ПР1

Сервер и клиент. Сервер (программное обеспечение) - программный компонент вычислительной системы, выполняющий сервисные (обслуживающие) функции по запросу клиента, предоставляя ему доступ к определённым ресурсам или услугам

Сервер (аппаратное обеспечение) - выделенный или специализированный компьютер для выполнения сервисного программного обеспечения без непосредственного участия человека.

Клиент - это аппаратный или программный компонент вычислительной системы, посылающий запросы серверу.

База данных - это информационная модель, позволяющая упорядоченно хранить данные об объекте или группе объектов, обладающих набором свойств, которые можно категорировать. Базы данных функционируют под управлением систем управления базами данных (сокращенно СУБД).

Сервис - легко заменяемый компонент сервисноориентированной архитектуры со стандартизированными интерфейсами

API (в клиент-сервере) - описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой.

Архитектура «Клиент-Сервер» предусматривает разделение процессов предоставление услуг и отправки запросов на них на разных компьютерах в сети, каждый из которых выполняют свои задачи независимо от других.

Виды сервисов: Файл-серверы, Файрволы (брандмауэры), Серверы баз данных, Серверы приложений, Почтовые серверы

Вертикальная масштабируемость увеличение производительности компонентов серверной системы в интересах повышения производительности всей системы

Горизонтальная масштабируемость разбиение системы на более мелкие структурные компоненты и разнесение их, так и увеличение количества компонентов, параллельно выполняющих одну и ту же функцию

Паттерн MVC: Model-View-Presenter. — шаблон проектирования, производный от MVC, который используется в основном для построения пользовательского интерфейса.

ПР2

1. Конфигурационный файл php.ini.

   Файл конфигурации (php.ini) считывается при запуске PHP. Для версий серверных модулей PHP это происходит только один раз при запуске веб-сервера.

2. Как написать простой скрипт на php.
   Создать php файл и внутри <?php ?> писать скрипт.
3. Основные правила, связанные с переменными в php.
   a. Имя переменной чувствительно к регистру.
   b. Правильное имя переменной должно начинаться с буквы или символа подчёркивания и состоять из букв, цифр и символов подчёркивания в любом количестве.
   c. По умолчанию переменные всегда присваиваются по значению.

4. Основные типы данных в php.

   a. bool (логический тип)
   b. int (целые числа)
   c. float (дробные числа)
   d. string (строки)
   e. array (массивы)
   f. object (объекты)
   g. callable (функции)
   h. mixed (любой тип)
   i. resource (ресурсы)
   j. null (отсутствие значения)

5. Какие существуют функции для работы с переменными в php вне зависимости от типа данных.
   a. boolval — Возвращает логическое значение переменной
   b. debug_zval_dump — Выводит строковое представление внутренней структуры zval
   c. doubleval — Псевдоним floatval
   d. empty — Проверяет, пуста ли переменная
   e. floatval — Возвращает значение переменной в виде числа с плавающей точкой
   f. get_debug_type — Возвращает имя типа переменной в виде, подходящем для отладки

g. get_defined_vars — Возвращает массив всех определённых переменных
h. get_resource_id — Возвращает целочисленный идентификатор для данного ресурса
i. get_resource_type — Возвращает тип ресурса
j. gettype — Возвращает тип переменной
k. intval — Возвращает целое значение переменной
l. is_array — Определяет, является ли переменная массивом
m. is_bool — Проверяет, является ли переменная булевой
n. is_callable — Проверяет, что значение может быть вызвано как функция в текущей области видимости
o. is_countable — Проверить, что содержимое переменной является счётным значением
p. is_double — Псевдоним is_float
q. is_float — Проверяет, является ли переменная числом с плавающей точкой
r. is_int — Проверяет, является ли переменная целым числом
s. is_integer — Псевдоним is_int
t. is_iterable — Проверяет, является ли переменная итерируемой
u. is_long — Псевдоним is_int
v. is_null — Проверяет, является ли значение переменной равным null
w. is_numeric — Проверяет, является ли переменная числом или строкой, содержащей число
x. is_object — Проверяет, является ли переменная объектом
y. is_real — Псевдоним is_float
z. is_resource — Проверяет, является ли переменная ресурсом
aa. is_scalar — Проверяет, является ли переменная скалярным значением
bb. is_string — Проверяет, является ли переменная строкой
cc. isset — Определяет, была ли установлена переменная значением, отличным от null
dd. print_r — Выводит удобочитаемую информацию о переменной
ee. serialize — Генерирует пригодное для хранения представление переменной
ff. settype — Задаёт тип переменной
gg. strval — Возвращает строковое значение переменной
hh. unserialize — Создаёт PHP-значение из хранимого представления
ii. unset — Удаляет переменную
jj. var_dump — Выводит информацию о переменной
kk. var_export — Выводит или возвращает интерпретируемое строковое представление переменной

6. Предопределенные переменные в php.
   PHP предоставляет всем скриптам большое количество предопределённых переменных. Эти переменные содержат всё, от внешних данных до переменных среды окружения, от текста сообщений об ошибках до последних полученных заголовков.
   Например - $_GET — Переменные HTTP GET
   $_POST — Переменные HTTP POST
7. Переменные переменных в php.
   Переменная переменной - имя переменной, которое может быть определено и изменено динамически. Обычная переменная ъ таким выражением:
   <?php
   $a = 'hello';
   ?>
   Переменная переменной берет значение переменной и рассматривает его как имя переменной. В вышеприведённом примере hello может быть использовано как имя переменной при помощи двух знаков доллара. То есть:
   <?php
   $$a = 'world';
   ?>
   еперь в дереве символов PHP определены и содержатся две переменные: $a, содержащая "hello" и $hello, содержащая "world".

8. Выражения в php.
   Выражение – это "все что угодно, имеющее значение". Основными формами выражений являются константы и переменные. Немного более сложными примерами выражений являются функции.

9. Арифметические операторы в php.
   - +$a Идентичность
   - -$a Отрицание
   - $a + $b Сложение
   - $a - $b Вычитание
   - $a * $b Умножение
   - $a / $b Деление
   - $a % $b Деление по модулю
   - $a ** $b Возведение в степень

10. Битовые операции в php.
    - $a & $b И

- $a | $b Или
- $a ^ $b Исключающее или
- ~ $a Отрицание
- $a << $b Сдвиг влево - Все биты переменной $a сдвигаются на $b позиций влево (каждая позиция подразумевает "умножение на 2")
- $a >> $b Сдвиг вправо - Все биты переменной $a сдвигаются на $b позиций вправо (каждая позиция подразумевает "деление на 2")

11.Оператор присваивания в php.
Базовый оператор присваивания обозначается как "=".

```
<?php
$a = ($b = 4) + 5; // $a теперь равно 9, а $b было присвоено 4.
?>
```

В дополнение к базовому оператору присваивания имеются "комбинированные операторы" для всех бинарных арифметических операций, операций объединения массивов и строковых операций, которые позволяют использовать некоторое значение в выражении, а затем установить его как результат данного выражения.

12.Операторы сравнения в php.
- $a == $b - Равно true если $a равно $b после преобразования типов.
- $a === $b - Тождественно равнотrue если $a равно $b и имеет тот же тип.
- $a != $b - Не равно true если $a не равно $b после преобразования типов.
- $a <> $b - Не равно true если $a не равно $b после преобразования типов.
- $a !== $b - Тождественно не равно true если $a не равно $b, или они разных типов.
- $a < $b - Меньше true если $a строго меньше $b.
- $a > $b - Больше true если $a строго больше $b.
- $a <= $b - Меньше или равно true если $a меньше или равно $b.
- $a >= $b - Больше или равно true если $a больше или равно $b.
- $a <=> $b - Космический корабль (spaceship) Число типа int меньше, больше или равное нулю, когда $a соответственно меньше, больше или равно $b.

13.Логические операторы в php.
- $a and $b - И
- $a or $b - Или

- $a xor $b - Исключающее или
- ! $a - Отрицание.
- $a && $b - И
- $a || $b – Или

14.Условная конструкция в php.

Конструкция if (условие) проверяет истинность некоторого условия, и если оно окажется истинным, то выполняется блок выражений, стоящих после if. Если же условие ложно, то есть равно false, тогда блок if не выполняется. Например:

```
<?php
$a = 4;
if($a>0){
echo "Переменная а больше нуля";
}
echo "<br>конец выполнения программы";
?>
```

Блок выражений ограничивается фигурными скобками. И так как в данном случае условие истинно (то есть равно true): значение переменной $a больше 0, то блок инструкций в фигурных скобках также будет выполняться. Если бы значение $a было бы меньше 0, то блок if не выполнялся.
Если блок if содержит всего одну инструкцию, то можно опустить фигурные скобки:

```
<?php
$a = 4;
if($a>0)
echo "Переменная а больше нуля";
echo "<br>конец выполнения программы";
?>
```

Можно в одной строке поместить всю конструкцию:

```
if($a>0) echo "Переменная а больше нуля";
```

15.Циклы в php.

Циклы позволяют повторять определенное (и даже неопределенное - когда работа цикла зависит от условия) колличество раз различные операторы. Данные операторы называются телом цикла. Проход цикла называется итерацией.

PHP поддерживает 4 вида циклов:

- Цикл с предусловием (while);
- Цикл с постусловием (do-while);
- Цикл со счетчиком (for);
- Специальный цикл перебора массивов (foreach).

При использовании циклов есть возможность использования операторов break и continue. Первый из них прерывает работу всего цикла, а второй - только текущей итерации.

16. Конструкции switch и match в php.

Конструкция switch..case является альтернативой использованию конструкции if..elseif..else. Оператор switch получает некоторое выражение и сравнивает его с набором значений.
После ключевого слова switch в скобках идет сравниваемое выражение. Значение этого выражения последовательно сравнивается со значениями, помещенными после операторов case. И если совпадение будет найдено, то будет выполняться определенный блок case.
Начиная с версии 8.0 в PHP была добавлена поддержка другой, похожей конструкции - match. Она позволяет оптимизировать конструкцию switch. Конструкция match также принимает некоторое выражение и сравнивает его с набором значений.

17. Include и require в php.

В PHP есть две функции, которые используются для помещения содержимого файла, содержащего исходный код PHP, в другой файл PHP. Это функции Include() и Require(). Обе функции одинаковы, но они имеют одно различие. Разница в том, что функция include() выдает предупреждение, но скрипт продолжит выполнение, а функция require() выдает предупреждение и фатальную ошибку, т.е. скрипт не будет продолжать выполнение. Эти две функции используются для помещения данных файла в другой файл PHP перед его выполнением сервером.

18. Функции в php.

Внутри функции можно использовать любой корректный PHP-код, в том числе другие функции и даже объявления классов.

Имена функций следуют тем же правилам, что и другие метки в PHP. Корректное имя функции начинается с буквы или знака подчёркивания, за

которым следует любое количество букв, цифр или знаков подчёркивания. В качестве регулярного выражения оно может быть выражено так: ^[a-zA-Z_\x80-\xff][a-zA-Z0-9_\x80-\xff]*$. Все функции и классы PHP имеют глобальную область видимости - они могут быть вызваны вне функции, даже если были определены внутри и наоборот.

ПР3

1. Что такое веб-сервер?

Веб-сервер — сервер, принимающий HTTP-запросы от клиентов, обычно веб-браузеров, и выдающий им HTTP-ответы, как правило, вместе с HTML-страницей, изображением, файлом, медиа-потоком или другими данными.

2. Что такое сервер приложения и чем он отличается от веб-сервера?

Основное различие между веб-сервером и сервером приложений заключается в том, что веб-сервер предназначен для обслуживания статических страниц, например HTML и CSS, тогда как сервер приложений отвечает за генерацию динамического содержимого путем выполнения кода на стороне сервера, например, JSP, сервлета или EJB.

3. Кратко опишите историю развития интернета в рамках развития веб-серверов.

Первый сервер ARPANET был установлен 1 сентября 1969 года в Калифорнийском университете в Лос-Анджелесе. Компьютер Honeywell DP-516 имел 24 Кб оперативной памяти.

29 октября 1969 года в 21:00 между двумя первыми узлами сети ARPANET, находящимися на расстоянии в 640 км — в Калифорнийском университете Лос-Анджелеса (UCLA) и в Стэнфордском исследовательском институте (SRI) — провели сеанс связи. Чарли Клайн (Charley Kline) пытался выполнить удалённое подключение к компьютеру в SRI. Успешную передачу каждого введённого символа его коллега Билл Дювалль (Bill Duvall) из SRI подтверждал по телефону.

В первый раз удалось отправить всего три символа «LOG», после чего сеть перестала функционировать. LOG должно было быть словом LOGON (команда входа в систему). В рабочее состояние систему вернули уже к 22:30

и следующая попытка оказалась успешной. Именно эту дату можно считать днём рождения Интернета. К 1971 году была разработана первая программа для отправки электронной почты по сети. Эта программа сразу стала очень популярна. В 1973 году к сети были подключены через трансатлантический телефонный кабель первые иностранные организации из Великобритании и Норвегии, сеть стала международной. В 1970-х годах сеть в основном использовалась для пересылки электронной почты, тогда же появились первые списки почтовой рассылки, новостные группы и доски объявлений. Однако в то время сеть ещё не могла легко взаимодействовать с другими сетями, построенными на других технических стандартах. К концу 1970-х годов начали бурно развиваться протоколы передачи данных, которые были стандартизированы в 1982—83 годах. Активную роль в разработке и стандартизации сетевых протоколов играл Джон Постел. 1 января 1983 года сеть ARPANET перешла с протокола NCP на TCP/IP, который успешно применяется до сих пор для объединения (или, как ещё говорят, «наслоения») сетей. Именно в 1983 году термин «Интернет» закрепился за сетью ARPANET. В 1984 году была разработана система доменных имён (англ. Domain Name System, DNS).

В 1984 году у сети ARPANET появился серьёзный соперник: Национальный научный фонд США (NSF) основал обширную межуниверситетскую сеть NSFNet (англ. National Science Foundation Network), которая была составлена из более мелких сетей (включая известные тогда сети Usenet и Bitnet) и имела гораздо бóльшую пропускную способность, чем ARPANET. К этой сети за год подключились около 10 тыс. компьютеров, звание «Интернет» начало плавно переходить к NSFNet. В 1988 году был разработан протокол Internet Relay Chat (IRC), благодаря чему в Интернете стало возможно общение в реальном времени (чат). В 1989 году в Европе, в стенах Европейского совета по ядерным исследованиям (фр. Conseil Européen pour la Recherche Nucléaire, CERN) родилась концепция Всемирной паутины. Её предложил знаменитый британский учёный Тим Бернерс-Ли, он же в течение двух лет разработал

протокол HTTP, язык HTML и идентификаторы URI. В 1990 году сеть ARPANET прекратила своё существование, полностью проиграв конкуренцию NSFNet. В том же году было зафиксировано первое подключение к Интернету по телефонной линии (т. н. «дозво́н» — англ. Dialup access). В 1991 году Всемирная паутина стала общедоступна в Интернете, а в 1993 году появился знаменитый веб-браузер NCSA Mosaic. Всемирная паутина набирала популярность. Можно считать что существует две ясно различимые эры в истории Web: [до браузера Mosaic] Марка Андриссена и после. Именно сочетание веб-протокола от Тима Бернерс-Ли, который обеспечивал коммуникацию, и браузера (Mosaic) от Марка Андриссена, который предоставил функционально совершенный пользовательский интерфейс, создало условия для наблюдаемого взрыва (интереса к Веб). За первые 24 месяца, истекшие после появления браузера Mosaic, Web прошел стадию от полной неизвестности (за пределами считанного числа людей внутри узкой группы ученых и специалистов лишь одного мало кому известного профиля деятельности) до полной и абсолютно везде в мире его распространенности. A Brief History of Cyberspace, Mark Pesce, ZDNet, 15 октября 1995В 1995 году NSFNet вернулась к роли исследовательской сети, маршрутизацией всего трафика Интернета теперь занимались сетевые провайдеры, а не суперкомпьютеры Национального научного фонда.

В том же 1995 году Всемирная паутина стала основным поставщиком информации в Интернете, обогнав по трафику протокол пересылки файлов FTP. Был образован Консорциум всемирной паутины (W3C). Можно сказать, что Всемирная паутина преобразила Интернет и создала его современный облик. С 1996 года Всемирная паутина почти полностью подменяет собой понятие «Интернет». В 1990-е годы Интернет объединил в себе большинство существовавших тогда сетей (хотя некоторые, как Фидонет, остались обособленными). Объединение выглядело привлекательным благодаря отсутствию единого руководства, а также благодаря открытости технических

стандартов Интернета, что делало сети независимыми от бизнеса и конкретных компаний. К 1997 году в Интернете насчитывалось уже около 10 млн компьютеров, было зарегистрировано более 1 млн доменных имён. Интернет стал очень популярным средством для обмена информацией. В настоящее время подключиться к Интернету можно через спутники связи, радио-каналы, кабельное телевидение, телефон, сотовую связь, специальные оптико-волоконные линии или электропровода. Всемирная сеть стала неотъемлемой частью жизни в развитых и развивающихся странах.
В течение пяти лет Интернет достиг аудитории свыше 50 миллионов пользователей.

