Четвертая лекция
java for web
WEB-контейнер

Понятие Web-компонента

Согласно спецификации J2EE Web-компонентами являются

- Сервлеты
- Фильтры
- Слушатели событий
- JSP-страницы,

которые могут отвечать на HTTP-запросы Web-клиентов.

Управление работой web-компонентов возлагается на Web-контейнер

Понятие Web-контейнера

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

Функции Web-контейнера

- Управление жизненным циклом компонентов
- Перенаправление запросов
- Управление конкурентным доступом
- Управление безопасностью

Понятие сервлета

Сервлет - это самостоятельный Web-компонент, который, согласно спецификации J2EE, функционирует под управлением Web-контейнера.

Сервлет в ответ на полученный от клиента запрос динамически генерирует HTML-страницу, или другой документ.

Сервлеты являются компонентами, работающими на стороне сервера и предоставляющими мощный механизм для разработки серверных программ.

Сервлет с точки зрения Java

Все сервлеты реализуют общий интерфейс **Servlet** из пакета **javax.servlet.** Для обработки HTTP-запросов можно воспользоваться в качестве базового класса абстрактным классом HttpServlet из пакета **javax.servlet.http.**

Для этого подключим зависимости в Maven.

```
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
  <version>3.0.1</version>
</dependency>
```

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

- Возможность генерации бинарных данных

Установка параметров HTTP-ответа

Возвращение ответа клиенту

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 сервлета
- Инициализирует экземпляр сервлета, вызывая метод init()

После этого сервлет можно считать запущенным, он находится в ожидании запросов от клиентов.

Появившийся запрос обслуживается методом service(HttpServletRequest req, HttpServletResponse res) сервлета, а все параметры запроса упаковываются в объект req класса HttpServletRequest, передаваемый в сервлет.

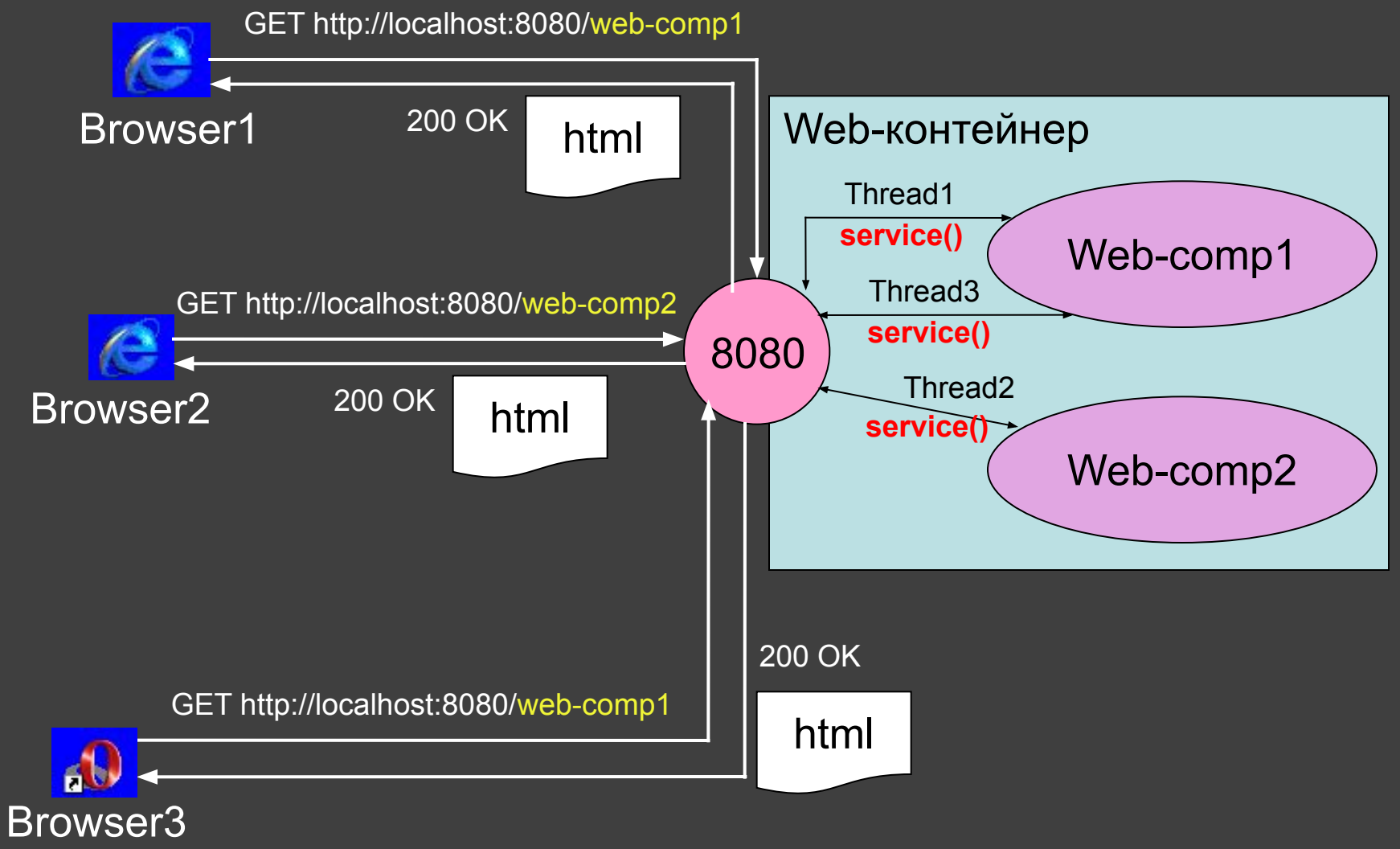
Еще одним параметром этого метода является объект res класса HttpServletResponse, в который загружается информация для передачи клиенту.

