Мультимедийный учебный курс

Программирование на Java

Часть 2. Программирование клиент-серверных Java-приложений

Лекция 1 ч.1

Сервлеты (базовые средства)

Автор:

- Борисенко В. П.

Лаборатория Интернет-технологий

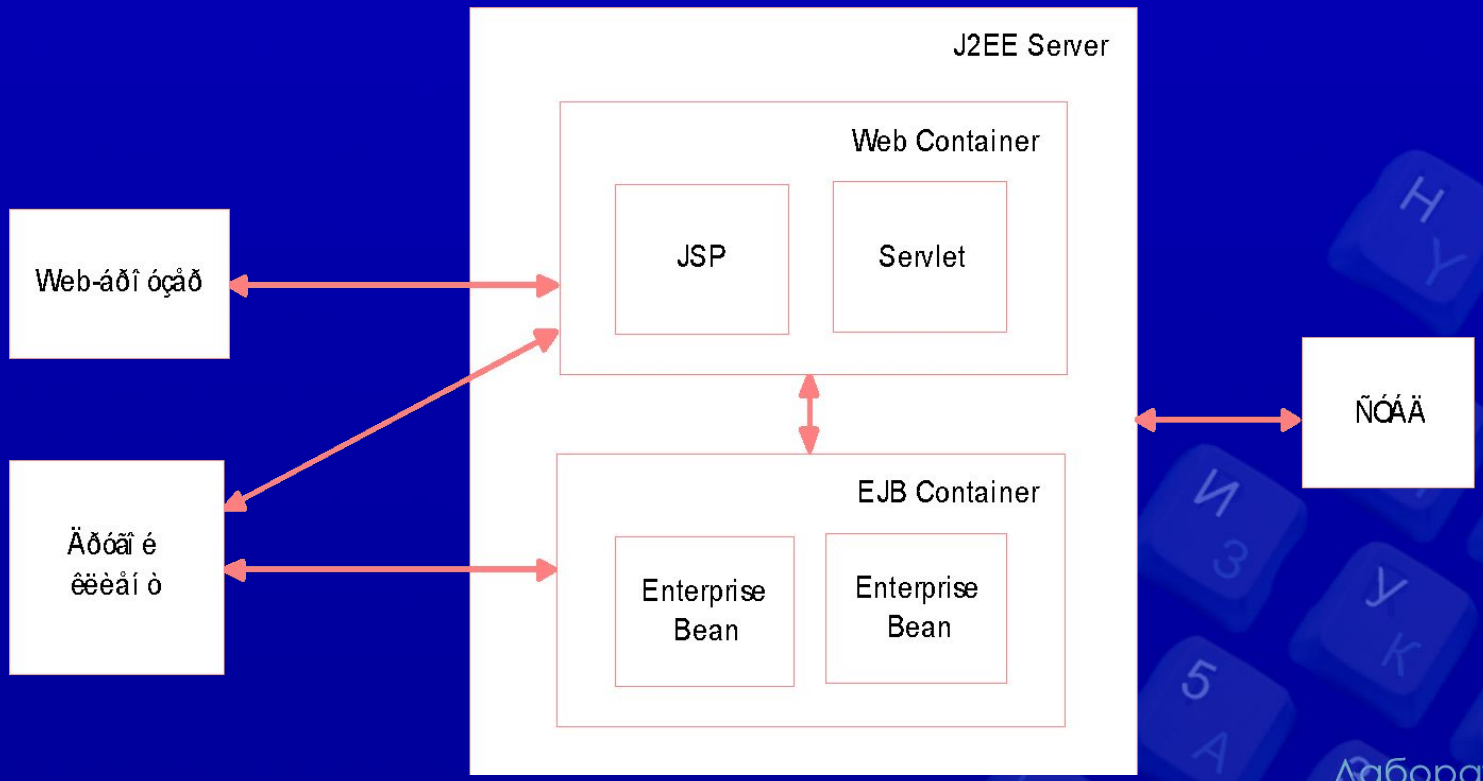
Понятие Web-компонента

- Согласно спецификации J2EE Web-компонентами являются
  - Сервлеты
  - JSP-страницы
  - Фильтры
  - Слушатели событий
- Управление работой web-компонентов возлагается на Web-контейнер (сервер Web-приложений)
- Спецификация J2EE содержит основные понятия и сведения о технологиях, применяемых в рамках J2EE-платформы. Спецификация доступна по адресу:

  http://www.oracle.com/technetwork/java/javaee/downloads/index.html

Что такое Web-контейнер?