4. Кратко опишите протокол HTTP.

HTTP — это протокол, позволяющий получать различные ресурсы, например HTML-документы. Протокол HTTP лежит в основе обмена данными в Интернете. HTTP является протоколом клиент-серверного взаимодействия, что означает инициирование запросов к серверу самим получателем, обычно веб-браузером (web-browser).

5. Опишите механизм взаимодействия HTTP-сервера, HTTP-клиента и пользователя.

Когда пользователь хочет перейти на страницу, браузер отправляет HTTP-запрос GET с указанием URL-адреса его HTML-страницы. Сервер извлекает запрошенный документ из своей файловой системы и возвращает HTTP-ответ, содержащий документ и код состояния HTTP Response status code 200 OK (успех). Сервер может вернуть другой код состояния, например, «404 Not Found», если файл отсутствует на сервере или «301 Moved Permanently», если файл существует, но был перемещён в другое место.

6. Опишите цели и задачи веб-сервера.

Цель веб-сервера проста - обслуживать одновременно большое количество клиентов, максимально эффективно используя hardware. Главная задача веб сервера принимать HTTP-запросы от пользователей, обрабатывать их, переводить в цифровой компьютерный код. Затем выдавать HTTP-ответы,

преобразуя их из миллионов нолей и единичек в изображения, медиа-потоки, буквы, HTML страницы.Любой веб сервер, для удобства его использования пользователями, должен иметь удобный веб-браузер. Он передает веб серверу запросы, преобразованные в URL-адреса интернет - ресурсов.Наряду со стандартными функциями, некоторые веб серверы имеют дополнительные. Так, к примеру, соответствующее программное обеспечение может фиксировать число обращений пользователей к тому или иному ресурсу, записывать их в отдельный журнал. А еще они могут поддерживать HTTPS, что не маловажно для защищенного соединения между сайтами и пользователями. Зачастую веб-сервер устанавливается вместе с мейл-сервером. Это позволяет пользователям быстро переходить на страничку почты прямо с сайта, нажав всего лишь на одну гиперссылку.

7. Опишите технологию SSI.

SSI – это простая и удобная технология организации динамических страничек. SSI экономит место на сервере, и одновременно делает администрирование сайта удобней в десятки раз.

8. Что такое система управления контентом?

Система управления контентом (CMS) — это программное обеспечение, которое работает в вашем браузере. Она позволяет создавать, управлять и изменять веб-сайт и его содержимое, не имея никаких знаний в области программирования. Система управления контентом предоставляет вам графический интерфейс пользователя. В нём вы можете управлять всеми аспектами вашего сайта.

9. Верно ли, что сервер приложения умеет работать с протоколом HTTP?

Верно.

HTTP-клиентами чаще всего являются браузеры — Google Chrome, Mozilla Firefox, Safari, Opera, Yandex Browser и другие. А серверами являются веб-сервера. Вот эта приставка «веб-» и указывает на то, что это не просто какой-то сервер, а сервер, который умеет принимать запросы и отвечать на них по протоколу HTTP.

10. Что такое CGI?

CGI (от англ. Common Gateway Interface — «общий интерфейс шлюза») — стандарт интерфейса, используемого для связи внешней программы с веб-сервером. Программу, которая работает по такому интерфейсу совместно с веб-сервером, принято называть шлюзом

11. Как работает система с использованием интерфейс шлюза - CGI?

Сам интерфейс разработан таким образом, чтобы можно было использовать любой язык программирования, который может работать со стандартными устройствами ввода-вывода. Такими возможностями обладают даже скрипты для встроенных командных интерпретаторов операционных систем, поэтому в простых случаях могут использоваться даже командные скрипты.

12. Назовите достоинства и недостатки CGI.

- CGI не налагает особых условий на платформу и web - сервер, поэтому работает на всех популярных платформах и web - серверах. Также технология не привязана к конкретному языку программирования и может быть использована на любом языке, работающем со стандартными потоками ввода/вывода.
- Производительность CGI - программ не высока. Основной причиной этого является то, что при очередном обращении к серверу для работы CGI – программы создается отдельный процесс, что требует большого количества системных ресурсов.
- Встроенных средств масштабируемости технология не предусматривает.
- CGI - программа представляет из себя готовый к исполнению файл, что препятствует легкому расширению системы.

13.Что такое FastCGI?

Интерфейс FastCGI — клиент-серверный протокол взаимодействия веб-сервера и приложения, дальнейшее развитие технологии CGI. По сравнению с CGI является более производительным и безопасным.

14.Назовите основные отличия CGI от FastCGI.

CGI-программа, запущенная в цикле. Если обычная CGI-программа заново запускается для каждого нового запроса, то в FastCGI-программе используется очередь запросов, которые обрабатываются последовательно.

15.Что такое менеджер процессов?

Поток process_manager отвечает за генерацию новых экземпляров процесса, за связь со средой и между экземплярами процесса, а также за выполнение экземпляров процесса.

16.Что такое PHP-FPM?

PHP-FPM — это альтернативная реализация PHP FastCGI с несколькими дополнительными возможностями, которые обычно используются для высоконагруженных сайтов.

17.Что такое Spawn-fcgi?

spawn-fcgi — одна из составных частей проекта Lighttpd. Предназначен он для того, что бы запустить php, как FastCGI сервер, ну а с этим сервером может работать потом практически любой http сервер.

18.Что такое Lighttpd?

lighttpd это веб-сервер с открытым исходным кодом, оптимизированный для критически важных сред, отвечает за предоставление доступа через HTTP или HTTPS протокол к статическому контенту.

19. Что такое chroot окружение?

Chroot окружение — способ запуска программ, системный вызов и просто команда, позволяющая изменить корневой каталог в системе.

Chroot используется для создания jail и изоляции процессов, а также для сборки пакетов методом debootstrap. Командой chroot с указанием каталога запускается механизм, создающий систему директорий в этом каталоге идентичную той, что существует в корне. Команда выполняется только от имени суперпользователя.

20. Опишите механизм взаимодействия сервисов с использованием FastCGI

FastCGI — открытый унифицированный стандарт, расширяющий

интерфейс CGI и позволяющий создавать высокопроизводительные web-приложения без использования специфичных API web-сервера. Цель данной спецификации — с точки зрения FastCGI-приложения описать интерфейс взаимодействия между ним и web-сервером, также реализующим интерфейс FastCGI. Связь между web-сервером и FastCGI-процессом осуществляется через один сокет, который процесс должен слушать на предмет входящих подключений от web-сервера.
После приема соединения от web-сервера FastCGI-процесс обменивается данными с использованием простого протокола, решающего две задачи: организация двунаправленного обмена в рамках одного соединения (для эмуляции STDIN, STDOUT, STDERR) и организация нескольких независимых FastCGI-сессий в рамках одного соединения.

21.Опишите процесс выбора встроенного или внешнего менеджера процессов.

Менеджер процессов тесно взаимодействует с микроядром, чтобы обеспечить услуги, составляющие сущность операционной системы. Менеджер процессов отвечает за создание новых процессов в системе и за управление основными ресурсами, связанными с процессом. Все эти услуги предоставляются посредством сообщений.
Встроенный менеджер процессов – удобная программа, с помощью которой можно определять запущенные на ПК приложения, процессы и службы, управлять вышеперечисленными компонентами (запускать, останавливать или завершать). С
помощью нее можно оценивать и влиять на быстродействие компьютера, а также выполнять другие задачи.
К сожалению, встроенный менеджер процессов для анализа и обработки запущенных процессов не всегда удобен и имеет очень ограниченные возможности, он не в полной мере отображает процессы, запущенные на компьютере, а средства анализа запущенных процессов вообще весьма

ограничены. Из-за недостатков в применении менеджера процессов для анализа процессов, запущенных на компьютере, приходится прибегать к внешним менеджерам процессов. Они более качественно отображают информацию и позволяют лучше контролировать процессы.

22. Что такое интерфейс шлюза?

Сетевой шлюз - аппаратный маршрутизатор или программное обеспечение для сопряжения компьютерных сетей, использующих разные протоколы (например, локальной и глобальной). Сетевой шлюз конвертирует протоколы одного типа физической среды в протоколы другой физической среды (сети).

23. Что такое SCGI?

SCGI (Simple Common Gateway Interface) - простой общий интерфейс шлюза - разработан как альтернатива CGI и во многом аналогичен FastCGI, но более прост в реализации.

24. Что такое PCGI?

PCGI (Perl Common Gateway Interface) — библиотека к языку программирования Perl для работы с интерфейсом CGI. Библиотека позволяет с высокой скоростью обрабатывать входящий поток данных. Основное достоинство заключается в том, что библиотека позволяет совершенно безопасно принимать сколь угодно крупные объёмы данных, при этом очень экономично потребляя оперативную память.

25. Что такое PSGI?

PSGI (Perl Web Server Gateway Interface) – это спецификация, предназначенная для отделения среды веб-сервера от кода веб-фреймворка. PSGI не является программным интерфейсом (API) для веб-приложений.

26. Что такое WSGI?

WSGI — стандарт взаимодействия между Python-программой, выполняющейся на стороне сервера, и самим веб-сервером, например Apache. [WSGI предоставляет простой и универсальный интерфейс между большинством веб-серверов и веб-приложениями или фреймворками]. Это протокол, спецификация, которая была предложена в PEP 3333. Этот

протокол предназначен для решения проблем совместимости многих веб-платформ и программного обеспечения веб-сервера. Благодаря WSGI вам больше не нужно выбирать конкретное программное обеспечение веб-сервера из-за используемой веб-платформы.

27. Опишите механизм взаимодействия серверов Apache и PHP

Когда компьютер обращается к web-серверу Apache, то он запускает интерпретатор PHP. Он выполняет скрипт записанный в файле index.php. То есть Apache это сервер, который взаимодействует с клиентом (принимает и отвечает на запросы клиента), а затем сервер выполняет запрос клиента, используя PHP

28.Опишите преимущества веб-сервера Apache.

Основные достоинства Apache - надежность, безопасность и гибкость настройки. Apache позволяет подключать различные модули, добавляющие в него новые возможности - например, можно подключить модуль, обеспечивающий поддержку PHP или любого другого Web-ориентированного языка программирования.

29.Опишите недостатки веб-сервера Apache.

Недостатки - отсутствие удобного графического интерфейса администратора. Настройка Apache осуществляется путем редактирования его конфигурационного файла. В Интернете можно найти простые конфигураторы Apache, но их возможностей явно не хватает для настройки всех функций Web-сервера.

30.Опишите архитектуру веб-сервера Apache.

Apache состоит из ядра и динамической модульной системы. Параметры системы изменяются с помощью конфигурационных файлов. (В архитектуру Apache входит: простое ядро, платформо-зависимый уровень (APR), и модули.)

31.Опишите функции ядра веб-сервера Apache.

Ядро Apache включает в себя основные функциональные возможности,

такие как обработка конфигурационных файлов, протокол HTTP и система загрузки модулей. Ядро (в отличие от модулей) полностью разрабатывается Apache Software Foundation, без участия сторонних программистов. Теоретически ядро apache может функционировать в чистом виде, без использования модулей. Однако функциональность такого решения крайне ограничена.

Ядро Apache полностью написано на языке программирования C.

32.Опишите конфигурацию веб-сервера Apache.

Система конфигурации Apache основана на текстовых конфигурационных файлах. Имеет три условных уровня конфигурации:

- Конфигурация сервера
- Конфигурация виртуального хоста
- Конфигурация уровня каталога

33.Что такое URI, URL и чем они различаются.

URL — это URI, который, помимо идентификации ресурса, предоставляет ещё и информацию о местонахождении этого ресурса. A URN — это URI, который только идентифицирует ресурс в определённом пространстве имён (и, соответственно, в определённом контексте), но не указывает его местонахождение.

(URI: Это лишь обобщенное понятие (множество) идентификации ресурса, включающее в нашем случае как URL, так и URN, как по отдельности, так и совместно. Т.е. мы можем считать, что: URI = URL или URI = URN или URI = URL + URN)

ПР4

Что такое HTTP-запрос?

HTTP запросы - это сообщения, отправляемые клиентом, чтобы инициировать реакцию со стороны сервера.

2. Опишите существующие HTTP-запросы

Особенности GET запроса:

-может быть закэширован

-остается в истории браузера

-может быть закладкой в браузере

-не должен использоваться при работе с крайне -важными данными

-имеет ограниченную длину

-должен применяться только для получения данных (ред.)

Особенности POST запроса:

-не кэшируется

-не может быть закладкой в браузере

-не остаётся в истории браузера

-нет ограничений по длине запроса

3. Опишите обработку запроса на PHP. Что нужно использовать, как вычленить параметры запроса?

Внутри PHP-скрипта имеется несколько способов получения доступа к данным, переданным клиентом по протоколу HTTP. До версии PHP 4.1.0 доступ к таким данным осуществлялся по именам переданных переменных (напомним, что данные передаются в виде пар "имя переменной, символ "=", значение переменной"). Таким образом, если, например, было передано first_name=Nina, то внутри скрипта появлялась переменная $first_name со значением Nina. Если требовалось различать, каким методом были переданы данные, то использовались ассоциативные массивы $HTTP_POST_VARS и $HTTP_GET_VARS , ключами которых являлись имена переданных переменных, а значениями – соответственно значения этих переменных. Таким образом, если пара first_name = Nina передана методом GET , то $HTTP_GET_VARS["first_name"]="Nina".

Использовать в программе имена переданных переменных напрямую небезопасно. Поэтому было решено начиная с PHP 4.1.0 задействовать для обращения к переменным, переданным с помощью HTTP-запросов, специальный массив – $_REQUEST . Этот массив содержит данные,

переданные методами POST и GET , а также с помощью HTTP cookies. Это суперглобальный ассоциативный массив, т.е. его значения можно получить в любом месте программы, используя в качестве ключа имя соответствующей переменной (элемента формы).

4. Опишите создание HTML-форм на PHP.

Вставка формы осуществляется напрямую в HTML—код страницы. Главный элемент формы называется <form>. Уже внутрь него добавляются все остальные элементы – текстовые поля, «чекбоксы», переключатели и т.д. У элемента <form> имеется несколько атрибутов, один из которых является обязательным. Он называется action. В action указывается, где именно будет приниматься и обрабатываться информация, переданная посредством формы. Как правило, обработка происходит в стороннем PHP—файле. Пример использования атрибута– action=»obrabotchik.php«. Атрибут method позволяет задать метод передачи информации. По умолчанию (если не прописывать атрибут) будет указан метод GET. В данном случае информация передается напрямую через URL—адрес. Для каждого элемента формы будет создана пара следующего вида – «имя элемента = значение, которое в нем лежит». Все эти пары, разделенные знаком «амперсанд» будут перечислены в адресной строке. Если прописать method=»POST» (регистр не важен), то данные будут передаваться не через URL, а через тело запроса (в скрытом режиме)

5. Что такое API?

API — описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой. Обычно входит в описание какого-либо интернет-протокола, программного каркаса или стандарта вызовов функций операционной системы.

6. Опишите API как средство интеграции приложений.

Если программу (модуль, библиотеку) рассматривать как чёрный ящик, то API — это набор «ручек», которые доступны пользователю данного ящика и которые он может вертеть и дёргать.

Программные компоненты взаимодействуют друг с другом посредством API. При этом обычно компоненты образуют иерархию — высокоуровневые компоненты используют API низкоуровневых компонентов, а те, в свою очередь, используют API ещё более низкоуровневых компонентов. По такому принципу построены протоколы передачи данных по Интернету. Стандартный стек протоколов (сетевая модель OSI) содержит 7 уровней (от физического уровня передачи бит до уровня протоколов приложений, подобных протоколам HTTP и IMAP). Каждый уровень пользуется функциональностью предыдущего («нижележащего») уровня передачи данных и, в свою очередь, предоставляет нужную функциональность следующему («вышележащему») уровню.

Понятие протокола близко по смыслу к понятию API. И то, и другое является абстракцией функциональности, только в первом случае речь идёт о передаче данных, а во втором — о взаимодействии приложений.

API библиотеки функций и классов включает в себя описание сигнатур и семантики функций.

7. Что такое Web API?

