

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ИНСТИТУТ НЕПРЕРЫВНОГО И ДИСТАНЦИОННОГО ОБРАЗОВАНИЯ

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

ОЦЕНКА

ПРЕПОДАВАТЕЛЬ

канд. техн. наук, доцент
должность, уч. степень, звание

подпись, дата

П.А. Степанов
инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

Разработка простого серверного приложения J2EE с
использованием сервлетов

по дисциплине: Технология разработки серверных информационных систем

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. №

Z1431
номер группы

подпись, дата

М.Д.Быстров
инициалы, фамилия

Студенческий билет №

2021/3572

Санкт-Петербург 2025

Оглавление

Задание	3
Описание разрабатываемого продукта	4
Текст основных фрагментов кода.....	5
Вывод.....	10

Задание

Вариант 4: Расписание поездов, самолетов, кораблей

Выполните следующие задачи.

1. В соответствии со своим вариантом разработайте набор из минимум двух экранных форм приложения
2. Соберите проект веб-приложения (war) на Maven (можно без использования Spring)
3. реализуйте формы средствами сервлетов. Проект должен как минимум содержать формы просмотра, добавления и удаления данных. **Не используйте технологию JSP (Java Server Pages)**
4. Аргументируйте почему были выбраны те или иные запросы HTTP.
5. Использовать базу данных можно, но не обязательно

Описание разрабатываемого продукта

Веб-приложение – онлайн-табло отправлений.

Две формы – форма просмотра отправлений, форма создания отправлений.

На форме просмотра отправлений расположена таблица с данными отправлений, где показаны данные об отправлениях: номер, пункт отправления, пункт назначения, время отправления, время прибытия, тип транспорта, кнопка удаления (рисунок 1).

[Добавить](#)

№	От	До	С	ПО	Вид	
1	SPB	MSK	2024-02-12T00:00	2024-02-12T01:00	Поезд	Удалить
2	123	123	2025-03-19T23:13	2025-03-19T23:15	Корабль	Удалить

Рисунок 1 форма просмотра отправлений

На форме создания отправлений расположены поля ввода для реквизитов отправлений (рисунок 2).

Создать отправление

[К отправлениям](#)

От

До

Отправление В

дд.мм.гггг --:--

Прибытие В

дд.мм.гггг --:--

Тип

--Выберите--

Создать

Рисунок 2 форма создания отправлений

Текст основных фрагментов кода

1. Метод doGet главного сервлета (форма просмотра данных)

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

    ScheduleDataSource ds = ScheduleDataSource.getInstance();

    PrintWriter writer = response.getWriter();

    Utils.appendHead(writer);

    writer.append("<a href=\"createDeparture\">Добавить</a>");

    writer.append("<table border=\"1px\" class=\"table table-bordered\">");

    writer.append("<tr>");

    writer.append("<th>№</th>");
    writer.append("<th>От</th>");
    writer.append("<th>До</th>");
    writer.append("<th>С</th>");
    writer.append("<th>ПО</th>");
    writer.append("<th>Вид</th>");

    writer.append("</tr>");

    ArrayList<Departure> deps = ds.getDepartures();

    for (Departure dep : deps){
        writer.append("<tr>");

        writer.append("<td> " + new Integer(dep.getId()).toString() +
"</td>");

        writer.append("<td> " + (dep.getSource()).toString() + "</td>");
        writer.append("<td> " + (dep.getDestination()).toString() +
"</td>");

        writer.append("<td> " + (dep.getFromTimestamp()).toString() +
"</td>");

        writer.append("<td> " + (dep.getToTimeStamp()).toString() +
"</td>");

        writer.append("<td> " + (dep.getDepartureType()).toString() +
"</td>");

        writer.append("<td>");

        writer.append(String.format("<form
action=\"deleteDeparture?id=%d\" method=\"post\">\r\n"
+ "    <button type=\"submit\" name=\"your_name\"
value=\"your_value\" class=\"btn-link\">Удалить</button>\r\n"
+ "</form>", dep.getId()));

        writer.append("</td>");

        writer.append("</tr>");
    }

    writer.append("</table>");
```

```

        Utils.appendTail(writer);
    }

```

2. Методы doGet, doPost сервлета создания отправления (форма создания отправлений)

```

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

    PrintWriter writer = response.getWriter();

    Utils.appendHead(writer);

    writer.append("<h1>Создать отправление</h1>");

    writer.append("<a href=\"/webapp/MainServlet\">K
отправлениям</a><br><br>");

    response.getWriter().append(
        "<form method=\"POST\" enctype=\"multipart/form-
data\"><br>\r\n"
        + "<label for=\"source\">ОТ</label>"
        + " <input name=\"source\"><br>\r\n<br>"
        + "<label for=\"destination\">ДО</label>"
        + " <input name=\"destination\"><br>\r\n"
        + "<label for=\"fromTimestamp\">Отправление В</label>"
        + " <input name=\"fromTimestamp\" type=\"datetime-
local\"><br>\r\n"
        + "<label for=\"toTimestamp\">Прибытие В</label>"
        + " <input name=\"toTimestamp\" type=\"datetime-
local\"><br>\r\n"
        + "<label for=\"departureType\">Тип</label>"
        + "<select name=\"departureType\"><br>\r\n"
        + " <option value=\"None\">--Выберите--</option><br>\r\n"
        + " <option value=\"Train\">Поезд</option><br>\r\n"
        + " <option value=\"Plane\">Самолет</option><br>\r\n"
        + " <option value=\"Ship\">Корабль</option><br>\r\n"
        + "</select>"
        + "<button>Создать</button>"
        + "</form>");