- Web-контейнер – это инструментальный программный модуль, который управляет сервлетами и JSP-страницами, который работает в среде J2EE-сервера



Понятие Web-контейнера

- Web-контейнер –стандартизованный компонент, который занимается системной поддержкой прикладных программных компонентов и обеспечивает их жизненный цикл в соответствии с правилами, определенными в спецификациях

- Функции Web-контейнера
  - Управление жизненным циклом компонентов
  - Управление безопасностью
  - Управление конкурентным доступом
  - Перенаправление запросов

Jakarta Tomcat 6.x, 7.x и 8.x

- входит в состав Sun Reference Implementation

- Линейка 6.x (version 6.0.43, 14.01.2014)
  - Спецификация Servlets – 2.5
  - Спецификация JSP – 2.1

- Линейка 7.x (version 7.0.59, 02.04.2014)
  - Спецификация Servlets – 3.0
  - Спецификация JSP – 2.2

- Линейка 8.x (version 8.0.32, 02.04. 2016)
  - Спецификация Servlets – 3.1
  - Спецификация JSP – 2.3

Получение и установка Tomcat

- http://jakarta.apache.org
- Полностью написан на Java
- Доступен:
  - Binary Distributions
    - zip
    - tar.gz
    - 32-bit Windows zip
    - 64-bit Windows zip
    - 64-bit Itanium Windows zip
    - 32-bit/64-bit Windows Service Installer
  - Full documentation:
    - tar.gz

