**HTTP** (*HyperText Transfer Protocol* — «протокол передачи гипертекста») — протокол прикладного уровня передачи данных (изначально — в виде гипертекстовых документов в формате HTML, в настоящий момент используется для передачи произвольных данных).

Именно этот протокол мы используем, когда просматриваем странички в браузере.

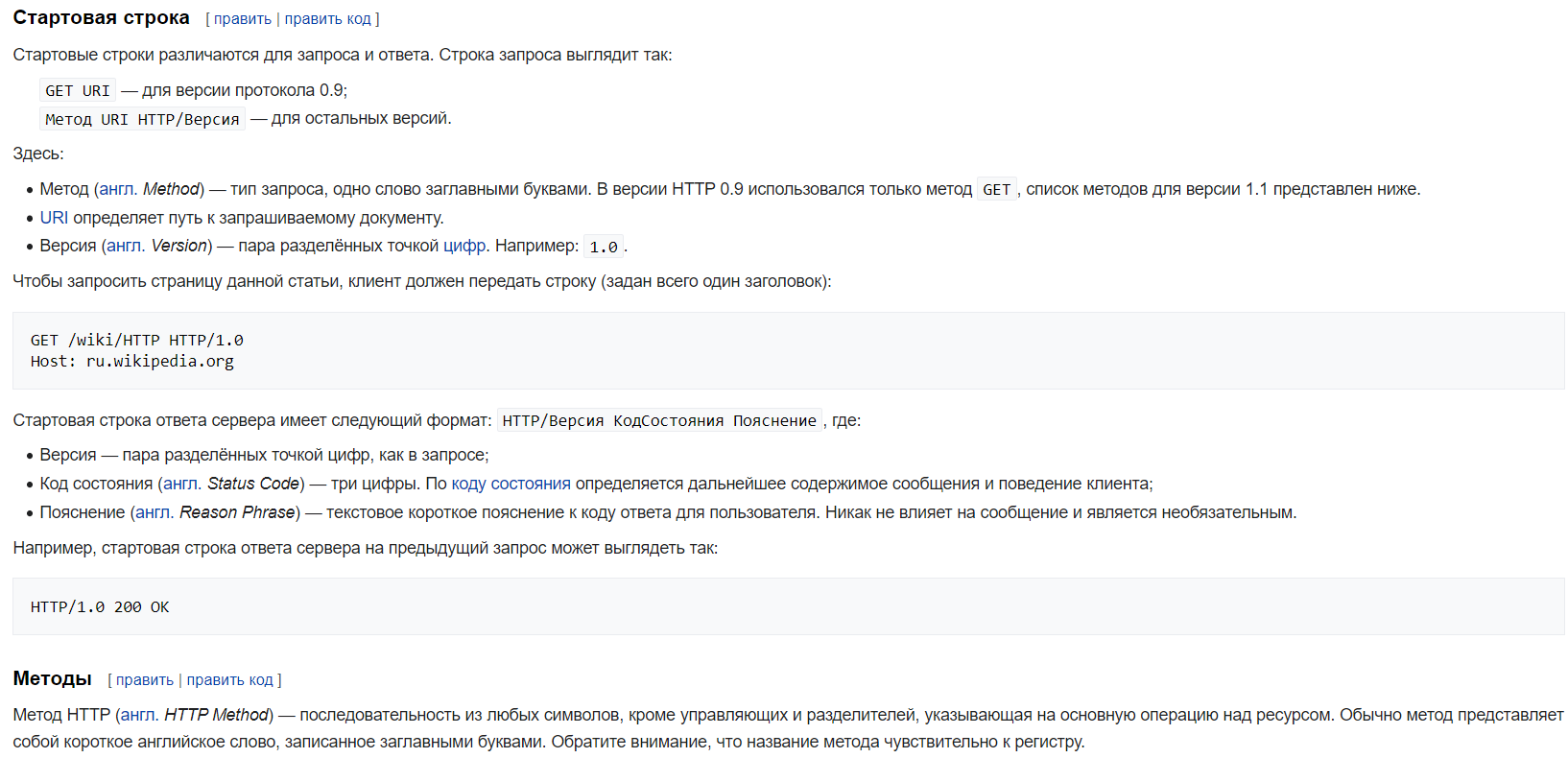
Но только два из них пользуются особой популярностью, а именно **GET** и **POST**.  Метод GET вызвать легко, достаточно ввести в браузере ссылку и перейти по ней, а вот POST не так просто выполнить, один из простых способов используя **form** тег указать ему **action**.