    Utils.appendTail(writer);
}

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

    Map<String, String[]> parameters = request.getParameterMap();

    if (parameters.size() > 0) {
        Departure dep = Departure.fromParameterMap(parameters);

        ScheduleDataSource.getInstance().addDeparture(dep);
    }

    response.setStatus(HttpServletResponse.SC_MOVED_PERMANENTLY);
    response.setHeader("Location", "/webapp/MainServlet");
}

```

```
}
```

3. Метод doPost сервлета удаления отправления

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

    ScheduleDataSource sds = ScheduleDataSource.getInstance();

    String idStr = request.getParameter("id");

    if (idStr == null) {
        throw new ServletException("id is not present in query");
    }

    int id = Integer.parseInt(idStr);
    if (!sds.removeDeparture(id)) {
        throw new ServletException("Can't remove departure");
    }

    response.setStatus(HttpServletResponse.SC_MOVED_PERMANENTLY);
    response.setHeader("Location", "/webapp/MainServlet");
}
```

4. Методы класса Utils

```
public static void appendHead(PrintWriter writer) {
    writer.append("<!doctype html>\r\n"
        + "<html lang=\"en\">\r\n"
        + "    <head>\r\n"
        + "        <meta charset=\"utf-8\">\r\n"
        + "        <meta name=\"viewport\" content=\"width=device-
width, initial-scale=1\">\r\n"
        + "        <title>Расписание</title>\r\n"
        + "        <link
href=\"https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css\"
rel=\"stylesheet\" integrity=\"sha384-
rbsA2VBKQhggwzxH7pPCaAqO46MgnOM80zW1RWuH61DGLwZJEdK2Kadq2F9CUG65\"
crossorigin=\"anonymous\">\r\n"
        + "    </head>\r\n"
        + "    <body>\r\n"
    );
}

public static void appendTail(PrintWriter writer) {
    writer.append("</body>\r\n"
        + "</html>"
    );
}
```

5. Класс Departure

```
public class Departure {
    private int id;
    private String source;
    private String destination;
```

```

private LocalDateTime fromTimestamp;
private LocalDateTime toTimestamp;
private DepartureType departureType;

public int getId() {
    return id;
}
public void setId(int id) {
    this.id = id;
}
public String getSource() {
    return source;
}
public void setSource(String source) {
    this.source = source;
}
public String getDestination() {
    return destination;
}
public void setDestination(String destination) {
    this.destination = destination;
}
public LocalDateTime getFromTimestamp() {
    return fromTimestamp;
}
public void setFromTimestamp(LocalDateTime fromTimestamp) {
    this.fromTimestamp = fromTimestamp;
}
public LocalDateTime getToTimestamp() {
    return toTimestamp;
}
public void setToTimestamp(LocalDateTime toTimestamp) {
    this.toTimestamp = toTimestamp;
}

public DepartureType getDepartureType() {
    return departureType;
}
public void setDepartureType(DepartureType departureType) {
    this.departureType = departureType;
}

public static Departure fromParameterMap(Map<String, String[]> parameterMap)
{
    Map<String, String> props = new HashMap<String, String>();

    parameterMap.forEach((k, v) -> {
        if (v.length > 0) {
            props.put(k, v[0]);
        }
    });

    return fromPropertyMap(props);
}

public static Departure fromPropertyMap(Map<String, String> parameterMap)
{
    Departure dep = new Departure();

    if (parameterMap.containsKey("source")) {
        dep.setSource(parameterMap.get("source"));
    }
}

```



```

    }

    if (parameterMap.containsKey("destination")) {
        dep.setDestination(parameterMap.get("destination"));
    }

    if (parameterMap.containsKey("fromTimestamp")) {
        String timestamp = parameterMap.get("fromTimestamp");

        LocalDateTime dateTime = LocalDateTime.parse(timestamp);

        dep.setFromTimestamp(dateTime);
    }

    if (parameterMap.containsKey("toTimestamp")) {
        String timestamp = parameterMap.get("toTimestamp");

        LocalDateTime dateTime = LocalDateTime.parse(timestamp);

        dep.setToTimestamp(dateTime);
    }

    if (parameterMap.containsKey("departureType")) {
        DepartureType type = Enum.valueOf(
            DepartureType.class,
            parameterMap.get("departureType"));

        dep.setDepartureType(type);
    }

    return dep;
}
}

```

Вывод

В ходе выполнения первой лабораторной работы было разработано веб-приложение из двух форм с использованием сервлетов Java EE. Проект создан с использованием системы сборки maven, распространение приложения осуществляется в виде пакета war.

Для получения данных используется метод HTTP GET, для модификации данных метод POST. Выбор соответствующих методов обусловлен их назначением в стандарте HTTP, а также кешированием/его отсутствием (к примеру, удаление либо модификация данных методом GET, а не POST в современных браузерах может не работать, т.к. полностью совпадающий запрос может быть отдан из локального кеша, а не получен с помощью отправки запроса на сервер).

В качестве веб-сервера используется Apache Tomcat.