Запуск/останов Tomcat

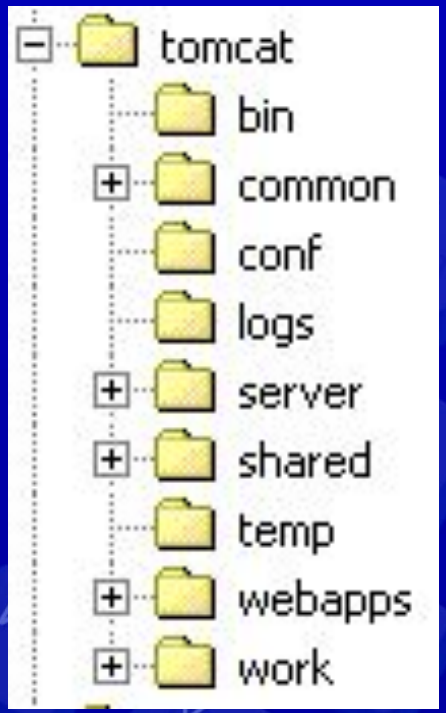
- JAVA_HOME=C:\jdk1.7

- /bin/startup.bat
- /bin/catalina.bat start

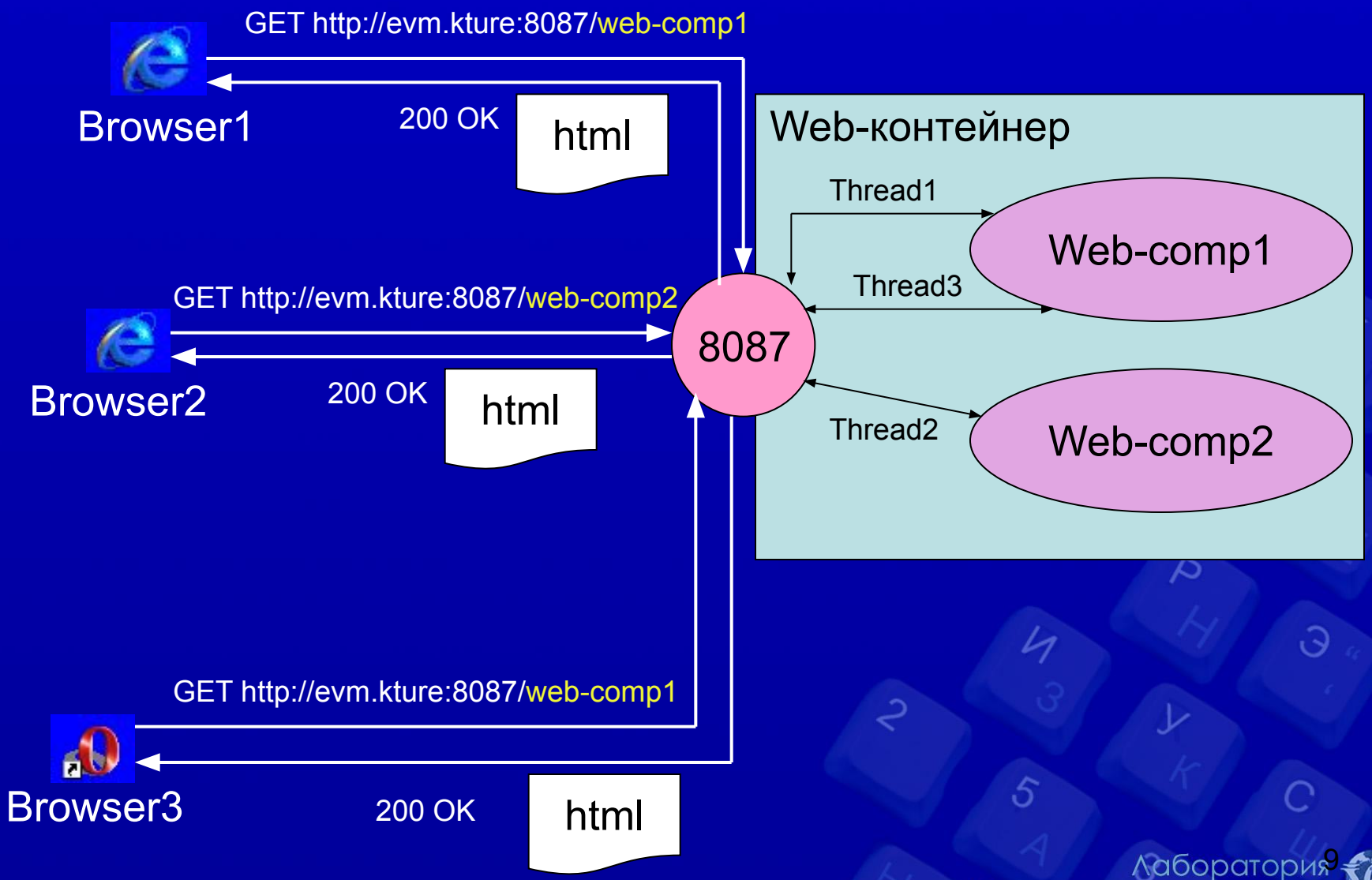
- /bin/shutdown.bat

Структура каталогов

- **/bin** – содержит исполняемые jar'ы и соответствующие скриптовые/командные файлы для windows (.bat) и для *nix (.sh);
- **/common** – классы и библиотеки (jar'ы), которые доступны как внутренним классам Tomcat'а, так и всем Web-приложениям
- **/conf** – содержит конфигурационные файлы для настройки Tomcat'а (основная конфигурация в файле **server.xml**)
- /logs – журнальные файлы
- **/server** – классы и библиотеки, необходимые для работы Tomcat'а
- **/shared** – классы и библиотеки, общие для всех Web-приложений
- **/temp** – используется JVM для хранения временных файлов
- **/webapps** – каталог по умолчанию для размещения пользовательских Web-приложений
- **/work** – хранит временные (рабочие) файлы Tomcat'а, в том числе скомпилированные JSP



Работа нескольких Web-компонентов в одном Web-контейнере



Сервлет

- Сервлет - это самостоятельный Web-компонент, который, согласно спецификации J2EE, функционирует под управлением Web-контейнера
- Сервлет в ответ на полученный от клиента HTTP-запрос динамически генерирует HTTP-ответ: HTML-страницу, XML-документ или другой документ, контент которого корректно отображается в стандартном браузере
- Спецификация технологии сервлетов находится по адресам:
  http://www.oracle.com/technetwork/java/index-jsp-135475.html
  http://jcp.org/en/jsr/detail?id=315
  - Последняя версия спецификации – 3.2

