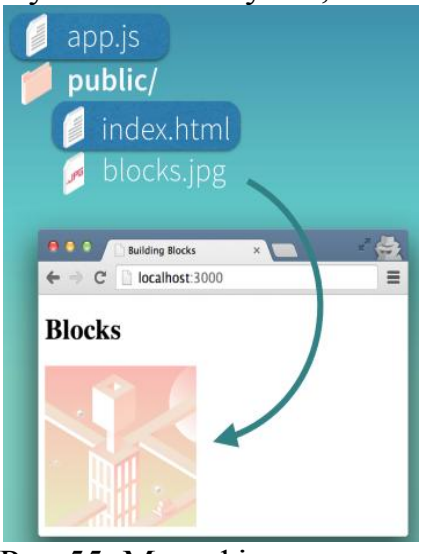


Рис.55. Модифікована сторінка прикладу

Маршрутизація в Express

Маршрутизація визначає, як додаток відповідає на клієнтський запит до конкретної адреси, і певного методу HTTP запиту.

Кожен маршрут може мати кілька обробників, які виконуються при співставленні маршруту.

Для визначення маршруту слід використати наступну структуру:

```
app.METHOD(path, callback) , де
```

app – екземпляр Express;
METHOD – метод HTTP запиту (GET, POST і т.д.);
path – шлях на сервері;
callback – функція-обробник на указаний шлях.

Генератор структури застосунків Express

Можна використовувати інструмент express-generator, для швидкого створення каркасу застосунку.

Встановлюється express-generator наступною командою:

```
npm install express-generator -g
```

З параметром -h можна проглянути доступні опції:

```
express -h
```

В наступному прикладі створюється каркас застосунку Express з іменем myapp в поточній директорії:

```
express –view=pug myapp
```

Після чого треба встановити залежності:

```
cd myapp
npm install
```

На Windows запустіть застосунок такою командою:

```
set DEBUG=myapp:* & npm start
```

Згенерований застосунок має наступну структуру директорій (Рис.56):

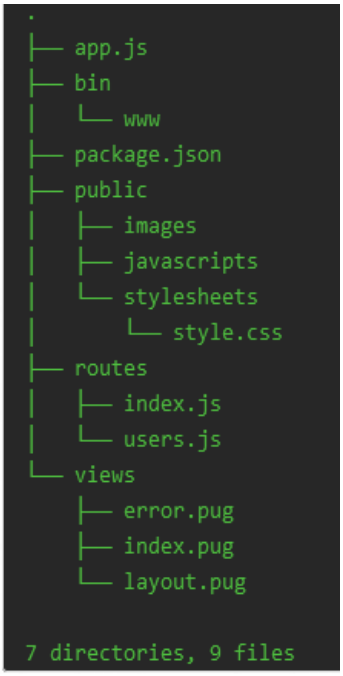


Рис.56. Згенерована структура каталогів застосунку

Модуль body-parser

Модуль, ймовірно, є найбільш важливим з усіх third-party middleware модулів. Він дозволяє розробникам обробляти вхідні дані, такі як вміст тіла та заголовки HTTP-запиту, транслюючі їх в сукупність об'єктів JavaScript/Node.js.

Інсталяція модуля:

```
npm install body-parser@1.8.1
```

Головні сервіси модуля body-parser:

- json() : Processes JSON data; e.g., {"name": "value", "name2": "value"};
- urlencoded() : Processes URL-encoded data; e.g., name=value&name2=value2;
- raw() : Returns body as a buffer type;
- text() : Returns body as string type.

Движки шаблонів (Template engines)

Це бібліотеки, які дозволяють нам використовувати різні мови шаблонів (EJS, Handlebars, Pug/Jade, Mustache) для представлення даних в Express.js.

Мови шаблонів мають спеціальні інструкції для движка, що показують як обробляти дані.

Процес об'єднання даних з шаблонами називається rendering (візуалізація)

Використання Jade/Pug

Jade перейменували в Pug через порушення торгового знаку. Систаксис трохи відрізняється.

Сайти: http://jade-lang.com/ та https://pugjs.org/

Інсталяція:
```
npm install jade --save
npm install pug --save
```

Монтувати движок шаблонів до застосунку необхідно в методі app.set, вказавши директорію та ім'я рушія, наприклад:

```
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'pug');
```

При обробці маршруту на сторінку index.pug можна передати параметри, наприклад:

```
res.render('index', {title: 'Express'});
```

Синтаксис Jade/Pug покажемо на прикладі.

