Java Servlets

Лекция 10

Введение

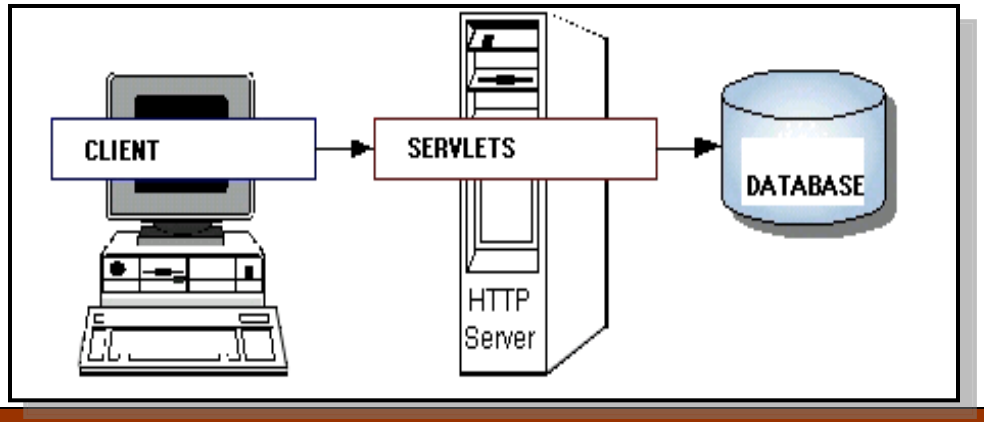
- **Сервлет** – это Java-приложение, выполняемое на стороне сервера и взаимодействующее с клиентами посредством принципа запрос-ответ .
- **Сервлет** – это компонент приложений Java Enterprise Edition, выполняющийся на стороне сервера, способный обрабатывать клиентские запросы и динамически генерировать ответы на них. Наибольшее распространение получили сервлеты, обрабатывающие клиентские запросы по протоколу HTTP.

Альтернативы сервлетам

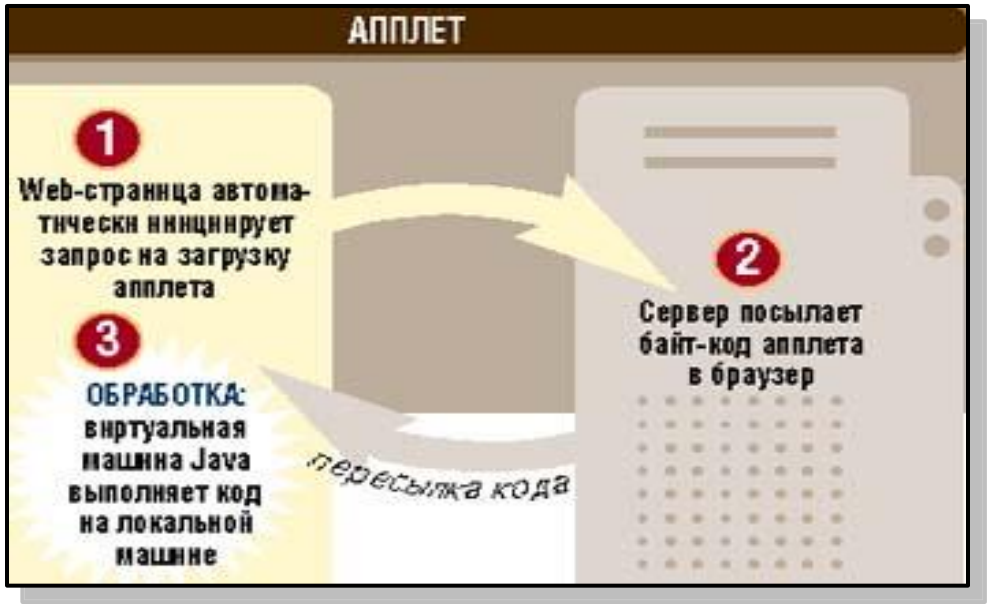
- Сервлеты не являются единственным методом работы с Web-страницами.
- До появления сервлетов существовали альтернативные возможности создания Web-приложений:
  - **Common Gateway Interface (CGI)**
  - **ASP**
- Сервлеты являются альтернативой всем этим технологиям и имеют несколько преимуществ:
  - Они также платформо-независимы, как 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 (thread) для запросов, а не отдельные процессы.
  - Существует распространенная отраслевая поддержка сервлетов, включая *контейнеры* для наиболее популярных Web-серверов и серверов приложений.

