JavaServer Pages

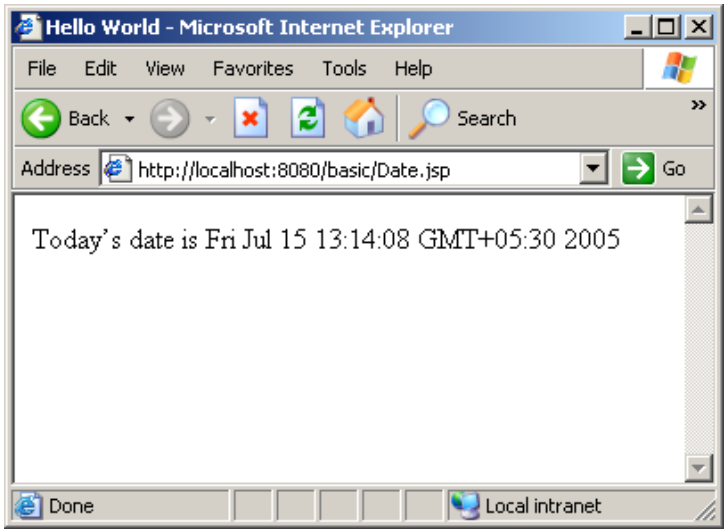
Лекция 11

Страницы JavaServer Pages (JSP)

- JSP – это технология стороны сервера, основанная на сервлетах.
- Позволяет создавать кроссплатформные Web-приложения, управляемые базами данных.
- Библиотека тэгов в JSP упрощает задачу создания динамического Web-контента.
- Хранятся в файлах с расширением .jsp
- Страницы JSP компилируются в сервлеты, представляющие собой Java-классы, которые выполняются на сервере и обслуживают клиентские запросы. Эти технологии дополняют друг друга.

JSP – Пример 1

- JSP-страница может быть создана с использованием HTML-кода



Вывод страницы HelloWorld.jsp

```
<html>
    <head>
        <title>Hello World</title>
    </head>
    <body>
        Today's date is <%= new java.util.Date() %>
    </body>
</html>
```

HelloWorld.jsp

→ Вывод текущей даты

JSP – Пример 2

- Код Java может быть встроен в JSP-страницу

Вывод страницы Display_Num.jsp

Display_Num.jsp

```
<html>
  <body>
       <h1>Displaying Numbers</h1>
        <%
       for (int i=0; i<10; i++) {
              out.println(i);
        }
       %>
   </body>
</html>
```

Код Java для вывода чисел от 1 до 10

Тэги JSP

- Различные тэги JSP:
  - Комментарии (Comments)
  - Директивы (Directives)
  - Стандартные операции (Standard Actions)
  - Элементы скриптов (Scripting Elements)

Директивы

- **Директивы** управляют структурой сервлета, посылая сообщения из страницы JSP в JSP-контейнер.
- Директивы являются сообщениями для JSP-контейнера, обеспечивают глобальную информацию, касающихся конкретных запросов, и предоставляют сведения, необходимые на стадии трансляции.

Директивы

- Синтаксис директив JSP:

  <%@ директива имяАтрибута="значение" %>

- Синтаксис директив JSP на XML:

<jsp:directive.директива имяАтрибута="значение" />

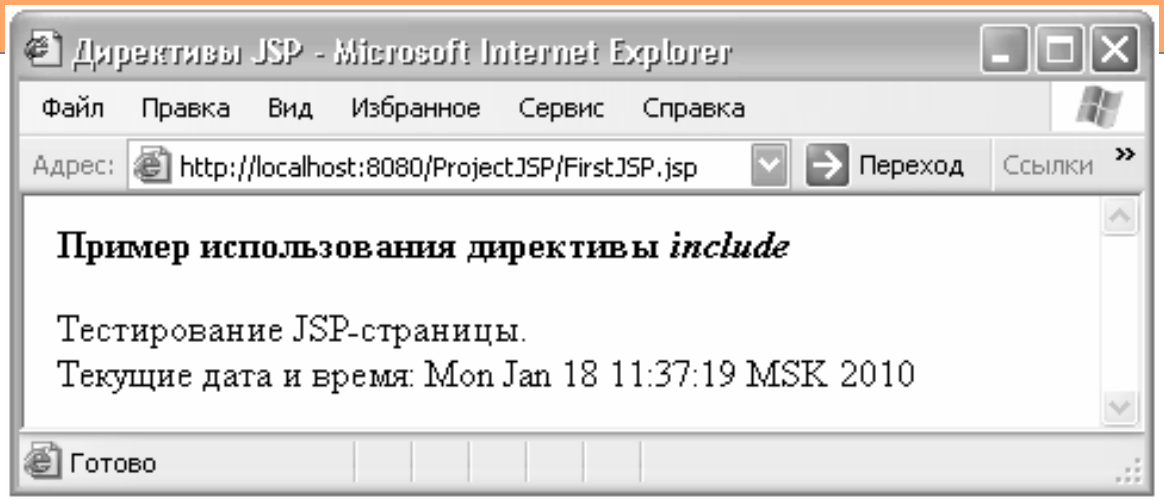
- Существует три типа директив:
  - page (страница) ;
  - taglib (библиотека тегов);
  - include (включить).

Директивы

- Типы директив JSP:
  - **`page`** определяет общие свойства страницы JSP, такие как кодировка, создание сеанса, работа с буфером, обработка ошибок и др.;
  - **`include`** посылает сообщение в JSP-контейнер для включения содержимого одного файла в другой;
  - **`taglib`** – позволяет использовать специализированные тэги в странице JSP.