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Pug</title>
</head>
<body>
<h1>Pug Examples</h1>
<div class="container">
<p>Cool Pug example!</p>
</div>
</body>
</html>
```

Цей файл на Pug буде виглядати лаконічніше:

```
doctype html
html(lang='en')
  head
     title Pug
  body
     h1 Pug Examples
    .container
        p Cool Pug example!
```

Існує сайт (http://html2jade.org/) для конвертації html в jade.

Тема 12. Мова Python в web-програмуванні. Фреймворк Django.

Мова програмування Python

Python – інтерпретована, мультипарадигменна мова програмування високого рівня з динамічною типізацією, автоматичним управлінням

пам'яттю і зручними високорівневими структурами даних. Розроблена в 1991 році Гвідо ван Россумом. Підтримує обробку винятків, паралельні обчислення. Підтримує декілька парадигм програмування, зокрема: об'єктно-орієнтовану, процедурну, функціональну та аспектно-орієнтовану.

Імперативна - описує процес обчислення у вигляді інструкцій, що змінюють стан програми. Об'єктно-орієнтована - основною концепцією є поняття об'єкта, що ототожнюється з об'єктом предметної області. Функціональна - в ній процес обчислення трактується як обчислення значень функцій в математичному розумінні (спосіб вирішення задачі описується у вигляді залежності функцій одна від одної).

Динамічна типізація: немає попереднього оголошення типів – тип змінної виводиться в процесі виконання.

Застосування Python

- Інтерактивна консоль - потужний «калькулятор» (робота з числами, матрицями, файлами, зображеннями, статистичного аналізу та ін.).
- Мова програмування для невеликих скриптів (обробка зображень, створення резервних копій).
- Мова програмування для прототипування (швидке створення шаблону програми з UI, швидка перевірка роботи алгоритму).
- Мова програмування для повноцінних програм (Gajim, BitTorrent, Dropbox, EVE Online).
- Мова програмування для web-застосунків (фреймворки Django, Flask, TurboGears, web2py, Pylons, Zope, WebWare підтримують швидке створення повнофункціональних і якісних сайтів на основі Python).
- Вбудована в додатки мова програмування (вбудована Python-консоль в якій можна оперувати з об'єктами додатків мовою Python: 3D моделювання - Blender, Maya, обробка зображень - GIMP, робота з ГІС даними - ESRI ArcGIS, математичні пакети - Sage, IPython Notebook, офісний пакет OpenOffice).

Транслятор — установка

Завантажити транслятор для будь-якої платформи можна на сайті: http://www.python.org . Актуальна версія – Python 3.6.5 (квітень 2018).

Python IDE (Integrated Development Environment)

Найбільш популярні:
- IDLE - поставляється з компілятором, середовище розробки;
- Eclipse + PyDev (http://www.pydev.org/);
- PyCharm from JetBrains (https://www.jetbrains.com/pycharm/);
- Sublime Text (https://www.sublimetext.com/);
- Wingware’s Wing IDE 101 (http://wingware.com/downloads/wingide-101).

Синтаксис Python

Відступи в коді

- Функції в Pyton не мають явних операторів begin/end або фігурних дужок для позначення початку й кінця коду функції.
- Єдиним розділювачем є двокрапка (:) та відступи в коді.
- Блоки коду описуються їх відступами.
- “Блок коду“ – це функції, умовні оператори, тіла циклів і так далі.
- Відступ позначає початок блоку, а прибирання відступу - його закінчення.
- Явні дужки чи ключові слова не потрібні.

Стандарт PEP 8

PEP (python enhanced proposal - заявки на поліпшення мови python). Цей стандарт – посібник з написання коду на Python.

- Використовуйте 4 пробіли на кожен рівень відступу
- Python 3 забороняє змішування табуляції і пробілів в відступах
- Обмежте довжину рядка максимум 79 символами
- Відокремлюйте функції верхнього рівня і визначення класів двома порожніми рядками
- Кодування Python 3 повинно бути UTF-8 (ASCII в Python 2)
- Слід уникати використання пробілів в наступних ситуаціях:
  - Усередині круглих, квадратних і фігурних дужок animals(tiger[2], {eagle: 6})
  - Перед комою, крапкою з комою і двокрапкою if x == y:
  - Відразу перед відкриваючою дужкою, після якої слідує індекс - dict['key'] = list[index]
- Завжди оточуйте наступні бінарні оператори одним пропуском з кожного боку: оператори присвоювання (=, + =, - = і інші), оператори порівняння (==, <,>,! =, <>, <=,> =, In, not in, is, is not), логічні оператори (and, or, not)
- Не використовуйте пробіли навколо знаку =, якщо він використовується для позначення іменованого аргументу або повернуться до стандартних значень.

Приклад. Демонстрація операторів цикла for та while.

Файл for_while.py