Для каждого нового клиента при обращении к сервлету создается независимый поток, в котором производится вызов метода service().

Метод service() предназначен для одновременной обработки множества запросов.

Если контейнеру нужно удалить сервлет, он его финализирует, вызывая метод destroy()

Работа нескольких Web-компонентов в одном Web-контейнере



Класс HttpServlet

При разработке сервлетов в качестве базового класса в большинстве случаев используется не интерфейс Servlet, а класс HttpServlet, отвечающий за обработку запросов HTTP. Этот класс уже имеет реализованный метод service().

Метод service() класса HttpServlet служит диспетчером для других методов, каждый из которых обрабатывает методы доступа к ресурсам.

В спецификации HTTP определены следующие методы: GET, HEAD, POST, PUT, DELETE, OPTIONS и TRACE. Наиболее часто употребляются методы GET и POST, с помощью которых на сервер передаются запросы, а также параметры для их выполнения.

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.

Интерфейс HttpServletRequest

При каждом вызове методы doGet и doPost класса HttpServlet принимают в качестве параметра объект, который реализует интерфейс HttpServletRequest. Web-сервер, который исполняет сервлет, создает объект HttpServletRequest и передает его методу service сервлета (который в свою очередь передает его методу doGet или doPost). Данный объект содержит запрос, поступивший от клиента.

Ряд ключевых методов, использованных в примерах, представлены в таблице. Полный список методов интерфейса HttpServletRequest можно найти в документации компании Sun.

String getParameter(String name) Получение из запроса значения параметра. Наименование параметра определено значением name.

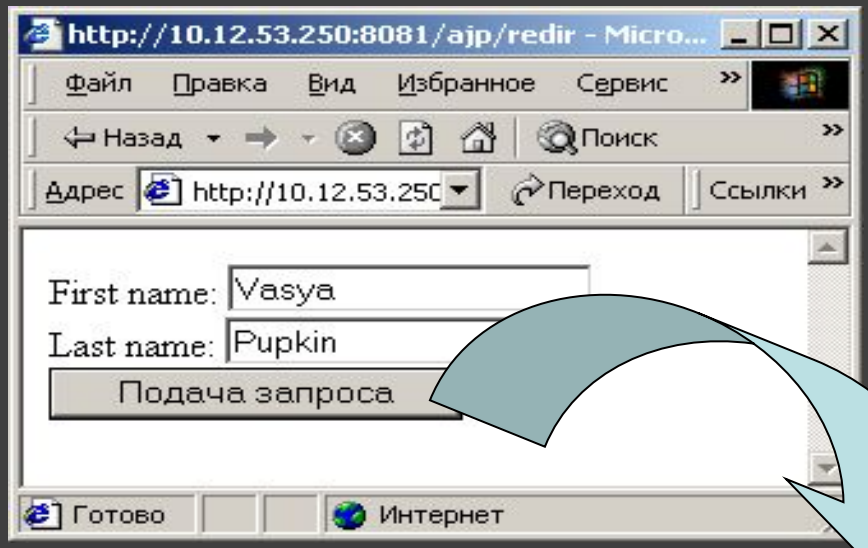
Enumeration getParameterNames() Получение из запроса имен всех параметров.

String[ ] getParameterValues(String name) Для параметра с несколькими значениями данный метод возвращает строковый массив.

Cookie[ ] getCookies () Получение массива объектов Cookie, сохраненных на компьютере клиента. Cookie могут быть использованы для уникальной идентификации клиента сервером.

HttpSession getSession(boolean create) Возвращает объект HttpSession текущего сеанса клиента. Если параметр create равен true и объект HttpSession не существует, то создается новый объект HttpSession.

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"

Использование других ресурсов сервера

Чтобы Ваш сервлет получил доступ к другим ресурсам сервера, таким как другой сервлет, страница JSP, и т.д вы можете сделать запрос ресурса с помощью объекта RequestDispatcher, если ресурс доступен на сервере, на котором запущен сервлет.

