1. Что такое сервлет ?

**Сервлет** является интерфейсом [Java](https://ru.wikipedia.org/wiki/Java" \o "Java), реализация которого расширяет функциональные возможности [сервера](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_(%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5)). Сервлет взаимодействует с клиентами посредством принципа запрос-ответ. Сервлет - это класс, который расширяет функциональность класса **HttpServlet** и запускается внутри контейнера сервлетов. Servlet - серверный web-компонент JavaEE-приложения, предназначенный для обработки запроса, формирования и отправки ответа.

2. Для чего нужен web-сервер?

**Веб-сервер** – это сервер, который принимает запросы от пользователя и выдает им ответы — документ, страницу или сайт.

Его главная задача – хранение информации. Страницы, файлы, изображения, текстовый контент.  
Задачи:

* Получать запросы.
* Запускать программы на специальных языках программирования (серверных).
* Формировать веб страницы.
* Отправлять их назад пользователю.
* Защищать информацию.
* Проводить идентификацию посетителей.
* Вести журнал обращений.
* Обслуживать запросы других типов: mailto, [FTP](https://semantica.in/blog/chto-takoe-ftp-server.html) и т.д.

3. Расскажите о принципах построения и работы сервлета? Объясните общее взаимодействие клиентской и серверной сторон.

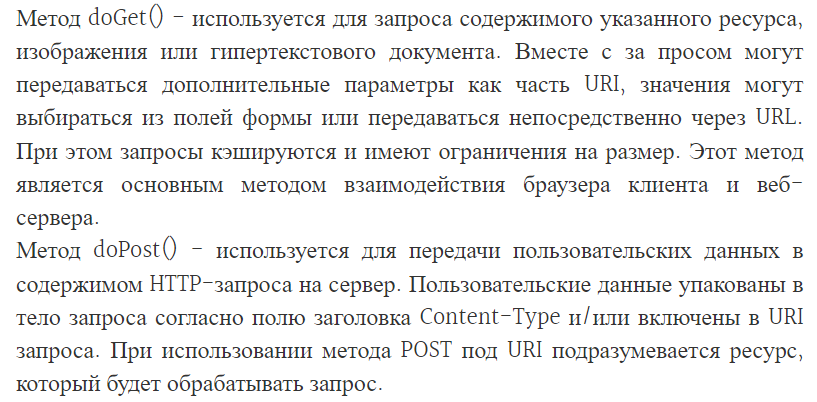
В основе работы приложения лежит так называемая модель взаимодействия клиент-сервер, которая позволяет разделять функционал и вычислительную нагрузку между клиентскими приложениями (заказчиками услуг) и серверными приложениями (поставщиками услуг).

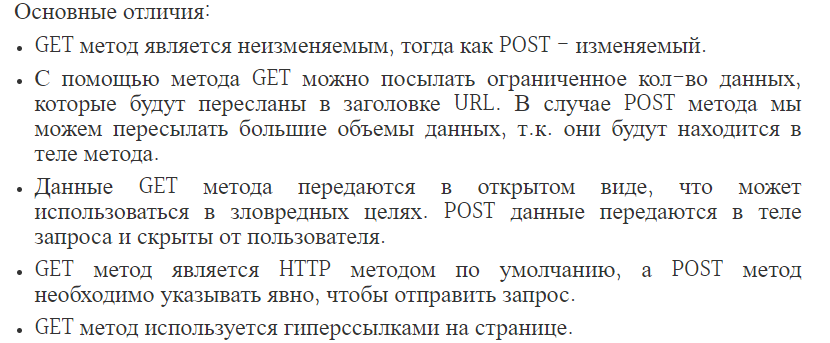
Request- запрос; Response- отклик;

4. Чем отличаются методы doGet и doPost друг от друга? В чем разница между методами GET и POST?

**Различия** **между** **методами** **doGet** () **и** **doPost** () заключаются в том, что они вызываются в HttpServlet, который ваш сервлет расширяет **методом** service (), когда он получает запрос **GET** или **POST** из запроса протокола HTTP. **GET**-запрос – это запрос на получение ресурса с сервера.

В **отличие** от него, метод HTTP **GET** предназначен для получения информации от сервера. В рамках **GET**-**запроса** некоторые данные могут быть переданы в строке **запроса** URI, указывающие, например, условия поиска, диапазоны дат, или другую информацию, определяющую **запрос**. В рамках **POST** **запроса** произвольное количество данных любого типа может быть **отправлено** на сервер в теле сообщения **запроса**.





