1. Сервлет - это Java-класс, который используется для обработки HTTP-запросов и генерации HTTP-ответов на сервере. Он работает внутри контейнера сервлетов, который является частью Java-технологии Java Servlet API, и обеспечивает динамическую генерацию контента на сервере, включая HTML, XML, JSON и другие форматы данных.
2. Web-сервер - это программное обеспечение, которое обрабатывает **HTTP-запросы от клиентов** (обычно веб-браузеров) и отправляет **HTTP-ответы** с веб-страницами или другими ресурсами. Он является основным компонентом веб-инфраструктуры и обеспечивает обработку запросов, управление сеансами, безопасность, аутентификацию, авторизацию и другие функции, связанные с веб-приложениями.
3. Принципы построения и работы сервлета включают следующие шаги:

* Клиент отправляет **HTTP-запрос на сервер**, адресуя **его к сервлету** по URL-адресу.
* Запрос передается веб-серверу, который передает его **контейнеру сервлетов**.
* Контейнер сервлетов инициализирует сервлет, вызывая его метод **init()** один раз при старте приложения.
* Контейнер сервлетов вызывает **метод service()** сервлета для обработки запроса, передавая ему объекты HttpServletRequest и HttpServletResponse, которые представляют запрос и ответ соответственно.
* Сервлет обрабатывает запрос, **генерирует HTTP-ответ,** который отправляется обратно на **веб-сервер.**
* Веб-сервер отправляет HTTP-ответ **обратно клиенту**, который отображает результат веб-страницы или выполняет другие действия в зависимости от содержимого ответа.

1. Методы doGet() и doPost() - это методы HTTP-запросов, которые могут быть обработаны сервлетом. Они отличаются **по способу передачи данных от клиента к серверу**. Метод **doGet**() передает данные **в URL-адресе запроса**, что может быть видимо в адресной строке браузера, тогда как метод doPost() передает **данные в теле HTTP-запроса**, что делает их скрытыми **отображения на клиентской стороне**. Разница между методами GET и POST заключается в способе передачи данных и их использования. GET используется для запросов только на чтение, когда клиент просто запрашивает данные, а POST используется для отправки данных на сервер
2. Методы жизненного цикла сервлета включают:

* **init**(): вызывается при инициализации сервлета и выполняется один раз. Здесь происходит инициализация ресурсов, настройка параметров и другие подготовительные действия.
* service(): вызывается при каждом HTTP-запросе и обрабатывает запросы клиента. Здесь происходит основная обработка запросов и генерация HTTP-ответов.
* **destroy**(): вызывается при остановке или перезагрузке контейнера сервлетов и выполняется один раз. Здесь происходит освобождение ресурсов, завершение работы и другие заключительные действия.