Пример

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<html>
    <head>
        <meta http-equiv="Content-Type“ content="text/html; charset=UTF-8">
        <title> Директивы JSP </title>
    </head>
    <body>
        <h4>Пример использования директивы <i> include </i></h4>
        <%@include file="index.jsp" %>
    </body>
</html>
```



Стандартные теги

- **Стандартные теги** - тэги, влияющие на поведение страницы JSP во время выполнения и отправки ответа обратно клиенту.
- Набор стандартных тегов JSP довольно прост и включает в себя следующие элементы:
  - **declarations (объявления);**
  - **scriptlets (скриптлеты);**
  - **expressions (выражения).**
- Синтаксис элементов сценария:

```
<%! объявление %>
<%  скриплет   %>
<%= выражение  %>
```

Стандартные теги

- **Объявления (declarations)** предназначены для определения переменных и методов на языке скриптов, которые в дальнейшем будут использоваться на странице JSP.

<%! объявление %>

- **Скриптлеты (scriptlets)** состоят из фрагментов Java-кода и символов разметки скриптовых элементов. Скриптлет может содержать программный код и декларации локальных переменных, которые будут использованы для обработки запросов клиентов.

<% скриплет %>

- **Выражения (expressions) -** это исполняемые выражения, написанные на языке Java, предназначенные для конвертации данных в выходящий поток в виде строковых значений результатов выполняемых методов.

<%= выражение %>

Комментарии

- Комментарии игнорируются сервлетом во время компиляции
- Типы комментариев:
  - **HTML** – Записываются в HTML-коде и не выводятся вообще

    <!-- HTML-комментарий -->
  - **JSP** – Записываются в коде страницы JSP и не выводятся в Web-браузере.

    <%-- JSP-комментарий --%>
  - **Язык скриптов** – Записываются в скриптлетах.

    // все Java - комментарии

Пример

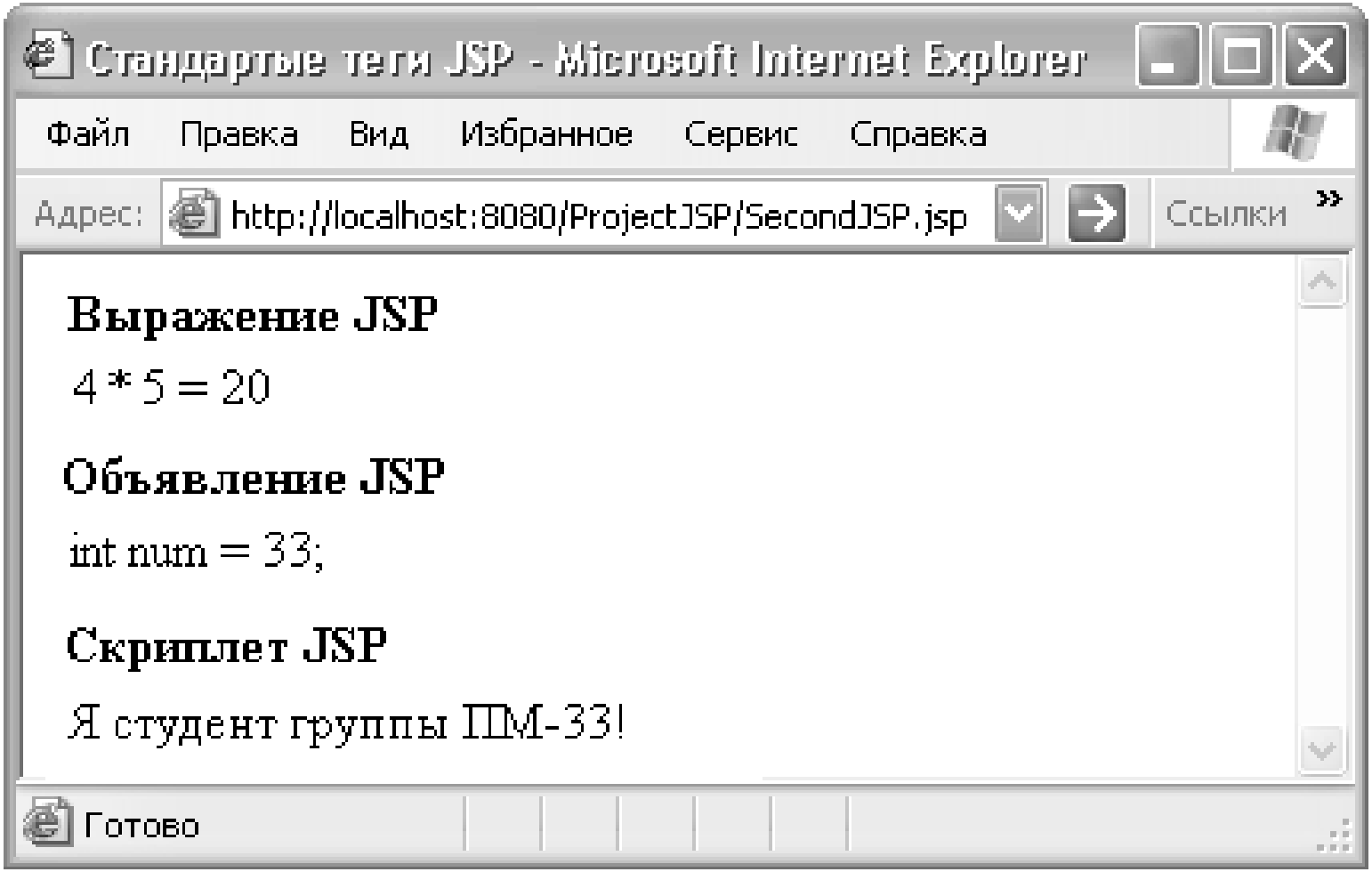
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<html>
    <head>
        <meta http-equiv="Content-Type“ content="text/html; charset=UTF-8">
        <title> Стандартые теги JSP </title>
    </head>
    <body>

        <h4> Выражение JSP </h4>
        4*5 = <%= 4*5 %>

        <h4> Объявление JSP </h4>
        <%! int num = 33; %> int num = 33;

        <h4> Скриптлет JSP </h4>
        <% out.println("Я студент группы ПМ-" + num + "!"); %>
    </body>
</html>
```