Чтобы получить доступ к объекту RequestDispatcher, используйте метод getRequestDispatcher класса ServletContext. Этот метод в качестве аргумента берет URL запрашиваемого ресурса. Формат этого аргумента последовательность имен директорий разбитых знаком слэш /, и именем ресурса на конце.

Как только Вы получаете объект RequestDispatcher, Вы можете дать возможность ассоциированному с ним ресурсу отвечать на запрос клиента. Перенаправление очень полезно, например, когда сервлет производит запрос, и ответ носит общий характер, так что он может быть передан другому ресурсу.

```
request.getRequestDispatcher("/index.jsp").forward(request, response);
```

Пример обработки данных формы

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
@WebServlet(name = "StudentCreate", urlPatterns = {"/student-create"})
public class StudentCreate extends HttpServlet {

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


            Student student = new Student();

            student.setNameStudent(request.getParameter("name"));
            student.setSurnameStudent(request.getParameter("toname"));
            student.setGroupe(request.getParameter("groupe"));
  }
```

Интерфейс HttpServletResponse

При каждом обращении к сервлету методы doGet и doPost класса HttpServlet принимают объект, который реализует интерфейс HttpServletResponse. Web-сервер, который исполняет сервлет, создает объект HttpServletResponse и передает его методу service сервлета (который в свою очередь передает его методу doGet или doPost). Объект HttpServletResponse описывает ответ клиенту.

Имеется множество методов, дающих возможность сервлету сформировать ответ клиенту. Полный список методов интерфейса HttpServletResponse можно найти в документации компании Sun.

void addCookie (Cookie cookie) Метод используется для добавления Cookie в заголовок ответа клинту. Установленный максимальный возвраст Cookie, а также разрешение клиентом хранения Cookie определяют, будут ли Cookies сохранены на клиенте и время их хранения.

ServletOutputStream getOutputStream() Получение бинарного потока вывода для отправления бинарных данных клиенту.

Установка данных ответа

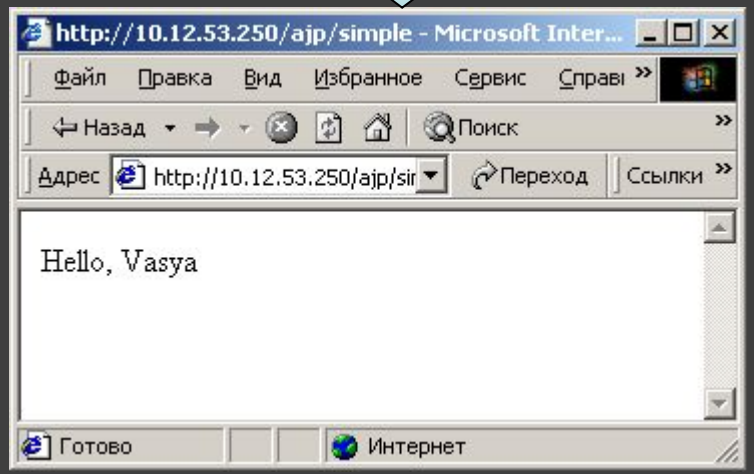
```
HTTP/1.1 200 OK
Content-Type: text/html;charset=ISO-8859-1
Content-Language: de-DE
Content-Length: 44
Date: Tue, 15 Feb 2005 16:00:49 GMT
Server: Apache-Coyote/1.1
Connection: close

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

response.setContentType("text/html")

response.setLocale(Locale.GERMAN);

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



Установка Web-приложения

Перед запуском скомпилированного Web-приложения, его необходимо внедрить (deploy) в Web-контейнер.

Для этого необходим дескриптор поставки (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.) упаковывают в Web-ARchive (WAR)

Web-контейнер содержит **контексты**, в которые помещаются Web-приложения.

Содержимое Web-приложения

/ - корневой каталог, содержит

- JSP-файлы
- HTML-файлы
- картинки
- др. ресурсы
- каталог WEB-INF (недоступен для клиента)
  - /classes – каталог с необходимыми 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
- Страница по умолчанию
- Страница, которая будет отображаться в случае возникновения ошибки
- Сервлеты и “маппинг” к ним
- Параметры инициализации
- Параметры ограничений доступа к ресурсам и т.д.

Спецификации сервлетов 3.0 притерпела ряд лучшений - программирование и развертывание сервлетов упростилось главным образом за счет введения аннотаций для декларирования сервлет (@WebServlet), фильтров (@WebFilter), листнеров (@WebListener) .

Таким образом, и дескриптор развертывания web.xml стал опциональным элементом.