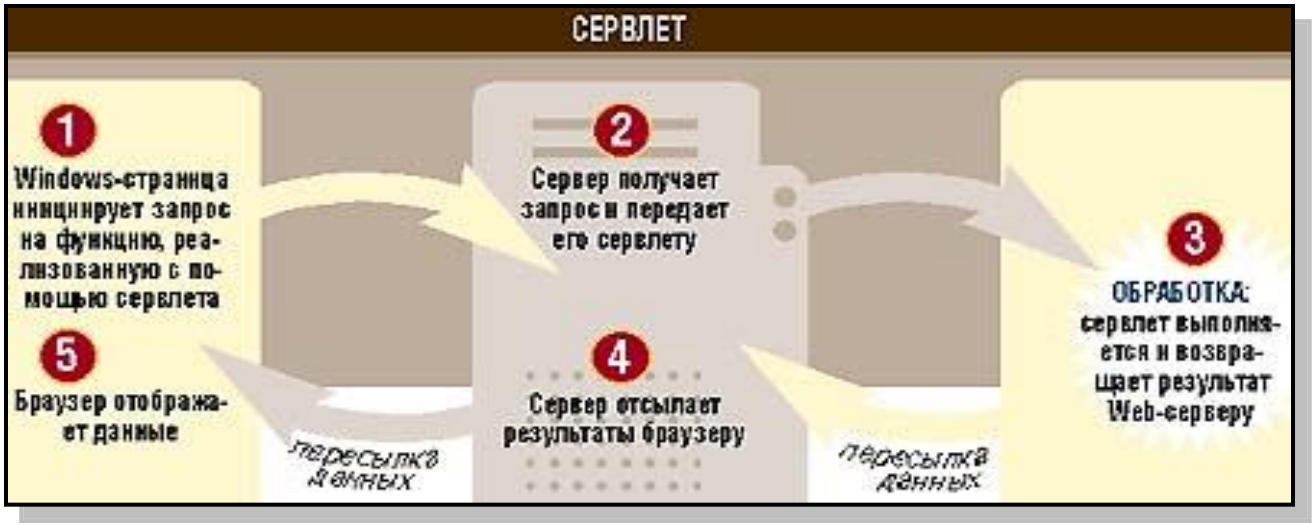
Web-приложения

- **Апплеты** - это программы на Java, которые, как правило, предназначены для того, чтобы загружаться посредством браузера, а затем работать в окне браузера.
- Апплеты могут использоваться для создания богатых графикой и интерактивными возможностями пользовательских интерфейсов, которые не способны выразить средствами обычного языка разметки HTML.
- **Сервлеты** - программы на Java, которые работают на Web-серверах Java или серверах приложений Java.
- Как и программы CGI, сервлеты могут доставлять Web-службы непосредственно в браузер или действовать как промежуточное ПО, которое связывает браузер с серверными службами.



Web-приложения





Работа сервлетов

- Большинство Java-сервлетов предназначены для ответов на HTTP-запросы в контексте Web-приложения. Для этого необходимо использовать HTTP-классы из пакетов **javax.servlet** и **javax.servlet.http**.
- При создании Java-сервлета обычно создается подкласс **HttpServlet**, который имеет методы, предоставляющие доступ к конвертам запроса и ответа для обработки запросов и создания ответов.
- **HTTP-протокол**, естественно, не связан с Java. Это просто спецификация, определяющая, какими должны быть запросы и ответы.