1. Имя (name) и URL (web URL) сервлета определяются в конфигурации развертывания (deployment configuration) в файле **web.xml** или аннотациями на самом классе сервлета. Имя сервлета используется для его идентификации внутри контейнера сервлетов, а URL сервлета определяет адрес, по которому клиенты могут обращаться к нему.
2. Для получения значения параметра формы в сервлете, можно использовать объект HttpServletRequest, который передается в метод service() или другие методы сервлета. Методы, такие как **getParameter**() или **getParameterValues**(), могут быть использованы для извлечения данных из запроса, включая параметры формы.
3. Сервлет возвращает результат своей работы на сторону клиента с помощью объекта **HttpServletResponse**, который также передается в метод service() или другие методы сервлета. Методы, такие как getWriter() или getOutputStream(), могут быть использованы для записи данных в тело HTTP-ответа, которые будут отправлены клиенту.
4. Развертывание сервлета происходит путем **упаковки сервлета в архив веб-приложения** (war-файл) и размещения его **в директории развертывания** (deployment directory) веб-сервера или контейнера сервлетов. Затем **веб-сервер или контейнер сервлетов** автоматически **обрабатывает развертывание**, и сервлет становится доступным для обработки HTTP-запросов.
5. Интерфейс ServletContext предоставляет информацию о контексте развертывания (deployment context) веб-приложения, такую как параметры и атрибуты контекста, доступные ресурсы, настройки и другие данные, которые могут быть использованы сервлетами и другими компонентами веб-приложения. Он также предоставляет методы
6. Атрибут контекста - это объект, который позволяет хранить данные в области контекста веб-приложения, доступные для всех сервлетов и JSP-страниц в данном приложении.
7. Из контекста сервлета можно получить информацию, такую как настройки конфигурации приложения, ресурсы базы данных, параметры инициализации, а также другие данные, общие для всех компонентов веб-приложения.
8. Интерфейс ServletConfig предоставляет доступ к конфигурационным настройкам сервлета, таким как параметры инициализации, имя сервлета, контекст веб-приложения и другие.
9. Интерфейс HttpServletRequest содержит информацию о HTTP-запросе, такую как параметры запроса, заголовки, метод запроса, URL, атрибуты запроса, информация о сессии и другие данные, связанные с текущим HTTP-запросом.
10. Интерфейс HttpServletResponse используется для формирования HTTP-ответа, такого как установка заголовков ответа, запись данных в тело ответа, управление сессией, отправка перенаправлений и других действий, связанных с формированием ответа на клиентскую сторону.
11. Переадресация запросов выполняется на сервере и предполагает отправку клиентского запроса на другой ресурс с использованием нового URL, в то время как перенаправление откликов происходит на стороне клиента с использованием JavaScript или HTML-мета тегов, и не влечет изменения URL на сервере. Основная разница заключается в том, что при переадресации запроса сервер выполняет полноценный новый запрос, в то время как при перенаправлении отклика клиентский браузер выполняет новый запрос на основе информации из ответа на предыдущий запрос.
12. Для сохранения данных приложения между запросами можно использовать атрибуты контекста (ServletContext), атрибуты сессии (HttpSession), базы данных, файловую систему или другие внешние хранилища данных.
13. Для получения объекта HttpSession можно использовать метод getSession() объекта HttpServletRequest. HttpSession представляет сессию клиента на сервере и содержит данные, специфичные для сессии, такие как атрибуты сессии, идентификатор сессии, время создания и последнего доступа к сессии, а также другие атрибуты, связанные с текущей сессией пользователя.
14. Файлы Cookie - это небольшие текстовые файлы, хранящиеся на стороне клиента,

продолжи

и передаваемые сервером в HTTP-заголовках ответа. Они используются для сохранения информации на стороне клиента и передачи ее обратно на сервер при каждом HTTP-запросе. Для создания и размещения информации в Cookie можно использовать методы HttpServletResponse, такие как addCookie() или setCookie(), и передавать соответствующие значения, такие как имя, значение, срок действия и другие параметры.

1. Интерфейс Filter используется для обработки запросов и ответов перед тем, как они достигнут сервлетов или после того, как они вернулись от сервлетов. Он позволяет выполнять определенные действия на этапе обработки запросов и ответов, такие как аутентификация, авторизация, фильтрация, логирование, компрессия и другие манипуляции с данными HTTP-трафика.
2. Методы жизненного цикла фильтра включают:

* init(FilterConfig config) - выполняется при инициализации фильтра и позволяет настроить фильтр, получив доступ к его конфигурационным настройкам.
* doFilter(ServletRequest request, ServletResponse response, FilterChain chain) - выполняется при каждом HTTP-запросе, перед тем, как запрос достигнет сервлета или после того, как ответ вернется от сервлета. Здесь происходит основная обработка запроса и ответа, и фильтр может принять решение о передаче запроса дальше в цепочку фильтров или сервлету, или же выполнении других действий.
* destroy() - выполняется при остановке фильтра и позволяет освободить ресурсы, занятые фильтром, и завершить его работу.