Структура 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 поставки

Что такое JSP

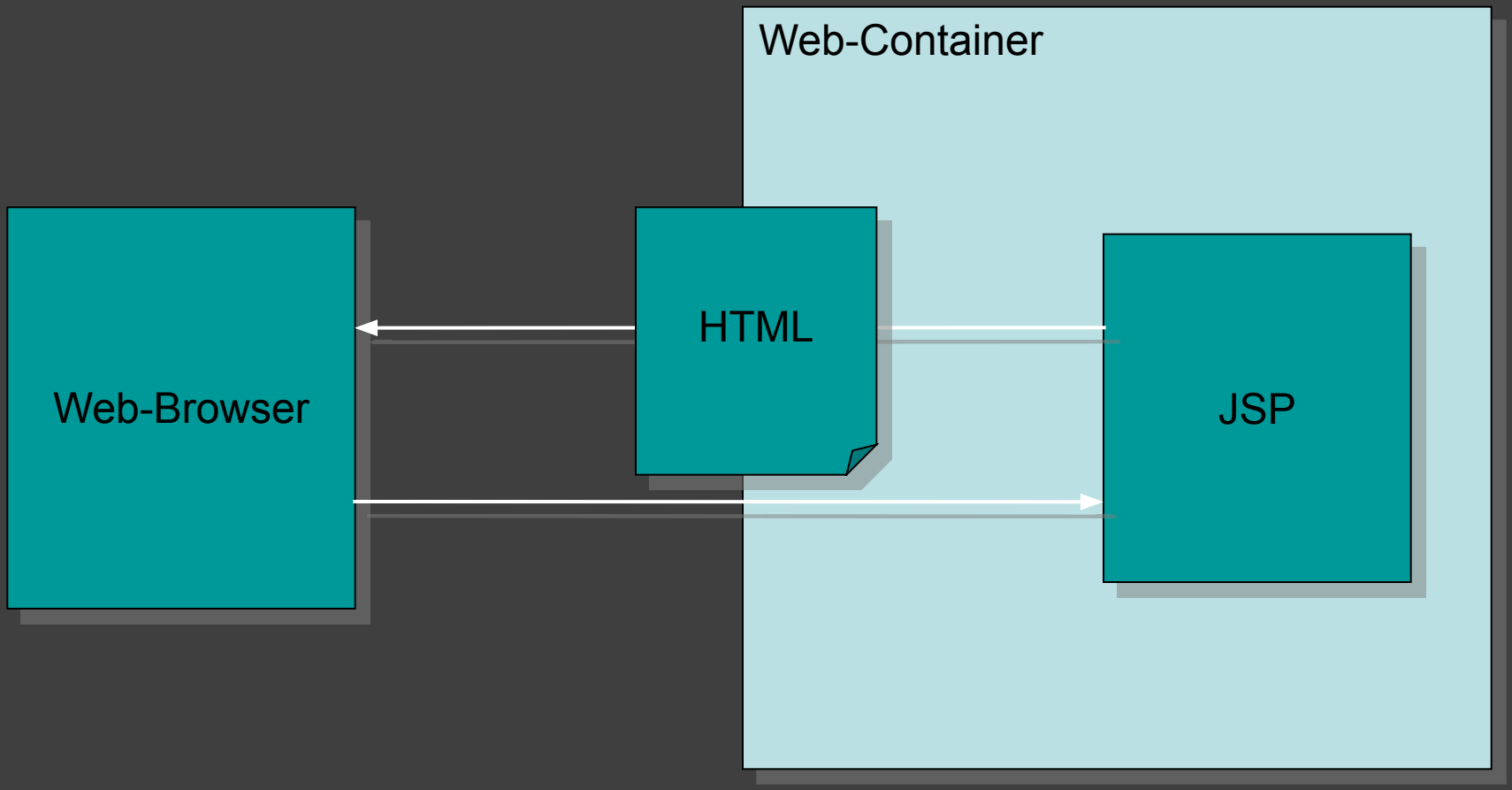
JSP — это документы текстового вида, которые описывают создание отклика на запрос клиента. Технология JSP позволяет комбинировать в одном файле HTML-содержимое web-страницы и программный код на языке Java

JSP определяются спецификацией, разработанной международным сообществом JCP (Java Community Process) и поддерживаемой компанией Sun Microsystems Inc в качестве официального стандарта