5. Назовите методы жизненного цикла сервлета. В каком порядке они вызываются?

Жизненный цикл любого сервлета состоит из 4 этапов:

Инициализация, которая происходит при вызове метода init()

Обработка клиентского запроса после вызова метода service()

Уничтожение сервлета с помощью вызова метода destroy()

Полное уничтожение сервлета сборщиком мусора Java (Garbage Collector)

6. Как определить name и URL сервлета?

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException { response.setContentType("text/html"); response.getWriter().print("This is " + this.getClass().getName() + ", using the GET method"); } protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException { response.setContentType("text/html"); response.getWriter().print("This is " + this.getClass().getName() + ", using the POST method"); } }

request.getAttribute("javax.servlet.forward.request\_uri")

7. Как получить значение параметра формы в сервлете?

Сервлеты автоматически обрабатывают данные, используя следующие методы в зависимости от ситуации:

getParameter () — Вы вызываете метод request.getParameter (), чтобы получить значение параметра формы

getParameterValues ​​() — Вызовите этот метод, если параметр появляется более одного раза и возвращает несколько значений, например, флажок

getParameterNames () — вызовите этот метод, если вы хотите получить полный список всех параметров в текущем запросе

8. Как сервлет возвращает результат своей работы на сторону клиента?

Просто вызов URL-адреса сервлета приведет вас к стороне клиента (браузеру). Вам не нужно делать ничего конкретного.

9. Как происходит развертывание сервлета?

Перед развертыванием наши сервлеты необходимо упаковать в war-архив.

10.Для чего нужен интерфейс ServletContext?

**Интерфейс** javax.**servlet**.**ServletContext** определяет ряд методов, которые сервлет использует, чтобы связаться с его контейнером сервлета, например, получить тип MIME файла, диспетчеризировать запросы, или записать в файл журнала.

11. Что такое атрибут контекста?

серверный объект, предназначенный для хранения информации об одном web-приложении.

► формируется сразу при загрузке web-сервера, основные данные (параметры приложения) копируются из конфигурационного файла приложения, ► общий для всех сессий приложения, ► формат ключ/значение

12. Какую информацию можно получить из контекста сервлета?

Получить информацию из контекста сервлета

String getRealPath(String filename) — определение истинного маршрута файла относительно каталога, в котором сервер хранит документы;

String getServerInfo() — возвращает имя и номер версии контейнера сервлетов;

String getContextPath() — возвращает имя веб-приложения, находящегося в папке контейнера сервлетов;

String getServletContextName() — возвращает имя веб-приложения, заданное в дескрипторе приложения в теге ;

void log(String msg) — запись лога с сообщением msg;

void log(String msg, Throwable t) — запись лога с текстовым сообщением msg и сообщением, извлекаемым из исключения.

13.Опишите интерфейс ServletConfig?

Экземпляр ServletConfig - содержит параметры инициализации сервлета

ServletConfig sc = this.getServletConfig();

Методы интерфейса:

Enumeration getInitParameterNames() //спискок имен параметров инициализации сервлета;

String getInitParameter(String name) // определение значения параметра по его имени;

String getServletName() // определение имени сервлета

String getServletContext() // получение экземпляра контекста

14. Какую информацию содержит интерфейс HttpServletRequest?

Объект запроса – содержит информацию от клиента веб-приложения:

► информация о сервлете

► доступ к сессии и параметрам запроса

► управление атрибутами и кодировками

► детали протокола HTTP

15. Как можно использовать интерфейс HttpServletResponse?

Объекты, реализующие интерфейс ServletResponse пересылают клиенту из сервера-контейнера, генерируемые сервелатами данные

►ServletResponse HttpServletResponse

void setContentType(String type) //установка MIME-типа генерируемых документов;   
void addCookie(Cookie cookies) // добавление файла cookie для пересылки на клиентска;   
void sendError(int sc, String msg) // сообщение о возникших ошибках   
void sendRedirect(String location) // переход по указанному адресу, //без передачи экземпляра request.

ServletOutputStream getOutputStream() // извлечение ссылки на поток // для передачи бинарной информации;

PrintWriter getWriter() // извлечение ссылки на поток // для передачи символьной информации;

►Позволяет буферизацию

►Устанавливают + добавдяют заголовки

16. Как выполняется переадресация запросов и как перенаправление откликов? В чем принципиальная разница?