Сервлет

- Сервлет не имеет привязки к определенному серверу приложений
- Взаимодействие с клиентом происходит только через контейнер
- Сервлеты могут быть применены для обработки запросов любого вида
- Обычно сервлеты применяют для обработки HTTP запросов

Сервлет

- Функциональность сервлета программируют в классе сервлета

- Для того, чтобы класс был сервлетом он должен реализовывать интерфейс javax.servlet.Servlet

Сервлет с точки зрения Java

- Главный класс сервлета должен реализовывать интерфейс javax.servlet.Servlet или расширять класс, реализующий данный интерфейс

- Библиотека Java Servlet API предлагает два класса, реализующих интерфейс Servlet - это классы javax.servlet.GenericServlet (протоколонезависимый сервлет) и javax.servlet.http.HttpServlet (HTTP-сервлет)

Сервлет с точки зрения Java

- Для работы и компиляции требуются библиотечные классы поддержки сервлетов (j2ee.jar или servlet.jar)

Пример сервлета

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

public class HelloServlet extends HttpServlet {

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

      response.setContentType("text/html"); // Установка типа документа
      PrintWriter out = response.getWriter();
      out.println("<html> <body> Доброе Утро !!! </body></html>");
      out.close();
    }

}
```

Жизненный цикл сервлета

- Жизненный цикл сервлета управляется контейнером, в котором сервлет был развернут
- Когда запрос отображается на сервлет, контейнер выполняет следующие шаги:
  - Если экземпляр сервлета не существует, Web-контейнер
    - Загружает класс сервлета
    - Создает экземпляр класса (объект) сервлета
    - Инициализирует экземпляр сервлета, вызывая метод void init(ServletConfig config)
  - Вызывает метод

    void service(ServletRequest req, ServletResponse res),

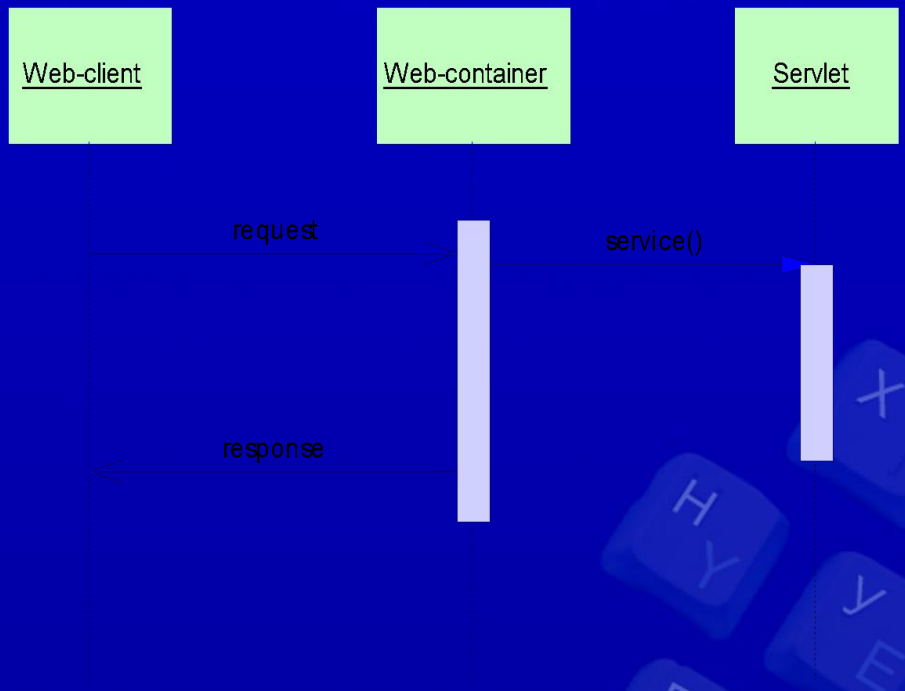
    передавая ему объекты запроса и ответа