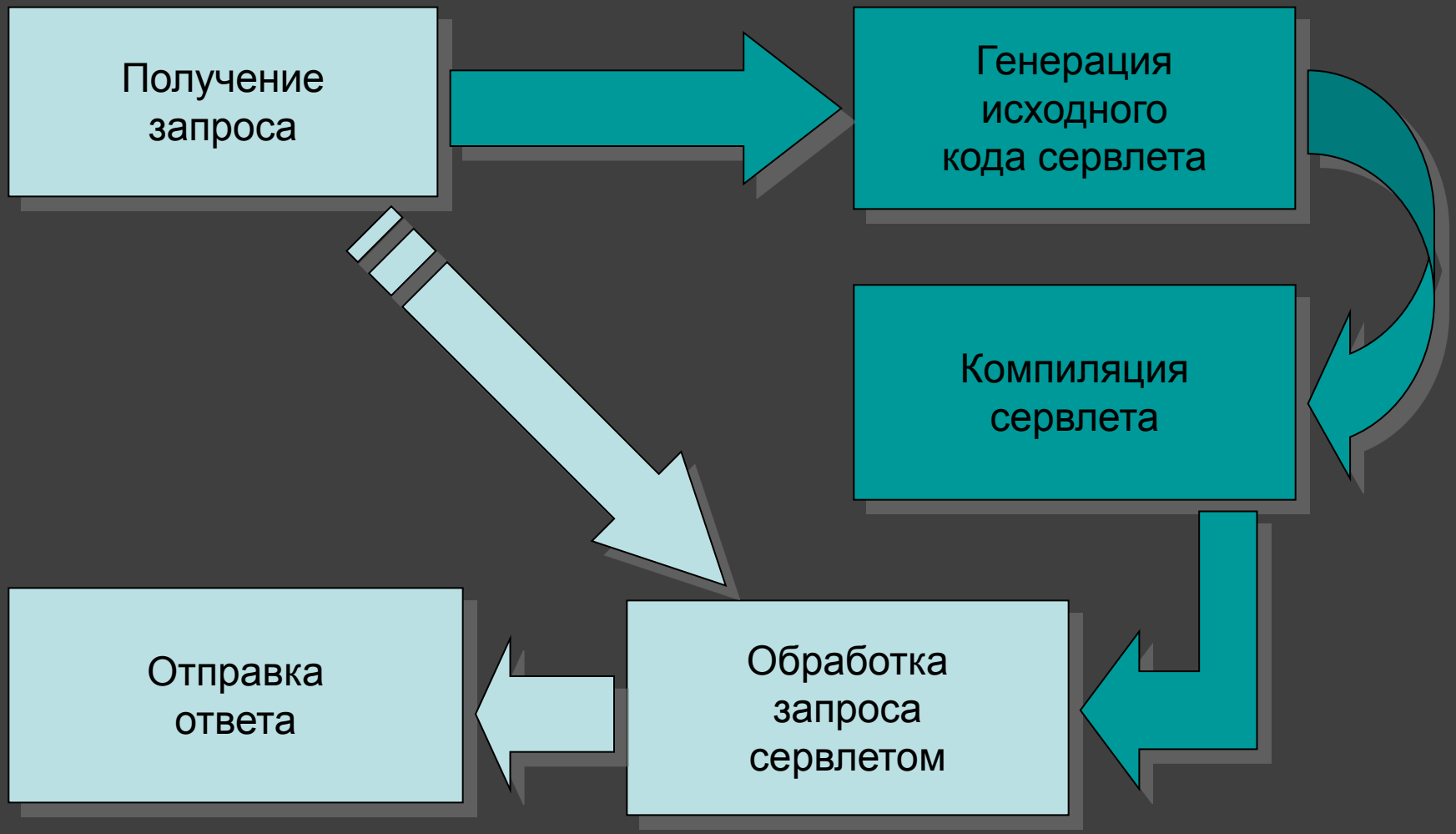
JSP в соответствии со спецификацией представляют собой технологию, входящую в состав J2EE . И JSP, и сервлеты в рамках технологии J2EE получили статус Web-компонентов

Страницы JSP не зависят от конкретной реализации Web-контейнера, что обеспечивает возможность их повторного использования

Общий принцип работы



Обработка запроса к JSP



Вставка кода в JSP-страницах

Выражения (expressions)

Выражения Java вычисляются, конвертируются в строку и вставляются в страницу. Эти вычисления происходит во время выполнения (то есть при при запросе страницы), а потому существует полный доступ к информации о самом запросе. Например, следующий код служит для отображения даты и времени запроса данной страницы:

Текущее время: <%= new java.util.Date() %>

Скриптлеты (scriptlets) – Если вы хотите сделать что-то большее чем вставка простых выражений, скриплеты JSP дадут вам возможность вставить любой код в метод сервлета, который будет создан при обработке данной страницы. Скриплеты имеют следующий вид: <% System.out.println(“Hello”); %>

Объявления (declarations) – участки кода, вставляемые в тело сервлета вне его методов.

<%! declarations %>

Комментарии

<%-- This text will not appear in the response --%>

Объявления

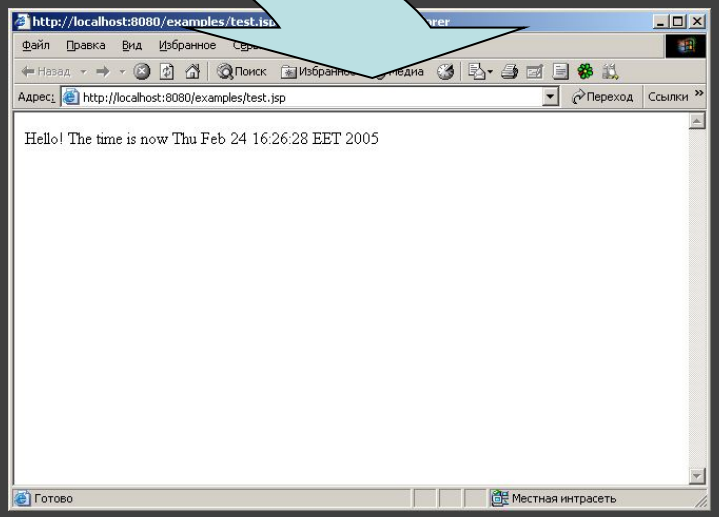
Объявления:

