АРХИТЕКТУРА WEB ПРИЛОЖЕНИЙ НА ОСНОВЕ ТЕХНОЛОГИЙ JSP И СЕРВЛЕТОВ

Сервлеты являются специализированным механизмом Java для создания WEB ресурсов. Фактически, это модули обработки HTTP и FTP запросов, используемые для построения порталов (web-gates).

Основой этих порталов является собственно WEB-сервер — программа, которая держит сокет сервера, принимает и передаёт данные. Чаще всего, для ускорения работы, сервер бывает написан не на Java, а на каком-либо другом языке программирования. Иногда используется специальный сервер, например Apache Tomcat, который называют *контейнером сервлетов*.

В связке с сервером работает базовый сервлет. Именно ему отправляет сервер данные и от него же получает ответ, отправляемый клиенту. Фактически, базовый сервлет является "мозгом" сервера. Основная функция этого сервлета — прочитать запрос клиента, расшифровать его и, в соответствии с расшифровкой, передать работу другому сервлету, отвечающему за этот тип запрашиваемой информации. Зачастую, для достижения скорости, роль базового сервлета играет сам сервер (именно по такой схеме работает Apache Tomcat).

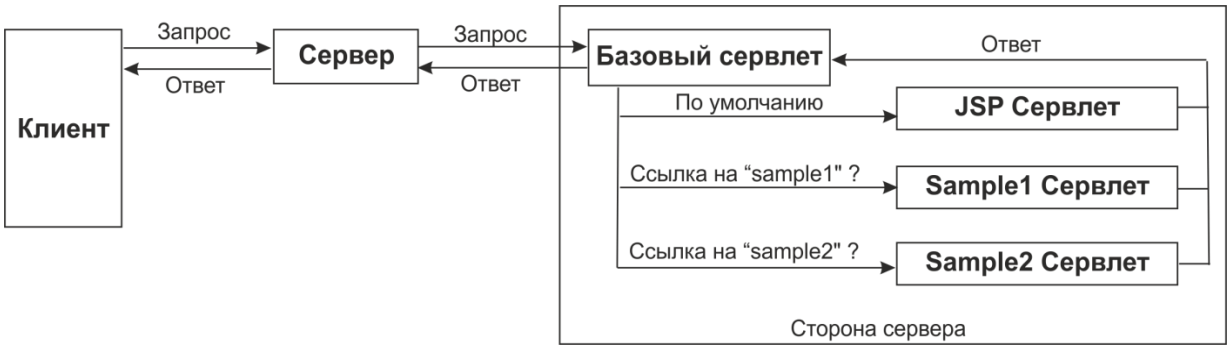

Рис. 1 Схема работы сервлетов и сервера

На рисунке 1 изображена схема передачи вызовов и ответов между сервером и сервлетами. Данная схема изображает работу HTTP сервера, который имеет несколько JSP страниц и два ресурса "/sample1" и "/sample2", за обработку которых отвечает два сервлета - "Sample1 Servlet" и "Sample2 Servlet" соответственно.

Таким образом, достигается разбиение задачи обработки запроса на логические части, за каждую из которых отвечает свой модуль. На самом деле, ступеней в обработке запроса может быть гораздо больше. К примеру, за методы GET и POST могут отвечать разные модули.

Сервлеты входят в пакеты языка Java: javax.servlet, javax.servlet.http, javax.servlet.jsp. Пакеты эти, в свою очередь, принадлежат набору Java Servlet API, который входит в архитектуру Java 2 Enterprise Edition. Для работы с сервлетами требуется либо J2EE, либо Java Servlet API, поставляемый вместе с такими WEB серверами, как Apache Tomcat.

Сервлеты не являются единственным методом работы с WEB-страницами. Одной из самых ранних технологий для этой цели являлся CGI (common gateway interface), но он инициализировал отдельный процесс для каждого запроса, что не было очень эффективно. Существовали также патентованные серверные расширения, например Netscape Server API (NSAPI). В мире Windows существует стандарт ASP (active server pages). Сервлеты являются альтернативой всем этим технологиям и предлагают несколько преимуществ:

- Они также платформенно-независимы, как и язык Java.
- Они предоставляют полный доступ ко всем API языка Java, включая библиотеки для доступа к данным (например, JDBC).
- Они (в большинстве случаев) в своей основе более эффективны, чем CGI, поскольку сервлеты порождают новые потоки (нити) для запросов, а не отдельные процессы.
- Существует распространенная отраслевая поддержка сервлетов, включая контейнеры для наиболее популярных WEB-серверов и серверов приложений.