```
lastName="Smith"
count=0
for letter in lastName:
    print(letter," ",count)
    count+=1
print("-------")
count=0
while (count<5):
    print(lastName[count]," ",count)
```

```
    count+=1
```

Введення даних

```
>> age = input("Please, enter your age :  ")
Please, enter your age :  24
>> print( type(age))
<class 'str'>
>> print ("You age is   ", age)
You age is  24
```

Файл sum_int.py

```
     n1 = int(input("Enter n1 : "))
     n2 = int(input("Enter n2 : "))
     sum = n1 + n2
     print("n1 + n2 = ", sum)
```

Замість int можна використати float.

Рядки в Python

```
>>> S = 'Spam'
>>> len(S)
4
>>> S[0]
'S'
>>> S[-1]
'm'
>>> S[-2]
'a'
>>> S[1:3]    # Slice - "зріз"
'pa'
>>> S + 'xyz'    # Concatenation
'Spamxyz'
>>> S              # S is unchanged
'Spam'
>>> S * 8         # Repetition
'SpamSpamSpamSpamSpamSpamSpamSpam'
>>> S = 'Spam'
>>> S.find('pa')
1
>>> S.replace('pa', 'XYZ')
'SXYZm'
>>> S
'Spam'
>>> S = 'shrubbery'
>>> L = list(S)              # Expand to a list: [...]
>>> L
['s', 'h', 'r', 'u', 'b', 'b', 'e', 'r', 'y']
>>> L[1] = 'c'                # Change it in place
>>> ''.join(L)                # Join with empty delimiter
'scrubbery'
>>> x = 3.4
>>> str(x)
'3.4'
>>> format(x, "0.5f")
'3.40000'
>>> smiles = "C(=N)(N)N.C(=O)(O)O"
>>> smiles.split(".")
['C(=N)(N)N', 'C(=O)(O)O']
```

```
if "Br" in "Brother":
     print ("contains brother")
email_address = "clin"
if "@" not in email_address:
     email_address += "@brandeis.edu"
```

Приклад: Чи є рядок паліндромом?

```
def isPalindrome(s):
    if len(s) <= 1:
        return True
    return s[0] == s[len(s) - 1] and isPalindrome(s[1 : len(s) -
1])

def main():
    line = input("Enter a string:  ")
    if isPalindrome(line):
        print("That was a palindrome !")
    else:
        print("That is not a palindrome !")
main()
```

Кортежи (tuple)

Кортеж в Python - це упорядкований набір об'єктів, в який можуть одночасно входити об'єкти різних типів. Кортеж задається перерахуванням його елементів у круглих дужках через кому, наприклад,

t = (12 , 'b' , 34.6 , 'derevo')

Елементам кортежу можна відразу зіставити які-небудь змінні:

t = (x , s1 , y , s2) =(12 , 'b' , 34.6 , 'derevo')

Основні операції з кортежами: len(t) , t1 + t2 , t ∗ n , t[i] , t[i : j :k] , min(t) , max(t).

>>> s= 'amamam '

>>> t = tuple(s)

t → (' a ' , 'm' , ' a ' , 'm' , ' a ' , 'm')

Списки (Lists)

Список в Python - це упорядкований набір об'єктів, в список можуть одночасно входити об'єкти різних типів.

lst = [12 , 'b' , 34.6 , 'derevo']

Основні операції зі списками: len(lst) , lst1 + lst2 , lst ∗ n , lst[i] , lst[i : j :k] , min(lst) , max(lst) , lst [i]=x , del lst [i] , lst [i : j]=x.

Основні методи списків:

Додавання елементу lst .append(x) .

Додавання кортежу або списку lst .extend(t).

lst .count(x) , lst .index(x) , lst .remove(x) , lst .pop(i) ,

lst .insert(i ,x) , lst.sort() , lst.reverse().

Набори (sets)

Набір використовується для зберігання невпорядкованого набору об'єктів. Щоб створити набір, використовуйте функцію set(), або фігурні дужки { }

>>> s = set([3, 9, 2, 15])

```
>>> t = set ( “Hello”)
>>> t
{ ‘e’, ‘l’, ‘H’, ‘o’}
>>> {1, 2, 3, 4, 5}
```

Операції над наборами

```
a = t | s      # Union of t and s
b = t & s    # Intersection of t and s
c = t – s     # Set difference (items in t, but not in s)
d = t ^ s     # Symmetric difference (items in t or s, but not both)
t.add('x')            # Add a single item
s.update([10,37,42])   # Adds multiple items to s
t.remove('H')
```

Словники (Dictionaries)

Словники в Python - невпорядковані колекції довільних об'єктів з доступом по ключу. Їх іноді ще називають асоціативними масивами або хеш-таблицями.