Объявления JSP позволят вам задать методы или поля, для вставки в тело класса сервлета (вне метода service, обрабатывающего запрос). Они имеют следующую форму:

Поскольку объявления не осуществляют вывода, обычно они используются совместно с JSP выражениями или скриплетами. В приведенном в качестве примера фрагменте JSP отображается количество запросов к данной странице с момента загрузки сервера (или с момента последнего изменения и перезагрузки сервлета):

Пример:

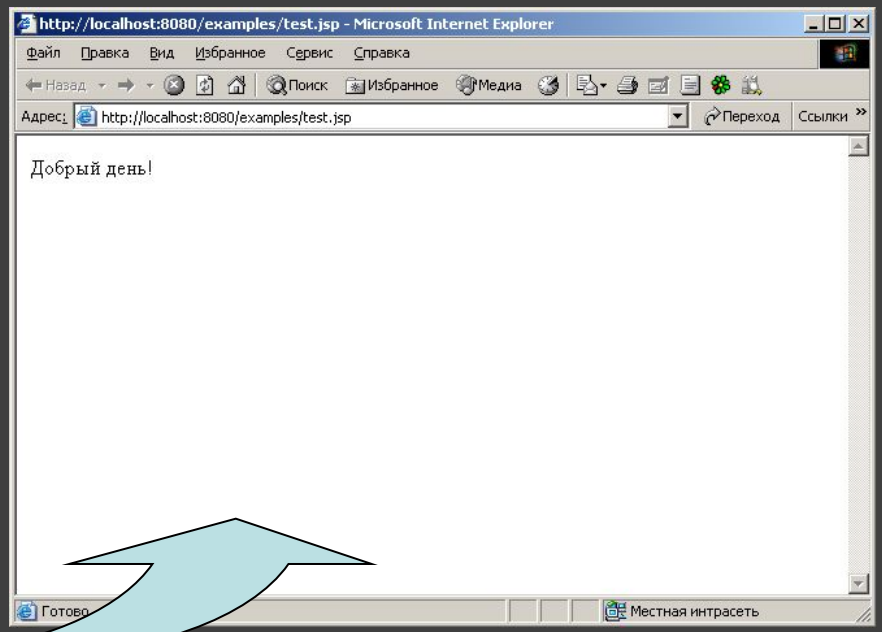
```
<%@ page import="java.util.*" %>
<HTML><BODY>
<%!
    Date theDate = new Date();
    Date getDate() {
        System.out.println( "In getDate() method" );
        return theDate;
    }%>
Hello!  The time is now <%= getDate() %>
</BODY></HTML>
```



Пример JSP-страницы

создать страницу, которая возвращает приветствие в зависимости от времени суток

```
 <%@ page contentType="text/html; charset=windows-1251" %>
<% java.util.Calendar rightNow = java.util.Calendar.getInstance();
  int hour = rightNow.get(java.util.Calendar.HOUR_OF_DAY);
  String greeting;
  if (hour < 12)
    greeting = "Доброе утро!";
  else if (hour < 17)
    greeting = "Добрый день!";
  else if (hour < 22)
    greeting = "Добрый вечер!";
  else
    greeting = "Доброй ночи!";
%>
<html><body>
<%=greeting%>
</body></html>
```



Предопределенные переменные

Для упрощения кода в выражениях JSP и скриплетах, вам предоставлен набор их восьми автоматически определенных переменных, иногда называемых неявными объектами. Доступные переменные это request, response, out, session, application, config, pageContext и page

request – переменная класса HttpServletRequest, предоставляет доступ к текущему запросу

response – переменная класса HttpServletResponse

session – переменная класса HttpSession – доступ к объекту сеанса данного пользоателя.

out – переменная класса JspWriter (буферизированная версия PrintWriter), может использоваться для отправки сформированного выходного документа

Директивы

Директивы JSP воздействуют на всю структуру класса сервлета. Обычно они имеют следующую форму:

Директивы JSP должны заключаться символами <%@ … %>

<%@ page pageEncoding="windows-1251" %> - задание кодировки страницы

<%@ page contentType=“appication/vnd.ms-excel" %> - задание типа содержимого

<%@ page import="java.util.List" %> - импорт класса

<%@ page session=“true”%> - задает, будет ли использоваться сессия

<%@ page buffer=“Xkb”%> - задает размер буфера

<%@ page isThreadSafe=“true" %> - задание SingleThreadModel

<%@ page errorPage=“relative URL" %> - задание страницы, на которую буде осуществлен переход в случае возникновения исключительной ситуации