- Если контейнеру нужно удалить сервлет, он его финализирует, вызывая метод void destroy()

Метод init()

- Метод void init(ServletConfig config) вызывается один раз в момент загрузки сервлета контейнером
- Создавая Java класс для нового сервлета, разработчик может переопределить метод init() с тем, чтобы тот выполнил определенные подготовительные работы

Метод service(…)

- Метод service() вызывается при каждом обращении клиента к сервлету
- Метод service() получает из web-контейнера запрос клиента в виде объекта ServletRequest и строит соответствующий ответ в объекте класса ServletResponse
- Создание указанных объектов на основе запроса клиента, а также передача клиенту результата обработки, содержащегося в объекте ServletResponse, возлагаются на web-контейнер



Метод destroy()

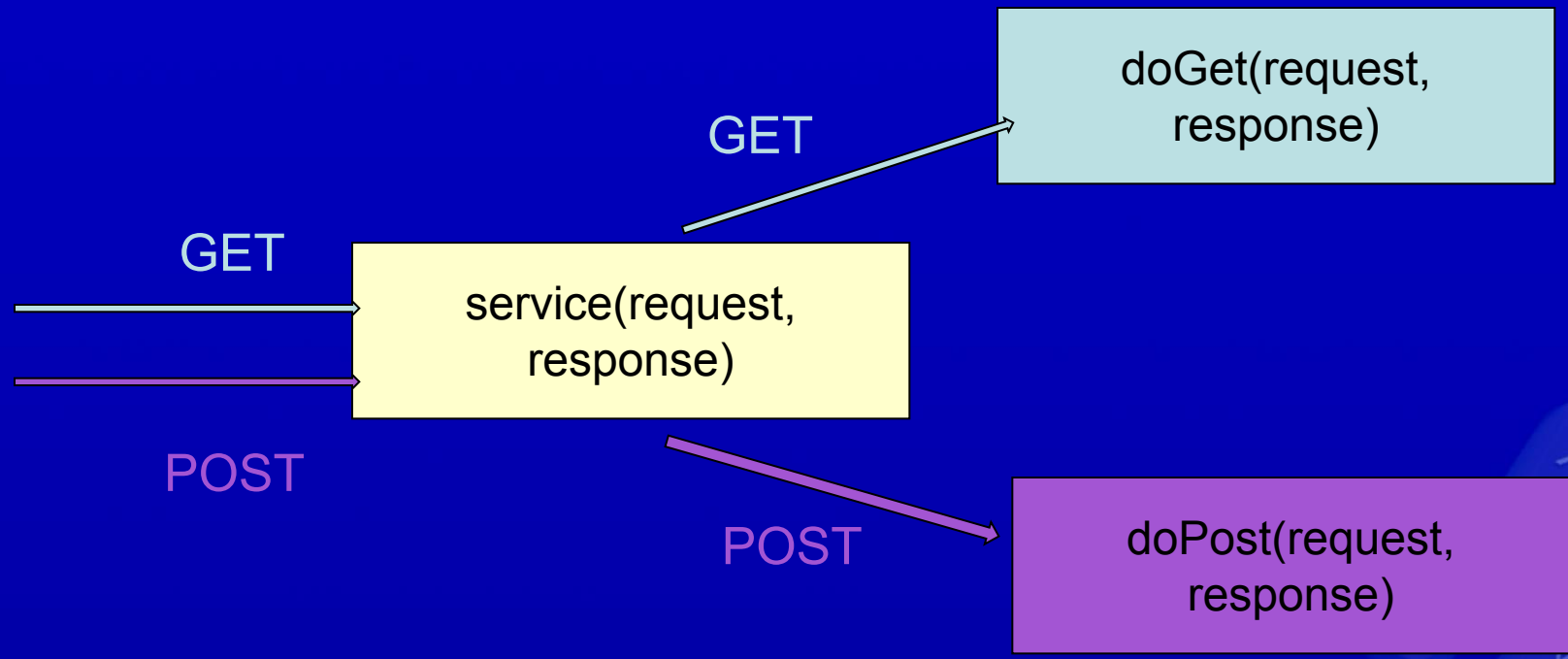
- Метод destroy() вызывается контейнером в момент уничтожения экземпляра сервлета

- Разработчик может, переопределив этот метод, произвести некоторые операции, такие как освобождение выделенных ресурсов и т.п.

Класс HttpServlet

- Предназначен для обработки запросов по протоколу HTTP
- Метод service() класса HttpServlet вызывает один из методов doXxx(), в зависимости от типа запроса:
  - doGet(HttpServletRequest req, HttpServletResponse resp) - предназначен для обработки GET-запросов
  - doPost(HttpServletRequest req, HttpServletResponse resp)- предназначен для обработки POST-запросов
  - doPut(…)
  - doDelete(…), и др.

Одинаковая обработка запросов GET и POST



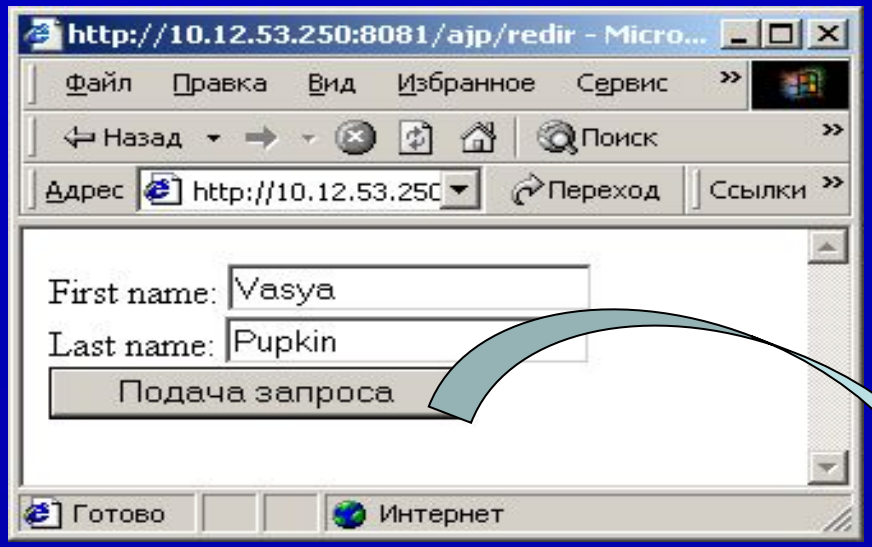
- В методе service() класса HttpServlet в зависимости от типа запроса происходит вызов методов doGet(), doPost() или др.
- Чтобы обеспечить одинаковую обработку запросов сервлетом не рекомендуется переопределять метод service(). Для этого лучше поместить обработчик в doGet(), а в doPost() просто вызвать doGet()