```
>>> d = {}
>>> d
{}
>>> d = {'dict': 1, 'dictionary': 2}
>>> d
 {'dict': 1, 'dictionary': 2}
>>> d = dict(short='dict', long='dictionary')
>>> d
{'short': 'dict', 'long': 'dictionary'}
>>> d = dict([(1, 1), (2, 4)])
>>> d
{1: 1, 2: 4}
>>> D = { ‘a’: 1, ‘b’: 2, ‘c’: 3}
>>> D.keys()
        dict_keys([‘b’, ‘a’, ‘c’])
>>> D.values()
        dict_values([2, 1, 3])
>>> D.items()
        dict_items([(‘b’,2), (‘a’,1), (‘c’,3)])
>> D.copy()
{‘a’: 1, ‘b’: 2, ‘c’: 3}
D.clear()  - очистка словника
D.update(D2)
D.get(K)
D.pop(K)
```

Файли

```
f = open('data.dat')   #  f is a file object
for line in f:                #  Read each line as text
```

```
        print(line.strip())      #  Remove  trailing  newline  character
f.close()                        #  Close  the  file
input_file = open(“in.txt")
output_file = open(“out.txt", "w")
for line in input_file:
                output_file.write(line)
```

Функції

Функцію можна визначити двома способами :

- за допомогою ключового слова def
- lambda-виразом

```
def  remainder (a, b):
        q = a // b       # // is truncating division.
        r = a - q*b
        return r
```

Виклик функції : `result = remainder(37,15)`

Змінні, визначені всередині функції, є локальними

```
count = 0
...
def foo():
        global count
        count += 1            # Changes the global variable count
```

Приклад

```
def multi_table(a):
        for i in range(1, 11):
                print('{0} x {1} = {2}'.format(a, i, a*i))
if __name__ == '__main__':
        a = input('Enter a number: ')
        multi_table(float(a))
```

Можна використовувати кортежі, щоб повернути декілька значень

```
def divide(a,b):
        q = a // b        # If a and b are integers, q is integer
        r = a - q*b
        return (q,r)
```

Виклик : `q, r = divide( 1456, 33)`

Можна задавати значення аргументів за замовчуванням

```
def connect(hostname, port, timeout=300):
        # Function body
```

Виклик : `connect('www.python.org', 80)`

або так - `connect(port=80,hostname="www.python.org")`

Модулі

Модулі виконують дві важливих функції:

- Повторне використання коду.
- Управління адресним простором імен.

Python дозволяє помістити класи, функції або дані в окремий файл і використовувати їх в інших програмах. Такий файл називається модулем.

Підключити модуль можна за допомогою інструкції import

```
>>> import os
>>> os.getcwd()
'C:\\Python33‘
```

```
>>> import math
>>> math.e
2.718281828459045
```

Використання псевдонімів

```
>>> import math as m
>>> m.e
2.718281828459045
```

Інструкція from

from <Назва модуля> import <Атрибут 1> [as <Псевдонім 1>], [<Атрибут 2> [as <Псевдонім 2>] ...]

from <Назва модуля> import *

```
>>> from math import e, ceil as c
>>> e
2.718281828459045
>>> c(4.6)
5
```

Об’єкти та класи

Всі значення, використовувані програмами, є об’єкти.

Об’єкт містить деякі внутрішні дані та має методи, що дозволяють виконувати різні операції над цими даними.

Функція dir () виводить список всіх методів об'єкта.

```
>>> items = [37, 42]
>>> dir(items)
[‘__add__’, ‘__class__’, ‘__contains__’, ‘__delattr__’, ‘__delitem__’,
        ...
‘append’, ‘count’, ‘extend’, ‘index’, ‘insert’, ‘pop’,
‘remove’, ‘reverse’, ‘sort’]
>>>
```

Для визначення нових типів об'єктів використовується інструкція class

Приклад. Клас стека

```
class Stack(object):
   def __init__(self):    # Ініціалізація стека
      self.stack = [ ]
   def push(self,object):
      self.stack.append(object)
   def pop(self):
      return self.stack.pop()
   def length(self):
      return len(self.stack)
```

Клас Stack успадковує властивості і методи класу object.

self – аналог this (посилання на екземпляр класу).

Методи, імена яких починаються і закінчуються двома символами підкреслення, є спеціальними методами.