Это интерфейс прикладного программирования для веб-сервера или веб-браузера. Это концепция веб-разработки, обычно ограниченная клиентской стороной веб-приложения, и поэтому обычно не включает детали реализации веб-сервера или браузера, такие как SAPI или API, если они не доступны для общего доступа через удаленное веб-приложение.

8. Приведите пример API.

Каждый раз, когда пользователь посещает какую-либо страницу в сети, он взаимодействует с API удалённого сервера. API — это составляющая часть сервера, которая получает запросы и отправляет ответы.

9. Что такое REST?

Это архитектура, т.е. принципы построения распределенных гипермедиа систем, того что другими словами называется World Wide Web, включая универсальные способы обработки и передачи состояний ресурсов

по HTTP.

10.Как организована передача данных в архитектуре REST?

Архитектура REST требует соблюдения следующего условия. В период между запросами серверу не нужно хранить информацию о состоянии клиента и наоборот. Все запросы от клиента должны быть составлены так, чтобы сервер получил всю необходимую информацию для выполнения запроса. Таким образом и сервер, и клиент могут «понимать» любое принятое сообщение, не опираясь при этом на предыдущие сообщения.

11.Как организована работа REST?

Это набор принципов и ограничений взаимодействия клиента и сервера в сети интернет, использующий существующие стандарты (HTTP протокол, стандарт построения URL, форматы данных JSON и XML) в ходе взаимодействии

12.Что такое SOAP?

SOAP (Simple Object Access Protocol) — стандартный протокол по версии W3C.

13.Чем SOAP отличается от REST?

· SOAP обозначает простой протокол доступа к объектам, тогда как REST обозначает передачу представительного состояния.

· SOAP — это протокол, тогда как REST — это архитектурный паттерн.

· SOAP использует сервисные интерфейсы для предоставления своих функций клиентским приложениям, а REST использует унифицированные локаторы сервисов для доступа к компонентам на аппаратном устройстве.

· SOAP требует большей пропускной способности для его использования, тогда как REST не требует большой пропускной способности.

· SOAP работает только с форматами XML, тогда как REST работает с простым текстом, XML, HTML и JSON.

· SOAP не может использовать REST, тогда как REST может использовать SOAP.

14.Для чего нужен SOAP-процессор?

SOAP основан на языке XML и расширяет некоторый протокол прикладного уровня — HTTP, FTP, SMTP и т.д. Как правило чаще всего используется HTTP. Вместо использования HTTP для запроса HTML-страницы, которая будет показана в браузере, SOAP отправляет посредством HTTP-запроса XML-сообщение и получает результат в HTTP-отклике. Для правильной обработки XML-сообщения процесс-«слушатель» HTTP (напр. Apache или Microsoft IIS) должен предоставить SOAP-процессор, или, другими словами, должен иметь возможность обрабатывать XML

15.Опишите общую структуру SOAP-сообщения.

Сообщение SOAP состоит из заголовка — элемент SOAP-ENV:Header и основной части — элемент SOAP-ENV:Body. Заголовок может содержать метаданные, относящиеся к сообщению в целом. В теле сообщения передается элемент params с входными параметрами метода. Формат элемента params отличается для разных методов.

16.Что такое и что содержит Конверт (SOAP Envelope)?

Является самым «верхним» элементом SOAP сообщения. Содержит корневой элемент

XML-документа. Описывается с помощью элемента Envelope с обязательным

пространством имен http://www.w3.org/2003/05/soap-envelope для версии 1.2 и

http://schemas.xmlsoap.org/soap/ для версии 1.1.

У элемента Envelope могут быть атрибуты xmlns, определяющие пространства

имен, и другие атрибуты, снабженные префиксами.

Envelope может иметь необязательный дочерний элемент Header с тем же

пространством имен — заголовок. Если этот элемент присутствует, то он должен быть

3

Рисунок 2. Структура

SOAP сообщения

Сервис-ориентированные технологии интеграции информации. Автор - Фастовский Э.Г. 2011 г.

первым прямым дочерним элементом конверта.

Следующий дочерний элемент конверта должен иметь имя Body и то же самое

пространство имен - тело. Это обязательный элемент и он должен быть вторым прямым

дочерним элементом конверта, если есть заголовок, или первым — если заголовка нет.

Версия 1.1 позволяла после тела сообщения записывать произвольные элементы,

снабженные префиксами. Версия 1.2 это запрещает.

Элементы Header и Body могут содержать элементы из различных пространств

имен.

Конверт изменяется от версии к версии. SOAP-процессоры, совместимые с версией

1.1, при получении сообщения, содержащего конверт с пространством имен версии 1.2,

будут генерировать сообщение об ошибке. Аналогично для SOAP-процессоров, совместимых

с версией 1.2. Ошибка — VersionMismatch.

17.Что такое и что содержит Заголовок SOAP (SOAP Header)?

Заголовок может содержать метаданные, относящиеся к сообщению в целом.Заголовок содержит элемент locale, устанавливающий русский язык для ответных сообщений, и элемент token — авторизационный токен.

18.Что такое и что содержит Тело SOAP (SOAP Body)?

Тело SOAP-сообщения является обязательным элементом внутри env:Envelope, содержащим основную информацию SOAP-сообщения, которая должна быть передана из начальной точки пути сообщения в конечную

19.Опишите SOAP-сообщение с вложением.

SOAP-сообщение представляет собой XML-документ; сообщение состоит из трех

основных элементов: конверт (SOAP Envelope), заголовок (SOAP Header) и тело (SOAP

Body).

20.Что такое graphql?

Язык запросов и обработки данных с открытым исходным кодом для API и среда выполнения для выполнения запросов с существующими данными.

21.Что такое Распознаватели (resolvers) в graphql?

Resolver или распознаватель — функция, которая возвращает данные для определённого поля. Resolver’ы возвращают данные того типа, который определён в схеме. Распознаватели могут быть асинхронными.

22.Из чего состоит экосистема graphql, что нужно, чтобы использовать данную технологию?

Она состоит из двух взаимосвязанных объектов: TypeDefs и Resolvers. Выше были описаны основные типы GraphQL. Чтобы сервер мог с ними работать, эти типы необходимо определить. Объект typeDef определяет список типов, которые доступны в проекте.

23.Что такое валидация данных и для чего она нужна?

Валидация – это проверка продукта, процесса или системы на соответствие требованиям клиента.

24.Где и когда выполнять валидацию данных?

Валидация проводится тогда, когда невозможно оценить соответствие продукта, процесса или системы требованиям клиента до того, как клиент начнет этим продуктом пользоваться. Например, если речь идет о

программном обеспечении, в него встраивается валидационный код. Этот код клиент вводит, если продукт полностью соответствует его ожиданиям и выполняет нужные задачи. В противном случае доступ к продукту прекращается и проводятся его доработки либо исполнитель возвращает деньги.

25.Как выполнять валидацию данных?

Перед отправкой данных на сервер важно убедиться, что все обязательные поля формы заполнены данными в корректном формате. Это называется валидацией на стороне клиента и помогает убедиться, что данные, введённые в каждый элемент формы, соответствуют требованиям.

26.Приведите пример с поэтапной валидацией данных.

Регистрация и авторизация: пользователь вводит валидные данные email и валидные данные пароль

27.Что такое запрос и мутация в graphql и чем они отличаются?

К первому виду относятся запросы на чтение данных, которые в терминологии GraphQL называются просто запросами (query) и относятся к букве R (reading, чтение) акронима CRUD. Запросы второго вида — это запросы на изменение данных, которые в GraphQL называют мутациями (mutation).

ПР5

1. Сессии являются механизмом, который использует для отслеживания "состояния" между сайтом и каким-либо браузером. Сессии позволяют вам хранить произвольные данные браузера и получать их в тот момент, когда между данным браузером и сайтом устанавливается соединение. Данные получаются и сохраняются в сессии при помощи соответствующего "ключа".

2. это небольшой фрагмент данных, отправляемый сервером на браузер пользователя, который тот может сохранить и отсылать обратно с новым запросом к данному серверу. Это, в частности, позволяет узнать, с одного ли браузера пришли оба запроса (например, для аутентификации пользователя). Они запоминают информацию о состоянии для протокола HTTP, который сам по себе этого делать не умеет.

3. Перевод веб-страниц между сервером и браузером происходит посредством протокола передачи гипертекста (http). Когда пользователь вводит URL в адресную строку браузера, браузер берет ее и отправляет запрос на сервер, запрашивая веб-страницы, заданные пользователем. Далее, сервер посылает страницу, запрашиваемую браузером, в виде http-ответа. Ответ передается в виде пакетов из текста, который может содержать заявление с просьбой, чтобы браузер сохранил куки. Это делается посредством заявления, "настройка cookie: имя = значение". Браузер спрашивает, сохранить значение-строка 'имя' и вернуть его на сервер при каком-либо дальнейших к нему обращений. В любой последующий запрос к одному серверу, даже при запросе другой веб-страницы с использованием данного сервера, браузер отправляет серверу значение куки. Сервер идентифицирует эту информацию и выполняет запрос, без необходимости пользователю выполнять процесс аутентификации еще раз.

4. Опишите простой пример работы сессий в PHP.

Сессии являются простым способом хранения информации для отдельных пользователей с уникальным идентификатором сессии. Это может использоваться для сохранения состояния между запросами страниц. Идентификаторы сессий обычно отправляются браузеру через сессионный cookie и используются для получения имеющихся данных сессии. Отсутствие идентификатора сессии или сессионного cookie сообщает PHP о том, что необходимо создать новую сессию и сгенерировать новый идентификатор сессии.

5. Опишите способы защиты сессии пользователя.

По умолчанию вся информация о сессии, включая ID, передается в cookie. Но так бывает не всегда. Некоторые пользователи отключают cookie в своих браузерах. В таком случае браузер будет передавать идентификатор сессии в URL.

Здесь ID передается в открытом виде, в отличие от сессии через cookie, когда информация скрыта в HTTP-заголовке. Самым простым способом защиты от этого будет запрет передачи идентификатора сессии через адресную строку. Сделать это можно, прописав следующее в конфигурационном файле Apache-сервера .htaccess:

php_flag session.use_only_cookies on

6. Верно ли, что можно хранить данные сессии в БД?

Сессия как правило - это необходимые горячие данные пользователя, что нужно сохранить между запросами. БД - одно из самых медленных хранилищ этих данных. В файлах кстати тоже не стоит хранить. При большой нагрузке io будет подтормаживать. Длительный период сессии обычно не хранятся, вместо этого на клиент задается токен, по которому человек через много времени может автоматически авторизироваться 7. Что такое Web API?
Это интерфейс прикладного программирования для веб-сервера или веб-браузера. Это концепция веб-разработки, обычно ограниченная клиентской стороной веб-приложения, и поэтому обычно не включает детали реализации веб-сервера или браузера, такие как SAPI или API, если они не доступны для общего доступа через удаленное веб-приложение.

7) Жизненный цикл сессии проходит несколько этапов :

1) Абсолютно для каждого нового запроса на сервер (неважно, разные это клиенты или один) ASP.NET генерирует уникальный идентификатор сессии. Идентификатор сессии представляет собой случайно сгенерированное число, закодированное с помощью специального алгоритма в строку длиной 24 символа. Строка состоит из литералов от A до Z в нижнем регистре, а также чисел от 0 до 5. Пример идентификатора - hjnyuijl1pam3vox2h5i41in

2)Если в течение текущего запроса данные клиента НЕ сохраняются для дальнейшей работы с ним, то и время жизни сессии этого клиента заканчивается (фактически не начавшись). При этом ранее сгенерированный идентификатор сессии становится недействительным (так как не был использован). В ответ на такой запрос клиент не получает ничего, чтобы связало его с новой сессией.

3)Если же данные клиента (например, имя, адрес доставки товара) сохраняются на сервере, ASP.NET связывает сохраненные данные с ранее сгенерированным идентификатором сессии. Далее создается специальная сессионная куки, и в нее записывается также этот идентификатор. Эта куки добавляется в ответ на запрос и сохраняется в браузере клиента. Таким образом, создается связь клиента и его персонализированной информации на сервере. Новая сессия для данного клиента создана.

4)При каждом следующем запросе клиент передает на сервер персональный идентификатор сессии через куки. Сервер сопоставляет идентификаторы и «узнает» клиента в рамках текущей сессии.

5)До тех пор пока клиент передает свой персональный ключ, сессия считается активной. Сессия может закончиться по разным причинам, например, вручную

на стороне сервера или по истечении какого-то установленного времени (таймаут).

8. Обработка сессии это ключевой приём в PHP, что позволяет хранить данные пользователя на всех страницах веб-сайта или приложения, так что нет.

9. Система управления сессиями поддерживает ряд опций, которые могут быть указаны в файле php.ini. Ниже приводится краткий обзор.

session.save_handler string

session.save_handler определяет имя обработчика, который используется для хранения и извлечения данных, связанных с сессией. По умолчанию имеет значение files. Следует обратить внимание, что некоторые модули могут зарегистрировать собственные обработчики (save_handler). Текущие зарегистрированные обработчики отображаются в phpinfo(). Смотрите также session_set_save_handler().

session.save_path string

session.save_path определяет аргумент, который передаётся в обработчик сохранения. При установленном по умолчанию обработчике files, аргумент содержит путь, где будут создаваться файлы. Смотрите также session_save_path().

У этой директивы также существует дополнительный аргумент N, определяющий глубину размещения файлов сессии относительно указанной директории. Например, указание '5;/tmp' может в конечном итоге привести к такому размещению файла сессии: /tmp/4/b/1/e/3/sess_4b1e384ad74619bd212e236e52a5a174If . Для того, чтобы использовать аргумент N, необходимо предварительно создать все эти директории. Помочь в этом может небольшой скрипт, расположенный в ext/session. Версия для bash называется mod_files.sh, а Windows-версия - mod_files.bat. Также следует учитывать, что если N определён и больше 0, то автоматическая сборка мусора не выполняется, подробнее смотрите информацию в файле php.ini. Кроме того, если используется N, необходимо удостовериться, что значение session.save_path указано в кавычках, поскольку разделитель (;) в php.ini используется как знак комментария.

Модуль хранения файлов создаёт файлы с правами 600 по умолчанию. Это можно изменить с помощью необязательного аргумента MODE: N;MODE;/path, где MODE - восьмеричное представление режима доступа к файлу. Установка MODE не затрагивает umask.

10 Опишите директивы конфигурации файловой системы и потоков в PHP

Имя Значение по умолчанию Область изменения

allow_url_fopen "1" PHP_INI_SYSTEM

user_agent NULL PHP_INI_ALL

default_socket_timeout "60" PHP_INI_ALL

from NULL ??

auto_detect_line_endings "Off" PHP_INI_ALL

allow_url_fopen boolean

Данная директива включает поддержку упаковщиков URL (URL wrappers), которые позволяют работать с объектами URL, как с обычными файлами. Упаковщики, доступные по умолчанию, служат для работы с удаленными файлами с использованием протокола ftp или http. Некоторые расширения, например, zlib, могут регистрировать собственные упаковщики.

ser_agent string

Устанавливает строку "User-Agent" для использования ее PHP при запросах к удаленным серверам.

default_socket_timeout integer

Значение таймаута (в секундах) для потоков, использующих сокеты.

Замечание: Данная директива стала доступна с версии PHP 4.3.0

from="joe@example.com" string

Устанавливает пароль для анонимного доступа к серверу ftp (ваш адрес электронной почты).

auto_detect_line_endings boolean

Когда данная директива включена, PHP проверяет данные, получаемые функциями fgets() и file() с тем, чтобы определить способ завершения строк (Unix, MS-Dos или Macintosh).

Данная директива позволяет PHP взаимодействовать с системами Macintosh, однако, по умолчанию эта директива выключена, поскольку при ее использовании возникает (несущественная) потребность в дополнительных ресурсах для определения символа окончания первой строки, а также потому, что программисты, использующие в системах Unix символы перевода строки в качестве разделителей, столкнутся с обратно-несовместимым поведением PHP.

11 Какой тип ресурса использует файловая система. Опишите данный тип

Файловая система. На каждом носителе информации (гибком, жестком или лазерном диске) может храниться большое количество файлов. Порядок хранения файлов на диске определяется используемой файловой системой.

Каждый диск разбивается на две области: область хранения файлов и каталог. Каталог содержит имя файла и указание на начало его размещения на диске. Если провести аналогию диска с книгой, то область хранения файлов соответствует ее содержанию, а каталог - оглавлению. Причем книга состоит из страниц, а диск - из секторов.

Для дисков с небольшим количеством файлов (до нескольких десятков) может использоваться одноуровневая файловая система, когда каталог (оглавление диска) представляет собой линейную последовательность имен файлов. Такой каталог можно сравнить с оглавлением детской книжки, которое содержит только названия отдельных рассказов.