```
public void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
    doPost(request, response);
}
```

Интерфейс HttpServletRequest

- Потомок ServletRequest, содержит информацию о запросе клиента и передается в виде параметра методам doXxx()
- Методы:
  - String getParameter(String name)
  - get/setAttribute(String)
  - String getHeader(String)
  - InputStream getInputStream()
  - getRemoteHost()
  - getRequestURL()

Излечение данных из запроса




```
POST http://10.12.53.250:8081/ajp/redir HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel, applica
Referer: http://10.12.53.250:8081/ajp/redir
Accept-Language: ru
Content-Type: application/x-www-form-urlencoded
Proxy-Connection: Keep-Alive
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0; Lucky Net, Kiev, Ukraine)
Host: 10.12.53.250
Content-Length: 31
Pragma: no-cache

firstname=Vasya&lastname=Pupkin
```

getRequestURL()

request.getHeader("User-Agent"): "Mozilla/4.0 (compa…"

request.getParameter("lastname"): "Pupkin"

Интерфейс HttpServletResponse

- Этот интерфейс является наследником ServletResponse и содержит информацию об ответе клиенту
  - Объект с этим интерфейсом передается в виде параметра методам doXxx()

- Методы HttpServletResponse
  - addHeader(String name, String val)
  - setHeader(String name, String val)
  - getOutputStream()
  - getWriter()
  - reset()
  - sendError(int code)
  - setContentLength(int len)
  - setContentType(String type)
  - sendRedirect(String location)

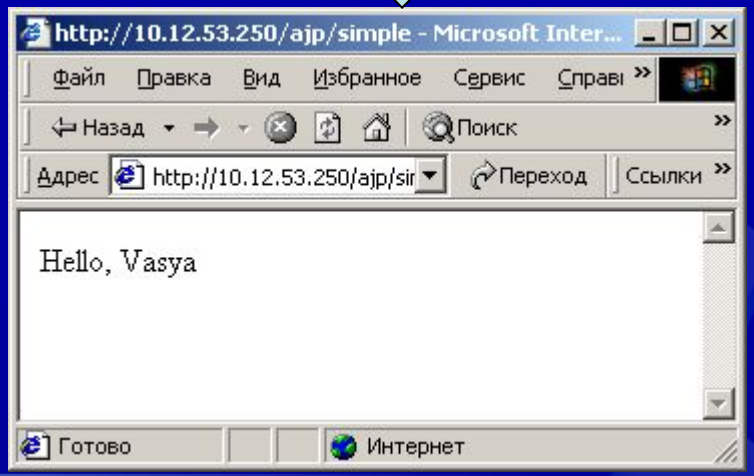
Установка данных ответа

