

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

Кафедра вычислительной техники

**Отчет по лабораторной работе №6**  
**по дисциплине «Web-программирование»**

**Тема: ОРГАНИЗАЦИЯ ПЕРЕДАЧИ ДАННЫХ МЕЖДУ ЗАПРОСАМИ**  
**ПОЛЬЗОВАТЕЛЕЙ**

Студент гр. 2310

Альсакма О.С.М

Преподаватель

Павловский М.Г.

Санкт-Петербург

2024

**Цель работы:** знакомство с методами передачи информации между соединениями, открываемыми в рамках одного сеанса работы пользователя

## Передача данных через файл Cookie

Была реализована страница welcome.jsp, которая представляет из себя форму для заполнения значения зарплаты. В значение формы value помещается информация, которая была в предыдущем куке (если она там была).

```
<%@page import="java.net.URLDecoder"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Приветствую</title>
<link rel="stylesheet" href="style.css">
</head>
<body>

    <form method = get action = "SalaryProcessor">
        <h2>Введите минимальную<br>зарплату футболиста</h2>
        <input type = "text" name = "salary" placeholder = "Пропуск, если не надо"
        <%
            Cookie [] c = request.getCookies();
            if(c != null)
                for(int i = 0; i < c.length; i++)
                    if("salary".equals(c[i].getName())) {
                        // Запись значения в поле ввода, если найден Cookie
                        out.print(" value=" + URLDecoder.decode(c[i].getValue(),
"UTF-8") + " ");
                        break;
                    }
        %>
        <br>
        <input type = "submit" value = "Ввод">
    </form>

</body>
</html>
```

После заполнения формы информация передается сервлету SalaryProcessor.java. Полученную из формы информацию сохраняем в куку, которая потом отправляется на сервер.

```
import java.io.IOException;
import java.net.URLEncoder;

import jakarta.servlet.ServletException;
import jakarta.servlet.http.Cookie;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

/**
```

```

* Servlet implementation class SalaryProcessor
*/
public class SalaryProcessor extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * Default constructor.
     */
    public SalaryProcessor() {
        // TODO Auto-generated constructor stub
    }

    protected void processRequest(HttpServletRequest request, HttpServletResponse
        response) throws ServletException, IOException {
        response.setContentType("text/html; charset=UTF-8");
        request.setCharacterEncoding("utf-8");
        // Получение параметра из строки запроса
        String parameter = request.getParameter("salary"); //может быть null
        // Сохранение зарплаты в сессии
        request.getSession().setAttribute("salary", parameter);
        // Сохранение зарплаты в Cookie
        Cookie c = new Cookie("salary", URLEncoder.encode(parameter, "UTF-8"));
        // Установка времени жизни Cookie в секундах
        c.setMaxAge(100);
        response.addCookie(c);
        // Перенаправление на страницу
        response.sendRedirect(response.encodeRedirectURL(request.getContextPath() +
            "/TeamTitle.jsp"));
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
        IOException {
        // TODO Auto-generated method stub
        response.getWriter().append("Served at: ").append(request.getContextPath());
        Cookie cookie = new Cookie("salary", "17000");
        response.addCookie(cookie);
        processRequest(request, response);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException {
        // TODO Auto-generated method stub
        processRequest(request, response);
    }
}

```

Затем происходит переход на следующую страницу TeamTitle.jsp, которая уже работает с кукой

```

<%@ page import="java.util.ResourceBundle"%>
<%@ page import="java.util.Locale"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">