Работа сервлетов

- Когда пользователь инициирует запрос через URL, классы Java-сервлетов преобразуют их в **HttpServletRequest** и передают адресату, указанному в URL, как определено в конфигурационных файлах конкретного контейнера сервлетов, который вы используете.
- Когда сервер завершает свою работу с запросом, Java Runtime Environment упаковывает результаты в **HttpServletResponse** и затем передает HTTP-ответ обратно клиенту, пославшему запрос.
- Взаимодействуя с Web-приложением, вы обычно посылаете несколько запросов и получаете несколько ответов. Все они работают в контексте *сессии*, которая в языке Java заключается в объект **HttpSession**.
- *Контейнер*, например **Tomcat**, управляет средой исполнения для сервлетов.
- Во время работы вашего приложения контейнер загружает и инициализирует ваш сервлет и управляет его **жизненным циклом**.

Жизненный цикл сервлета

- Когда мы говорим о том, что сервлеты имеют жизненный цикл, это просто означает, что при активизации сервлета все работает предсказуемо.
- Для любого сервлета всегда будут вызываться определенные методы в определенном порядке:
  - Класс сервлета загружается контейнером (контейнер загружает класс и создает экземпляр объекта).
  - Контейнер вызывает метод **init()** сервлета. Этот метод инициализирует сервлет и вызывается в первую очередь, до того, как сервлет сможет обслуживать запросы. За весь жизненный цикл метод **init()** вызывается только однажды.
  - 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.

Преимущества использования сервлетов

- **Переносимость**

  Сервлеты являются переносимыми между операционными системами и Веб-серверами. Можно разработать сервлет на машине с Windows, на которой работает Java Web-сервер, а затем развернуть этот сервлет на Unix-сервере.

- **Эффективность**

  Сервлеты обладают высокой эффективностью. Однократно загруженные, они практически постоянно остаются в пространстве памяти сервера.

  Таким образом, сервлет поддерживает и сохраняет свое состояние, следовательно, состояние внешних ресурсов(соединения с базами данных, которые в противном случае требуют дополнительного времени на установку). Кроме того, в отличие от CGI, сервлет может одновременно выполнять многочисленные запросы с помощью потоков.

Преимущества использования сервлетов

- **Мощные функциональные возможности**

  Сервлетам присущи в полной мере все сильные стороны ядра прикладного интерфейса Java API: поддержка сетевых концепций, многопоточность, обработка изображений, сжатие данных, поддержка соединений с базами данных, RMI, Corba и т.д.

- **Безопасность**

  Сервлеты обеспечивают безопасную технологию программирования. Автоматический механизм сборки мусора языка Java и отсутствие указателей защищает сервлеты от проблем с управлением памятью, таких как некорректные ссылки, указатели на неиспользуемую память и т.д. Они также могут обеспечить безопасность при обработке ошибок благодаря механизму обработки исключения языка Java.

- **Способность к интеграции**

  Сервлеты прочно связаны с сервером. Следовательно, они могут работать совместно с сервером гораздо эффективнее, чем CGI-программы. Сервлет может использовать сервер для преобразования путей к файлам, для входной регистрации, для выполнения проверок при авторизации и т.д.

Java Server Pages

- Сервлеты и **JSP** представляют собой **Java-**технологии, предназначенные для разработки серверных web-сценариев.
- Технология **Java Server Pages** позволяет смешивать обычный, статический HTML-код с динамически сгенерированным содержимым сервлетов (подобно технологиям **PHP** и **ASP**).
- Для работы с сервлетами и **JSP** требуется установить web-сервер, поддерживающий **Java**. Некоторые серверы специально разработаны с поддержкой **Java**, другие позволяют устанавливать для этой цели дополнительные модули (plug-ins).

Что нужно для разработки сервлетов

- Рассмотрим известный web-сервер **Apache Tomcat**.
  - Для его установки нужно просто распаковать архив в некоторый каталог, например, **C:\TomCat**.
  - По умолчанию этот сервер использует порт 8080, поэтому для просмотра страниц из браузера нужно использовать адрес **http://localhost:8080**
- Для запуска сервера используется пакетный файл **startup.bat** из каталога **BIN**.
- Сервер **TomCat** работает на Java-платформе, поэтому на компьютере также должен быть установлен JDK.
  - Чтобы указать серверу, где именно установлен JDK, в файле **startup.bat** нужно инициализировать переменную **JAVA_HOME**, например, так (если JDK установлен в каталог C:\j2sdk1.4.0): set JAVA_HOME=C:\j2sdk1.4.0
- Корневым каталогом для **TomCat** является

  C:\TOMCAT\WEBAPPS\ROOT
- Наберите в адресной строке браузера

  http://localhost:8080
- Браузер загрузит страницу **index.html** из этого каталога.

