МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра "ПОВТ и АС"

Серверные страницы Java

Методические указания к лабораторной работе по дисциплине "Объектно-ориентированное программирование"

Ростов-на-Дону 20 г.

Составитель: к.ф.-м.н., доц. Габрельян Б.В.

УДК 512.3

Серверные страницы Java: методические указания – Ростов н/Д: Издательский центр ДГТУ, 20 . – 8 с.

Рассмотрена технология JSP, схема создания, установки и использования серверных страниц Java. Даны задания по выполнению лабораторной работы. Методические указания предназначены для студентов направлений 090304 "Программная инженерия", 020303 "Математическое обеспечение и администрирование информационных систем".

Ответственный редактор:

Издательский центр ДГТУ, 20

Серверные страницы Java (JSP) - это EE технологии для создания приложений, генерирующих динамический web-контент, такой как HTML, XHTML, XML, текстовая спецификация для динамической генерации ответа на запрос.

При первом обращении JSP преобразуется в сервлет. При этом создается класс, производный от HttpJspBase с переопределенным методом _jspService(HttpServletRequest req, HttpServletResponce resp) – аналогом метода service сервлета, который и вызывается для обработки запросов. Полученный сервлет компилируется модулем JSP Engine.

1. Базовые концепции

1. Стандартные директивы
2. Элементы сценария
3. Стандартные действия
4. Теговый механизм расширения
5. Шаблонное содержимое

Стандартные директивы: синтаксис собственный или xml

<%@ директива атрибуты %> <jsp:directive.директива атрибуты />

Примеры

<%@ page import="java.util.*" %> <jsp:directive.page import="java.util.*" />

атрибуты page: language, session (true/false), errorPage, isErrorPage, contentType, pageEncoding, …

<%@ include file= "/part1.html" %> <jsp:directive.page import="java.util.*" />

путь в include относительно jsp-страницы (вставка на этапе **трансляции**)

<%@ taglib uri="http://www.mysite.ru/mytaglib" prefix="c" %>
Использование <c:mytag> … </c:mytag>

<jsp:directive.taglib uri= "http://www.mysite.ru/mytaglib" prefix= "c" />

Элементы сценария

1. Выражение (expression)

<%= выражение %>	<jsp:expression> выражение </jsp:expression>
Пример <%= 2*2 %>	<jsp:expression> 2*2 </jsp:expression>

2. Скриптлет (scriptlet)

<% код %>	<jsp:scriptlet> код </jsp:scriptlet>
Пример <% int x; ++x; %>	<jsp:scriptlet> int x; ++x; </jsp:scriptlet>

3. Объявление (declaration)

<%! объявление %>	<jsp:declaration> объявление </jsp:declaration>
Пример	
<% private String name; %>	<jsp:declaration> private String name; </jsp:declaration>

Стандартные действия

<jsp:useBean>

<jsp:setProperty>

<jsp:getProperty>

<jsp:include> **динамически**

<jsp:forward>

Пример

<jsp:useBean id=“x” class=“mypackage.MyBean”/>

<jsp:setProperty name=“x” property=“name” value=“Ivanov I.I.” />

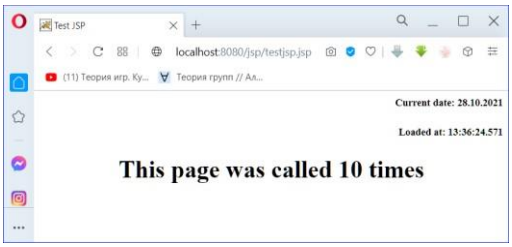
Пример. JSP показывающая текущие дату, время и число обращений к ней. На сервреpe Apache Tomcat в каталоге webapps создан подкаталог jsp, в нем размещена JSP – текстовый файл с названием, например, tetsjsp.jsp. Тогда локальное обращение к странице можно выполнить так

http://localhost:8080/jsp/testjsp.jsp

```
<html>
  <head><title>Test JSP</title></head>
<body>
  <%-- JSP-директива --%>
  <%@ page import="java.util.*" %>
  <%-- JSP-объявление метода --%>
  <%!
    public String getDate() {
      Calendar calendar = Calendar.getInstance();
      String date = calendar.get(Calendar.DAY_OF_MONTH) + "."
      + (calendar.get(Calendar. MONTH)+1) + "."
      + calendar.get(Calendar.YEAR);
      return date;
    }
  %>
  <%-- JSP-объявление поля --%>
  <%! private int counter; %>
                                    <%-- JSP-выражение --%>
  <h5 align="right">Current date: <%= getDate() %> </h5>
<%-- JSP-объявления --%>
```

```
<%!
private String time;

    public void calculateTime() {
      Calendar calendar = Calendar.getInstance();
       time =  calendar.get(Calendar.HOUR_OF_DAY) + ":"
                + calendar.get(Calendar. MINUTE) + ":"
                + calendar.get(Calendar.SECOND) + "."
                + calendar.get(Calendar.MILLISECOND);
    }
  %>
  <%-- JSP-скриптлет --%>
  <% calculateTime(); %>
                                <%-- JSP-выражение --%>
  <h5 align="right">Loaded at: <%= time %></h5>
                                                    <%-- JSP-выражение --%>
  <h1 align="center">This page was called <%= ++counter %> times
  </h1>
</body>
</html>
```