```

```

<title>Список футболистов</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
<%
request.setCharacterEncoding("UTF-8");
//String salary = request.getParameter("salary");
String salary = (String)session.getAttribute("salary");
String lang = request.getParameter("lang");

if (lang == null) lang = "ru";
if (!"en".equalsIgnoreCase(lang) && !"ru".equalsIgnoreCase(lang)) {
    response.sendError(HttpServletResponse.SC_NOT_ACCEPTABLE,
        "Параметр lang может принимать значения ru или en вместо \"\" + lang + "\"");
    return;
}
if (salary.equals("")) salary = null;
ResourceBundle res = ResourceBundle.getBundle("team", new Locale(lang));
%>
<%//@include file="Header.jsp"%>
<h1>
    <%=res.getString("title") %>
    <%= (salary == null)? " " : (res.getString("condition") + salary + "$") %>
</h1>
<%
Object[][] team = new Object[][] {
    {"Билли Херрингтон", 0, "Махачкала", 15000},
    {"Антон Чехов", 1, "Санкт-Петербург", 30000},
    {"Илья Антонов", 2, "Екатеринбург", 25000},
    {"Андрей Сачков", 3, "Вологда", 19000}
};
String[] roles = new String[] {"Вратарь", "Нападающий", "Полузащитник", "Защитник"};
%>
<table border='1'>
    <tr>
        <td><b><%=res.getString("name") %></b></td>
        <td><b><%=res.getString("spec") %></b></td>
        <td><b><%=res.getString("city") %></b></td>
        <td><b><%=res.getString("salary") %></b></td>
    </tr>
    <%
    for (Object[] temp : team)
        if (salary == null || (int)temp[3] >= Integer.parseInt(salary))
            out.println("<tr><td>" + temp[0] + "</td><td>" + roles[(int)temp[1]] + "</td><td>"
                + temp[2] + "</td><td>" + Integer.toString((int)temp[3]) + "</td></tr>");
    %>
</table>
<%@include file="footer.jsp"%>
</body>
</html>

```

## Демонстрация результатов

Приложение приветствует формой, которую можно не заполнять, если необходимо вывести полную таблицу

← → ↻ localhost:8081/TestServlet/

## Введите минимальную зарплату футболиста

Ввод

После ввода значения 17000 выводится таблица с игроками, чья зарплата больше 17000

← → ↻ localhost:8081/TestServlet/TeamTitle.jsp

## Вот мои футболисты с зарплатой $\geq 17000\$$

Имя Фамилия	Специализация	Город	Зарплата
Антон Чехов	Нападающий	Санкт-Петербург	30000
Илья Антонов	Полузащитник	Екатеринбург	25000
Андрей Сачков	Защитник	Вологда	19000

ru ▾ Submit

Если закрыть и снова открыть стартовую страницу, то в форме появляется предыдущее вводимое значение, значит информация в виде куки сохранилась

← → ↻ localhost:8081/TestServlet/

## Введите минимальную зарплату футболиста

Ввод

Остальной функционал работает так же, как и в предыдущих лабораторных: например, если ввести неверный формат зарплаты, то выдаст ошибку, и если поменять язык, то он успешно поменяется

---

## Некорректно введенная зарплата

Значение должно быть целым числом без литералов



## Вывод

В ходе выполнения лабораторной работы были изучены и освоены методы передачи информации между соединениями в рамках одного сеанса работы пользователя. В частности, были выполнены следующие шаги:

### 1. Передача данных через файлы Cookie:

- Была создана страница `welcome.jsp`, которая содержит форму для ввода зарплаты. Значение из предыдущего Cookie автоматически заполняется в форме.
- Сервлет `SalaryProcessor.java` обрабатывает данные из формы, сохраняет их в Cookie и перенаправляет пользователя на страницу `TeamTitle.jsp`.

### 2. Использование Cookie для сохранения данных:

- Значение зарплаты сохраняется в Cookie, что позволяет сохранять данные между запросами.
- При повторном открытии страницы `welcome.jsp` предыдущее значение зарплаты автоматически подставляется в форму.

### 3. Демонстрация работы:

- Приложение успешно работает с использованием Cookie для сохранения данных.

- Остальные функции, такие как обработка ошибок и интернационализация, продолжают работать как в предыдущих лабораторных работах.

Таким образом, в результате выполнения лабораторной работы были успешно реализованы методы передачи данных через Cookie, что позволяет сохранять информацию между запросами пользователя.