Основы HTTP Servlet

- HTTP генерирует HTML-страницы.
- Вы можете встроить HTTP-сервлет в HTML-страницу.
- HTTP является протоколом без сохранения состояния сеанса.
- Используются методы **get()** и **post()**.
- Сервлеты используют классы и интерфейсы из двух пакетов **javax.servlet** и **javax.servlet.http**.
- Пакет **javax.servlet** содержит классы, которые поддерживают общие, универсальные, независимые от протокола сервлеты.
- Эти классы расширяются в пакете **javax.servlet.http** для обеспечения функциональности протокола HTTP.

Сервлеты

- Любой класс сервлета, используемый для генерации HTML-страницы, должен реализовывать метод **doGet()** или **doPost().**
- Класс **HttpServlet** предоставляет для каждого типа HTTP-запроса свой do-метод.
- Методы:
  - protected void **processRequest**(HttpServletRequest request, HttpServletResponse response) {…}
  - protected void **doGet**(HttpServletRequest request, HttpServletResponse response) {…}
  - protected void **doPost**(HttpServletRequest request, HttpServletResponse response) {…}

Пример сервлета

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {
   public void doGet (HttpServletRequest request, HttpServletResponse response)
                                                        throws ServletException, IOException
     {
          response.setContentType("text/html");
          PrintWriter out = response.getWriter();
          out.println(
                    "<HTML>\n" +
                     "<HEAD><TITLE>Hello, World! "+
                     "</TITLE></HEAD>\n" +
                     "<BODY>\n" +
                     "<H1>Hello, World!</H1>\n" +
                     "</BODY></HTML>");
     }
}
```

Разбор примера

- Сначала в этом примере импортируются дополнительные библиотеки классов для сервлетов. Эти библиотеки находятся в архиве servlet.jar, расположенном в каталоге LIB сервера TomCat.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
```

- Любой сервлет является производным от класса **HttpServlet**.

```
public class HelloWorld extends HttpServlet {
```

- Обычно в нем переопределяется метод **doGet,** или **doPost**, или оба эти метода (можно переопределить один из них и вызвать его из другого). Эти методы могут получать параметры запроса (с помощью своего первого параметра – объекта **request**), а затем формируют ответ сервера (с помощью второго параметра – объекта **response**).

```
public void doGet (HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException
```

Разбор примера

- Первой строкой ответа должно быть формирование заголовка **Content-Type**:

```
response.setContentType("text/html");
```

- Затем получаем ссылку на выходной поток сервлета:

```
PrintWriter out = response.getWriter();
```

и выводим любую информацию в этот выходной поток (в данном случае - простой HTML-код).

```
out.println(
        "<HTML>\n" +
         "<HEAD><TITLE>Hello, World! "+
         "</TITLE></HEAD>\n" +
         "<BODY>\n" +
         "<H1>Hello, World!</H1>\n" +
         "</BODY></HTML>");
```

Отправка информации

- Сервлеты возвращают обычный HTML-ответ.
- Они возвращают клиенту следующие данные:
  - Одиночный код состояния
  - Некоторое количество HTTP-заголовков
  - Тело ответа
- Код состояния является целочисленным значением, которое описывает состояние ответа. Он сообщает об успешности или отказе (сбое) при формировании ответа.

Коды состояния

- Наиболее часто используемые коды состояния определены, как мнемонические константы в классе **HttpServletResponse**. В таблице ниже показаны некоторые коды состояния:

Мнемонические константы	Код состояния	Сообщение
SC_OK	200	Ok (Документ успешно загружен)
SC_NO_CONTENT	204	No Content (Нет содержимого)
SC_MOVED_PERMANENTLY	301	Moved Permanently (Документ перемещён постоянно)
SC_MOVED_TEMPORARILY	302	Moved Temporarily (Документ перемещён временно)
SC_NOT_FOUND	404	Not found (Документ не найден)
SC_UNAUTHORIZED	401	Unauthorized (Несанкционированный доступ)

Контроль сеанса (Session Tracking)

- Это механизм, который сервлеты используют для сохранения и обслуживания состояния при работе с последовательностью запросов от одного и того же пользователя (точнее, запросов, исходящих от одного и того же экземпляра браузера) в течение определённого интервала времени.