```
s = Stack()
s.push("Dave")
s.push(42)
s.push([3,4,5])
x = s.pop()       # x отримає значення [3,4,5]
y = s.pop()       # y отримає значення 42
del s              # Знищує об'єкт s
```

Всі методи класу Stack можуть застосовуватися тільки до екземплярів даного класу (тобто до створених об'єктів).

Статичні методи

```
class EventHandler(object):
   @staticmethod
   def dispatcherThread():
      while (1):
         # Чекання запиту
...
EventHandler.dispatcherThread() # Виклик метода
```

В даному випадку @staticmethod оголошує наступний за ним метод статичним.

@staticmethod - це приклад використання декоратора

Приклад класу

```
class Car:
        def  __init__( self, make, model, year)
            self.make = make
            self.model = model
            self.year = year
            print("Instance object of the class is created")
        def  displayParameters(self) :
            print("Make  : ", self.make)
            print("Model  : ", self.model)
            print("Year  : ", self.year)
            print()

    car1 = Car("A", "B", 2015)
    car1.displayParameters()
```

Class variable

```
class Car:
        totalNumber = 0
        def  __init__( self, make, model, year)
            Car.totalNumber +=1
            self.make = make
            self.model = model
```

```
                    self.year = year
                    print("Total number of car is ", Car.totalNumber)
                def  displayParameters(self) :
                    print("Make  : ", self.make)
                    print("Model  : ", self.model)
                    print("Year  : ", self.year)
                    print()
        car1 = Car("A", "B", 2015)
        car2 = Car("C", "D", 2014)
```

Деструктор

```
        class Car:
                totalNumber = 0
                def  __init__( self, make, model, year) :
                    Car.totalNumber +=1
                    self.make = make
                    self.model = model
                    self.year = year
                def  __del__ (self) :
                    Car.totalNumber -=1
                def  displayParameters(self) :
                    print("Make  : ", self.make)
                    print("Model  : ", self.model)
                    print("Year  : ", self.year)
        car1 = Car("A", "B", 2015)
        car2 = Car("C", "D", 2014)
        del car2
```

Приватні поля класу та властивості

Ми можемо приховати атрибути класу (зробити їх приватними), якщо перед іменем атрибуту поставимо два підкреслення.

```
    class Gadget:          # Файл  Gadget.py
        """A class used for modelling Gadgets in a web shop."""
        __weight = 100
        __operating_system = None
        __battery_capacity = 2000
        __screen_size = 1
        def __init__(self, weight, operating_system,
battery_capacity, screen_size):
            self.__weight = weight
            self.__operating_system = operating_system
            self.__battery_capacity = battery_capacity
    self.__screen_size = screen_size
        def get_weight(self):
            return self.__weight
        def set_weight(self, weight):
            self.__weight = weight
    weight = property(get_weight, set_weight)
        @property
        def operating_system(self):
```

```
        return self.__operating_system
    @operating_system.setter
    def operating_system(self, new_os):
        self.__operating_system = new_os
```

Застосування

```
>>> from Gadget import Gadget
>>> my_iphone = Gadget(240,'iOS',1980,4)
>>> my_iphone.weight
240
>>> my_iphone.weight = 255
>>> my_iphone.weight
255
>>> my_iphone.operating_system
'iOS'
>>> my_iphone.operating_system = 'iOS 8.1'
>>> my_iphone.operating_system
'iOS 8.1'
```

Одиночне успадкування в Python

В Python успадкування здійснюється за допомогою наступного синтаксису:

class MySubClass(MyBaseClass).

```
class Animal:
    __name = None
    __age = 0
    __is_hungry = False
    __nr_of_legs = 0
    def __init__(self, name, age, is_hungry, nr_of_legs):
        self.name = name
        self.age = age
        self.is_hungry = is_hungry
        self.nr_of_legs = nr_of_legs
    def eat(self, food):
        print("{} is eating {}.".format(self.name, food))
    @property
    def name(self):
        return self.__name
    @name.setter
    def name(self,new_name):
        self.__name = new_name
    @property
    def age(self):
        return self.__age
    @age.setter
    def age(self,new_age):
        self.__age = new_age
    @property
    def is_hungry(self):
        return self.__is_hungry
    @is_hungry.setter
    def is_hungry(self,new_is_hungry):
        self.__is_hungry = new_is_hungry
    @property
    def nr_of_legs(self):
        return self.__nr_of_legs
```

```
        @nr_of_legs.setter
        def nr_of_legs(self,new_nr_of_legs):
            self.__nr_of_legs = new_nr_of_legs
    class Snake(Animal):
        __temperature = 28
        def __init__(self, name, temperature):
            super().__init__(name, 2, True, 0)
            self.temperature = temperature
        def eat(self, food):
            if food == "meat":
                super().eat(food)
            else:
                raise ValueError
        @property
        def temperature(self):
            return self.__temperature
        @temperature.setter
        def temperature(self,new_temperature):
            if new_temperature < 10 or new_temperature > 40:
                raise ValueError
            self.__temperature = new_temperature
```

Multiple Inheritance in Python