Если на диске хранятся сотни и тысячи файлов, то для удобства поиска используется многоуровневая иерархическая файловая система, которая имеет древовидную структуру. Такую иерархическую систему можно сравнить, например, с оглавлением учебника, которое представляет собой иерархическую систему разделов, глав, параграфов и пунктов.

Начальный, корневой каталог содержит вложенные каталоги 1-го уровня, в свою очередь, каждый из последних может содержать вложенные

каталоги 2-го уровня и так далее. Необходимо отметить, что в каталогах всех уровней могут храниться и файлы.

12 Как открыть и закрыть файл с помощью PHP

Для открытия файла используется функция fopen().

Ее синтаксис: int fopen(string filename, string mode [, int use_include_path])

Принимаемые аргументы:

string filename – имя файла или абсолютный путь к нему. Если путь к файлу не будет указан, то будет произведен его поиск в текущем каталоге. При отсутствии искомого файла система выведет сообщение об ошибке.

string mode – указывает режим открытия файла. Принимаемые аргументом значения:

r – файл открыт только для чтения, файловый указатель устанавливается в начале;

r+ – файл открыт для чтения и записи;

w – создается новый файл только для записи. Если файл с таким именем уже существует, в нем происходит автоматическое удаление всех данных;

w+ — создается новый файл для записи и чтения. При существовании такого файла происходит полная перезапись его данных на новые;

a – файл открыт для записи. Указатель устанавливается в конце. То есть запись в файл php начнется не с начала, а с конца;

a+ – открытие файла в режиме чтения и записи. Запись начнется с конца;

b – режим работы с файлом, содержащим в себе двоичные данные (в двоичной системе исчисления). Этот режим доступен только в операционной системе Windows.

Для закрытия доступа к файлу служит функция fclose ().

Ее синтаксис: int fclose (int file), где int file – дескриптор сайта, который нужно закрыть.

После каждого чтения или записи файл нужно закрывать этой функцией. Иначе остается открытым поток, созданный для файла. А это ведет к лишнему расходу серверных мощностей.

13. Как производится чтение и запись файлов в PHP

//Открытие тестового файла $file = fopen('test.txt', 'wt');

//Запись строки в файл fwrite($file, 'Текущая дата и время: '. date('d.m.y H:i:s')); //Закрытие файла fclose($file).

14. Опишите как считать только часть файла, как считывать файл последовательно и считать весь файл целиком.

Если нам надо прочитать файл полностью, то мы можем облегчить себе жизнь, применив функцию **file_get_contents()**:

```
<?php
$str = htmlentities(file_get_contents("form.php"));
echo $str;
?>
```

Также можно провести поблочное считывание, то есть считывать определенное количество байт из файла с помощью функции **fread()**:

```
<?php
$fd = fopen("form.php", 'r') or die("не удалось открыть файл");
while(!feof($fd))
```

```
{
   $str = htmlentities(fread($fd, 600));
   echo $str;
}
fclose($fd);
?>
```

15. Как производится создание и удаление файлов с помощью PHP

Для создания и открытия файла на PHP используют функцию fopen():

```
fopen(имя_файла, режим_файла);
```

- имя_файла – здесь нужно указать название и расширение файла, которое нужно создать или открыть. Например, «bloggood-ru.txt».
- режим_файла – здесь нужно указать режим, другими словами параметры. Например, что вы хотите сделать с этим файлом: дописать текст или вставить новый и т.д

Для **удаления файла** в **PHP** предусмотрена функция unlink(f). Она принимает единственный строковой параметр, который указывает директорию и имя **файла**, который следует **удалить**. Пример: if (unlink(«D:\documents\copy.txt»)) echo «**Файл** успешно **удален**»; elseecho «Ошибка!

16. С помощью каких функций и какую информацию о файле можно получить с помощью PHP?

fstat — Получает информацию о файле, используя открытый файловый указатель. fsync — Синхронизирует изменения в файле (включая метаданные). ftell — Возвращает текущую позицию указателя чтения/записи файла.

17. Что такое DOM?

DOM означает объектную модель документа. Это программный интерфейс, который позволяет нам создавать, изменять или удалять элементы из документа. Мы также можем добавлять события к этим элементам, чтобы сделать нашу страницу более динамичной. Модель DOM рассматривает документ HTML как дерево узлов.

18. Как создать документ и работать с ним с помощью модуля DOM?

Можно представить HTML как набор вложенных коробок. Теги вроде <body> и </body> включают в себя другие теги, которые в свою очередь включают теги, или текст. Структура данных, использующаяся браузером для представления документа, отражает его форму. Для каждой коробки есть объект, с которым мы можем взаимодействовать и узнавать про него разные данные – какой тег он представляет, какие коробки и текст содержит. Это представление называется Document Object Model (объектная модель документа), или сокращённо DOM.

Мы можем получить доступ к этим объектам через глобальную переменную document. Её свойство documentElement ссылается на объект, представляющий тег . Он также предоставляет свойства head и body, в которых содержатся объекты для соответствующих элементов.

19. Что такое JSON?

JSON (JavaScript Object Notation) — это формат для хранения и обмена информацией, доступной для чтения человеком. Файл содержит только текст и использует расширение. json.

20. Как декодировать строку JSON и вернуть JSON-представление данных?

Метод JSONDecoder.raw_decode() декодирует JSON-документ из строки s в формате JSON и возвращает двойной кортеж представление данной строки в Python и индекс в строки s, где документ закончился. Метод JSONDecoder.raw_decode() может быть использован
для декодирования документа JSON из строки, которая в конце содержит посторонние данные.

21. Как проанализировать и выявить ошибки при кодировании и декодировании JSON?

В некоторых случаях вам может не потребоваться Codable для поддержки двунаправленного кодирования и декодирования.

Например, некоторым приложениям требуется только вызывать удаленные сетевые API, и им не нужно декодировать ответы, содержащие тот же тип.

Encodable Если вам нужна только поддержка кодирования данных Encode, заявите о соответствии. И наоборот, Decodable заявляет о соответствии, если вам нужно только читать данные определенного типа.

22)

можно создать XML-документ и считывать его функцией
simplexml_load_file().
simplexml_load_file — Интерпретирует XML-файл в объект

```
simplexml_load_file(
string $filename,
?string $class_name = SimpleXMLElement::class,
int $options = 0,
string $namespace_or_prefix = "",
bool $is_prefix = false
): SimpleXMLElement|false
```

Преобразует правильно сформированный XML-документ в указанном файле в объект.

23)

драйверы на основе субд используются с такими источниками данных, как Oracle или SQL Server, которые предоставляют автономное ядро субд для использования драйвером. Эти драйверы обращаются к физическим данным через автономный модуль. то есть они отправляют SQLные инструкции и получают результаты из подсистемы.

24)

SELECT x,y,z FROM table1

UPDATE table1 SET x = 'b' WHERE x = 'a'

25. Постоянное HTTP-соединение — использование одного TCP-соединения для отправки и получения множественных HTTP-запросов и ответов вместо открытия нового соединения для каждой пары запрос-ответ.

Идея постоянных подключений состоит в том, чтобы соединение между клиентским процессом и базой данных можно было использовать повторно, особенно когда требуется создавать и закрывать соединения множество раз. Это бы позволило снизить накладные расходы на создание новых подключений каждый раз, когда они требуются, за счёт использования существующих кешированных подключений, свободных для повторного использования.

В модуле есть встроенный функционал, осуществляющий очистку соединений и переводящий их в состояние пригодное для использования. Код очистки, реализованный в mysqli включает следующие операции:

· Откат активных транзакций

· Закрытие и удаление временных таблиц

· Снятие блокировки с таблиц

· Сброс переменных сессии

· Закрытие подготовленных запросов (всегда происходит в PHP)

· Закрытие обработчиков

· Снятие блокировок, установленных функцией GET_LOCK()

Модуль mysqli делает очистку соединений автоматически путём вызова C-API функции mysql_change_user().

26. MongoDB — это ориентированная на документы база данных NoSQL с открытым исходным кодом, которая использует для хранения структуру JSON. Модель данных MongoDB позволяет представлять иерархические отношения, проще хранить массивы и другие более сложные структуры.

MongoDB отлично подходит для:

· кэширование данных;

· электронная коммерция, каталоги товаров, соцсети, новостные форумы и другие похожие сценарии, где много контента, в том числе видео и изображений;

· новый проект или стартап, если неизвестна итоговая структура данных или же вы точно знаете, что у вас будут слабо связанные данные без четкой схемы хранения;

· геоаналитика (обработка геопространственных данных — данных на основе местоположения);

· хранение данных с датчиков и устройств, собранных с решений интернета вещей, в том числе промышленных;

· работа с большими данными в машинном обучении;

· исследования в ритейле и других отраслях.

27. Процесс добавления новой записи в СУБД MongoBD

Все данные хранятся в бд в формате BSON, который близок к JSON, поэтому нам надо также вводить данные в этом формате. При добавлении в нее данных она автоматически создается.

Для добавления в коллекцию могут использоваться три ее метода:

· insertOne(): добавляет один документ

Все поля в документе представляют из себя набор пар ключ, значение.

Для хранения публикаций коллекция будет содержать поля title, description, rate, languages. Таким образом title, description, rate, languages - будут являться ключами.

Добавим публикацию с заголовком, описанием, рейтингом и списками языков, относящимися к данной публикации:

db.posts.insertOne({" title": "Заголовок публикации", "description": "Какое то описание публикации", rate: 20, languages: ["Английский", "Русский", "Итальянский"]})

· insertMany(): добавляет несколько документов

Добавим еще 2 демонстративные публикации.

Для этого передадим в insertMany массив, в котором перечислены добавляемые записи через запятую.

db.posts.insertMany([{" title": "Заголовок второй публикации", "description": "Описание второй публикации", rate: 1, languages: ["Английский", "Португальский", "Итальянский"]},{" title": "Заголовок третьей публикации", "description": "Описание третьей публикации", rate: 22, languages: ["Русский"]}])

· insert(): может добавлять как один, так и несколько документов.

Добавление одной записи:

b.posts.insert({" title": "Заголовок для insert метода", "description": "Описание для публикации с insert методом", rate: 20, languages: ["Английский", "Русский", "Португальский", "Польский"]})

Добавление нескольких записей:

db.posts.insert([{" title": "Заголовок для insert метода", "description": "Описание для публикации с insert методом", rate: 44, languages: ["Английский", "Русский", "Португальский", "Польский"]},{" title": "Новый заголовок для insert метода", "description": "Новое описание для публикации с insert методом", rate: 30, languages: ["Французский", "Немецкий"]}])

ПР6

1. Composer - это менеджер для подключения и управления этими сторонними библиотеками или пакетами в PHP-проекте. Еще его называют пакетный менеджер. Composer управляет этими пакетами, чтобы они подключились и хорошо работали.

2. Календарь:

· cal_days_in_month — Возвращает количество дней в месяце для заданного года и календаря

· cal_from_jd — Преобразует дату, заданную в юлианском календаре, в дату указанного календаря

· cal_info — Возвращает информацию о заданном календаре

· cal_to_jd — Преобразует заданную дату в юлианскую

· easter_date — Получить метку времени Unix, соответствующую полуночи на Пасху в заданном году

· easter_days — Получить количество дней между 21 марта и Пасхой в заданном году

· frenchtojd — Преобразует дату Французского республиканского календаря в

количество дней в Юлианском летоисчислении

· gregoriantojd — Преобразует дату по григорианскому календарю в количество дней в юлианском летоисчислении

· jddayofweek — Возвращает день недели

· jdmonthname — Возвращает название месяца

· jdtofrench — Переводит число дней в юлианском летоисчислении в дату по французскому республиканскому календарю

· jdtogregorian — Переводит число дней в юлианском летоисчислении в дату по Григорианскому календарю

· jdtojewish — Переводит количество дней из юлианского календаря в дату по еврейскому календарю

· jdtojulian — Переводит число дней в юлианском летоисчислении в дату по юлианскому календарю

· jdtounix — Переводит число дней в юлианском летоисчислении в метку времени Unix

· jewishtojd — Переводит дату по еврейскому календарю в число дней в юлианском летоисчислении

· juliantojd — Переводит дату по юлианскому календарю в число дней в юлианском летоисчислении

· unixtojd — Переводит метку времени Unix в юлианский день

3. Серверное время - это время региона (часового пояса), где находится сервер.

Dater — определяет часовой пояс, локализует и форматирует время в PHP.

Dater, и его основные возможности:

· Биндинг форматов

· Локализация текстов и форматов

· Расширение списка опций форматирования

· Автоопределение часового пояса

· Конвертация времени с учётом часового пояса

· Автоматическая конвертация времени в $_GET, $_POST, $_REQUEST с учётом часового пояса

· Автоматическая конвертация часового пояса в шаблоне отправляемых данных

4. Опишите способ получения времени заката и рассвета с использованием языка программирования PHP:

date_sunrise — Возвращает время рассвета для заданных дня и местоположения

date_sunset — Возвращает время захода солнца для заданных дня и местоположения

5. Опишите константы, используемые в модуле “Время и дата”:

SUNFUNCS_RET_TIMESTAMP (int)

Время в секундах с начала эпохи Unix

SUNFUNCS_RET_STRING (int)

Часы:минуты (например: 08:02)

SUNFUNCS_RET_DOUBLE (int)

Часы как число с плавающей точкой (например: 8.75)

DateTime::ATOM

DATE_ATOM

Atom (пример: 2005-08-15T15:52:01+00:00)

DateTime::COOKIE

DATE_COOKIE

HTTP Cookies (пример: Monday, 15-Aug-05 15:52:01 UTC)

DateTime::ISO8601

DATE_ISO8601

ISO-8601 (пример: 2005-08-15T15:52:01+0000)

Замечание: Этот формат не совместим с ISO-8601, но остаётся для обратной совместимости. Вместо него используйте DateTime::ATOM или DATE_ATOM для совместимости с ISO-8601.

DateTime::RFC822

DATE_RFC822

RFC 822 (пример: Mon, 15 Aug 05 15:52:01 +0000)

DateTime::RFC850

DATE_RFC850

RFC 850 (пример: Monday, 15-Aug-05 15:52:01 UTC)

DateTime::RFC1036

DATE_RFC1036

RFC 1036 (пример: Mon, 15 Aug 05 15:52:01 +0000)

DateTime::RFC1123

DATE_RFC1123

RFC 1123 (пример: Mon, 15 Aug 2005 15:52:01 +0000)

DateTimeInterface::RFC7231

DATE_RFC7231

RFC 7231 (с версии PHP 7.0.19 и 7.1.5) (пример: Sat, 30 Apr 2016 17:52:13 GMT)

DateTime::RFC2822

DATE_RFC2822

RFC 2822 (пример: Mon, 15 Aug 2005 15:52:01 +0000)

DateTime::RFC3339

DATE_RFC3339

Тоже, что и DATE_ATOM

DateTime::RFC3339_EXTENDED

DATE_RFC3339_EXTENDED

Формат RFC 3339 EXTENDED (пример: 2005-08-15T15:52:01.000+00:00)

DateTime::RSS

DATE_RSS

RSS (пример: Mon, 15 Aug 2005 15:52:01 +0000)

DateTime::W3C

DATE_W3C

World Wide Web Consortium (пример: 2005-08-15T15:52:01+00:00)

6. Приведите принципы арифметики даты и времени

Пример #1 DateTimeImmutable::add/sub добавляет интервалы, охватывающие прошедшее время

Добавление PT24H через переход DST приведёт к добавлению 23/25 часов (для большинства часовых поясов).

```
<?php $dt = new DateTimeImmutable("2015-11-01 00:00:00", new DateTimeZone("America/New_York")); echo "Начало: ", $dt->format("Y-m-d H:i:s P"), PHP_EOL; $dt = $dt->add(new DateInterval("PT3H")); echo "Конец: ", $dt->format("Y-m-d H:i:s P"), PHP_EOL; ?>
```

Результат выполнения данного примера:

```
Начало: 2015-11-01 00:00:00 -04:00
Конец: 2015-11-01 02:00:00 -05:00
```

Пример #2 DateTimeImmutable::modify и strtotime увеличит или уменьшит значения индивидуальных компонентов

Добавление +24 часов через переход DST добавит точно 24 часов (вместо учёта перехода на зимнее или летнее время).

```
<?php $dt = new DateTimeImmutable("2015-11-01 00:00:00", new DateTimeZone("America/New_York")); echo "Начало: ", $dt->format("Y-m-d H:i:s P"), PHP_EOL; $dt = $dt->modify("+24 hours"); echo "Конец: ", $dt->format("Y-m-d H:i:s P"), PHP_EOL; ?>
```

Результат выполнения данного примера:

```
Начало: 2015-11-01 00:00:00 -04:00
Конец: 2015-11-02 00:00:00 -05:00
```

Пример #3 Добавление или вычитание времени может уменьшить или увеличить дату

Например, 31 января + 1 месяц вернёт 2 марта (високосный год) или 3 марта (обычный год).

```
<?php echo "Обычный год:\n"; // В феврале 28 дней $dt = new DateTimeImmutable("2015-01-31 00:00:00", new DateTimeZone("America/New_York")); echo "Начало: ", $dt->format("Y-m-d
```

H:i:s P"), PHP_EOL; $dt = $dt->modify("+1 month"); echo "Конец: ", $dt->format("Y-m-d H:i:s P"), PHP_EOL; echo "Високосный год:\n"; // В феврале 29 дней $dt = new DateTimeImmutable("2016-01-31 00:00:00", new DateTimeZone("America/New_York"));
echo "Начало: ", $dt->format("Y-m-d H:i:s P"), PHP_EOL; $dt = $dt->modify("+1 month"); echo "Конец: ", $dt->format("Y-m-d H:i:s P"), PHP_EOL; ?>

Результат выполнения данного примера:

Обычный год:

Начало: 2015-01-31 00:00:00 -05:00

Конец: 2015-03-03 00:00:00 -05:00

Високосный год:

Начало: 2016-01-31 00:00:00 -05:00

Конец: 2016-03-02 00:00:00 -05:00

Для получения последнего дня следующего месяца (то есть чтобы предотвратить переполнение) существует директива last day of.