Результат


Стандартные теги JSP - Microsoft Internet Explorer
Файл Правка Вид Избранное Сервис Справка
Адрес: http://localhost:8080/ProjectJSP/SecondJSP.jsp
Ссылки
Выражение JSP
4 * 5 = 20
Объявление JSP
int num = 33;
Скриплет JSP
Я студент группы ПМ-33!
Готово

Жизненный цикл страницы JSP

- Жизненный цикл страницы JSP состоит из следующих этапов:
  - трансляция;
  - компиляция;
  - загрузка и конкретизация класса;
  - вызов метода jspInit();
  - вызов метода _jspService();
  - вызов метода jspDestroy().

Жизненный цикл страницы JSP

- **Трансляция:** страница JSP прочитывается, синтаксически анализируется, проверяется. В случае отсутствия ошибки, создается java-файл, содержащий java-класс сервлета.
- **Компиляция**: java-файл, созданный во время фазы трансляции, компилируется в class-файл. На данном этапе код Java проверяется на корректность, в случае обнаружения ошибки появляется необходимое сообщение.
- **Загрузка и конкретизация класса**: в случае успешной компиляции, класс сервлета загружается в память и конкретизируется.

Жизненный цикл страницы JSP

- **Метод jspInit()** запрашивается только один раз за жизненный цикл и используется для того, чтобы задать все начальные условия, необходимые для данного сервлета.
- **Метод _jspService()** соответствует телу JSP-страницы, получает объекты запроса и ответа. В этом методе обрабатываются скриптлеты и выражения JSP.
- **Метод jspDestroy()** запрашивается, когда экземпляр сервлета страницы JSP необходимо удалить. Любые операции по очистке, такие как освобождение ресурсов, могут производиться в рамках этого метода.