```
    class Phone(object):
        def __init__(self):
            print("Phone constructor invoked.")
        def call_number(self, phone_number):
            print("Calling number {}".format(phone_number))
    class Computer(object):
        def __init__(self):
            print("Computer constructor invoked.")
        def install_software(self, software):
            print("Computer installing the {}
software".format(software))
    class SmartPhone(Phone,Computer):
        def __init__(self):
            Phone.__init__(self)
            Computer.__init__(self)
    . . . . . .
    >>> my_iphone = SmartPhone()
    Phone constructor invoked.
    Computer constructor invoked.
    >>> my_iphone.call_number("123456789")
    Calling number 123456789
```

Обробка винятків

```
    try:
        Value1 = int(input("Type the first number: "))
        Value2 = int(input("Type the second number: "))
        Output = Value1 / Value2
    except  ValueError:
        print("You must type a whole number!")
    except  KeyboardInterrupt:
        print("You pressed Ctrl+C!")
```

```
except  ZeroDivisionError:
    print("Attempted to divide by zero!")
except  ArithmeticError:
    print("An undefined math error occurred.")
else:
    print(Output)
TryAgain = True
while TryAgain:
    try:
        Value = int(input("Type a whole number. "))
    except  ValueError:
        print("You must type a whole number!‟)
        try:
            DoOver = input("Try again (y/n)? ")
        except:
            print("OK, see you next time!")
            TryAgain = False
        else:
            if (str.upper(DoOver) == "N"):
                TryAgain = False
    except KeyboardInterrupt:
        print("You pressed Ctrl+C!")
        print("See you next time!")
        TryAgain = False
    else:
        print(Value)
        TryAgain = False
```

Генерація винятків

Приклад 1

```
try:
    raise ValueError
except ValueError:
    print("ValueError Exception!")
```

Приклад 2