<?php echo "Обычный год:\n"; // Февраль содержит 28 дней $dt = new DateTimeImmutable("2015-01-31 00:00:00", new DateTimeZone("America/New_York")); echo "Начало: ", $dt->format("Y-m-d H:i:s P"), PHP_EOL; $dt = $dt->modify("last day of next month"); echo "Конец: ", $dt->format("Y-m-d H:i:s P"), PHP_EOL; echo "Високосный год:\n"; // Февраль содержит 29 дней $dt = new DateTimeImmutable("2016-01-31 00:00:00", new DateTimeZone("America/New_York")); echo "Начало: ", $dt->format("Y-m-d H:i:s P"), PHP_EOL; $dt = $dt->modify("last day of next month"); echo "Конец: ", $dt->format("Y-m-d H:i:s P"), PHP_EOL; ?>

Результат выполнения данного примера:

Обычный год:

Начало: 2015-01-31 00:00:00 -05:00

Конец: 2015-02-28 00:00:00 -05:00

Високосный год:

Начало: 2016-01-31 00:00:00 -05:00

Конец: 2016-02-29 00:00:00 -05:00

7. Чем отличается фреймворк от библиотеки? Приведите пример

В отличие от библиотеки, которая объединяет в себе набор близкой функциональности, — «фреймворк» может содержать в себе большое число разных по тематике библиотек.

Другим ключевым отличием «фреймворка» от библиотеки может быть инверсия управления: пользовательский код вызывает функции библиотеки (или классы) и получает управление после вызова.

8. Опишите возможные форматы даты и времени и примеры их использования

Данные даты (date)

Дата при использовании типа date определяется в формате "ГГГГ-ММ-ДД", где:

ГГГГ – год; ММ — месяц; ДД – день

Данные времени (time)

Время определяется в формате "чч:мм:сс", где:

чч – часы; мм – минуты; сс – секунды

Тип данных dateTime

Тип данных dateTime используется для определения даты и времени.

Значения типа dateTime имеют формат "ГГГГ-ММ-ДДТчч:мм:сс", где:

ГГГГ – год; ММ – месяц; ДД – день; Т — указывает на начало данных времени; чч – час; мм – минуты; сс – секунды

Данные о продолжительности

Типы данных о продолжительности используются для определения интервалов времени.

Интервал времени определяется в формате "PnYnMnDTnHnMnS", где:

P указывает период (обязателен); nY указывает число лет; nM указывает число месяцев; nD указывает число дней; T указывает на начало раздела с временем (обязателен, если будут определяться часы, минуты или секунды); nH указывает количество часов; nM указывает количество минут; nS указывает количество

секунд.

9. Опишите работу с датой и временем в подходе ООП.

PHP предоставляет специализированный класс DateTime для работы с датой и временем.

Вот несколько причин, почему предпочтительнее использовать класс DateTime вместо strtotime и date:

Класс DateTime может работать с большим числом форматов даты и времени по сравнению с strtotime.

Работать с объектами легче, чем с функциями. Даты, являющиеся объектами класса DateTime можно сравнивать напрямую, как обычные числа. Тогда как для сравнения двух дат с помощью функции strtotime нам необходимо сначала преобразовать их во временные метки и только затем сравнить.

Объектно-ориентированный интерфейс DateTime скрывает много внутренней логики работы с датой и обеспечивает понятный и однозначный интерфейс.

10. Опишите использование класса DateTime

Класс DateTime может работать с большим числом форматов даты и времени. Даты, являющиеся объектами класса DateTime можно сравнивать напрямую, как обычные числа. Объектно-ориентированный интерфейс DateTime скрывает много внутренней логики работы с датой и обеспечивает понятный и однозначный интерфейс.

DateTime::add — Добавляет заданное количество дней, месяцев, лет, часов, минут и секунд к объекту DateTime

DateTime::__construct — Конструктор класса DateTime

DateTime::createFromFormat — Разбирает строку с датой согласно указанному формату

DateTime::createFromImmutable — Возвращает объект DateTime инкапсулирующий заданный объект DateTimeImmutable

DateTime::createFromInterface — Возвращает новый объект DateTime, созданный из переданного объекта, реализующего интерфейс DateTimeInterface

DateTime::getLastErrors — Возвращает предупреждения и ошибки

DateTime::modify — Изменение временной метки

DateTime::__set_state — Обработчик __set_state

DateTime::setDate — Устанавливает дату

DateTime::setISODate — Устанавливает дату в формате ISO

DateTime::setTime — Устанавливает время

DateTime::setTimestamp — Устанавливает дату и время на основе метки времени Unix

DateTime::setTimezone — Устанавливает часовой пояс для объекта класса DateTime

DateTime::sub — Вычитает заданное количество дней, месяцев, лет, часов, минут и секунд из времени объекта DateTime

Пример: $now = new DateTime();

11. Опишите использование класса DateTimeImmutable

Представление даты и времени. Данный класс ведет себя аналогично классу DateTime, за исключением того, что он никогда не изменяет себя и всегда возвращает новый объект.

DateTime::diff — Возвращает разницу между двумя объектами DateTime

DateTime::format — Возвращает дату, отформатированную согласно переданному формату

DateTime::getOffset — Возвращает смещение часового пояса

DateTime::getTimestamp — Возвращает временную метку Unix

DateTime::getTimezone — Возвращает часовой пояс относительно текущего значения DateTime

DateTime::__wakeup — Обработчик __wakeup

Пример: $date = new DateTimeImmutable();

12. Опишите использование класса DateTimeZone

Представление часового пояса.

DateTimeZone :: __ construct - Создает новый объект DateTimeZone

DateTimeZone :: getLocation - Возвращает информацию о местоположении для часового пояса

DateTimeZone :: getName - Возвращает название часового пояса

DateTimeZone :: getOffset - Возвращает смещение часового пояса от GMT

DateTimeZone :: getTransitions - Возвращает все переходы для часового пояса

DateTimeZone :: listAbbreviations - Возвращает ассоциативный массив, содержащий dst, смещение и имя часового пояса

DateTimeZone :: listIdentifiers - Возвращает числовой индексный массив со всеми идентификаторами часовых поясов.

const integer DateTimeZone::AFRICA = 1 ;

const integer DateTimeZone::AMERICA = 2 ;

const integer DateTimeZone::ANTARCTICA = 4 ;

const integer DateTimeZone::ARCTIC = 8 ;

const integer DateTimeZone::ASIA = 16 ;

const integer DateTimeZone::ATLANTIC = 32 ;

const integer DateTimeZone::AUSTRALIA = 64 ;

const integer DateTimeZone::EUROPE = 128 ;

const integer DateTimeZone::INDIAN = 256 ;

const integer DateTimeZone::PACIFIC = 512 ;

const integer DateTimeZone::UTC = 1024 ;

const integer DateTimeZone::ALL = 2047 ;

const integer DateTimeZone::ALL_WITH_BC = 4095 ;

const integer DateTimeZone::PER_COUNTRY = 4096 ;

Пример: date_default_timezone_set(«Europe/Oslo»);

13.Какие библиотеки используются для работы с изображениями в PHP.

Imagine, Php Graphic Works, Zebra Image, PhpThumb,GD.

14.Опишите основные возможности библиотеки GD.

Библиотека GD, позволяет создавать новые изображения, редактировать уже существующие, копировать одни изображения на другие, изменять размеры, а также наносить текст на изображения.

15.Как использовать Composer для подключения библиотек к проекту?

Composer упрощает не только установку библиотек, но и их использование. Он берёт на себя подключение всех необходимых файлов классов библиотеки. За это отвечает специальный сценарий autoload.php.

Сценарий autoload.php — единственный файл, который необходимо подключить для использования любых библиотек

16.Что такое PEAR? В чём разница работы PEAR и Composer?

PEAR — это библиотека классов PHP с открытым исходным кодом, распространяемых через одноименный пакетный менеджер. В стандартную поставку PHP входит система управления классами PEAR, которая позволяет легко скачивать и обновлять их.

Использование PEAR более обременительно для сопровождающих пакетов. Поэтому большая часть кода на PEAR устарела. Разработчику необходимо получить пакет "PEAR-reviewed", прежде чем он может быть опубликован на PEAR, поэтому количество доступных пакетов мало по сравнению с количеством пакетов, доступных в Composer. Кроме того, в PEAR нет возможности установить пакет для одного проекта. Все пакеты устанавливаются глобально. В Composer вы можете устанавливать пакеты для каждого проекта или глобально. Ну и еще в PEAR отсутствует управление зависимостями, что, откровенно говоря, должно быть единственной вещью, которую менеджер пакетов делает хорошо.

17.Как использовать PEAR для установки библиотек?

Сам пакетный менеджер pear не входит в состав дистрибутива PHP, поэтому необходимо, чтобы он был предварительно установлен у хостера.

Основная проблема — это чтобы php-скрипт «видел» откуда ему брать тот или иной компонент (PHP как всегда идёт «своим» путём). Для этого должна быть определена настройка include_path, например, для пользователя vasya, так:

include_path=".:/home/vasya/pear"

Перед установкой модулей PEAR следует сообщить утилите pear, что мы хотим ставить компоненты в свой домашний каталог командой:

pear create-config $HOME .pearrc

Будет создан конфигурационный файл, используемый pear в дальнейшем.

Конечно же вместо $HOME можно выбрать любое другое место, не забыв отразить это в значении include_path.

Теперь, если пакет существует в списке пакетов PEAR, можно просто устанавливать требуемые пакеты. Например:

pear install -o PEAR

установит базовый компонент системы PEAR с зависимостями.

Если пакет выложен на другом канале, вам нужно сначало сделать discover этого канала и затем указать его во время установки.

18.Как использовать Composer для обработки зависимостей PEAR?

Если вы уже используете Composer и желаете установить какой-то код из PEAR, вы можете использовать Composer для обработки зависимостей PEAR. Этот пример установит код из pear2.php.net:

```
{

"repositories": [

{

"type": "pear",

"url": "http://pear2.php.net"

}

],

"require": {

"pear-pear2/PEAR2_Text_Markdown": "*",

"pear-pear2/PEAR2_HTTP_Request": "*"

}

}
```

Первый раздел "repositories" даст понять Composer, что он должен сделать "initialise" (или "discover" в терминологии PEAR) репозиторий pear. Затем секция require укажет именам пакетов префикс, как ниже:

pear-channel/Package

Префикс "pear" жестко ограничен, чтобы избежать любых конфликтов, так как каналы Pear могут быть схожи с другими поставщиками пакетов,

например, вместо короткого имени (или полного URL) может быть использовано для объявления в каком канале находится пакет.

Когда код будет установлен он будет доступен в вашей папке vendor и автоматически доступен через автозагрузчик (файл Autoload) Composer.

vendor/pear-pear2.php.net/PEAR2_HTTP_Request/pear2/HTTP/Request.php

Чтобы использовать этот пакет PEAR просто объявите, как ниже:

$request = new pear2\HTTP\Request();

19. Что такое PECL?

PECL - это репозиторий нативных расширений, написанных на C. Обычно из используют, когда что-то нельзя реализовать на голом PHP, например перегрузку функций или операторов. 20. Как декодировать строку JSON и вернуть JSON-представление данных?

Метод JSONDecoder.raw_decode() декодирует JSON-документ из строки s в формате JSON и возвращает двойной кортеж представление данной строки в Python и индекс в строки s, где документ закончился. Метод JSONDecoder.raw_decode() может быть использован
для декодирования документа JSON из строки, которая в конце содержит посторонние данные.

20. Как называется репозиторий, содержащий Composer-совместимые библиотеки?

21. С помощью какой библиотеки можно получить детальный отчет о работе приложения?

PHP Benchmark

\PHPBenchmark\Monitor::instance()->snapshot('Plugins loaded');

22. С помощью какой библиотеки можно упростить себе работу с регулярными выражениями в PHP?

PCRE

23. С помощью какой библиотеки возможны быстрые и эффективные запросы на PHP, данная библиотека является аналогом jQuery.

Simple HTML DOM — PHP-библиотека, позволяющая парсить HTML-код с помощью удобных jQuery-подобных селекторов. Она лишена главного недостатка XPath — библиотека умеет работать даже с невалидным HTML-кодом, что значительно упрощает работу.

24. С помощью какой библиотеки возможно простое и эффективное генерирование документов в формате PDF?

С помощью какой библиотеки возможно простое и эффективное генерирование документов в формате PDF

pdftk, DOMPDF

25. С помощью какой библиотеки возможен эффективный парсинг HTML/XML?

С помощью какой библиотеки возможен эффективный парсинг HTML/XML

phpQuery, PHP Simple HTML DOM Parser, Nokogiri.

26. С помощью какой библиотеки возможно создание диаграмм на движке GOOGLE?

С помощью какой библиотеки возможно сканирование конфигурационного файла PHP на предмет безопасности

SaltStack

27. С помощью какой библиотеки возможно сканирование конфигурационного файла PHP на предмет безопасности?

RIPS

28. С помощью какой библиотеки возможен анализ HTML и удаление вредоносного кода для защиты от XSS атак. HTMLPurifier

29. С помощью какой библиотеки возможно построение графиков,

диаграмм и другого структурированного контента на PHP?

PChart — это PHP-библиотека для создания графиков, гистограмм и диаграмм.

30. С помощью какой библиотеки облегчается процесс загрузки и

валидации файлов на PHP? Upload

ПР7

1. Назовите основные признаки ООП.

Абстракция – это способ придания объектам определённых характеристик, которые отличают его от всех остальных объектов.
Инкапсуляция – один из принципов ООП, позволяющее объединить данные и методы, работающие с ними, в классе.
Наследование – один из принципов ООП, позволяющий описать новый класс на основе уже существующего с частично или полностью заимствованной функциональностью.
Полиморфизм – один из принципов ООП, позволяющий использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

2. Опишите как определить класс в PHP.

Для примера представлен класс машины.

```
<?php
class Car {
   public $brand;
   public $color;
   public function __construct($brand, $color = "blue")
   {
      $this->brand = $brand;
      $this->color = $color;
   }
   public function getBrand()
   {
      return $this->brand;
   }
   public function getColor()
   {
      return $this->color;
   }
}
?>
```

3. Как создать экземпляр класса в PHP.

Создадим экземпляр класса машины. Ниже представлен пример.
$car = new Car(“Mercedes”, “black”);

4. Опишите механизм наследования в PHP.

Ниже приведён примера наследования в PHP.

```
<?php
class A {
	// code here
}
class B extends A {
	// code here
}
class C extends B {
	// code here
}
?>
```

5. Опишите правила совместимости сигнатур.

При переопределении метода его сигнатура должна быть совместима с родительским методом. В противном случае выдаётся фатальная ошибка или, до PHP 8.0.0, генерируется ошибка уровня E_WARNING. Сигнатура является совместимой, если она соответствует правилам контравариантности, делает обязательный параметр необязательным и если какие-либо новые параметры являются необязательными. Это известно как принцип подстановки Барбары Лисков или сокращённо LSP. Правила совместимости не распространяются на конструктор и сигнатуру private методов, они не будут выдавать фатальную ошибку в случае несоответствия сигнатуры. Ниже представлен пример.

```
<?php
class Base
{
   public function foo(int $a) {
      echo "Допустимо\n";
   }
}
class Extend1 extends Base
{
   function foo(int $a = 5)
   {
      parent::foo($a);
   }
}
class Extend2 extends Base
{
   function foo(int $a, $b = 5)
   {
      parent::foo($a);
   }
}
$extended1 = new Extend1();
$extended1->foo();
$extended2 = new Extend2();
$extended2->foo(1);
```

6. Опишите методы и свойства Nullsafe.