Каждое HTTP-сообщение состоит из трёх частей, которые передаются в указанном порядке:

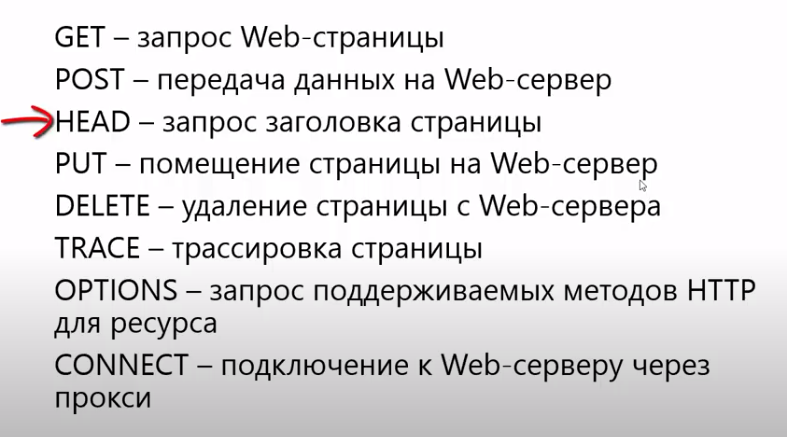
1. Стартовая строка ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Starting line*) — определяет тип сообщения;
2. Заголовки ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Headers*) — характеризуют тело сообщения, параметры передачи и прочие сведения;
3. Тело сообщения ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Message Body*) — непосредственно данные сообщения. Обязательно должно отделяться от заголовков пустой строкой.

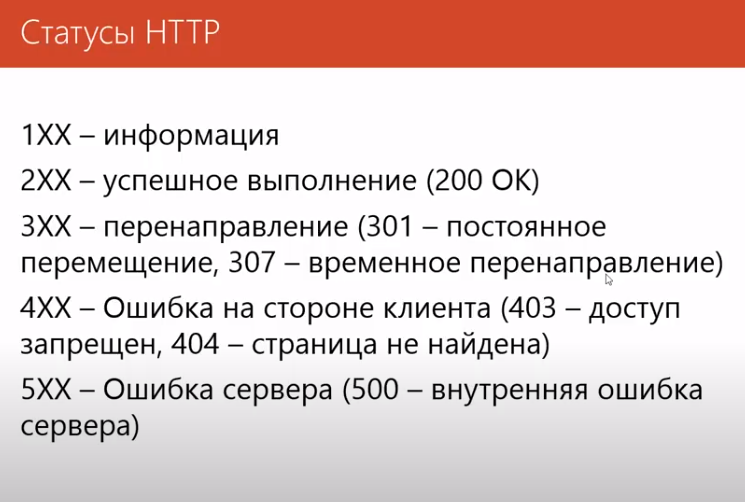
Тело сообщения может отсутствовать, но стартовая строка и заголовок являются обязательными элементами. Исключением является версия 0.9 протокола, у которой сообщение запроса содержит только стартовую строку, а сообщения ответа — только тело сообщения.