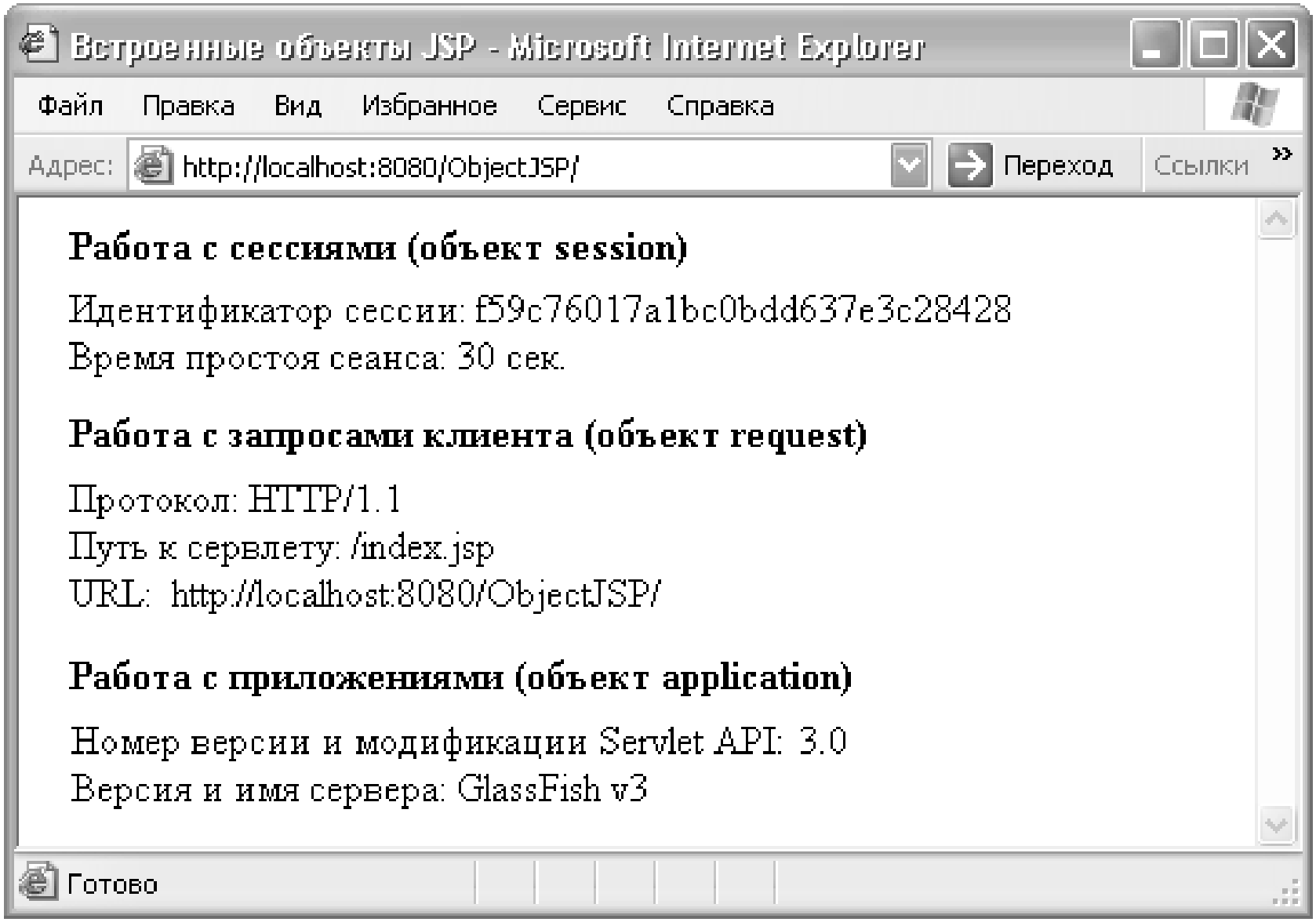
Встроенные объекты JSP

- **Встроенные объекты JSP (implicit objects)** - это объекты, автоматически доступные как часть стандарта JSP без их специального объявления или импорта:
  - **request** - запрос клиента (ServletRequest или HttpServletRequest);
  - **response** - ответ сервера (ServletResponse или HttpServletResponse);
  - **config** - конфигурация (ServletConfig);
  - **application** - приложение (ServletContext);
  - **session** - сеанс (HttpSession);
  - **pageContext** - контекст страницы (PageContext);
  - **page** - произвольный объект (Object);
  - **out** - выходной поток (JspWriter);
  - **exception** - исключение (Throwable).

Пример

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="java.util.*" %>
<html>
    <head> <meta http-equiv="Content-Type“ content="text/html; charset=UTF-8">
    <title> Встроенные объекты JSP </title> </head>
    <body>
        <H4> Работа с сессиями (объект session) </H4>
        Идентификатор сессии: <%= session.getId()%> <br>
        Время простоя сеанса: 30 сек. <% session.setMaxInactiveInterval(30);%>
        <H4> Работа с запросами клиента (объект request)</H4>
        Протокол: <%= request.getProtocol()%> <br>
        Путь к сервлету: <%= request.getServletPath()%> <br>
        URL: <%= request.getRequestURL()%>
        <H4> Работа с приложениями (объект application)</H4>
        Номер версии и модификации Servlet API:
        <%= application.getMajorVersion() + ".”+ application.getMinorVersion()%> <br>
        Версия и имя сервера: <%= application.getServerInfo()%>
    </body>
</html>
```

Результат


Встроенные объекты JSP - Microsoft Internet Explorer
Файл Правка Вид Избранное Сервис Справка
Адрес: http://localhost:8080/ObjectJSP/
Переход
Ссылки
Работа с сессиями (объект session)
Идентификатор сессии: f59c76017a1bc0bdd637e3c28428
Время простоя сеанса: 30 сек.
Работа с запросами клиента (объект request)
Протокол: HTTP/1.1
Путь к сервлету: /index.jsp
URL: http://localhost:8080/ObjectJSP/
Работа с приложениями (объект application)
Номер версии и модификации Servlet API: 3.0
Версия и имя сервера: GlassFish v3
Готово

Стандартные элементы action

- Стандартные элементы **action** выглядят как обычные тэги, название которых начинается с сочетания символов **jsp:** , что согласно терминологии XML означает принадлежность данному пространству имен.

- Синтаксис элементов **action**:

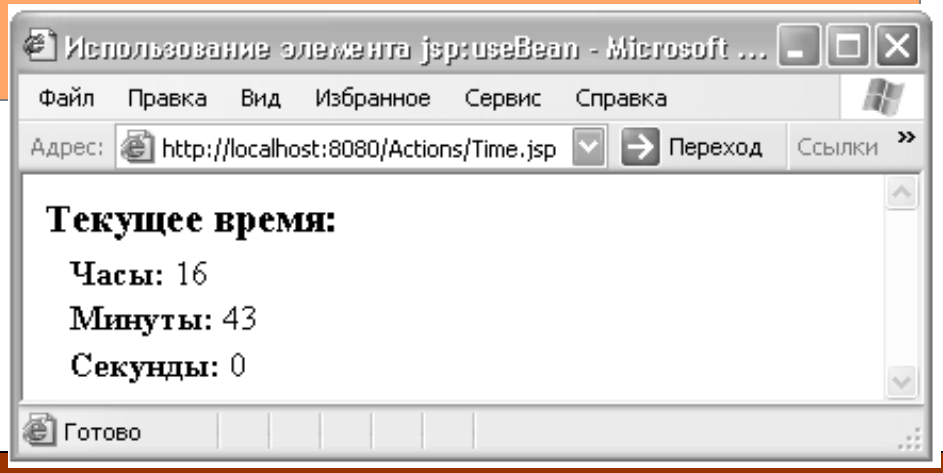
<jsp:action attributes />

Стандартные элементы action

- **jsp:useBean** – позволяет использовать экземпляр компонента JavaBean;
- **jsp:setProperty** - устанавливает значения полей - свойств объекта;
- **jsp:getProperty** - получает значения полей указанного объекта, преобразует его в строку и отправляет в неявный объект out;
- **jsp:include** - позволяет включать файлы в генерируемую страницу при запросе;
- **jsp:text** - вывод текста;
- **jsp:forward** - позволяет перенаправить запрос другой странице;
- **jsp:plugin** - заменяет соответствующие клиентскому браузеру конструкции OBJECT или EMBED, которые приводят к выполнению компонентов Applet или JavaBean;
- **jsp:params** – группирует параметры внутри тега jsp:plugin;
- **jsp:param** – используется для предоставления информации ключ/значение, например, в элементах forward, include, plugin;
- **jsp:fallback** – указывает содержимое, которое будет использоваться браузером клиента, если подключаемый модуль элемента plugin не сможет запуститься.

Пример