- Чтобы осуществлять контроль сеанса необходимо выполнить последовательность шагов, описанных ниже:
  - Инициализировать сеанс (объект **HttpSession**) для пользователя
  - Сохранить данные или получить данные из объекта **HttpSession**
  - Завершить сеанс (необязательная операция)

Контроль сеанса (Session Tracking)

- **Сеанс** (**сессия**) – это процесс взаимодействия, протекающий между клиентом и web-приложением.
- Основу средств создания сеанса с клиентом составляет интерфейс **HttpSession**.
- Объект типа **HttpSession** формируется контейнером сервлета при получении запроса, а создать его можно методом

  public HttpSession getSession (boolean create),

  описанным в интерфейсе **HttpServletRequest**.
- Чтобы осуществлять контроль сеанса необходимо выполнить последовательность шагов:
  - Инициализировать сеанс (объект **HttpSession**) для пользователя
  - Сохранить данные или получить данные из объекта **HttpSession**
  - Завершить сеанс (необязательная операция)

Использование закладок Cookies

- Серверы (или сервлеты, как часть серверов) используют закладки **cookies** для отсылки информации клиенту с целью её сохранения и последующего извлечения (считывания) у того же клиента.
- **Cookie** – это текстовые данные, которые посылаются сервлетами как часть ответов клиентам. В зависимости от максимального возраста cookie, браузер либо хранит их в течение сеанса (сессии), либо сохраняет на компьютере клиента для последующего использования.
- **Cookie** – это механизм, который использует сервлет для того, чтобы у клиентов сохранялось некоторое количество информации состояния, ассоциируемой с пользователем.
- Поля cookie представляют собой фрагменты данных, управляемые браузером, обычно с целью управления сессией. Поскольку HTTP-соединения не сохраняют состояния, можно использовать cookie для записи постоянной информации между несколькими HTTP-соединениями.

Использование закладок Cookies

- Используется **javax.servlet.http.Cookie**
- Вы также можете предоставить дополнительные атрибуты закладок cookies, например, комментарии.
- Чтобы послать закладку cookie, выполните шаги, описанные ниже:
  - Создать экземпляр объекта **cookie**
  - Установить его атрибуты
  - Послать закладку cookie
- Чтобы получить информацию из закладки cookie, выполните следующие шаги, описанные ниже:
  - Извлечь все закладки cookies из запроса пользователя.
  - Использовать стандартные приёмы программирования, чтобы найти закладку cookie или несколько закладок cookies с именем, которое вам нужно.

Интерсервлеты

- **Интерсервлеты** – это сервлеты, расположенные в пределах одного и того же сервера.
- Сервлеты обладают разнообразными способами осуществления обмена информацией друг с другом в пределах одного и того же сервера.
- Сервлеты используют специализированный обмен информацией между интерсервлетами (interservlet communication) по следующим причинам:
  - Любой сервлет может получить доступ к другим загруженным сервлетам, а следовательно может выполнять определённые задачи в этих сервлетах.
  - Любой сервлет может повторно (многократно) использовать свойства и возможности другого сервлета для выполнения определённых задач.
  - Два или более сервлетов могут совместно использовать информацию о состоянии.

Установка соединения сервлета с базой данных

- В настоящее время Web-сайты не ограничиваются выводом информации в виде статических HTML-страниц.
- Покупатель заполняет форму на оплату и доставку товаров.
- Информация, полученная от покупателя, сохраняется в базе данных с целью обслуживания записей.
- Современный бизнес приходит к этой модели обслуживания записей данных и выполнения транзакций.
- Сервлеты, с их продолжительным жизненным циклом, и JDBC (Java Database Connectivity), хорошо организованный, независимый от конкретной базы данных прикладной интерфейс (API) соединения, являются элегантным и эффективным решением для Web-программистов, которым необходимо связывать свои сайты с различными узлами баз данных.