Пр4

1. 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

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.

2. Опишите как определить класс в PHP.

Класс определяется с помощью ключевого слова class, за которым следует имя класса и пара фигурных скобок ({})

3. Как создать экземпляр класса в PHP.

Для создания экземпляра класса используется директива new

4. Опишите механизм наследования в PHP.

Наследование - это механизм объектно ориентированного программирования, который позволяет описать новый класс на основе уже существующего (родительского). Класс, который получается в результате наследования от другого, называется подклассом.

5. Опишите правила совместимости сигнатур.

6. Опишите методы и свойства Nullsafe.

Оператор nullsafe обеспечивает функциональность, аналогичную объединению null, но также поддерживает и вызовы методов

7. Опишите понятие автоматическая загрузка классов.

Автоматическая загрузка работает так. Где-то в начале приложения PHP вы создаете специальную функцию __autoload(). В последствии, если где-то в коде будет попытка создать объект класса, о котором ничего не известно, PHP автоматически вызовет данную функцию, передав ей в качестве параметра имя искомого класса. Вся работа функции заключается в том, чтобы найти нужный файл и подгрузить его к скрипту, тем самым загрузить сам класс. После этого PHP уже сможет создать объект данного класса.

8. Опишите конструкторы и деструкторы в PHP.

Конструкторы представляют специальные методы, которые выполняются при создании объекта и служат для начальной инициализации его свойств. Для создания конструктора надо объявить функцию с именем __construct (с двумя подчеркиваниями впереди)

Деструкторы служат для освобождения ресурсов, используемых программой - для освобождения открытых файлов, открытых подключений к базам данных и т.д. Деструктор объекта вызывается самим интерпретатором PHP после потери последней ссылки на данный объект в программе.

Деструктор определяется с помощью функции __destruct (два подчеркивания впереди)

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 или просто "двойное двоеточие" - это лексема, позволяющая обращаться к статическим свойствам, константам и переопределённым свойствам или методам 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.

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.

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.

Магическими эти методы называются по той причине, что они вызываются автоматически, если вы определите один из этих методов в классе. Вам остается лишь определить, какую задачу будет выполнять метод. Лучший пример – функция __construct(), которая вызывается автоматически каждый раз при создании экземпляра объекта.

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.

16.Что такое ковариантность и контравариантность?

В PHP 7.2.0 была добавлена частичная контравариантность путём устранения ограничений типа для параметров в дочернем методе. Начиная с PHP 7.4.0, добавлена полная поддержка ковариантности и контравариантности.

Ковариантность позволяет дочернему методу возвращать более конкретный тип, чем тип возвращаемого значения его родительского метода. В то время как контравариантность позволяет типу параметра в дочернем методе быть менее специфичным, чем в родительском.

17.Опишите понятие чистой архитектуры.

Смысл понятия в том, чтобы создавать архитектуру, которая не зависела бы от внешнего воздействия

18.Сформулируйте правило зависимостей.

19.Чем определяются сущности, чем они могут быть?

20.Что такое слой сценариев?

Сценарии — это слой, где реализуются случаи использования, которые возникают из-за того, что пользователям приложения необходимо что-то «делать» с субъектами базового Домена.

21.Что такое DTO?

Data Transfer Object (DTO) — один из шаблонов проектирования, используется для передачи данных между подсистемами приложения. Data Transfer Object, в отличие от business object или data access object не должен содержать какого-либо поведения.

22.Что является деталью в рамках чистой архитектуры?

23.Опишите принципы организации компонентов.

Release Equivalence Principle (REP) — Принцип эквивалентности повторного использования и выпуска Единица повторного использования равна единице выпуска. принцип REP гласит, что единица повторного использования, компонент, не может быть меньше единицы выпуска.

Принцип согласованного изменения (CCP) гласит: в один компонент должны включаться классы, изменяющиеся по одним причинам и в одно время.

Общий принцип повторного использования (CRP) гласит: «Классы в компоненте повторно используются вместе. Если вы повторно используете один из классов в компоненте, вы повторно используете их все».

24.Опишите принципы дизайна архитектуры.

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

Предметно-ориентированное проектирование (реже проблемно-ориентированное, англ. domain-driven design, DDD) — это набор принципов и схем, направленных на создание оптимальных систем объектов. Сводится к созданию программных абстракций, которые называются моделями предметных областей.

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.

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.

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)?

Это то, что делает организация, и среда, в которой она это делает. Разработчик программного обеспечения для организации обязательно работает в ее предметной области.

30.Что такое Пространство задач и пространство решений?

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.