```
try:
    Ex = ValueError()
    Ex.strerror = "Value must be within 1 and 10."
    raise Ex
except ValueError as e:
    print("ValueError Exception!", e.strerror)
```

Пакети

Пакети в мові Python забезпечують спосіб розподілу набора модулів по каталогам файлової системи та використання точкової нотації для доступу до модулів підпакетів.

Управління пакетами

Пакетами Python можуть бути інструменти, бібліотеки, фреймворки, застосунки. Існують десятки тисяч доступних пакетів, які можна використовувати для створення власних проектів.

Інструменти управління пакетами

Найбільш часто використовувані менеджери пакетів Python – це

- pip
- easy_install.

Дані інструменти допомагають виконати наступні завдання: скачування, установка, видалення пакетів; зборка пакетів; управління пакетами Python і багато іншого.

Основи web-фреймворку Django

Django (Джанго) – високорівневий відкритий Python-фреймворк для розробки веб-систем. Названий на честь джазмена Джанго Рейнхардта.

Архітектура Django подібна MVC. Рівневу архітектуру Django часто називають «Модель-Шаблон-Подання» (MTV).

Сайт https://www.djangoproject.com/

Має власний ORM ((Object-relational mapping)) для роботи з базами даних, вбудований інтерфейс адміністратора, диспетчер URL на основі регулярних виразів, шаблонізатор, бібліотеку для роботи з формами, інтернаціоналізацію.

Останній реліз - Django 2.0.3 (на квітень 2018). Розробники: Russell Keith-Magee, Adrian Holovaty, Simon Willison, Jacob Kaplan-Moss, Wilson Miner – перший реліз в 2003 році.

Веб-фреймворк Django використовується в таких великих і відомих сайтах, як Instagram, Bitbucket, Disqus, Mozilla Firefox, NASA, The Washington Times, Pinterest та ін. Також Django використовується в якості веб-компонента в різних проектах, таких як Graphite - система побудови графіків та моніторингу, FreeNAS - вільна реалізація системи зберігання та обміну файлами та ін. DjangoSites (https://djangosites.org/) - це сайт, який відстежує веб-сайти, створені за допомогою Django. Станом на березень 2018 р. налічується понад 5280 сайтів.

Як порівняти Django з іншими веб-фреймворками?

Станом на серпень 2017 року (https://devopedia.org/django), Django був шостим за популярністю веб-фреймворком (після ASP.NET, AngularJS, Ruby on Rails, ASP.NET MVC, React). Якщо розглядати тільки основані на Python фреймворки, то Django є на верхній позиції.

Серед основаних на Python фреймворків є наступні:

- Non-full-stack - Flask, Bottle, CherryPy, Pyramid.
- Full-stack - Django, Tornado, TurboGears, web2py.

Основні принципи Django: Django прагне бути “пітонічним”, DRY (Don’t Repeat Yourself), слабка залежність та гнучкість, швидка розробка.

Архітектура Django має наступний вигляд (Рис.57)

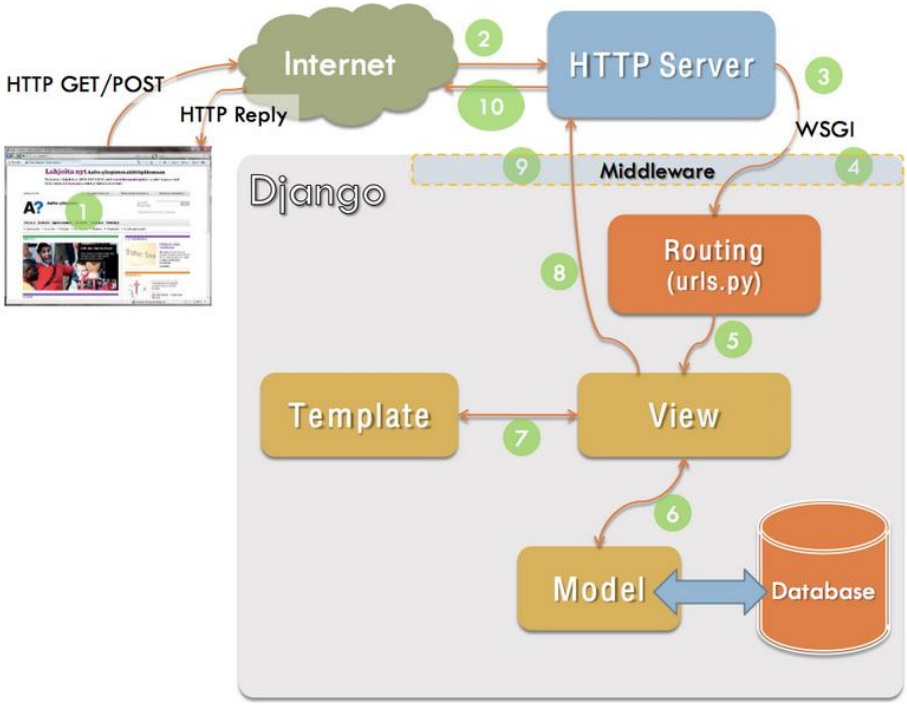


Рис.57. Архітектура Django.

Маємо наступні компоненти архітектури MTV:

Model – рівень доступу до даних. Це стосується доступу, перевірки та взаємовідносин даних. Моделі - це класи Python, які посереднюють між Django ORM та таблицями бази даних.

Template – презентаційний шар. Відповідає за те, як дані відображаються клієнту.

View – шар бізнес-логіки. Це міст між моделями та шаблонами. Перегляд (view) відкриває дані моделі та перенаправляє їх до шаблону для презентації. На відміну від визначення перегляду в традиційній архітектурі MVC, перегляд Django описує, які дані ви бачите, а не як ви бачите. Це стосується обробки, а не презентації.

Проект в Django

Проект в Django являє собою набір конфігураційних файлів і застосунків Django. Проект може містити множину застосунків Django.

Застосунок Django – набір файлів, що містять опис моделей та представлень, які є частиною одного пакета Python і являють собою повний застосунок. Застосунок може міститися у декількох проектах.

Створення та запуск Django проекта

Почати новий проект:

```
django-admin.py startproject example
```

Запустити локальний сервер:

```
python manage.py runserver
```

Мінімальна структура проекта Django

example/ - каталог проекта

example/ - пакет проекта

__init__.py

settings.py - налаштування проекта
urls.py - конфігурація URL адрес
wsgi.py - точка входу для wsgi
manage.py - командна утіліта Django

WSGI (Web Server Gateway Interface) - стандарт взаємодії між Python-програмою, що виконується на стороні сервера, і самим веб-сервером, наприклад Apache.

Middleware

Крім додатків і серверів, стандарт WSGI дає визначення middleware-компонентів (компонентів проміжного шару), що надають інтерфейси як додатку, так і серверу. Тобто для сервера middleware є додатком, а для додатка - сервером. Це дозволяє складати «ланцюжки» WSGI-сумісних middleware-компонентів.

Middleware можуть брати на себе такі функції (але не обмежуються цим): обробка сесій, аутентифікація / авторизація, управління URL (маршрутизація запитів), балансування навантаження, пост-обробка вихідних даних (наприклад, перевірка на валідність), зміна заголовків.

Створення Django застосунків