```
<%@ page language="java" contentType="text/html” pageEncoding="UTF-8"%>
<html>
    <head> <title> Использование элемента jsp:useBean </title></head>
    <body>
        <jsp:useBean id="clock" class="java.util.Date" />
        <h3> Текущее время:</h3>
        <b> Часы: </b>
        <jsp:getProperty name="clock" property="hours" /> <br>
        <b> Минуты: </b>
        <jsp:getProperty name="clock" property="minutes" /> <br>
        <b> Секунды: </b>
        <jsp:getProperty name="clock" property="seconds" /> <br>
    </body>
</html>
```



Интерфейс RequestDispatcher

- Перенаправляет запрос из JSP-страницы или сервлета в другие ресурсы.
- Другие ресурсы обрабатывают запрос и посылают ответ клиенту.
- Интерфейс **RequestDispatcher** инкапсулирует URL ресурса.
- Методы интерфейса **RequestDispatcher**:
  - **include()**
  - **forward()**

Методы интерфейса RequestDispatcher

- **include()**

  <jsp:include page="localURL" />

  - вызывает JSP-страницу или сервлет из другой JSP-страницы.

- **forward()**

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

  - перенаправляет запрос из JSP-страницы или сервлета в другую JSP-страницу.

Обработка исключений

- **Исключения** – это ошибки, которые могут возникать на JSP-странице.
- JSP-страница перехватывает и обрабатывает ошибки времени запроса.
- Необработанные исключения передаются на страницу ошибки.

<%@ page errorPage="errorpage.jsp" %>

- Необходимо установить для атрибута `isErrorPage` директивы `page` значение true, чтобы JSP-страница выполняла обработку ошибок.

<%@ page isErrorPage="true" %>

Обработка исключений

Виды ошибок

- **Во время трансляции** возникает, когда исходный файл JSP преобразуется в файл класса сервлета. Механизм JSP обрабатывает ошибки времени трансляции.
- **Во время запроса** возникает во время обработки запроса. Ошибки времени запроса являются ошибками времени выполнения, которые генерируют исключения.