Сервлетам, как и любым другим технологиям, работающим с протоколом HTTP, приходится принимать одну из двух схем работы с пользователями.

К примеру, пользователь заходит на страницу 1, где ему отправляются некоторые данные для страницы 2, а та сохраняет ещё какие-то вещи для страницы 3.

В принципе, на странице 1 можно выслать данные пользователю, потом получить их на странице 2, добавить что-то, выслать пользователю... Подобным образом придётся постоянно пересылать весь набор данных от клиента серверу и обратно, причём много раз. Кроме того, что такая пересылка не всегда удобна, она ещё и генерирует большой траффик (см. рисунок 2).

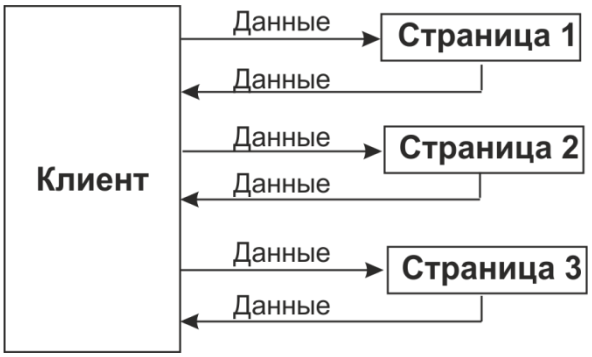

Рис. 2 Механизм работы сервера с клиентом по протоколу HTTP без сессий

Можно так же поступить иначе — использовать механизм сессий. Механизм этот работает следующим образом: данные, присланные пользователем, сервер сохраняет в отдельном файле — файле сессии. С содержимым этого файла и будет производиться вся работа по изменению данных. Клиенту же выдаётся "ключ сессии — уникальный указатель на файл, содержащий данные конкретно для этого пользователя. Теперь для того, чтобы получить все данные, касающиеся этого клиента, серверу необходимо знать лишь ключ сессии. Достоинством этого метода является удобство и скорость его использования (см. рисунок 3).

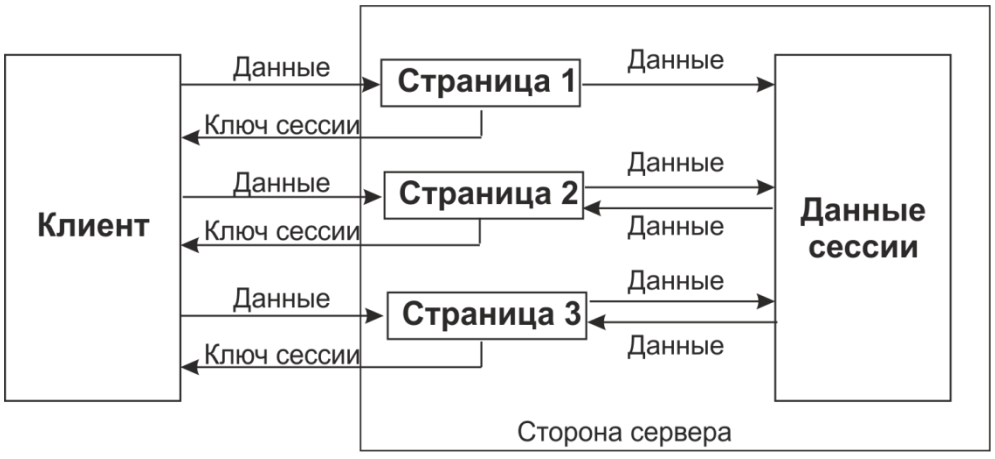

Рис. 3 Работа сервера с клиентом с помощью механизма сессий

Во время работы приложения контейнер загружает и инициализирует сервлет (сервлеты) и управляет его *жизненным циклом*.