Действия позволяют разместить на странице компоненты JavaBean. Требования к классу, представляющему JavaBean:

1. Реализация интерфейса Serializable

2. Наличие конструктора без аргументов

3. Свойства (get- и set-методы)

4. Обработка событий

Например,

файл TsetBean.java

```
package test;
public class TestBean implements java.io.Serializable {
        private String message;
        public String getMessage() { return message; }
        public void setMessage(String msg) { message = msg; }
}
```

файл с откомпилированным классом TestBean.class на сервере Apache Tomcat

помещается в каталог webapps\jsp\WEB-INF\classes, JSP, например, testbean.jsp в каталог webapps\jsp

```
<html> <head><title>Test JavaBean on JSP</title></head>
<body>
  <%@ page import="java.util.*,test.TestBean" %>
  <%-- JSP-действие --%>
  <jsp:useBean id="x" class="test.TestBean" />
  <jsp:setProperty name="x" property="message" value="Hello" />
  <h2 align="center"><jsp:getProperty name="x" property="message“ /></h2>
  <%-- использование в объявлении, как обычный класс --%>
  <%! TestBean tb = new TestBean(); %>
  <%-- использование в скриптлете --%>
  <% tb.setMessage(" World"); %>
  <%-- использование в выражении --%>
  <h2 align="center"><%= tb.getMessage() %></h2>
</body>
```

```
</html>
```

2. Встроенные объекты

1. out
2. request
3. response
4. config
5. application
6. session
7. pageContext
8. page
9. exception

3. Переадресация запроса от сервлета к JSP

```
class CommServlet extends HttpServlet {
        @Override
        public void doGet(HttpServletRequest req, HttpServletResponse resp) {
                String msg = "Vary important message";
        req.setAttribute("message", msg);
                // можно передавать целые коллекции
                RequestDispatcher disp = getServletConfig().getServletContext().
                                                getRequestDispatcher("jsptocall.jsp");
                disp.forward(req, res);
        }
}
```

Файл jsptocall.jsp

```
<html>
```

```
        <body>
        <%
                String m = (String)request.getAttribute("message");
        out.println("Message to JSP: " + m);
        %>
        </body>
</html>
```

ЗАДАНИЕ 1. Создайте простейшую информационную систему: на сервере в xml-файле хранится некоторая структурированная информация (например, об аудио- или видео файлах домашней мультимедиа библиотеки, о книгах домашней библиотеки и т.п. - выберите что-то на свой вкус), доступ к информации через браузер, можно запрашивать информацию отфильтрованную по какому-то фильтру (например, если это музыкальные произведения - по жанру, по исполнителю и т.д.), добавлять новую информацию, удалять ненужную, для этого нужно подготовить html-страницы с соответствующими формами. Нужно написать сервлет для обработки запросов пользователя и JSP для представления отображаемой в браузере информации.

ЗАДАНИЕ 2. Организуйте программу для задания 1 так, чтобы она соответствовала архитектуре MVC. Модель должна быть представлена

компонентами JavaBean, контролер сервлетом, вид - сервреной страницей (или страницами) Java.

ЗАДАНИЕ 3. Создайте JavaFX приложение для клиентской части информационной системы из задания 2.

ЗАДАНИЕ 4. Реализуйте игру с сервером в "крестики-нолики" для игрового поля 3x3. html-страница содержит игровую таблицу размером 3x3 с ноликами и единицами, отражающую текущее состояние игры. Первым играет человек, задает через поля ввода номер строки и столбца таблицы на пересечении которых он хочет поместить "крестик" и отсылает запрос на сервер. Сервлет, установленный на сервере генерирует свой ход и возвращает результат JSP странице, которая формирует для пользователя html-страницу с обновленной таблицей, содержащей выполненные ходы пользователя и сервера. При завершении игры отображается итоговая таблица и сообщение о результате.

ЗАДАНИЕ 5. Если это не так, то организуйте программу для задания 4 так, чтобы она соответствовала архитектуре MVC.

Литература

1. Jakarta EE. Jakarta Server Pages Specification. Version 3.0. Jakarta Server Pages Team, https://projects.eclipse.org/projects/ee4j.jsp. – 2020, 228 p.
2. В.В.Смелов "Оновы web-программирования на Java" : учеб.-метод. пособие.

– Минск: БГТУ. – 2009, 140 с.

3. Б.Перри "Java сервлеты и JSP: сборник рецептов", 3-е рус. изд. – М.:КУДИЦПресс. – 2009, 768 с.
4. Д.Хеффельфингер "Java EE 6 и сервер приложений GlassFish 3". – М.:ДМК-Пресс. – 2013, 416 с.
5. J.Murach, M.Urban "Murach's Java Servlets and JSP", 3-d ed. – Mike Murach & Associates Inc. – 2014, 767c.
6. J.Juneau "Java EE 8 Recipes", 2-d ed. – Apress. – 2018, 792c.

Редактор А.А. Литвинова

ЛР № 04779 от 18.05.01.	В набор	В печать
Объем 0,5 усл.п.л., уч.-изд.л.	Офсет.	Формат 60x84/16.
Бумага тип №3.	Заказ №	Тираж 120. Цена

Издательский центр ДГТУ

Адрес университета и полиграфического предприятия:

344010, г. Ростов-на-Дону, пл. Гагарина, 1.