```
python manage.py startup blog
```

Після створення нового Django застосунку необхідно зареєструвати його в списку застосунків, що належать проекту.

Файл settings.py:

```
INSTALLED_APPS = (
    . . . . . .
    ‘example’,
    ‘blog’
)
```

Мінімальна структура Django застосунку

blog/ - пакет за стосунку;
__init__.py
admin.py - налаштування панелі адміна;
migrations/ - пакет, що містить міграції даних;
__init__.py
forms.py - опис форм введення інформації;
models.py - опис таблиць БД;
tests.py - unit tests;
views.py - бізнес-логіка за стосунку.

Функція подання (view)

Являє собою Python функцію, яка породжує вміст сторінки.

```
from django.http import HttpResponse
                        ← django.http.Request
def hello(request):
```

```
return HttpResponse(“Hello, world!”)
```

Об’єкти запиту і відповіді (HttpRequest, HttpResponse)

Клас HttpRequest являє собою HTTP-запит, отриманий з сторінки клієнта, і містить:

- Інформацію про URL (path і т.д.).
- HTTP-заголовки запиту.
- Інформацію про відправлені дані (GET, POST та FILES).
- Інформацію про сесії та cookies (session та COOKIES).
- Інші змінні сервера.

Клас HttpResponse слугує для конструювання відповіді на запит.

```
HttpResponse(content = “”, mimetype = None, status = 200, content_type = DEFAULT_CONTENT_TYPE)
```

Але якщо необхідно додавати вміст поступово, можна використовувати об’єкт response як об’єкт файлу:

```
response = HttpResponse()
response.write(“<p>Here’s my Web Page</p>”)
response.write(“<p>Here’s another paragrath</p>”)
```

Підкласи HttpResponse

Модуль django.http містить декілька підкласів HttpResponse, які представляють різні види HTTP-відповідей: HttpResponseRedirect (302), HttpResponseNotFound (404), HttpResponseForbidden (403), HttpResponseServerError (500) та інші.

Конфігурування URLs

Побудовано на принципі слабкої зв’язності.

Кожна конфігурація повинна зберігатися у вигляді файлу з програмним кодом Python, зазвичай з іменем urls.py.

Такі файли повинні визначати об’єкт urlpatters, який отримується в результаті виклику функції django.conf.patterns.

Шлях до головного конфігуратора визначається в settings.py змінною ROOT_URLCONF.

Приклад.

```
from django.conf.urls import patterns, include, url
     from example.views import hello

     urlpatterns = patterns(“,
          #Examples
          (r’^hello/’, hello)
          (r’^$’, ‘example.views.home’, name=‘home’)
          (r’^example/’, include(‘example.blogs.urls’))
     )
```

Система шаблонів Django

Шаблон в Django являє собою рядок тексту в спеціальному форматі, призначений для відділення подання документу (view) від його даних.

Мова шаблонів в Django не має ніякого відношення до Python, і створена для веб-технологів та дизайнерів інтерфейсів.

Ідеологічні основи шаблонів Django

- Бізнес-логіку треба відділити від логіки подання.
- Систаксис не треба прив'язувати до HTML/XML.
- Передбачається, що дизайнери впевнено володіють HTML.
- Не передбачається, що дизайнери вміють програмувати на Python.
- Не ставилось завдання винайти нову мову програмування.

Конфігурування та завантаження Django шаблонів

Django пропонує зручний та потужний API для завантаження шаблонів з файлової системи.

- Спершу в settings.py треба задати:
  TEMPLATE_DIRS = (
  "/шлях до/папки що/містить шаблони/"
)
- Потім можна використовувати:
  from django.shortcuts import render_to_response
  def hello(request):
  return render_to_response('hello.html')

Модель даних в Django

Логічний опис структури даних, що зберігаються в БД, надається мовою Python. Всі моделі являють собою клас, що спадкує django.db.models.Model

```
#app/models.py  (загальний синтаксис):
from django.db import models
class CustomModel(models.Model):
    name = models.CharField(max_length=30)
    body = models.TextField()
    email = models.EmailField(blank=True)
    number = models.IntegerField()
```

Міграції в Django

Їх використовують для переносу змін в моделях (додавання/видалення поля або моделі і т.д.) на структуру бази даних.

Міграції – це дуже потужний механізм, який треба дуже обережно застосовувати в реальних умовах.

```
manage.py makemigrations [<app_label>]
```