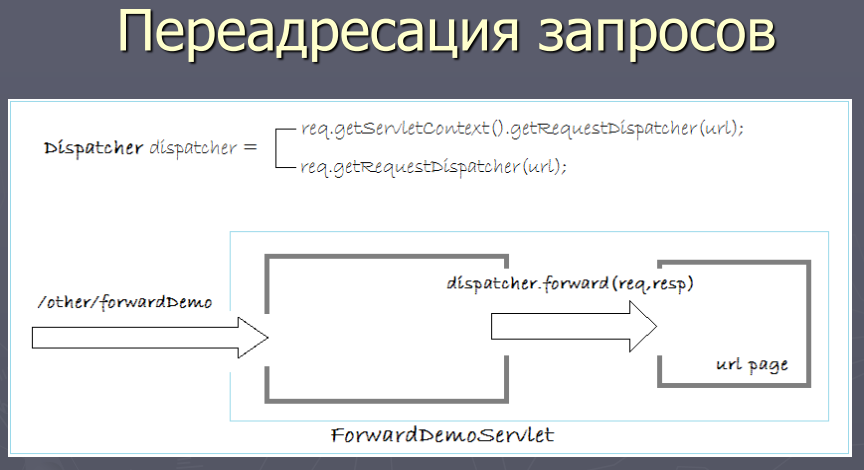
1) Запрос переадресуется (пересылается) (forwarded);

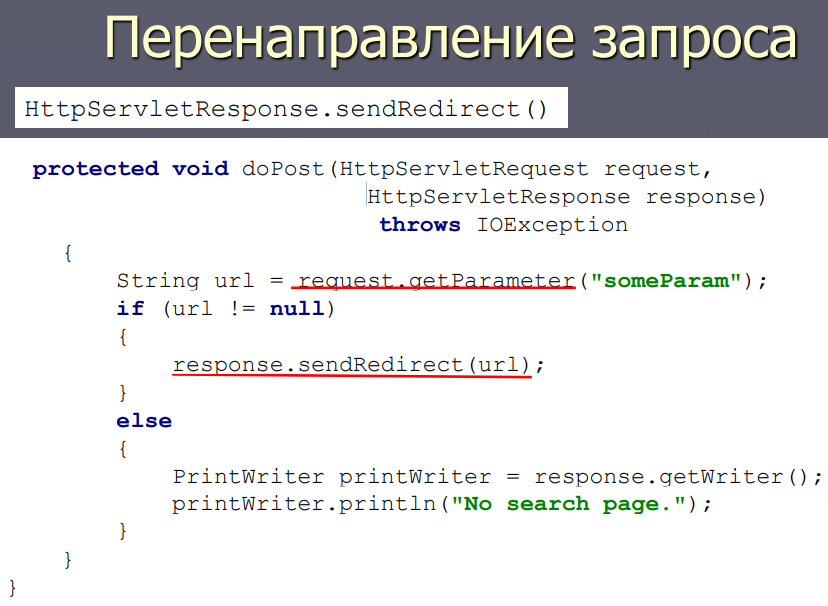
//переадресация запроса - до этого можно присоединить аттрибут request.getRequestDispatcher("SynchroServlet").forward(request,response);

б.д. в текущем контексте (war)

2) Отклик перенаправляется (redirected)

М.б. другой контекст или сервер //переадресация запроса - до этого можно присоединить аттрибут request.getRequestDispatcher("SynchroServlet").forward(request,response); Переадресация запросов





**Процесс пересылки:**

Сначала клиент отправляет запрос на сервер, который сопоставляет сервлет и указывает выполнение. Когда сервлет выполняется, он вызывает метод getRequestDispacther () и перенаправляет запрос в указанный тестовый сервлет. Весь процесс завершается на стороне сервера и выполняется в одном запросе, поэтому сервлет и тестовый сервлет совместно используют один и тот же запрос. Запрос, все что положено в ForwardServlet, можно вынести в testservlet. Поэтому testservlet может получить результат getAttribute (). После выполнения getAttribute () результат возвращается клиенту. Весь процесс представляет собой запрос и ответ.

**Процесс перенаправления:**

Клиент отправляет запрос на сервер, и сервер сопоставляет сервлет, который совпадает с переадресацией запроса. После обработки сервлета вызывается метод sendRedirect (), который является методом ответа. Итак, после обработки этого сервлета, посмотрите метод response.sendRedirect () и немедленно верните ответ клиенту. Строка ответа сообщает клиенту, что вы должны отправить еще один запрос для доступа к тестовому сервлету, и тогда клиент получит это Сразу после запроса отправляется новый запрос для запроса тестового сервлета. Два запроса не мешают друг другу и не зависят друг от друга. Ничего из setAttribute () в предыдущем запросе не может быть получено в следующем запросе. Поэтому в sendRedirect () есть два запроса и два ответа.