Включения файлов

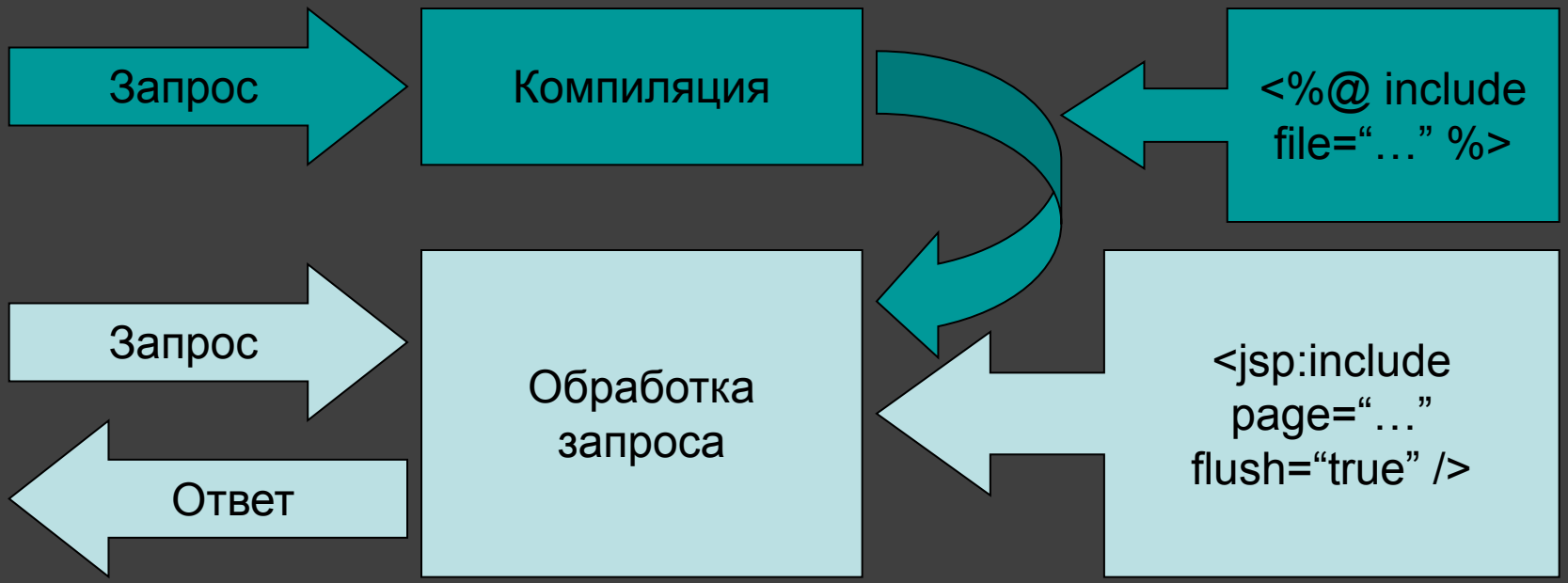
Существует 2 основных способа включения файлов в тело основной страницы:

На этапе трансляции страницы. В этом случае код включаемой страницы будет помещен в код сервлета основной страницы

```
<%@ include file="hello.jsp" %>
```

На этапе выполнения запроса. В этом случае страница будет включаться каждый раз динамически