Когда мы говорим о том, что сервлеты имеют жизненный цикл, это просто означает, что при активизации сервлета все работает предсказуемо —для любого создаваемого сервлета всегда будут вызываться определенные методы в определенном порядке (см. рисунок 4).

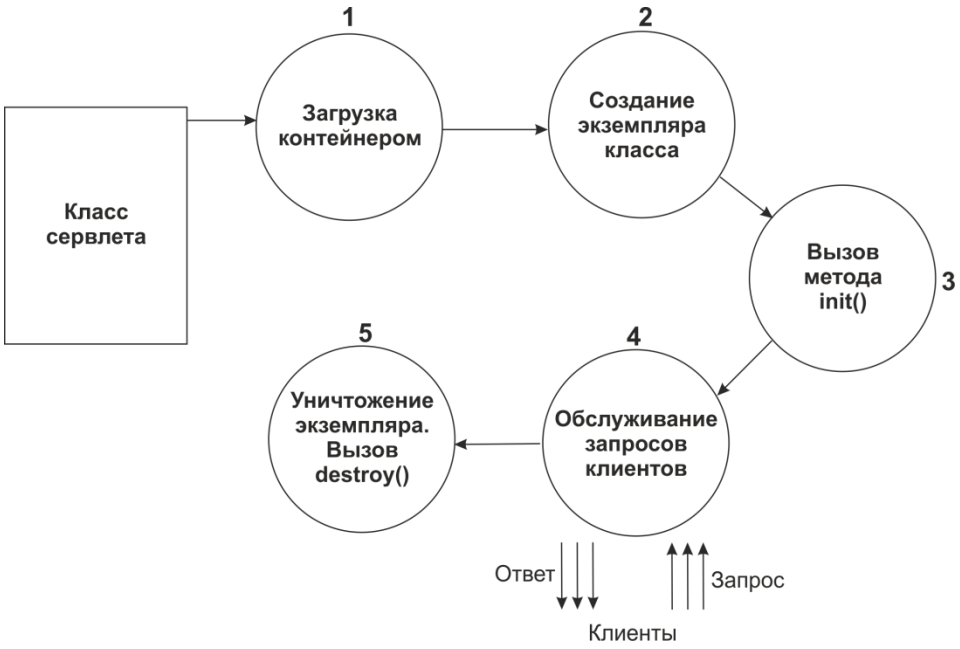

Рис. 4 Жизненный цикл сервлета

Вот обычный сценарий для жизненного цикла:

- Пользователь вводит URL в браузере. Конфигурационный файл Web-сервера указывает, что этот URL предназначен для сервлета, управляемого контейнером сервлетов на сервере.
- Если экземпляр сервлета еще не был создан (существует только один экземпляр сервлета для приложения), контейнер загружает класс и создает экземпляр объекта.
- Контейнер вызывает метод init() сервлета.
- Контейнер вызывает метод service() сервлета и передает HttpServletRequest и HttpServletResponse.
- Сервлет обычно обращается к элементам запроса, передает запрос другим серверным классам для выполнения запрошенной службы и для доступа к

таким ресурсам, как базы данных, а затем создает ответ, используя эту информацию.

- При необходимости, когда сервлет выполнил полезную работу, контейнер вызывает метод destroy() сервлета для его финализации.

Практически всегда вместе с технологией сервлетов используется технология JSP (Java server pages).

JSP — технология, позволяющая WEB-разработчикам легко создавать содержимое, которое имеет как статические, так и динамические компоненты. По сути, страница JSP является текстовым документом, который содержит текст двух типов: статические исходные данные, которые могут быть оформлены в одном из текстовых форматов HTML, SVG, WML, или XML, и JSP элементы, которые конструируют динамическое содержимое. Кроме этого могут использоваться библиотеки JSP тегов, а также EL — Expression Language — для внедрения Java-кода в статичное содержимое JSP-страниц. JSP позволяют отделить динамическую часть страниц от статического HTML. Процедура довольно проста, создаётся обычный код HTML (статический), а динамическая часть заключается в специальные теги "<% %>". JSP страницы имеют расширение .jsp. Их структура состоит из пяти конструкций: HTML, комментарии, скриптовые элементы, директивы и действия. JSP страница при компиляции преобразуется в обычный сервлет со статическим содержимым, которое направляется в поток вывода, связанный с методом service.