Начиная с PHP 8.0.0, к свойствам и методам можно также обращаться с помощью оператора "nullsafe": ?->. Оператор nullsafe работает так же, как доступ к свойству или методу, как указано выше, за исключением того, что если разыменование объекта выдаёт null, то будет возвращён null, а не выброшено исключение. Если разыменование является частью

цепочки, остальная часть цепочки пропускается.
Аналогично заключению каждого обращения в is_null(), но более компактный. Ниже представлен пример.

```
<?php
// Начиная с PHP 8.0.0, эта строка:
$result = $repository?->getUser(5)?->name;
// Эквивалентна следующему блоку кода:
if (is_null($repository)) {
   $result = null;
} else {
   $user = $repository->getUser(5);
   if (is_null($user)) {
      $result = null;
   } else {
      $result = $user->name;
   }
}
?>
```

7. Опишите понятие автоматическая загрузка классов.

Большинство разработчиков объектно-ориентированных приложений используют такое соглашение именования файлов, в котором каждый класс хранится в отдельно созданном для него файле. Одна из самых больших неприятностей - необходимость писать в начале каждого скрипта длинный список подгружаемых файлов (по одному для каждого класса).
Функция spl_autoload_register() позволяет зарегистрировать необходимое количество автозагрузчиков для автоматической загрузки классов и интерфейсов, если они в настоящее время не определены. Регистрируя автозагрузчики, PHP получает последний шанс для интерпретатора загрузить класс прежде, чем он закончит выполнение скрипта с ошибкой. Ниже представлен пример автоматической загрузки.

```
<?php
spl_autoload_register(function ($class_name) {
   include $class_name . '.php';
});
$obj = new MyClass1();
$obj2 = new MyClass2();
?>
```

8. Опишите конструкторы и деструкторы в PHP.

PHP позволяет объявлять методы-конструкторы. Классы, в которых объявлен метод-конструктор, будут вызывать этот метод при каждом создании нового объекта, так что это может оказаться полезным, например, для инициализации какого-либо состояния объекта перед его использованием. Конструктор задаётся следующим образом.

```
__construct(mixed ...$values = ""): void
```

PHP предоставляет концепцию деструктора, аналогичную с той, которая применяется в других ОО-языках, таких как C++. Деструктор будет вызван при освобождении всех ссылок на определённый объект или при завершении скрипта (порядок выполнения деструкторов не гарантируется).

```
__destruct(): void
```

9. Опишите понятие области видимости и модификаторы доступа в PHP.

Область видимости свойства, метода или константы (начиная с PHP 7.1.0) может быть

определена путём использования следующих ключевых слов в объявлении: public, protected или private. Доступ к свойствам и методам класса, объявленным как public (общедоступный), разрешён отовсюду. Модификатор protected (защищённый) разрешает доступ самому классу, наследующим его классам и родительским классам. Модификатор private (закрытый) ограничивает область видимости так, что только класс, где объявлен сам элемент, имеет к нему доступ.

10. Опишите оператор разрешения области видимости.

Оператор разрешения области видимости (также называемый "Paamayim Nekudotayim") или просто "двойное двоеточие" — это лексема, позволяющая обращаться к статическим свойствам, константам и переопределённым свойствам или методам класса. При обращении к этим элементам извне класса, необходимо использовать имя этого класса.

11. Опишите абстрактные классы и методы в PHP.

PHP поддерживает определение абстрактных классов и методов. На основе абстрактного класса нельзя создавать объекты, и любой класс, содержащий хотя бы один абстрактный метод, должен быть определён как абстрактный. Методы, объявленные абстрактными, несут, по существу, лишь описательный смысл и не могут включать реализацию.
При наследовании от абстрактного класса, все методы, помеченные абстрактными в родительском классе, должны быть определены в дочернем классе и следовать обычным правилам наследования и совместимости сигнатуры.
Ниже представлен пример наследования от абстрактного класса на PHP.

```
<?php
abstract class AbstractClass
{
  /* Данный метод должен быть определён в дочернем классе */
   abstract protected function getValue();
   abstract protected function prefixValue($prefix);

  /* Общий метод */
   public function printOut() {
      print $this->getValue() . "\n";
   }
}
class ConcreteClass1 extends AbstractClass
{
   protected function getValue() {
      return "ConcreteClass1";
   }
   public function prefixValue($prefix) {
      return "{$prefix}ConcreteClass1";
   }
}
?>
```

12. Опишите интерфейсы в PHP.

Интерфейсы объектов позволяют создавать код, который указывает, какие методы должен реализовать класс, без необходимости определять, как именно они должны быть реализованы. Интерфейсы разделяют пространство имён с классами и трейтами, поэтому они не могут называться одинаково.
Ниже приведён пример интерфейса и его имплементации.

```php
<?php
// Объявим интерфейс 'Template'
interface Template
{
   public function setVariable($name, $var);
   public function getHtml($template);
}

// Реализация интерфейса
// Это будет работать
class WorkingTemplate implements Template
{
   private $vars = [];
   public function setVariable($name, $var)
   {
      $this->vars[$name] = $var;
   }
   public function getHtml($template)
   {
      foreach($this->vars as $name => $value) {
         $template = str_replace('{' . $name . '}', $value, $template);
      }
      return $template;
   }
}
?>
```

13. Что такое трейт и как это используется?

Трейт — это механизм обеспечения повторного использования кода в языках с поддержкой только одиночного наследования, таких как PHP. Трейт предназначен для уменьшения некоторых ограничений одиночного наследования, позволяя разработчику повторно использовать наборы методов свободно, в нескольких независимых классах и реализованных с использованием разных архитектур построения классов. Семантика комбинации трейтов и классов определена таким образом, чтобы снизить уровень сложности, а также избежать типичных проблем, связанных с множественным наследованием и смешиванием (mixins).

14. Что такое магические методы? Приведите примеры.

Магические методы — это специальные методы, которые переопределяют действие PHP по умолчанию, когда над объектом выполняются определённые действия.
Например,__construct(), __destruct(), __call(), __callStatic(), __get(), __set(), __isset(), __unset(), __sleep(), __wakeup(), __serialize(), __unserialize(), __toString(), __invoke(), __set_state(), __clone() и __debugInfo().

15. Что такое позднее статическое связывание?

PHP реализует функцию, называемую позднее статическое связывание, которая может быть использована для того, чтобы получить ссылку на вызываемый класс в контексте статического наследования.
Если говорить более точно, позднее статическое связывание сохраняет имя класса указанного в последнем "неперенаправленном вызове". В случае статических вызовов это явно указанный класс (обычно слева от оператора ::); в случае не статических вызовов это

класс объекта. "Перенаправленный вызов" - это статический вызов, начинающийся с self::, parent::, static::, или, если двигаться вверх по иерархии классов, forward_static_call(). Функция get_called_class() может быть использована для получения строки с именем вызванного класса, а static:: представляет её область действия.
Само название "позднее статическое связывание" отражает в себе внутреннюю реализацию этой особенности. "Позднее связывание" отражает тот факт, что обращения через static:: не будут вычисляться по отношению к классу, в котором вызываемый метод определён, а будут вычисляться на основе информации в ходе исполнения. Также эта особенность была названа "статическое связывание" потому, что она может быть использована (но не обязательно) в статических методах.

16. Что такое ковариантность и контравариантность?

Ковариантность – это сохранение иерархии наследования исходных типов в производных типах в том же порядке. Ниже приведён пример ковариантности на языке PHP.

```
<?php
abstract class Animal
{
   protected string $name;
   public function __construct(string $name)
   {
      $this->name = $name;
   }
   abstract public function speak();
}
class Dog extends Animal
{
   public function speak()
   {
      echo $this->name . " лает";
   }
}
class Cat extends Animal
{
   public function speak()
   {
      echo $this->name . " мяукает";
   }
}
interface AnimalShelter
{
   public function adopt(string $name): Animal;
}
class CatShelter implements AnimalShelter
{
   public function adopt(string $name): Cat // Возвращаем класс Cat вместо Animal
   {
      return new Cat($name);
   }
}
class DogShelter implements AnimalShelter
{
   // Возвращаем класс Dog вместо Animal
```

```
    public function adopt(string $name): Dog
    {
        return new Dog($name);
    }
}
?>
```

Контравариантность – это обращение иерархии исходных типов на противоположную в производных типах. Ниже приведён пример контравариантности на языке PHP.

```
<?php
class Food {}
class AnimalFood extends Food {}
abstract class Animal
{
    protected string $name;
    public function __construct(string $name)
    {
        $this->name = $name;
    }
    public function eat(AnimalFood $food)
    {
        echo $this->name . " ест " . get_class($food);
    }
}
class Dog extends Animal
{
    public function eat(Food $food) {
        echo $this->name . " ест " . get_class($food);
    }
}
?>
```

17. Опишите понятие чистой архитектуры.

Понятие чистой архитектуры пошло из одноименной статьи Роберта Мартина 2012 года. Оно включает в себя несколько принципов:

- Независимость от фреймворков. Архитектура не должна полагаться на существование какой-либо библиотеки. Так вы сможете использовать фреймворки как инструменты, а не пытаться загнать свою систему в их ограничения;
- Тестируемость. Бизнес-логика должна быть тестируемой без любых внешних элементов вроде интерфейса, базы данных, сервера или любого другого элемента;
- Независимость от интерфейса. Интерфейс должен легко изменяться и не требовать изменения остальной системы. Например, веб-интерфейс должен заменяться на интерфейс консоли без необходимости изменения бизнес-логики;
- Независимость от базы данных. Ваша бизнес-логика не должна быть привязана и к конкретным базам данных;

- Независимость от любого внешнего агента. Ваша бизнес-логика не должна знать вообще ничего о внешнем мире.

18. Сформулируйте правило зависимостей.

Зависимости в исходном коде могут указывать только во внутрь. Ничто из внутреннего круга не может знать что-либо о внешнем круге, ничто из внутреннего круга не может указывать на внешний круг. Это касается функций, классов, переменных и т. д. Более того, структуры данных, используемых во внешнем круге, не должны быть использованы во внутреннем круге, особенно если эти структуры генерируются фреймворком во внешнем круге. Мы не используем ничего из внешнего круга, чтобы могло повлиять на внутренний.

19. Чем определяются сущности, чем они могут быть?

Сущность (entity) – это объект, который может быть идентифицирован неким способом, отличающим его от других объектов. Примеры: конкретный человек, предприятие, событие и т.д.
Сущности определяются бизнес-правилами предприятия. Сущность может быть объектом с методами или она может представлять собой набор структур данных и функций. Не имеет значения как долго сущность может быть использована в разных приложениях.
Если же вы пишете просто одиночное приложение, в этом случае сущностями являются бизнес-объекты этого приложения. Они инкапсулируют наиболее общие высокоуровневые правила. Наименее вероятно, что они изменятся при каких-либо внешних изменениях. Например, они не должны быть затронуты при изменении навигации по страницам или правил безопасности. Внешние изменения не должны влиять на слой сущностей.

20. Что такое слой сценариев?

В данном слое реализуется специфика бизнес-правил. Он инкапсулирует и реализует все случаи использования системы. Эти сценарии реализуют поток данных в и из слоя сущностей для реализации бизнес-правил.
Мы не ожидаем изменения в этом слое, влияющих на сущности. Мы также не ожидаем, что этот слой может быть затронут внешними изменениями, таких как базы данных, пользовательским интерфейсом или фреймворком. Этот слой изолирован от таких проблем. Мы, однако, ожидаем, что изменения в работе приложения повлияет на Сценарии. Если будут какие-либо изменения в поведении приложения, то они несомненно затронут код в данном слое.

21. Что такое DTO?

Data Transfer Object (DTO) — один из шаблонов проектирования, используется для передачи данных между подсистемами приложения.

22. Что является деталью в рамках чистой архитектуры?

Это части структуры, которые позволяют миру взаимодействовать с бизнес-правилами. Например: веб, фреймворк.

23. Опишите принципы организации компонентов.

Под понятием компонентов принято понимать единицы развертывания. Например, это может быть библиотека или исполняемый файл, в Java таковым назвать можно jar-файлы. Принципы организации компонентов в разработки ПО помогают сгруппировать классы в компоненты и сделать их более структурируемыми и управляемыми. Принципы разделяются на две группы: связность (component cohesion) (какие классы стоит поместить?) и сочетаемость (component coupling) компонентов (как должны

взаимодействовать друг с другом?).

Три принципа связности компонентов:

- Принцип эквивалентности повторного использования и выпусков. Единица повторного использования есть единица выпуска. Этот принцип требует, чтобы компоненты проходили процесс выпуска и получали версии. В один компонент должны объединяться классы с одной целью, которая будет выражена в очередном выпуске. По документации же пользователи и разработчики должны понимать, нужно ли им переходить на новую версию.
- Принцип согласованного изменения. В один компонент должны объединяться классы, изменяющиеся по одним причинам. Идея заключения вместе сущностей, которые могут изменяться в одно и то же время и по одним причинам, является одной из ключевых идей архитектуры ПО, поэтому она проходит красной нитью по всем структурным уровням.
- Принцип совместного повторного использования. Не вынуждайте пользователей компонента зависеть от того, чего им не требуется. Главная мысль этого принципа в объединении в компоненты тех классов, которые имеют множественные зависимости друг от друга. Кроме того, нужно избегать слабых зависимостей от других компонентов — даже зависимость от одного редко используемого класса наверняка потребует повторной компиляции и тестирования всего компонента в случае изменений в зависимом.

Теперь перейдем ко второй группе принципов. Сочетаемость компонентов:

- Принцип ацикличности зависимостей. Нельзя допускать циклов в графе зависимостей. Это позволяет разбить проект на компоненты, которые будут выпускаться независимо. При возникновении цикла между зависимостями для тестирования и выпуска новой версии придется отладить и подготовить все компоненты, входящие в цикл. Они превращаются в один большой компонент.
- Принцип устойчивых зависимостей. Зависимости должны быть направлены в сторону устойчивых компонентов. Нужно использовать ссылки на компоненты, которые будут редко меняться и избегать зависимостей от изменчивых компонентов. Это добавит гибкость в разработке, так как всегда нужны компоненты, которые можно легко изменять. Если же создать большое число зависимостей от такого компонента, эта возможность легких изменений испарится.
- Принцип устойчивости абстракций. Этот принцип проводит связь между устойчивыми и абстрактными компонентами. Компоненты, которые содержат интерфейс для высокоуровневой бизнес-логики, должны быть абстрактными и

почти не меняться. Реализации же этого интерфейса должны быть неустойчивыми — избегайте множественных зависимостей от таких компонентов.

24. Опишите принципы дизайна архитектуры.

Под принципами дизайна архитектуры понимаются SOLID принципы. Эта аббревиатура пяти основных принципов проектирования в объектно-ориентированном программировании — Single responsibility, Open-closed, Liskov substitution, Interface segregation и Dependency inversion. В переводе на русский: принципы единственной ответственности, открытости / закрытости, подстановки Барбары Лисков, разделения интерфейса и инверсии зависимостей).

- Принцип единственной обязанности / ответственности (single responsibility principle / SRP) обозначает, что каждый объект должен иметь одну обязанность и эта обязанность должна быть полностью инкапсулирована в класс. Все его сервисы должны быть направлены исключительно на обеспечение этой обязанности.
- Принцип открытости / закрытости (open-closed principle / OCP) декларирует, что программные сущности (классы, модули, функции и т. п.) должны быть открыты для расширения, но закрыты для изменения. Это означает, что эти сущности могут менять свое поведение без изменения их исходного кода.
- Принцип подстановки Барбары Лисков (Liskov substitution principle / LSP) в формулировке Роберта Мартина: «функции, которые используют базовый тип, должны иметь возможность использовать подтипы базового типа, не зная об этом».
- Принцип разделения интерфейса (interface segregation principle / ISP) в формулировке Роберта Мартина: «клиенты не должны зависеть от методов, которые они не используют». Принцип разделения интерфейсов говорит о том, что слишком «толстые» интерфейсы необходимо разделять на более маленькие и специфические, чтобы клиенты маленьких интерфейсов знали только о методах, которые необходимы им в работе. В итоге, при изменении метода интерфейса не должны меняться клиенты, которые этот метод не используют.
- Принцип инверсии зависимостей (dependency inversion principle / DIP) — модули верхних уровней не должны зависеть от модулей нижних уровней, а оба типа модулей должны зависеть от абстракций; сами абстракции не должны зависеть от деталей, а вот детали должны зависеть от абстракций.