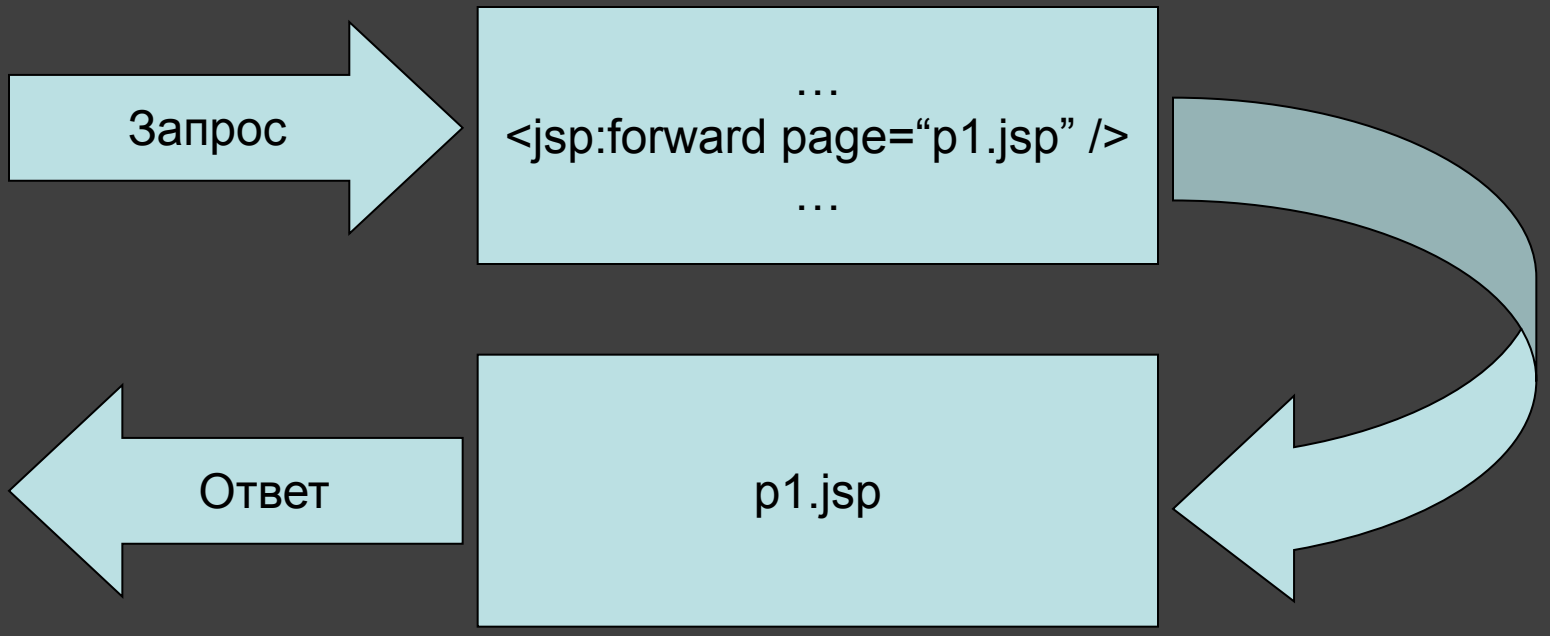
```
<jsp:include page="somePage.jsp" flush="true" />
```



Перенаправление запроса

Для перенаправления запроса предназначен тег

<jsp:forward page="…" />



Пример: forward.jsp

```
<html>

<%
   double freeMem = Runtime.getRuntime().freeMemory();
   double totlMem = Runtime.getRuntime().totalMemory();
   double percent = freeMem/totlMem;
   if (percent < 0.5) {
%>

     <jsp:forward page="/jsp/forward/one.jsp"/>

<% } else { %>

     <jsp:forward page="two.html"/>

<% } %>

</html>
```

forward.jsp

JSTL для написания JSP страниц

в JSP можно использовать Java вставки кода, но это является плохим тоном программирования а также не очень безопасно

Добавим необходимую зависимость :

```
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>jstl</artifactId>
  <version>1.1.2</version>
</dependency>
```

В начало JSP страницы вы должны подключить JSTL core:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
```

Синтаксис

Переменные:

```
<c:set var="myName" value="Alex"/>
```

В этом случае у нас будет создана переменная которая будет иметь в значение «Alex»

Вызвать эту переменную можно так:

```
<h1>${myName}</h1>
```

Операторы условий

```
<c:if test="${10 > 9}">
  <p>True<p>
</c:if>
<c:if test="${10 < 9}">
  <p>False<p>
</c:if>
```

Выше продемонстрировано пример if … else в виде jstl тегов, где:

test — это условие при котором оно будет выполнятся.

Это choose который дает возможность выбора, в java аналог switch, choose имеет вложенный тег when который отвечает за определенное действие при условии что test будет истинно.

```
<c:set var="num" scope="session" value="10"/>
<c:choose>
    <c:when test="${num <= 0}">
      Если num <= 0
    </c:when>
    <c:when test="${num > 1000}">
       Если num > 1000
    </c:when>
    <c:otherwise>
       Если не одно условие не есть верно.
    </c:otherwise>
</c:choose>
```

Тэг <c:forEach>

Позволяет, как это ни странно, сделать цикл. Пример использования:

```
<table>
    <c:forEach var=”movie” items=”${movieList}” >
        <tr> <td>${movie}</td> </tr>
    </c:forEach>
</table>
```

У тэга <c:forEach> есть опциональный параметр varStatus, с помощью которого, к примеру, можно реализовать счетчик (как “i” в обыкновенном цикле)

```
<c:forEach var=”movie” items=”${movieList}”
    varStatus=”movieLoopCount” >
    <tr> <td>count: ${movieLoopCount.count} </td> </tr>
    <tr><td>${movie}</td></tr>
</c:forEach>
```

Тэг <c:import>

Используется чтобы вложить одну страницу в другую. Синтаксис:

<c:import url=http://www.mysite/home.html />

Страница загружается динамически на этапе запроса. В качестве url может использоваться страница за пределами контейнера.

Если вы хотите передать какой-то параметр удаленной странице, то можно использовать тэг <c:param>. Пример:

```
<c:import url=”Header.jsp” >

    <c:param name=”subtitle” value=”My title” />

</c:import>
```