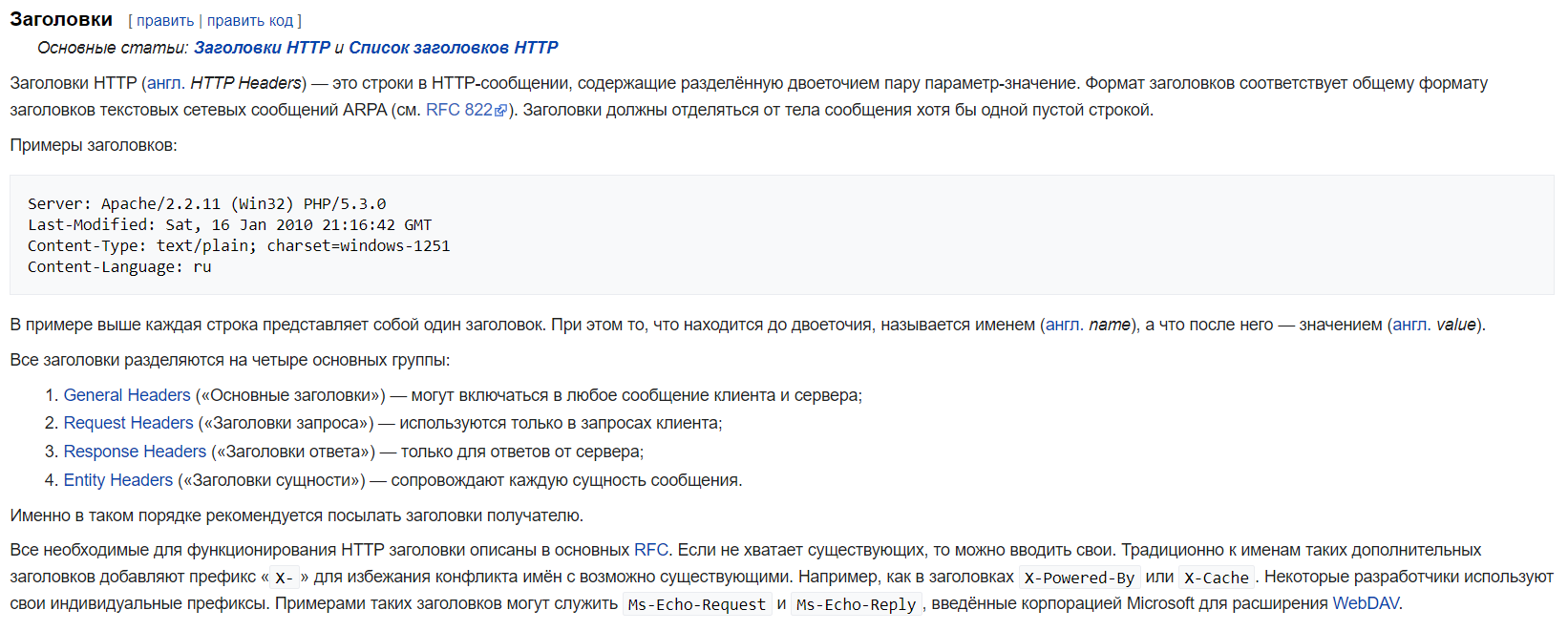
Для версии протокола 1.1 сообщение запроса обязательно должно содержать заголовок *Host*.