```
HTTP/1.1 200 OK
Content-Type: text/html;charset=ISO-8859-1
Content-Language: de-DE
Content-Length: 44
Date: Tue, 15 Feb 2005 16:00:49 GMT
Server: Apache-Coyote/1.1
Connection: close

<HTML><BODY>
Hello, Vasya
</BODY></HTML>
```

response.setContentType("text/html")

response.setLocale(Locale.GERMAN);

response.setDateHeader("Date",
Calendar.getInstance().getTimeInMillis());



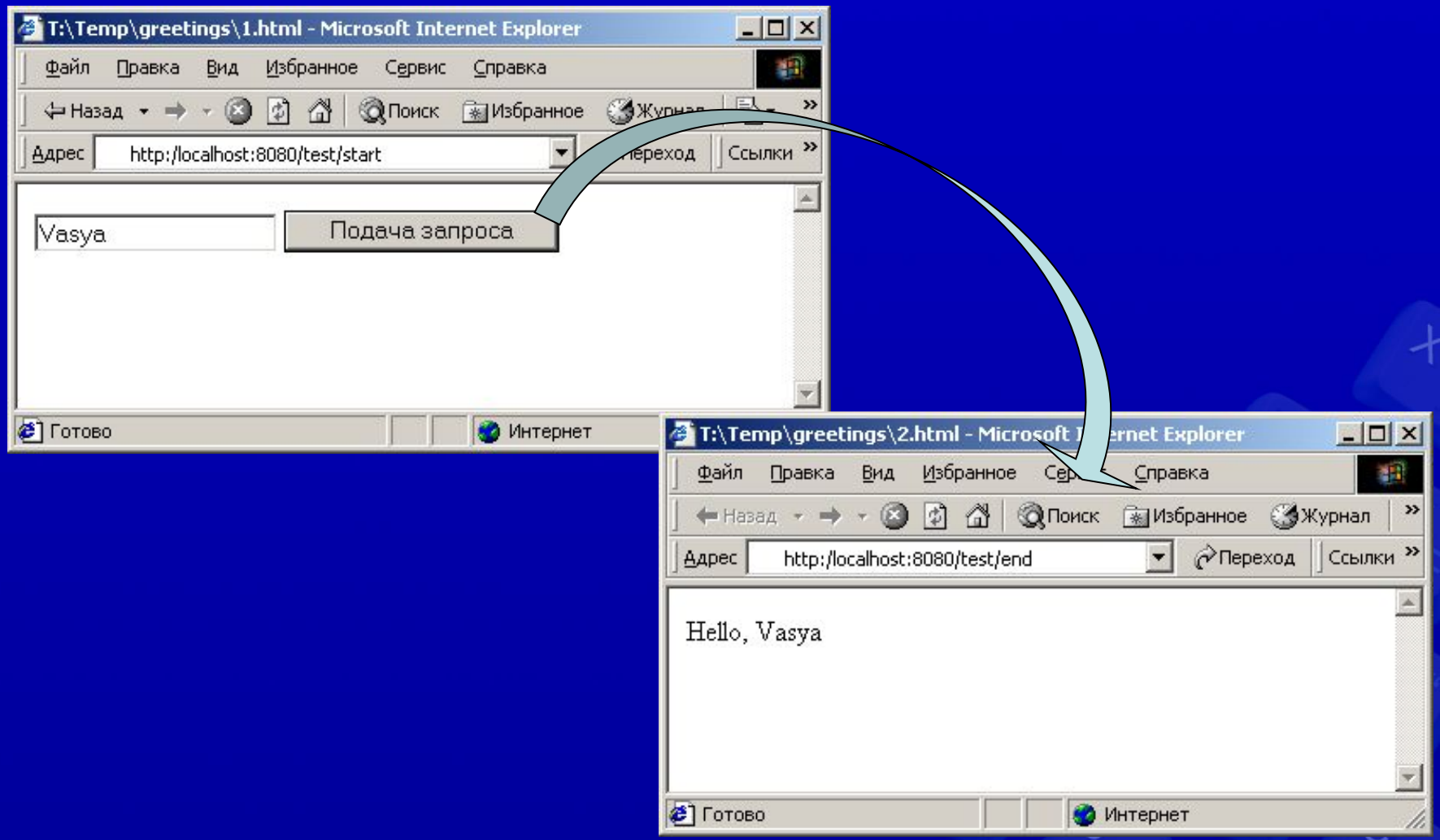
Пример обработки данных формы

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

public class MyServlet extends HttpServlet {

    /** Метод doGet служит для обработки GET-запросов */
  public void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {

     // Получаем поток вывода сервлета
   PrintWriter out = response.getWriter();
    // Генерируем форму
   out.println("<FORM method=\"POST\">");
   out.println("<INPUT type=\"text\" name=\"welcome\">");
   out.println("<input type=\"submit\"></FORM>");
  }
```

