Приложение Java, запускаемое и выполняемое в контейнере сервера прило- жений. Клиент общается с таким приложением посредством веб-браузера. Никаких дополнительных приложений на стороне клиента устанавливать не тре- буется. Сервлеты поддерживаются виртуальной машиной JVM, что предотвра- щает утечки памяти и обеспечивает функционирование garbage collection. Каждому клиенту сервлет выделяет независимый поток выполнения. Клиент посылает приложению HTTP-запрос, сервлет генерирует ответ и возвращает его клиенту в виде html-документа. Сервлет: — компонент приложений Java Enterprise Edition; — загружается веб-сервером в контейнер; — выполняется на стороне сервера; — обрабатывает клиентские запросы; — динамически генерирует ответы на запросы; — находится в состоянии ожидания, если запросы отсутствуют; — принимает запросы от других сервлетов (Servlet chaining); — поддерживает соединения с ресурсами. Наибольшее распространение получили сервлеты, обрабатывающие кли- ентские запросы по протоколу HTTP. Технология сервлетов является оболоч- кой протокола HTTP и поддерживает его как транспорт передачи данных от клиента серверу и обратно. Контейнер сервлетов поддерживает также про- токол HTTPS (HTTP и SSL) для защищаемых запросов. Сервлеты в промышленном программировании используются для: — приема входящих данных от клиента; — взаимодействия с бизнес-логикой системы; — динамической генерации ответа клиенту.

Страница JSP ответственна за формирование пользовательского интерфейса и отображение информации, пере- данной с сервера. Сервлет выполняет роль контроллера запросов и ответов, то есть принимает запросы от всех связанных с ним JSP-страниц, вызывает соответствую- щую бизнес-логику для их (запросов) обработки и в зависимости от результата вы- полнения решает, какую JSP поставить этому результату в соответствие

Технология Java Server Pages (JSP) была разработана компанией Sun Microsystems (в настоящее время поглощена компанией Oracle), чтобы облегчить создание страниц с динамическим содержанием. В то время как сервлеты наилучшим образом подходят для выполнения контролирующей функции приложения в виде обработки запросов и определе- ния содержания и вида ответа, страницы JSP выполняют функцию отображения результатов работы приложения в виде текстовых документов типа HTML, XML, WML и некоторых других. JSP поддерживают как JavaScript, так и HTML-теги. JavaScript обычно используется, чтобы добавить функциональные возможно- сти на уровне HTML-страницы. Принято разделять динамическое и статическое содержимое JSP. Динамические ресурсы. Результаты их деятельности изменяются во время выполнения приложения. Обычно представлены в виде выражений Expression Language, библиотек тегов и тегов разработчика. Статические ресурсы. Не изменяются сами в процессе работы (HTML, JavaScript, изображения и т. д.). Смысл разделения динамического и статического содержания в том, что статические ресурсы могут находиться под управлением HTTP-сервера в то время, как динамические нуждаются в движке (JSP Engine) и в большинст- ве случаев в доступе к уровню данных. Рекомендуется разделить и разрабатывать параллельно две части приложе- ния: часть, состоящяя только из динамических ресурсов, и часть, состоящяя только из статических ресурсов. Некоторые преимущества использования JSP-технологии над другими ме- тодами создания динамического содержания страниц: — разделение динамического и статического содержания. Возможность раз- делить логику приложения и дизайн веб-страницы снижает сложность раз- работки веб-приложений и упрощает их поддержку; ТЕХНОЛОГИИ РАЗРАБОТКИ WEB–ПРИЛОЖЕНИЙ 486 — независимость от платформы. Технологии Java не зависят от платформы, следовательно, JSP могут выполняться практически на любом веб-сервере. Разрабатывать JSP можно на любой платформе; — многократное использование компонентов. Использование JavaBeans и Enterprise JavaBeans (EJB) позволяет многократно использовать компоненты, что ускоря- ет создание веб-приложений; — теги. Спецификация JSP содержит библиотеку стандартных тегов JSTL, по- зволяет разработчику создавать собственные теги, кроме того, теги обеспе- чивают возможность использования JavaBean и обращение к классам биз- нес-логики. Содержимое Java Server Pages (теги HTML, теги JSP и скрипты) переводит- ся в сервлет код-сервером. Этот процесс ответствен за трансляцию как динамических, так и статических элементов, объявленных внутри файла JSP. Об архи- тектуре сайтов, использующих JSP/Servlet-технологии, часто говорят, как о thin-client (использование ресурсов клиента незначительно), потому что большая часть логики выполняется на сервере.

Жизненный цикл Процессы, выполняемые с файлом JSP при первом вызове: — браузер делает запрос к странице JSP; — JSP-engine анализирует содержание файла JSP и создает сервлет с кодом, основанным на исходном тексте файла JSP, при этом engine транслирует статическое содержимое в методы вывода и помещает его в метод \_jspService(). Полученный сервлет будет ответственен за генерацию статических элемен- тов, определенных во время разработки. Динамические элементы трансли- руются в java-код; — код сервлета компилируется в файл \*.class. и загружается в контейнер. В итоге сервлет на основе JSP установлен и готов к работе; — выполняется метод init() сервлета; — вызываeтся метод \_jspService(), и сервлет логически исполняется, форми- руя экземпляр response; — комбинация статического HTML и графики вместе с результатами исполне- ния динамических элементов, определенных в оригинале JSP, пересылают- ся браузеру через выходной поток объекта ответа HttpServletResponse. Следующие обращения к файлу JSP просто вызовут метод \_jspService() сер- влета. Сервлет используется до тех пор, пока сервер не будет остановлен или сер- влет не будет выгружен из контейнера. Результат работы JSP можно легко предста- вить, зная правила трансляции JSP в сервлет, в частности, в его метод \_jspService().

Взаимодействие JSP — сервлет — JSP

В большинстве приложений используются не сер- влеты или JSP по отдельности, а их архитектурное взаимодействие. Страница JSP представляет вид для результатов выполнения запроса клиента, а сервлет отвечает за вызов классов бизнес-логики и передачу результатов выполнения бизнес-логики в соответству- ющую JSP и ее вызов. Т.е. сервлеты не генерируют от- вета сами, а только выступают в роли контроллера за- просов. Такая архитектура построения приложений носит название MVC (Model/View/Controller). Model — классы бизнес-логики и длительного хранения, View — страницы JSP, Controller — сервлет. Применение шаблона на практике приводит к тому, что трехуровневая базовая модель распадается на мно- гоуровневую. Число и назначение слоев может существенно отличаться в зависимости от разработанного архитектурного решения. В предлагаемом ниже подходе добавляются: уро- вень служб, обрабатывающих объект запроса, уровень логики, имплементирующий конкретные бизнес-про- цессы приложения и уровень организации взаимодей- ствия с базой данных.