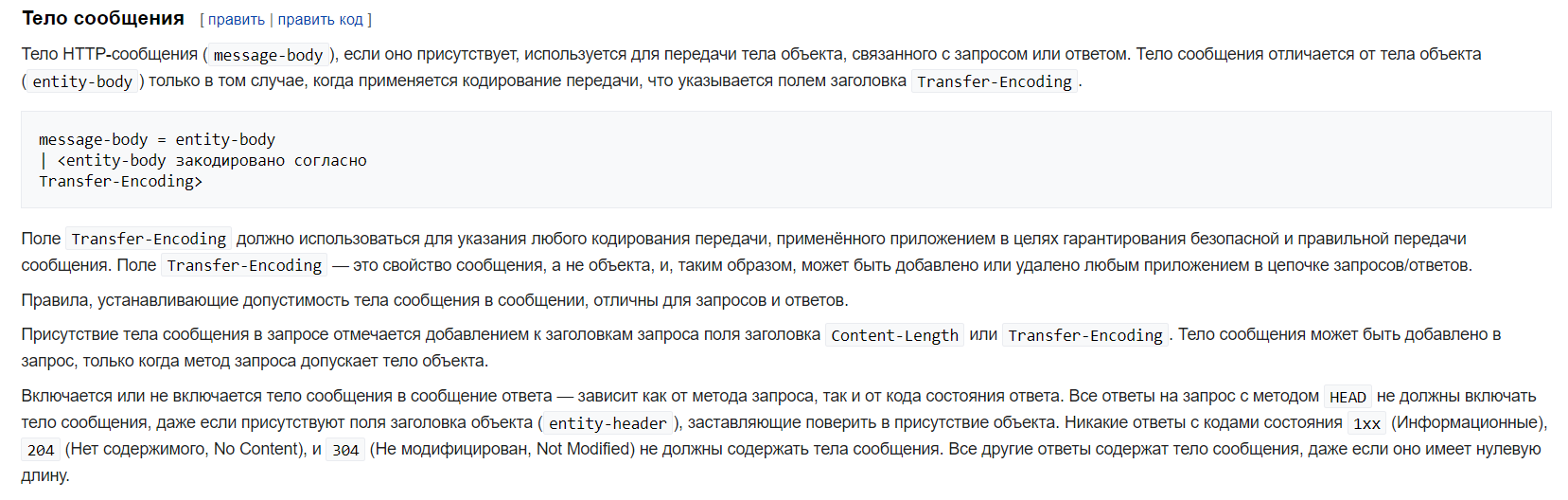


Известно, что HTTP протокол имеет 7 методов передачи данных:









1. **Что такое сервлет?**

**Servlet** - серверный web-компонент JavaEE приложения, предназначенный для обработки запроса, формирования и отправки ответа.

**Сервлет** – Java класс, написанный по определенным правилам и разворачиваемый в Java EE контейнере сервлета.

1. **Для чего нужен web-сервер?**

Веб-сервер — это сервер, который принимает HTTP-запросы от клиентов и выдающий им HTTP ответы (как правило, вместе с HTML страницей, изображением, файлом или другими данными). Запрашиваемые ресурсы обознаются URL-адресами.

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

Рассказать про HTTP и создание сервлета.

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

* GET метод является неизменяемым, тогда как POST — изменяемый.
* С помощью метода GET можно посылать ограниченное кол-во данных, которые будут пересланы в заголовке URL. В случае POST метода мы можем пересылать большие объемы данных, т.к. они будут находится в теле метода.
* Данные GET метода передаются в открытом виде, что может использоваться в зловредных целях. POST данные передаются в теле запроса и скрыты от пользователя.
* GET метод является HTTP методом по умолчанию, а POST метод необходимо указывать явно, чтобы отправить запрос.
* GET метод используется гиперссылками на странице.

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

Контейнер сервлетов управляет четырьмя фазами жизненного цикла сервлета:

* **Загрузка класса сервлета** — когда контейнер получает запрос для сервлета, то происходит загрузка класса сервлета в память и вызов конструктора без параметров.
* **Инициализация класса сервлета** — после того как класс загружен контейнер инициализирует объект ServletConfig для этого сервлета и внедряет его через init() метод. Это и есть место где сервлет класс преобразуется из обычного класса в сервлет.
* **Обработка запросов** — после инициализации сервлет готов к обработке запросов. Для каждого запроса клиента сервлет контейнер порождает новую нить (поток) и вызывает метод service() путем передачи ссылки на объект ответа и запроса.
* **Удаление из Service** — когда контейнер останавливается или останавливается приложение, то контейнер сервлетов уничтожает классы сервлетов путем вызова destroy() метода.

Можно описать как последовательность вызова методов: init(), service(), destroy().

* public void init(ServletConfig config) – используется контейнером для инициализации сервлета. Вызывается один раз за время жизни сервлета.
* public void service(ServletRequest request, ServletResponse response) – вызывается для каждого запроса. Метод не может быть вызван раньше выполнения init() метода.
* public void destroy() – вызывается для уничтожения сервлета (один раз за время жизни сервлета).

1. **Как определить name и URL сервлета?**

Для получения актуального пути сервлета на сервере можно использовать эту конструкцию: getServletContext().getRealPath(request.getServletPath())

1. Как получить значение параметра формы в сервлете?
2. **Как сервлет возвращает результат своей работы на сторону клиента?**

**HttpServletResponse** resp – это ответ который даст сервлет.

**9. Как происходит развертывание сервлета?**

Контейнер сервлетов обычно загружает сервлет при первом запросе клиента, но иногда необходимо загрузить сервлет прямо на старте приложения (например если сервлет объемный и будет долго грузиться). Для этого необходимо использовать элемент load-on-startup в дескрипторе (или аннотацию loadOnStartup), который укажет необходимость загрузки сервлета при запуске.

|  |  |
| --- | --- |
| 1  2  3  4  5 | <servlet>      <servlet-name>foo</servlet-name>      <servlet-class>com.foo.servlets.Foo</servlet-class>      <load-on-startup>5</load-on-startup>  </servlet> |

Значение должно быть int. Если значение отрицательное, то сервлет будет загружен при запросе клиента, а если 0 и далее, то загрузится на старте приложения. Чем меньше число, тем раньше в очереди на загрузку будет сервлет.