```
<html>
  <body>
     <%@ page isErrorPage="true" %>
     Detected Error: <br>
    <%= exception.getMessage() %>
 </body>
</html>
```

→ Для того, чтобы JSP-страница выполняла обработку ошибок

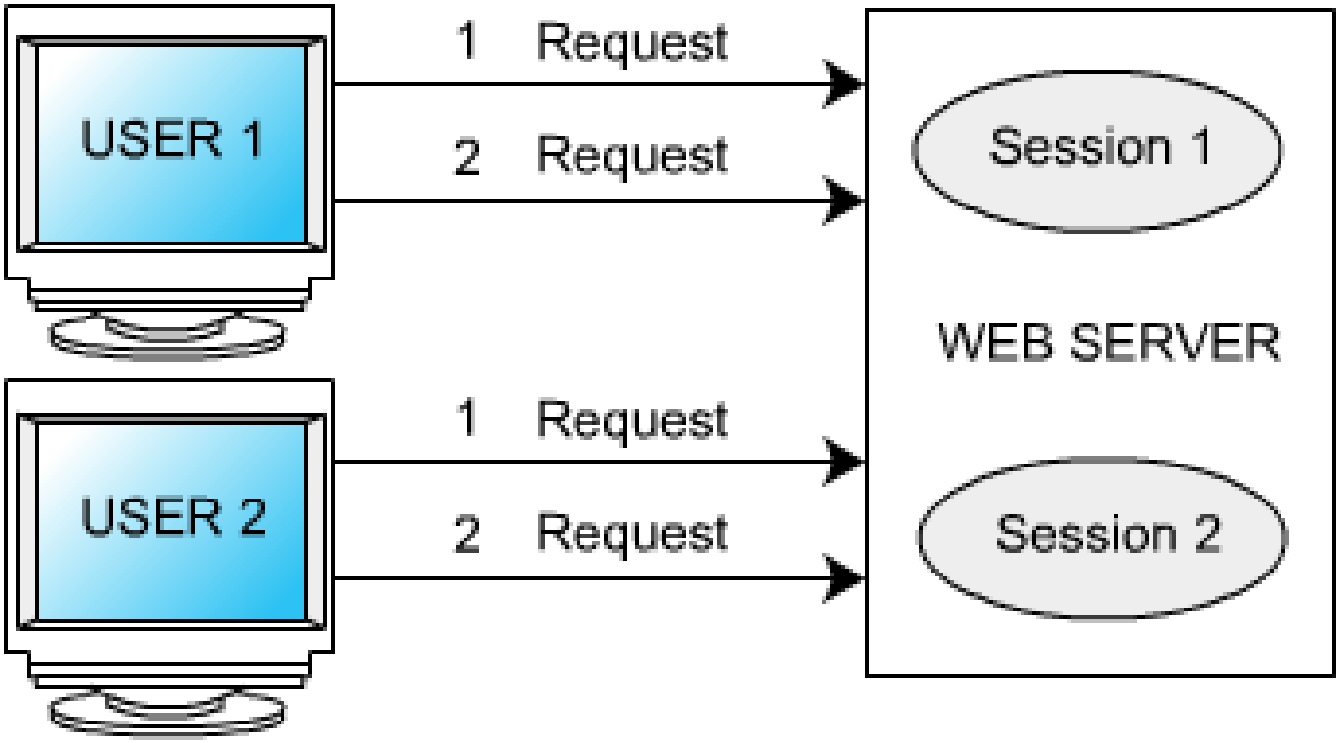
→ Возвращает сообщение об ошибке

Сессии в JSP (Session)

- **Сессия (сеанс)** - долговременное соединение, использующее уровень сеанса (session layer) сетевого протокола.
- Web-сервер определяет запросы и ответы через сетевое соединение, как единый рабочий сеанс.
- Сеанс работает, как канал связи между Web-сервером и событиями клиента.

Сессии в JSP (Session)

- Страница JSP использует сеансы для сохранения уникальных данных о конкретном клиенте, установившем соединение с Web-приложением.



Сеансы для двух Web-браузеров (Клиенты)

Методы в сеансе

Методы	Описание
`getAttribute()`	Возвращает объект с заданным именем, определённым в сеансе. Метод `getAttribute()` возвращает null-объект, если заданный объект не найден.
`getAttributeNames()`	Возвращает список объектов, определённых в сеансе
`getCreationTime()`	Возвращает время создания сеанса, как количество миллисекунд, прошедших с полночи 1 января 1970 г. по Гринвичу (GMT)
`getId()`	Возвращает уникальный идентификатор, который является идентификатором сеанса, в виде строки
`getLastAccessedTime()`	Возвращает время последнего запроса клиента в сеансе. Время возвращается, как количество миллисекунд, прошедших с полночи 1 января 1970 г. по Гринвичу (GMT)

Методы в сеансе

Методы	Описание
`getMaxInactiveInterval()`	Возвращает максимальный временной интервал сеанса. Сервлет-контейнер сохраняет сеанс открытым, пока пользователь имеет доступ к данному Web-сайту
`removeAttribute()`	Удаляет объект, связанный с заданной строкой, из сеанса
`setAttribute()`	Связывает объект с заданной ключевой строкой и сохраняет его в сеансе
`setMaxInactiveInterval()`	Задаёт интервал времени в секундах между запросами клиента, по истечении которого сервлет-контейнер будет считать данный сеанс недействительным

Обзор механизмов

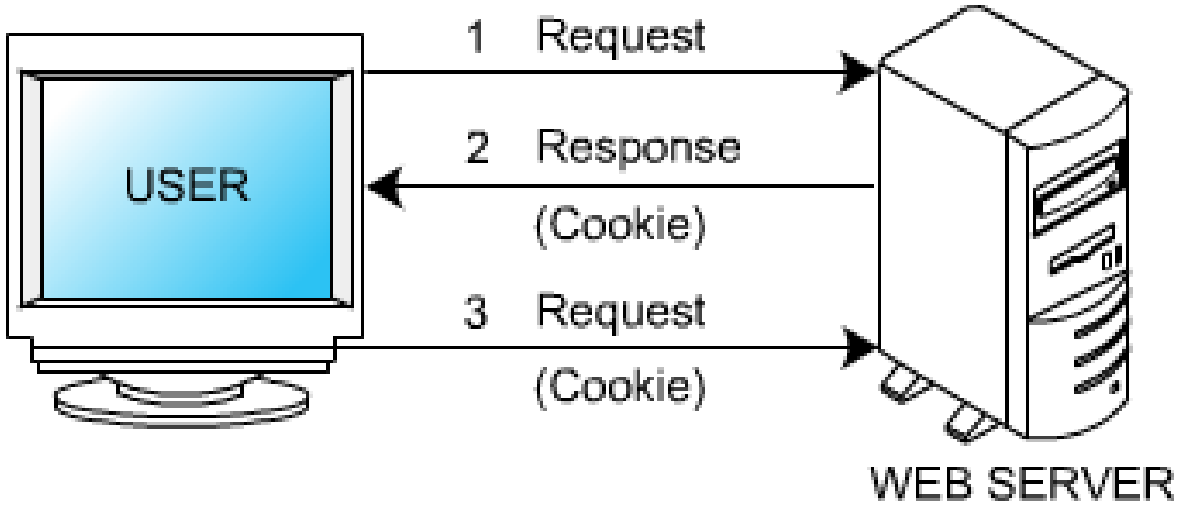
- Обслуживает сеанс в течение всего времени, пока пользователь просматривает данный Web-сайт.
- Используется в интерактивных Web-приложениях для сохранения информации о пользователе, зарегистрировавшемся на данном Web-сайте.
- Сохранённая информация используется для идентификации пользователя, посылающего запрос на Web-сервер.
- Отслеживание сеанса помогает обслуживать информацию о сеансе и контролировать многочисленные запросы, сделанные одним клиентом.

Отслеживание сеанса

- Информация посылается в браузер тремя способами, перечисленными ниже:
  - Закладки Cookies
  - Перезаписывание (Rewriting) URL
  - Метод скрытых полей формы

Закладки Cookies

- **Закладки Cookies** – это текстовые файлы, сохраняемые на компьютере пользователя и содержащие идентификатор сеанса (session Id) данного пользователя, посылаемый Web-сервером.
- Cookie включает имя, одно значение и дополнительные (необязательные) атрибуты.
- Cookies помогают обслуживать весь сеанс пользователя, просматривающего данный Web-сайт.



Cookies

- Преимущества закладок Cookies:
  - Запоминают пользовательские идентификаторы ID и пароли.
  - Помогают контролировать посетителей Web-сайта с целью улучшения их обслуживания и предоставления новых возможностей.