Пример обработки данных формы (продолжение)

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

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    // Получаем параметр запроса
    String welcome = request.getParameter("welcome");
    out.println("Hello, "+ welcome);
}
```

Демонстрация работы сервлета



Функции сервлетов

- Чтение данных, переданных пользователем
  - Например, из HTML-форм
- Просмотр информации о запросе, которая встроена в HTTP-запрос
  - Сведения о возможностях браузера, cookies, имени хоста клиента, делающего запрос, и т.д.
- Обращение к компонентам бизнес-логики и получение результатов
- Генерация ответа
  - В большинстве случаев предполагает формирование и вывод ответа в виде HTML-документа
- Установка параметров HTTP-ответа
- Возвращение ответа клиенту

Установка Web-приложения

- Перед запуском скомпилированного Web-приложения, его необходимо внедрить (развернуть, deploy) в Web-контейнер
- Для этого необходимо создать дескриптор развертования (deployment descriptor) – XML файл с информацией о web-приложении
- Для удобства все необходимые ресурсы для Web-приложения (классы, jsp и статические HTML-страницы, ресурсы, дескриптор поставки и т.п.) можно упаковать в Web-ARchive (WAR)
- Web-контейнер содержит **контексты**, в которые помещаются Web-приложения

Содержимое Web-приложения

- webaps/ - корневой каталог, содержит
  - JSP-файлы
  - HTML-файлы
  - Изображения
  - др. ресурсы
  - подкаталог WEB-INF (недоступен для клиента)
    - /classes – каталог с откомпилированными классами
    - /lib – каталог с библиотеками (Jar)
    - web.xml – дескриптор развертывания
- Создание WAR – так же, как JAR:

```
jar cvf archiveName.war *
```

Дескриптор развертывания web.xml

- В этом файле описывается
  - Название и описание web-приложения
  - Web-страница по умолчанию
  - Web-страница, которая будет отображаться в случае возникновения ошибки
  - Сервлеты и “маппинг” к ним
  - Параметры инициализации
  - Параметры ограничений доступа к ресурсам и т.д.

Структура дескриптора развертывния web.xml

- Файл web.xml – это дескриптор развертывания приложения. Он может содержать следующие теги:

```
<web-app>
    <display-name>My Web Application</display-name>
    <description>Description</description>

      <welcome-file-list>
        <welcome-file>myservlet</welcome-file>
      </welcome-file-list>

    <servlet> <!-- Объявление сервлета -->
    <servlet-name>MyServlet</servlet-name>
    <servlet-class>com.mycompany.MyServlet</servlet-class>
    </servlet>

    <!– Объявление “маппинга” сервлета – пути, по которому он будет доступен.
        В данном примере: http://host:port/context/myservlet -->
    <servlet-mapping>
    <servlet-name>MyServlet</servlet-name>
    <url-pattern>/myservlet</url-pattern>
    </servlet-mapping>
</web-app>
```

Здесь приведены лишь некоторые из возможных тегов дескриптора развертывания