1. **Для чего нужен интерфейс ServletContext?**

Интерфейс javax.servlet.ServletContext предоставляет доступ к параметрам веб приложения сервлету. Объект ServletContext является уникальным и доступен всем сервлетам веб приложения. Мы можем использовать объект ServletContext, когда нам необходимо предоставить доступ одному или нескольким сервлетам к инициализированным параметрам веб приложения. Для этого используется элемент <context-param> в web.xml. Объект ServletContext можно получить с помощью метода getServletContext() у интерфейса ServletConfig.

Контейнеры сервлетов так же могут предоставлять context объекты, уникальные для группы сервлетов. Каждая из групп будет связана со своим набором URL путей хоста.

ServletContext был расширен в спецификации Servlet 3 и предоставляет программное добавление слушателей и фильтров в приложение. Так же у этого интерфейса имеются множество полезных методов вроде getMimeType(), getResourceAsStream() и т.д..

1. **Что такое атрибут контекста?**

Атрибуты сервлетов ипользуются для внутренней коммуникации сервлетов. Мы можем использовать атрибуты set, get, remove в веб приложении. Существует три области видимости атрибутов — request scope, session scope, application scope.

Интерфейсы ServletRequest, HttpSession и ServletContext предоставляют методы для get(), set(), remove() атрибутов из request scope, session scope, application scope соответственно.

Есть несколько способов передачи данных из сервлета в jsp, которые заключаются в использовании определенного контекста или scope. Есть несколько контекстов для передачи данных:

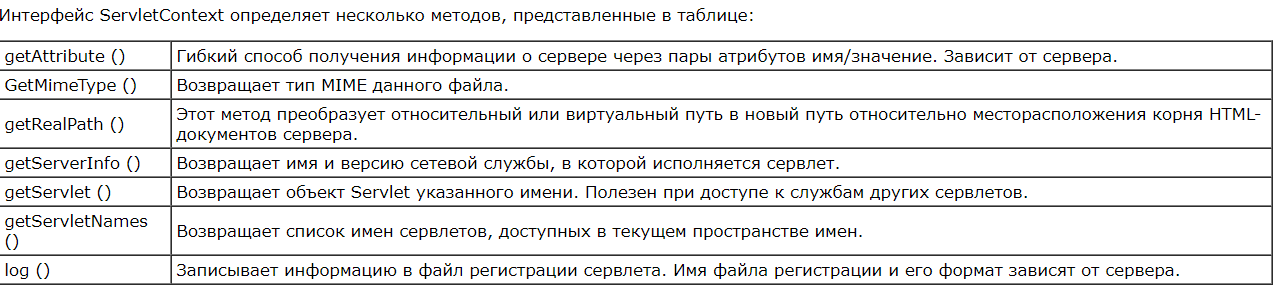
* request (контекст запроса): данные сохраняются в HttpServletRequest
* session (контекст сессии): данные сохраняются в HttpSession
* application (контекст приложения): данные сохраняются в ServletContext

Данные из контекста запроса доступны только в пределах текущего запроса. Данные из контекста сессии доступны только в пределах текущего сеанса. А данные из контекста приложения доступны постоянно, пока работает приложение.

Но вне зависимости от выбранного способа передача данных осуществляется с помощью метода setAttribute(name, value), где name - строковое название данных, а value - сами данные, которые могут представлять различные данные.

Наиболее распространенный способ передачи данных из сервлета в jsp представляют атрибуты запроса. То есть у объекта HttpServletRequest, который передается в сервлет, вызывается метод setAttribute(). Этот метод устанавливает атрибут, который можно получить в jsp.

1. **Какую информацию можно получить из контекста сервлета?**



1. **Опишите интерфейс ServletConfig?**

Интерфейс javax.servlet.ServletConfig используется для передачи конфигурационной информации сервлету. Каждый сервлет имеет свой собственный объект ServletConfig, за создание экземпляра которого ответственен контейнер сервлетов. Для установки параметров конфигурации используются init параметры в web.xml (или аннотации WebInitParam). Для получения объекта ServletConfig данного сервлета используется метод getServletConfig().