25. Опишите понятие DDD (Domain Driven Design, предметно-ориентированное проектирование).

Предметно-ориентированное проектирование — это набор принципов и схем,

направленных на создание оптимальных систем объектов. Сводится к созданию программных абстракций, которые называются моделями предметных областей. В эти модели входит бизнес-логика, устанавливающая связь между реальными условиями области применения продукта и кодом.

26. Что такое ограниченный контекст (Bounded Context)?

Ограниченный контекст (bounded context)– это граница, которая окружает ту или иную модель. Это держит знания внутри соответствующей границы, в то же время игнорируя помехи от внешнего мира. Это выгодно по ряду причин. Во-первых, вы можете моделировать различные аспекты проблемы, не контактируя с другими частями бизнеса. Используя предыдущий пример, объект Product в рамках складской системы должен быть связан с помощью методов и свойств этой единой системы, а не какой-либо другой коммерческой фирмы, что случается, когда пытаются соответствовать объекту под названием Product. Во-вторых, терминология в ограниченном контексте (Bounded Context) может иметь одно, четкое определение, что точно описывает конкретную проблему. Различные отделы по всей компании, как правило, имеют немного разные идеи и определения аналогичных условий, это часто может сорвать проект из-за отсутствия ясности и понимания, если сроки являются неоднозначными.

27. Что такое Ubiquitous Language (Единый язык)?

Этот коллективный язык терминов называется - единый язык. (Ubiquitous Language). Это один из основных и самых важных шаблонов предметного-ориентированного проектирования. Это не бизнес-жаргон, навязанный разработчикам, а настоящий язык, созданный целостной командой – экспертами в предметной области, разработчиками, бизнес-аналитиками и всеми, кто вовлечен в создание системы. Роль в команде не столь существенна, поскольку каждый член команды использует для описания проекта единый язык.

28. Что такое Смысловое ядро (Core domain)?

Смысловое ядро – это подобласть, имеющая первостепенное значение для организации. Со стратегической точки зрения бизнес должен выделяться своим смысловым ядром. Большинство DDD проектов сосредоточены именно на смысловом ядре.

29. Что такое Предметная область (Domain)?

Множество понятий и объектов, рассматриваемых в пределах отдельного рассуждения, исследования или научной теории. Включает объекты, изучаемые теорией, а также свойства, отношения и функции, которые принимаются во внимание в теории. В анализе данных в качестве предметной области может выступать компания, в интересах которой реализуется аналитический проект, внешнее окружение, сегмент рынка и т. д. Это понятие играет большую роль в анализе данных, поскольку используемые там подходы и методы оперируют объектами и терминами предметной области и, следовательно, зависят от нее. В хранилищах данных, которые являются предметно ориентированными, под предметной областью понимают устойчивую связь между именами, понятиями и объектами внешнего мира, не зависящую от самой информационной системы и круга ее пользователей. Введение в рассмотрение понятия предметной области ограничивает и делает обозримым пространство информационного поиска в хранилище данных и позволяет выполнять за конечное время даже сложные нерегламентированные запросы.

30. Что такое пространство задач и пространство решений?

Пространство задач и пространство решений в архитектуре подразумевает под собой набор определённых задач и их решений, в зависимости от которых будет зависеть построение целой архитектуры информационной системы/приложения.

ПР8

1. Что такое фреймворк?

   Программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.
2. Что такое фреймворк и чем он отличается от библиотеки?

   «Фреймворк» отличается от понятия библиотеки тем, что библиотека может быть использована в программном продукте просто как набор подпрограмм близкой функциональности, не влияя на архитектуру программного продукта и не накладывая на неё никаких ограничений. В то время как «фреймворк» диктует правила построения архитектуры приложения, задавая на начальном этапе разработки поведение по умолчанию — «каркас», который нужно будет расширять и изменять, согласно указанным требованиям.

   Также, в отличие от библиотеки, которая объединяет в себе набор близкой функциональности, — «фреймворк» может содержать в себе большое число разных по тематике библиотек.

   Другим ключевым отличием «фреймворка» от библиотеки может быть инверсия управления: пользовательский код вызывает функции библиотеки (или классы) и получает управление после вызова. Во «фреймворке» пользовательский код может реализовывать конкретное поведение, встраиваемое в более общий — «абстрактный» код фреймворка. При этом «фреймворк» вызывает функции (классы) пользовательского кода
3. Из каких этапов состоит разработка веб-приложения?

   Планирование, проектирование, тестирование, запуск.
4. Что такое “Гибкий” фреймворк?

Фреймворк, следующий гибкой методологии разработки.

5. Что такое “не гибкий” фреймворк?

   Фреймворк, не следующий гибкой методологии разработки.

6. Чем “гибкий” фреймворк отличается от “не гибкого” фреймворка?

   В основном – разные жизненные циклы приложения. Большая возможность соответствия декларируемым в agile-манифесте принципам.

7. Опишите фреймворк Laravel.

   Бесплатный веб-фреймворк с открытым кодом, предназначенный для разработки с использованием архитектурной модели MVC.

8. Какие приложения можно создавать с помощью Laravel?

   С помощью Laravel, как и на базе любого другого фреймворка, можно делать абсолютно разные типы сайтов, начиная с лендингов и заканчивая социальными сетями.

9. Что такое Eloquent ORM?

   Система объектно-реляционного отображения (ORM) Eloquent — красивая и простая реализация шаблона ActiveRecord в Laravel для работы с базами данных. Каждая таблица имеет соответствующий класс-модель, который используется для работы с этой таблицей. Модели позволяют запрашивать данные из таблиц, а также вставлять в них новые записи.

10. Что такое сервис-провайдеры (service providers)?

    Сервис-провайдеры лежат в основе первоначальной загрузки всех приложений на Laravel. И ваше приложение, и все базовые сервисы Laravel загружаются через сервис-провайдеры.

    Но что мы понимаем под "первоначальной загрузкой"? В общих чертах, мы имеем ввиду регистрацию таких вещей, как биндингов в IoC-контейнер (фасадов и т.д.), слушателей событий, фильтров

роутов и даже самих роутов. Сервис-провайдеры - центральное место для конфигурирования вашего приложения.

Если вы откроете файл config/app.php, поставляемый с Laravel, то увидите массив providers. В нём перечислены все классы сервис-провайдеров, которые загружаются для вашего приложения. Конечно, многие из них являются "отложенными" провайдерами, т.е. они не загружаются при каждом запросе, а только при необходимости.

11. Что такое Сервис-контейнер в Laravel?

    Сервис-контейнер в Laravel — это мощное средство для управления зависимостями классов и внедрения зависимостей. Внедрение зависимостей — это распространенный термин, который означает добавление других классов в этот класс через конструктор или, в некоторых случаях, метод-сеттер.

12. Что такое Контракты в Laravel?

    Контракты в Laravel — это набор интерфейсов, которые описывают основной функционал, предоставляемый фреймворком. Например, контракт Illuminate\Contracts\Queue\Queue определяет методы, необходимые для организации очередей, в то время как контракт Illuminate\Contracts\Mail\Mailer определяет методы, необходимые для отправки электронной почты.

13. Что такое Фасады в Laravel?

    Фасады предоставляют "статический" интерфейс к классам, доступным в сервис-контейнере. Laravel поставляется со множеством фасадов, которые предоставляют доступ практически ко всем функциям Laravel. Фасады Laravel служат "статическими прокси" для основополагающих классов в сервис-контейнере, предоставляя преимущество лаконичного, выразительного синтаксиса, сохраняя

при этом большую тестируемость и гибкость по сравнению с обычными статическими методами.

14. Опишите структуру Laravel-приложения по умолчанию.
    - Корневой каталог
        - Каталог app
        - Каталог bootstrap
        - Каталог config
        - Каталог database
        - Каталог public
        - Каталог resources
        - Каталог routes
        - Каталог storage
        - Каталог tests
        - Каталог vendor
    - Каталог пространства App
        - Каталог пространства Broadcasting
        - Каталог пространства Console
        - Каталог пространства Events
        - Каталог пространства Exceptions
        - Каталог пространства Http
        - Каталог пространства Jobs
        - Каталог пространства Listeners
        - Каталог пространства Mail
        - Каталог пространства Models
        - Каталог пространства Notifications
        - Каталог пространства Policies
        - Каталог пространства Providers
        - Каталог пространства Rules
15. Опишите преимущества фреймворка Laravel.

    Достаточно неплохая и понятная документация.

Вокруг фреймворка создана мощная экосистема. Различные курсы, конференции, обучающие материалы позволяют собрать вокруг фреймворка большое количество разработчиков и спонсоров, которые заинтересованы в развитии инструмента и принимают в этом участие. Да, здесь чувствуется запах маркетинга, и неплохой.

Одним из самых очевидных плюсов Laravel, является гибкая система маршрутизации, позволяющая составить самые разные проверки маршрута веб-приложения. Вы можете выделить маршруты в специальные группы, использовать пространство имен, указать параметры маршрута, использовать регулярные выражения, настроить поддоменную маршрутизацию и многое другое.

В Laravel много синтаксического сахара. Синтаксис API фреймворка достаточно простой и понятный. Здесь нет длинных и сложных конструкций, а только краткие и продуманные названия функций.

Laravel содержит удобный механизм обработки ошибок и исключений.

Фреймворк включает в себя встроенные механизмы аутентификации и авторизации пользователей, которую можно перенастроить под свои потребности.

Laravel предоставляет из коробки механизмы для кэширования веб-приложения с помощью Memcached и Redis. Кроме этого есть удобные функции для использования простого файлового кэширования данных.

Laravel предоставляет чистый и простой API поверх популярной библиотеки SwiftMailer с драйверами для SMTP, Mailgun, SparkPost, Amazon SES и sendmail, чтобы сделать отправку почты через локальную или облачную службу по выбору. В том числе есть механизм для построения очередей отправки почты.

Laravel Cashier обеспечивает выразительный, свободный интерфейс к сервисам биллинга по подписке Stripe и Braintree.

16. Опишите недостатки фреймворка Laravel.

Синтаксический сахар в Laravel как плюс, так может быть и минусом. Очень легко привыкнуть к нему и позабыть, как пишутся чистые запросы и функции.

Нарушение обратная совместимости между версиями фреймворка.

Не логичное расположение каталогов и файлов. Например, по умолчанию в прямо в каталоге /app расположена модель User.php, которую логичней было бы расположить в каталоге /app/Models. Каталог resources с файлами представления размещен в корне приложения, хотя логичней будет его разместить в /app/resources.

17. Что включает в себя фреймворк Laravel.

Laravel включает в себя встроенную поддержку для аутентификации, локализации, модели, представления, сессий, маршрутизации и других механизмов, поддержку контроллеров, которые сделали фреймворк полностью MVC-совместимым, встроенную поддержку для инверсии управления и шаблонизатор Blade, интерфейс командной строки (CLI) под именем «Artisan», встроенную поддержку нескольких систем управления базами данных, миграции баз данных в виде контроля версий, обработку событий, выгрузка таблиц базы данных для первоначальной популяции, поддержку очередей сообщений, встроенную поддержку отправки различных типов электронной почты и поддержку «мягкого» удаления записей базы данных, поддержку планирования периодически выполняемых задач через пакет Scheduler, слой абстракции Flysystem, который позволяет использовать удаленное хранилище так же, как и локальные файловые системы, улучшенную обработку активов пакета через Elixir и упрощенная аутентификация с внешней стороны через дополнительный пакет Socialite, Laravel Dusk, Laravel Mix, Blade Components и Slots, Markdown Emails, автоматические фасады, улучшения маршрута.

18. Опишите из чего состоит экосистема Laravel.

    PHP, Composer, Dotenv, PSR-4, Eloquent ORM, Flysystem, Elixir, HHVM, Homestead, Rocketeer

19. Опишите фреймворк Symfony.

    Свободный PHP фреймворк для быстрой разработки веб-приложений и решения рутинных задач веб-программистов. Разработка и поддержка фреймворка спонсируется французской компанией Sensio.

    Symfony состоит из набора не связанных между собой компонентов, которые можно использовать повторно в проектах.

    Symfony позволяет устанавливать сторонние пакеты, библиотеки, компоненты и настраивать их с помощью конфигурации в форматах YAML, XML, PHP, а также .env файлах.

    Symfony не обеспечивает компонент для работы с базой данных, но обеспечивает тесную интеграцию с библиотекой Doctrine.

    Symfony предоставляет функцию почтовой программы на основе популярной библиотеки Swift Mailer. Эта почтовая программа поддерживает отправку сообщений с ваших собственных почтовых серверов, а также с использованием популярных почтовых провайдеров, таких как Mandrill, SendGrid и Amazon SES.

    Механизм интернационализации позволяет установить и произвести перевод сообщений веб-приложения на основе выбранного языка или страны.

    Symfony предлагает систему логирования ошибок приложения, а также подключить библиотеку логирования Monolog.

20. Опишите преимущества фреймворка Symfony.

    Мощная экосистема вокруг фреймворка, с хорошим сообществом и множеством разработчиков.

    Хорошая и постоянно обновляемая документация для всех версий фреймворка.

Множество различных не связанных компонентов для повторного использования.

Предлагает механизм функциональных и модульных тестов для нахождения ошибок в веб-приложении.

Подходит для сложных и нагруженных веб-проектов.электронной коммерции.

21. Опишите недостатки фреймворка Symfony.

    Несмотря на хорошую документацию, фреймворк является сложным для изучения.

22. Опишите фреймворк Code Igniter.

    Это популярный PHP микро-фреймворк с открытым исходным кодом, для разработки веб-систем и приложений.

    В CodeIgniter компоненты загружаются и процедуры выполняются только по запросу, а не глобально. Система не делает никаких предположений относительно того, что может потребоваться помимо минимальных основных ресурсов, поэтому система по умолчанию очень легкая.

    Компоненты фреймворка слабо связаны между собой и не зависят друг от друга. Чем меньше компонентов зависит друг от друга, тем более гибкой и многоразовой становится система.

    Хотя CodeIgniter работает довольно быстро, объем динамической информации, отображаемой на страницах, будет напрямую зависеть от используемых ресурсов сервера, памяти и циклов обработки, которые влияют на скорость загрузки страниц.

    Поэтому CodeIgniter позволяет кэшировать страницы для достижения максимальной производительности. с помощью встроенного компонента кэширования.

23. Опишите преимущества фреймворка Code Igniter.

    Отличная документация и англоязычное сообщество.

    Высокая производительность фреймворка.

Небольшой размер фреймворка.

Предоставляет легкие и простые решения для разработки.

Подходит для быстрой разработки небольших сайтов и веб-приложений.

Структура фреймворка не требует строгих правил кодирования.

Не требует сложной настройки, почти нулевая конфигурация.

MVC-архитектура веб-приложения.

Слабая связанность компонентов.

Множество подключаемых библиотек и помощников.

24. Опишите недостатки фреймворка Code Igniter.

Задержка в развитии и переходе на новые технологии.

25. Опишите фреймворк Yii2.

Объектно-ориентированный компонентный фреймворк для PHP, реализующий парадигму MVC (Model-View-Controller). Yii является акронимом от “Yes It is”, на русском пишется и читается как “йии”. Yii2 является второй версией фреймворка Yii.

26. Опишите преимущества фреймфорка Yii2.

Фреймворк прост в понимании.
Легко адаптируется под большие и маленькие проекты.
Имеет большое количество решений рутинных задач из коробки. К примеру, шаблон advanced обладает механизмом авторизации и аутентификации. Это довольно нужный механизм и он не очень прост в реализации.
Имеет замечательную документацию, гайды по старту и различные рецепты.
Yii2 популярен и довольно стар (релиз-то был аж в 2014 году), поэтому на рынке в много вакансий yii2-разработчиков, а с помощью развитого сообщества ответы на 90% вопросов вы найдете при легком гуглеже.
С помощью шаблонов и gii фреймворк подсказывает начинающему разработчику, как правильно располагать файлы. Сначала разработчик начинает повторять за тем, как это сделано в фреймворке, а потом понимает почему это хорошо. Разработчики фреймворка будто делятся опытом с новичком

27. Опишите Недостатки фреймворка Yii2.

Наличие различных антипаттернов в проекте — например, одиночка

или божественный объект.

Встроенный класс User, являющийся потомком от ActiveRecord, показывает, как делать не нужно. Учит начинающих программистов, что классы, наследуемые от ActiveRecord, нужно раздувать различными методами, не связанными с работой с БД.

Сильная связность модулей в приложении. Говорят, эта проблема в Yii3 будет решена

Медленное развитие.

28. Опишите экосистему Spring.