Пересылка - это переход со стороны сервера, то есть клиент отправляет запрос на сервер, а сервер напрямую передает информацию о параметрах, связанных с запросом, в другой jsp или сервлет сервера для обработки. SendRedirect () - это переход клиента, сервер вернет заголовок ответа и новый URL-адрес клиента, исходная информация о параметрах не существует, если сервер не имеет специальной обработки, браузер получит доступ к сервлету или новый URL-адрес, указанный JSP, это может быть не веб-сервис на исходном сервере.

**Разница между перенаправлением и пересылкой**

1. Пересылка выполняется на стороне сервера, а перенаправление происходит на стороне клиента;

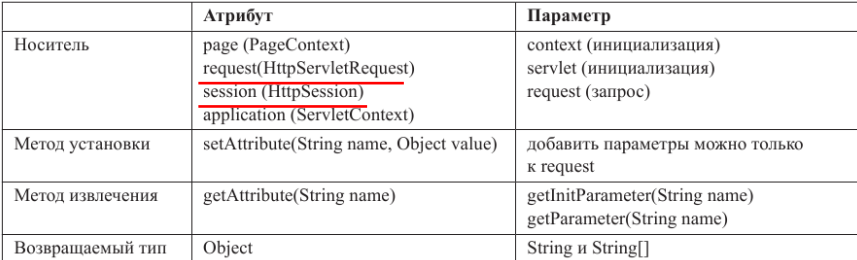
2. Переадресация - это один и тот же запрос, а перенаправление - два запроса;

3. Адресная строка пересылки не изменилась, а адресная строка переадресации изменилась;

4. Переадресация может осуществляться только в рамках одного веб-проекта, перенаправление может осуществляться в пределах одного веб-проекта или может быть перенаправлено в любое другое место;

17. Как можно сохранить данные приложения между запросами?

Объекты, которые могут получать, передавать, предоставлять информацию



Сохранение данных приложения между запросами (scope) 1) сохранение объекта в запросе HttpRequest.setAttribute() HttpRequest.getAttribute() 2) присоединение объекта к объекту сеанса(сессии в HttpSession) – не будут видимы для других пользователей 3) присоединение объекта к контексту сервлета (ServletContext) - будут видимы все пользователям

18.Как получить объект HttpSession и что она содержит?

HttpSession session = request.getSession();

► Сеанс есть сессия между клиентом и сервером, устанавливаемая на определенное время, за которое клиент может отправить на сервер сколько угодно запросов. ► для обеспечения хранения данных ► проблемы: ♣ поддержка распределенной сессии ♣ обеспечение безопасности; ♣ проблема инвалидации сессии (expiration), предупреждение пользователя об уничтожении сессии и возможность ее продления (watchdog). getSession(true) invalidate() Пример - хранение информации в сессии

19. Что такое файлы Cookie? Как его создать и поместить информацию информацию?

**Cookies** («куки», в переводе с англ. «печенье») — это небольшие текстовые документы, которые с помощью браузера сохраняет на компьютере пользователя веб-сервер (сайт). В эти **файлы** можно записать практически любую **информацию** о посетителе сайта: во сколько и с какого устройства человек заходил на страницу, какими товарами интересовался и так далее.

Cookie:фрагмент данных, оправленный web-сервером и хранимый web-клиентом. ► Используется для ♣ аутентификации, ► хранения пользовательских предпочтений, ♣ статистики, ► информации о сеансе ► имеет имя, содержащее URL, может иметь срок действия ► (RFC 6265) ► Для создания и пересылки Cookie применяются заголовки.

20. С какой целью используют интерфейс Filter?

Реализация **интерфейса** **Filter** позволяет создать объект, который может трансформировать заголовок и содержимое запроса клиента или ответа сервера.

21. Перечислите методы жизненного цикла фильтра

Метод init вызывается прежде, чем фильтр начинает работать, и определяет конфигурационные параметры фильтра. Метод **doFilter** выполняет непосредственно работу фильтра. Таким образом, сервер вызывает init один раз, чтобы запустить фильтр в работу, а затем вызывает **doFilter** столько раз, сколько запросов будет сделано непосредственно к данному фильтру. После того, как фильтр заканчивает свою работу, вызывается метод destroy.