1. **Какую информацию содержит интерфейс HttpServletRequest?**

**HttpServletRequest** req – это запрос, который пришел к сервлету;

Контейнер сервлета создает HttpServletRequest объект и передает его в качестве аргумента в службу сервлета методы (doGet,doPost, etc).

1. **Как можно использовать интерфейс HttpServletResponse?**

**HttpServletResponse** resp – это ответ который даст сервлет.

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

* RequestDispatcher forward() используется для проброски того же самого запроса к другому ресурсу, в то время как ServletResponse sendRedirect() это двухшаговый метод. Во втором методе веб приложение возвращает ответ клиенту с status code 302 (redirect) с ссылкой для отправки запроса. Запрос посылает полностью новый запрос.
* forward() обрабатывается внутри контейнера, а sendRedirect() обрабатывается браузером.
* Необходимо использовать forward() для организации доступа внутри одного и того же приложения, т.к. он быстрее sendRedirect(), которому требуется дополнительная сетевая работа.
* В методе forward() браузер не знает о фактически обрабатываемом ресурсе и URL в строке остается прежним. В sendRedirect() методе URL адрес изменяется на пробрасываемый ресурс.
* В методе forward() нельзя использовать для внедрения сервлета в другой контекст. Для этого можно использовать только sendRedirect().

1. **Как можно сохранить данные приложения между запросами?**

Куки + сессии. Все современные фреймворки умеют работать с различными хранилищами данных сессии: бд, файлы, мемкеш (имхо для сессий самое то). НО не надо сохранять всю инфу пользователя в сессию, она для этого не предназначена. Можно хранить в сессии ID пользователя и остальные ключи, что бы в нужный момент легко выдернуть нужные данные из бд. Если данных много и они не менятся каждый раз, при запросе от пользователя, то можно кешировать данные, например в мемкеш, при этом данные пользователей которые давно не посещали сервис должны вытесняться данными пользователей, посетившими сервис только что.

1. **Как получить объект HttpSession и что она содержит?**

Сессия позоляет сохранять некоторую информацию на время сеанса. Когда клиент обращается к сервлету или странице JSP, то движок сервлетов проверяет, определен ли в запросе параметр ID сессии. Если такой параметр неопределен (например, клиент первый раз обращается к приложению), тогда движок сервлетов создает уникальное значение ID и связанный с ним объект сессии. Объект сессии сохраняется на сервере, а ID оправляется в ответе клиенту и по умолчанию сохраняется на клиенте в куках. Затем когда приходит новый запрос от того же клиента, то движок сервлетов опять же может получить ID и сопоставить его с объектом сессии на веб-сервере.

Хотя по умолчанию ID сессии хранится в куках, но возможна ситуация, когда куки отключены на клиенте. Для решения этой проблемы есть ряд техник, в частности, добавление ID в адрес.

Для получения объекта сессии в сервлете у объекта HttpServletRequest определен метод getSession(). Он возвращает объект HttpSession.

Для управления сессией объект HttpSession предоставляет ряд методов:

* setAttribute(String name, Object o): сохраняет в сессии под ключом name
* getAttribute(String name): возвращает из сессии объект с ключом name. Если ключа name в сессии неопределено, то возвращается null
* removeAttribute(String name): удаляет из сессии объект с ключом name

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

Cookie - это небольшие блоки текстовой информации, которые сервер посылает клиенту для сохранения в файлах cookies. Клиент может запретить браузеру прием файлов cookies. Браузер возвращает информацию обратно на сервер как часть заголовка HTTP, когда клиент повторно заходит на тот же веб-ресурс.

1. **С какой целью используют интерфейс Filter?**

Реализация интерфейса Filter позволяет создать объект, который перехватывает запрос, может трансформировать заголовок и содержимое запроса клиента. Фильтры не создают запрос или ответ, а только модифицируют их. Фильтр выполняет предварительную обработку запроса, прежде чем тот попадает в сервлет, с последующей (если необходимо) обработкой ответа, исходящего из сервлета. Фильтр может взаимодействовать с разными типами ресурсов, в частности, и с сервлетами, и с JSP-страницами.

**21. Перечислите методы жизненного цикла фильтра?**