  - Закладки Cookies повышают эффективность обработки.

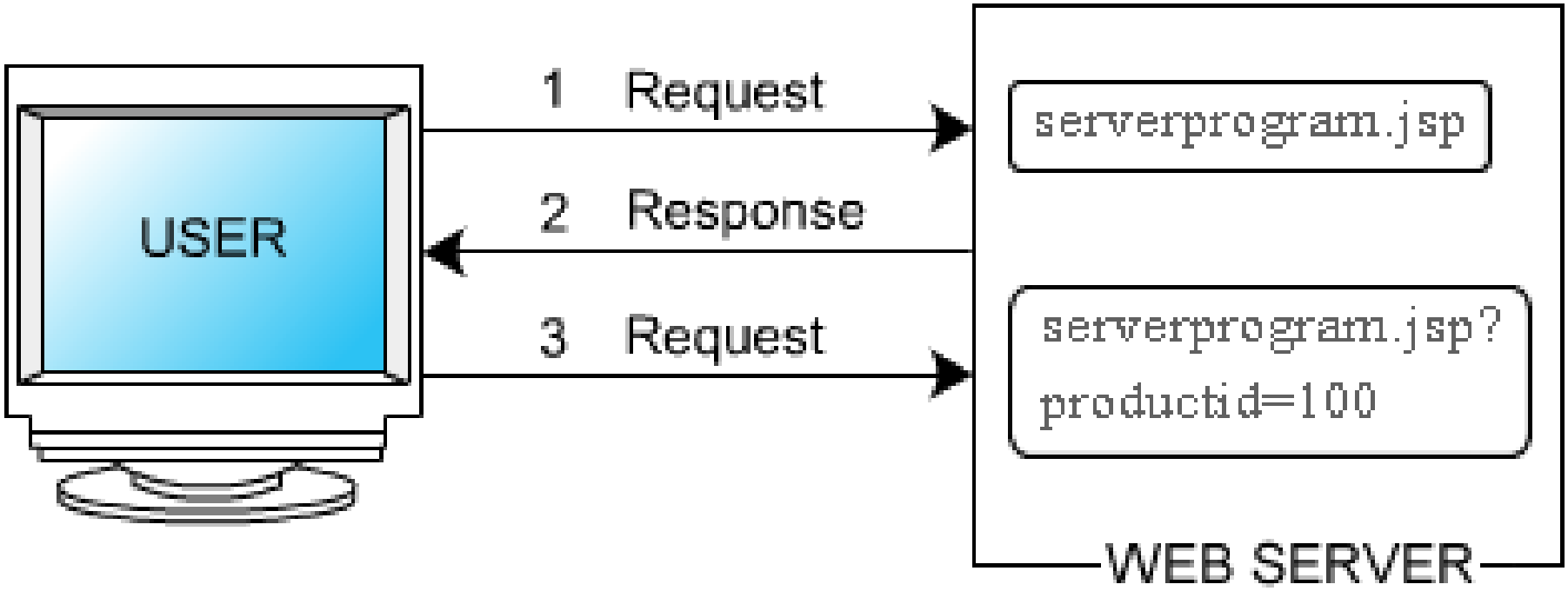
- Недостатки закладок Cookies:
  - Размер и количество сохраняемых закладок cookies ограничены.
  - Личная информация открыта для просмотра другими пользователями.
  - Закладки Cookies перестают работать, если в Web-браузере клиента установлен слишком высокий уровень безопасности.

Перезаписывание (Rewriting) URL

- Страница JSP скрывает подробности отслеживания сеанса, основанного на закладках cookie, и поддерживает механизм перезаписывания URL (URL rewriting).
- Перезаписывание URL работает с теми Web-браузерами, которые не поддерживают закладки cookies или закладки cookies запрещены Web-браузером.
- Каждая строка URL, которая вводится в Web-браузер, возвращается пользователю и содержит дополнительную информацию.

Перезаписывание (Rewriting) URL

- Идентификатор сеанса (session ID) кодируется в строке URL, которая создаётся JSP-страницами.



Перезаписывание (Rewriting) URL

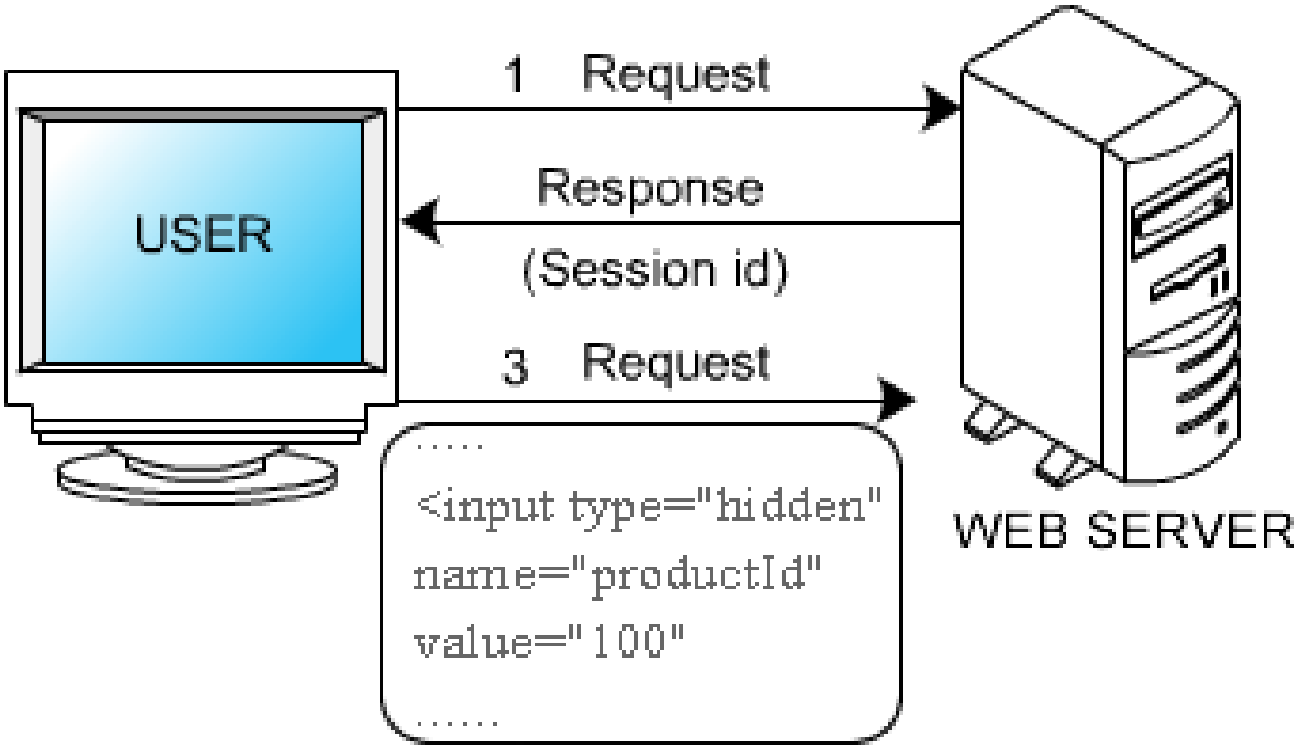
- Недостатки перезаписывания URL:
  - Трудоёмкость обработки на стороне сервера.
  - Каждая строка URL, которая возвращается пользователю, должна включать дополнительную информацию, добавленную в неё.
  - Если пользователь выходит из сеанса и открывает данную Web-страницу, используя ссылку (link) или закладку браузера (bookmark), то информация о сеансе теряется.

Метод скрытых полей формы

- Информация из Web-браузера возвращается на Web-сервер в виде HTTP-параметров.
- Используются скрытые поля в HTML-странице.
- Скрытые поля в форме используются для передачи информации в Web-браузер.
- Сохраняет информацию о сеансе.
- Помогает передавать информацию из одной HTML-страницы в другую.

Скрытые поля формы

- Когда пользователь переходит на следующую страницу, программа на стороне сервера считывает все параметры, которые этот пользователь передал в предыдущую форму.



Скрытые поля формы

- Преимущества скрытых полей формы:
  - Простейший способ реализации отслеживания сеанса
  - Ничего не выводит на HTML-странице, но может использоваться для хранения любого типа данных
  - Помогает организовать связь между двумя страницами

- Недостатки скрытых полей формы:
  - метод отслеживания сеанса даёт возможность любому пользователю просматривать конфиденциальную информацию.

Жизненный цикл сеанса

- Сервер присваивает уникальный идентификатор (ID) сеансу, созданному по конкретному запросу пользователя.
- Этот идентификатор сеанса (session ID) передаётся клиенту, как закладка cookie или как скрытая переменная.
- Сеанс считается новым до тех пор, пока клиент не вернёт идентификатор этого сеанса (session ID) серверу через закладку cookie или как часть запрашиваемого URL.
- Сеанс существует на сервере до тех пор, пока он не станет недействительным или пока сервер не будет остановлен.
- Объекты **HttpSession** используются для сохранения данных сеанса в текущем контексте сервлета.

Использование объекта Session

- Объект **Session** может использоваться для сохранения и чтения данных.
- Объект session работает почти как доска объявлений (bulletin board) на которую другие объекты могут записывать информацию или считывать с неё информацию

Использование объекта Session

- Значение объекта session может быть считано и преобразовано, то есть, приведено к соответствующему типу объекта.