    Spring Boot, Spring MVC, Spring Web Flow, Spring Web Services, Spring Security, Spring Integration, Spring Batch, Spring Social, Spring Mobile, Spring Dynamic Modules, Spring LDAP, Spring Rich Client, Spring.NET, Spring-Flex, Spring Roo

29. Что такое Spring MVC?

    Это веб-фреймворк Spring. Он позволяет создавать веб-сайты или RESTful сервисы (например, JSON/XML) и хорошо интегрируется в экосистему Spring, например, он поддерживает контроллеры и REST контроллеры в ваших Spring Boot приложениях.

30. Что такое Spring Boot?

    Spring Boot — это полезный проект, целью которого является упрощение создания приложений на основе Spring. Он позволяет наиболее простым способом создать web-приложение, требуя от разработчиков минимум усилий по его настройке и написанию кода.

31. Определите особенности Spring Boot и его роль в разработке приложений с помощью фреймворка Spring?

    Если Spring Framework фокусируется на предоставлении гибкости, то Spring Boot стремится сократить длину кода и упростить разработку web-приложения. Используя конфигурацию при помощи аннотаций и стандартного кода, Spring Boot сокращает время, затрачиваемое на разработку приложений.

32. Опишите процесс управления зависимостями с помощью Spring Boot.

    Spring Boot решает эту проблему путём предоставления набора зависимостей, облегчая жизнь разработчикам. Например, если вы желаете использовать Spring и JPA в целях доступа к базе данных, вам достаточно просто включить в проект зависимость spring-boot-starter-data-jpa.

33. Опишите процесс автоматической конфигурации на Spring Boot.

    Spring Boot автоматическая настройка пытается автоматически настроить приложение Spring на основе добавленных зависимостей jar. Например, если HSQLDB находится на вашем classpath, и вы не настроили вручную никаких компонентов подключения к базе данных, то автоматически настроится база данных в памяти.

34. Что такое HttpServlet?

    Сервлет является интерфейсом Java, реализация которого расширяет функциональные возможности сервера. Сервлет взаимодействует с клиентами посредством принципа запрос-ответ.

    Хотя сервлеты могут обслуживать любые запросы, они обычно используются для расширения веб-серверов. Для таких приложений технология Java Servlet определяет HTTP-специфичные сервлет классы.

35. Что делает Spring MVC DispatcherServlet?

    Это главный контроллер в приложении Spring MVC, который обрабатывает все входящие запросы и передает их для обработки в различные методы в контроллеры.

36. Что такое ViewResolver?

    Интерфейс, реализуемый объектами, которые способны находить представления View по имени View Name.

37. Опишите REST контроллеры.

Когда вы разрабатываете RESTFul сервисы, все немного по-другому. Ваш клиент, будь то браузер или другой веб-сервис, будет (обычно) создавать запросы JSON или XML. Клиент отправляет, скажем, запрос JSON, вы обрабатываете его, а затем отправитель ожидает возврата JSON.

Но на стороне Java (в вашей программе Spring MVC) вы не хотите иметь дело с JSON строками. Ни при получении запросов, как указано выше, ни при отправке ответов обратно клиенту. Вместо этого вы хотели бы просто иметь объекты Java, в которые Spring автоматически конвертирует JSON. i.e. DTO

Это также означает, что вам *не нужна* вся эта обработка модели и представления, которые вам приходилось делать при рендеринге HTML в ваших контроллерах. Для RESTful сервисов у вас нет библиотеки шаблонов, читающей шаблон HTML и заполняющей его данными модели, чтобы сгенерировать для вас ответ JSON.

Вместо этого вы хотите перейти непосредственно из HTTP запрос → Java объект и из Java объект → HTTP ответ.

Как вы уже догадались, это именно то, что Spring MVC обеспечивает при написании REST контроллера.

38. Что такое HttpMessageConverter?

HttpMessageConverter — это интерфейс с четырьмя методами (обратите внимание, я немного упростил интерфейс для более простого объяснения, так как он выглядит немного более продвинутым в реальной жизни).

- canRead (MediaType) → Может ли этот конвертер читать (JSON | XML | YAML | и т. д.)? Переданный здесь MediaType обычно является значением из заголовка запроса Content-Type.

- canWrite (MediaType) → Может ли этот преобразователь писать (JSON | XML | YAML | и т. д.)? Тип MediaType, переданный здесь, обычно является значением из заголовка запроса Accept.
- read(Object, InputStream, MediaType) → Читать мой Java-объект из (JSON | XML | YAML | и т. д.) InputStream
- write(Object, OutputStream, MediaType) → Записать мой Java-объект в OutputStream как (JSON | XML | YAML | и т. д.)

Короче говоря, MessageConverter должен знать, какие MediaTypes он поддерживает (например, application/json), а затем должен реализовать два метода для фактического чтения / записи в этом формате данных.

39. Какие есть HttpMessageConverters?

    AllEncompassingFormHttpMessageConverter

40. В чем разница между Spring MVC и Spring Boot?

    Spring MVC – это полный HTTP-ориентированный MVC-фреймворк, управляемый Spring Framework и основанный на сервлетах. Spring boot - это утилита для быстрой настройки приложений, предлагающая готовую конфигурацию для создания приложений на базе Spring. / «Нет никакой разницы, Spring Boot использует и строит приложение поверх Spring MVC.»

41. Какой тип ввода HTTP-запроса понимает Spring MVC?

    Spring MVC понимает практически все, что предлагает HTTP — с помощью сторонних библиотек.

    Это означает, что вы можете добавить в него тела запросов JSON, XML или HTTP (Multipart) Fileuploads, и Spring будет удобно конвертировать этот ввод в объекты Java.

42. Какие HTTP-ответы может создавать Spring MVC?

Spring MVC может записывать все что угодно в HttpServletResponse — с помощью сторонних библиотек.

Будь то HTML, JSON, XML или даже тела ответов WebSocket. Более того, он берет ваши объекты Java и генерирует эти тела ответов для вас.

43. В чем разница между контроллером и REST контроллером?

    Контроллер по умолчанию возвращают HTML пользователям с помощью библиотеки шаблонов, если вы не добавите аннотацию @ResponseBody к определенным методам, которые также позволяют возвращать XML / JSON.

    Исходный код REST контроллера показывает, что на самом деле это контроллер с добавленной аннотацией @ResponseBody. Что эквивалентно написанию контроллера с аннотацией @ResponseBody для каждого метода.

44. Как получить доступ к текущей HttpSession пользователя?

    В Spring MVC контроллере или REST контроллере вы можете просто указать HttpSession в качестве аргумента метода, и Spring автоматически вставит его (создав его, если он еще не существует).

45. Как получить доступ к HttpServletRequest?

    В вашем Spring MVC контроллере или REST контроллере вы можете просто указать HttpServletRequest в качестве аргумента метода, и Spring автоматически вставит его (создавая, если он еще не существует)

46. Как читать HTTP заголовки?

    Существует множество способов получить доступ к заголовкам запросов, в зависимости от того, хотите ли вы только один или карту со всеми из них. В любом случае вам нужно аннотировать их с помощью @RequestHeader.

47. Как получить IP-адрес пользователя?

Это вопрос с подвохом. Существует метод с именем httpServletRequest.getRemoteAddr(), который, однако, возвращает только IP-адрес пользователя или последнего прокси-сервера, отправившего запрос, в 99,99% случаев это ваш Nginx или Apache. Следовательно, вам нужно проанализировать заголовок X-Forwarded-For для получения правильного IP-адреса. Но что произойдет, если ваше приложение, кроме того, будет работать за CDN, например CloudFront? Тогда ваш X-Forwarded-For будет выглядеть так:

X-Forwarded-For: MaybeSomeSpoofedIp, realIp, cloudFrontIp

Проблема в том, что вы не можете прочитать заголовок слева направо, поскольку пользователи могут предоставить и, следовательно, подделать свой собственный заголовок X-Forwarded-For. Вам всегда нужно идти справа налево и исключать все известные IP-адреса. В случае CloudFront это означает, что вам необходимо знать диапазоны IP-адресов CloudFront и удалить их из заголовка. Ага!

Это приводит к довольно сложному коду, разрешающему IP. Угадайте, сколько проектов сделали это неправильно!

```
package com.marcobehler.springmvcarticle;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import javax.servlet.http.HttpServletRequest;
@RestController
public class IpController {

   private static final String[] HEADERS_TO_TRY = {
         "X-Forwarded-For",
         "Proxy-Client-IP",
         "WL-Proxy-Client-IP",
         "HTTP_X_FORWARDED_FOR",
         "HTTP_X_FORWARDED",
         "HTTP_X_CLUSTER_CLIENT_IP",
         "HTTP_CLIENT_IP",
         "HTTP_FORWARDED_FOR",
```

```
            "HTTP_FORWARDED",
            "HTTP_VIA",
            "REMOTE_ADDR"};

    @GetMapping("/ip")
    public String getClientIpAddress(HttpServletRequest request) {
        for (String header : HEADERS_TO_TRY) {
            String ip = request.getHeader(header);
            if (ip != null && ip.length() != 0 &&
!"unknown".equalsIgnoreCase(ip)) {
                return getRealClientIpAddress(ip);
            }
        }
        return request.getRemoteAddr();
    }

    /**
     * Goes through the supplied ip string (could be one or multiple).
Traverses it through the right side...
     * and removes any known ip address ranges
     *
     * @param ipString
     * @return
     */
    public String getRealClientIpAddress(String ipString) {
        String[] manyPossibleIps = ipString.split(",");

        for (int i = manyPossibleIps.length - 1; i >= 0; i--) {
            String rightMostIp = manyPossibleIps[i].trim();
            if (isKnownAddress(rightMostIp)) {
                continue; // skip this ip as it is trusted
            } else {
                return rightMostIp;
            }
        }

        return ipString;
    }

    private boolean isKnownAddress(String rightMostIp) {
        // do your check here..for cloudfront you'd need to download their
ip address ranges
        // from e.g. http://d7uri8nf7uskq.cloudfront.net/tools/list-
cloudfront-ips
        // and compare the current ip against them
```

```
        return false;
    }
}
```

48. Опишите фреймворк Django.

    Свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC. Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий фреймворка от некоторых других (например, Ruby on Rails). Один из основных принципов фреймворка — DRY (англ. Don’t repeat yourself).

49. Какое программное обеспечение позволяет разрабатывать django?

    Django Rest Framework (DRF) — это библиотека, которая работает со стандартными моделями Django для создания гибкого и мощного API для проекта.

50. Из каких частей состоит обычный проект на Django.

    Веб-приложение или проект Django состоит из отдельных приложений. Вместе они образуют полноценное веб-приложение. Каждое приложение представляет какую-то определенную функциональность или группу функциональностей. Один проект может включать множество приложений. Это позволяет выделить группу задач в отдельный модуль и разрабатывать их относительно независимо от других. Кроме того, мы можем переносить приложение из одного проекта в другой независимо от другой функциональности проекта.

    При создании проекта он уже содержит несколько приложений по умолчанию.

    - django.contrib.admin
    - django.contrib.auth
    - django.contrib.contenttypes
    - django.contrib.sessions

- django.contrib.messages
- django.contrib.staticfiles

51. Что такое Отображения (views) в рамках Django.

    Представления (views) — центральные персонажи Web-приложений на основе Django.

    В Django используются два вида представлений:

    1. *Представления-функции (view functions)*,
    2. *Представления-классы (class based views)*.

    Рассмотрим сначала более простой (но не менее мощный) вид - обычные функции, принимающие на входе *запрос* (объект класса HttpRequest) и возвращающие *ответ* (объект класса HttpResponse или ему подобных).

    Все представления этого вида наследуются от класса django.views.View. Метод get здесь работает точь-в-точь как view function. При этом на каждый запрос будет создан новый экземпляр этого класса, так что вы смело можете объявлять в классе методы, которые по ходу выполнения запроса будут менять его состояние. Регистрируется представление-класс с помощью метода (метода класса) as_view.

52. Что такое модель в рамках Django.

    Веб-приложения Django получают доступ и управляют данными через объекты Python, называемые моделями. Модели определяют структуру хранимых данных, включая типы полей и, возможно, их максимальный размер, значения по умолчанию, параметры списка выбора, текст справки для документации, текст меток для форм и т.д.

53. Как производится вывод данных с помощью Django.

    Да.

54. Как работает Административная панель в Django.

Она использует мета-данные модели чтобы предоставить многофункциональный, готовый к использованию интерфейс для работы с содержимым сайта.

55. Опишите реализацию аутентификации на Django.

Внутри locallibrary/locallibrary/settings.py:

INSTALLED_APPS = [

...

'django.contrib.auth', # Фреймворк аутентификации и моделей по умолчанию.

'django.contrib.contenttypes', # Django контент-типовая система (даёт разрешения, связанные с моделями).

....

MIDDLEWARE = [

...

'django.contrib.sessions.middleware.SessionMiddleware', # Управление сессиями между запросами

...

'django.contrib.auth.middleware.AuthenticationMiddleware', # Связывает пользователей, использующих сессии, запросами.

....

56. Опишите преимущества фреймворка Django.

Когда у вас возникает определенная мысль, трансформировать ее на языке программирования и предать ей реальную форму при помощи Django займет всего несколько минут. То, что Django находится в свободном доступе, дает возможность заметно упростить процесс веб разработки, так как разработчик может сфокусироваться на процессе дизайна и разработке функционала приложения. Таким образом, Django – это идеальный инструмент

для стартапов, когда веб дизайн должен отображать концепцию и цели компании.

Быстрота: Django был разработан, чтобы помочь разработчикам создать приложение настолько быстро, на сколько это возможно. Это включает в себя формирование идеи, разработку и выпуск проекта, где Django экономит время и ресурсы на каждом из этих этапов. Таким образом, его можно назвать идеальным решением для разработчиков, для которых вопрос дедлайна стоит в приоритете.

Полная комплектация: Django работает с десятками дополнительных функций, которые заметно помогают с аутентификацией пользователя, картами сайта, администрированием содержимого, RSS и многим другим. Данные аспекты помогают осуществить каждый этап веб разработки.

Безопасность: Работая в Django, вы получаете защиту от ошибок, связанных с безопасностью и ставящих под угрозу проект. Я имею ввиду такие распространенные ошибки, как инъекции SQL, кросс-сайт подлоги, clickjacking и кросс-сайтовый скриптинг. Для эффективного использования логинов и паролей, система пользовательской аутентификации является ключом.

Масштабируемость: фреймворк Django наилучшим образом подходит для работы с самыми высокими трафиками. Следовательно, логично, что великое множество загруженных сайтов используют Django для удовлетворения требований, связанных с трафиком.

Разносторонность: менеджмент контента, научные вычислительные платформы, даже крупные организации – со всем этим можно эффективно справляться при помощи Django.

57. Опишите недостатки фреймворка Django.

Использование шаблона маршрутизации с указанием URL

Django слишком монолитный

Все базируется на ORM Django

Компоненты развертываются совместно

Необходимо умение владеть всей системой для работы.

58. Опишите процесс масштабирования приложения с помощью какого-либо современного фреймворка.

    Фреймворки имеют разную сложность и рассчитаны на компании или подразделения разного размера. При этом большинство из них рассчитано на короткие цепочки создания ценности, когда одна кроссфункциональная команда делает продукт, поставляемый потребителям. Короткие цепочки являются естественным способом организации труда, способным быстро реагировать на изменения, в отличие от стабильных условий функционирования, которые ведут к специализации и образованию длинных цепочек из функциональных подразделений.

59. Опишите ситуации, когда для масштабирования системы, написанной с помощью фреймворка требуется реинжиниринг системы.

    К примеру, система плохо сынженирована.

60. Опишите как происходит процесс перехода с одного фреймворка на другой.

61. Когда требуется переход с одного фреймворка на другой?

    Когда ограничения изначального фреймворка начинают сильно мешать, а возможности изначального фреймворка больше не являются достаточными.

62. Что такое Пространство задач и пространство решений?

    Предметные области состоят из пространства задач и пространства решений. Пространство задач позволяет думать о стратегической бизнес проблеме, которая должна быть решена, а

пространство решений, сосредоточится на том, как реализуется программное обеспечение, чтобы решить бизнес проблему.

Пространство задач – части предметной области, которые необходимы, чтобы создать смысловое ядро. Это комбинация смыслового ядра и подобластей, которое это ядро должно использовать.

Пространство решений – один или несколько ограниченных контекстов, набор конкретных моделей программного обеспечения. Разработанный ограниченный контекст – это конкретное решение, представление реализации.

Идеальным вариантом является обеспечение однозначного соответствия между подобластями и ограниченными контекстами. Таким образом, объединяются пространство задач и пространство решений, выделяются модели предметной области в четко определенные области в зависимости от поставленных целей. Если система не разрабатывается с нуля, она часто представляет собой большой комок грязи, где подобласти пересекаются с ограниченными контекстами.