```
…
// Obtain a session object
HttpSession session =
request.getSession(true);
// Read the session data and cast it
to the appropriate object type
Integer sessionInt = (Integer)
session.getValue("session");
int count = sessionInt.intValue();
…
…
```

Получает объект session

Считывает значение session

Приводит тип значения session к соответствующему типу данных

Использование объекта Session

- Сеанс можно сделать недействительным, используя метод **`invalidate()`** объекта HttpSession.

```
<%
    String sessionval=(String)session.getAttribute("userid"));
    if(sessionval == null)
    {
        session.setAttribute("userid",sessionval);
        out.println(session.getAttribute("userid"));
    }
    else
   {
        out.println("User Session already created");
    }
%>
<b>click this link to
<a href="<%=session.removeAttribute("userid")%>">remove
session attribute</a></b><br/>
<b>click this link to <a href="<%=session.invalidate()%>">
invalidate the session</a></b><br/>
```

Принимает **userid**

Если sessionval равен нулю, то значением sessionval становится **userid**

Удаляет объект session

Делает сеанс недействительным

Использование объекта Session

- Привязывание объектов к объекту запроса request похоже на сохранение этого объекта в сеансе.
- Объект, связанный с запросом, доступен только во время существования этого конкретного запроса.
- Объект может быть привязан с помощью метода **`setAttribute(String key, Object obj)`** в интерфейсе HttpRequest.
- Объект может быть извлечён с использованием метода **`getAttribute(String key)`**.

JSTL

- **JSTL**(**JavaServer Pages Standard Tag Library**) — стандартная библиотека тегов JSP.
- Она расширяет спецификацию JSP, добавляя библиотеку JSP тегов для общих нужд, таких как разбор XML данных, условная обработка, создание циклов и поддержка интернационализации.
- JSTL является альтернативой скриптлетам, то есть прямые вставки Java кода.

Механизм расширения тэгов

- Позволяет создавать специализированные тэги.
- С помощью специализированных тэгов можно избежать написания Java-кода.
- Различными технологиями, относящимися к механизму расширения тэгов, являются:
  - Tag library – это набор специализированных операций и тэгов
  - Custom action – специализированная операция
  - Custom tag – инкапсулирует фрагмент многократно используемого кода

Специализированные тэги

- Скрывает детали выполняемой работы от Web-дизайнеров и разработчиков
- Специализированные тэги могут использоваться многократно
- Записываются с использованием синтаксиса языка XML
- Различные типы специализированных тэгов:
  - Empty tag (пустой тэг)
  - Tag with attributes (тэг с атрибутами)
  - Body tags (тэги с телом)
- Синтаксис:

```
…
< prefix : tagName >
…
```

Пользовательские теги

- В пакет JSTL входят пять пользовательских тегов :
  - core
  - xml
  - fmt
  - sql
  - fn
- Библиотека core записывается тегом:

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

- В нее входят теги, создающие подобие языка программирования. Они призваны заменить операторы java, тем самым устраняя скриптлеты.

core

Каталоги тегов:

<c:set var=“name” value=”$ {param.name}”>

<c: remove>

<c:url >

<c:if >