

ГУАП

КАФЕДРА № 43

ОТЧЕТ  
ЗАЩИЩЕН С ОЦЕНКОЙ  
ПРЕПОДАВАТЕЛЬ

ассистент

подпись, дата

Кочин Д.А.

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1

Разработка простого серверного приложения J2EE с  
использованием сервлетов

по дисциплине: ТЕХНОЛОГИИ РАЗРАБОТКИ СЕРВЕРНЫХ  
ИНФОРМАЦИОННЫХ СИСТЕМ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

4936

подпись, дата

Назаров М.Р.

фамилия, инициалы

Санкт-Петербург 2021

## Лабораторная работа №1

Разработка простого серверного приложения J2EE с использованием сервлетов

### Цель работы:

Разработать серверное веб-приложения на основе Maven с использованием сервлетов и возможным применением Spring и баз данных, разработать экранную форму и установить взаимодействие между формой и серверным приложением на основе GET и POST запросов.

### Задание на лабораторную работу:

1. В соответствии со своим вариантом разработайте набор экранных форм приложения (порядка 5)
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
4. Аргументируйте почему были выбраны HTTP методы GET, POST или оба
5. Использовать базу данных можно, но не обязательно

## Вариант №14

Сдача недвижимости в аренду

### Ход работы:

Создадим проект на основе Spring с помощью среды разработки. В разработке использовался язык программирования Kotlin и сборщик Gradle.

Разработаем необходимые экранные формы с помощью языка разметки html с использованием thymeleaf и Bootstrap, обеспечивающие все необходимые функции (добавление, удаление и просмотр информации соответствующей варианту задания).

С помощью функций фреймворка Spring создадим необходимые сервлеты:

Полный код программы содержится в Приложение 1.

```
package com.example.servlnazarov.controllers;  
  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.GetMapping;  
  
@Controller  
public class MainController {
```

```

@GetMapping("/")
String getHome() { return "home"; }

@GetMapping("/error")
String getError() { return "error"; }
}

```

Аннотация `@Controller` помечает выбранный класс в качестве сервлета, обрабатывающего запросы, приложение автоматически будет перенаправлять необходимые запросы к соответствующему методу обработчика с пометкой `@Get/PostMapping`. Аннотация `@GetMapping` помечает метод в качестве обработчика для GET запросов по выбранному url, а `PostMapping` соответственно POST запросы. `@RequestParam` используется для получения определенных полей из поступающих POST запросов на сервер.

GET запросы предназначены для просмотра данных и соответственно вывода пользователю, направлены именно на получение информации с возможностью при необходимости конкретизации с использованием параметров (в нашем случае ответом на такой запрос является одна из html форм).

POST запросы используются для передачи каких-либо данных из форм на сторону сервера в неявном виде для пользователя.

Разработаем модель данных, соответствующую варианту с полями идентификатора, типа, названия, описания и цены вещи.

```

public class Estate {

    private final Integer id;
    private final String name;
    private final String city;
    private final String address;
    private final Double area;
    private final Integer cost;

    public Estate(
        Integer id,
        String name,
        String city,
        String address,
        Double area,
        Integer cost) {

        this.id = id;
        this.name = name;
    }
}

```

```
this.city = city;
this.address = address;
this.area = area;
this.cost = cost;
}

public Integer getId() { return id; }
public String getName() { return name; }
public String getCity() { return city; }
public String getAddress() { return address; }
public Integer getCost() { return cost; }
public Double getArea() { return area; }
```

Проверим работу программы:

[Home](#) [Estates](#) [Create announcement](#)

## Сайт сдачи недвижимости в аренду

Просмотр данных (GET):

[Home](#) [Estates](#) [Create announcement](#)

### List of estates:

#### First apartments

27000₽

Saint Petersburg, Nevsky Prospect, 3

28.0m<sup>2</sup>



Добавление данных (POST):

Create your own estate announcement

name

city

address

area


cost

create

xfgfxg

sdfsd, sdfsd

5.0m<sup>2</sup>



25₽

Удаление данных (POST):

## List of estates:

<b>First apartments</b> Saint Petersburg, Nevsky Prospect, 3 28.0m <sup>2</sup> 	27000P
<b>Second apartments</b> Saint Petersburg, Nevsky Prospect, 27 31.8m <sup>2</sup> 	31000P
<b>Third apartments</b> Saint Petersburg, Bolshoy prospect, 73 70.5m <sup>2</sup> 	72999P
<b>Not apartments</b> Moscow, Lenin Street, 78 25.0m <sup>2</sup> 	55000P
<b>Good apartments</b> Vorkuta, Peace Street, 27 69.0m <sup>2</sup> 	26500P

**Вывод:** в ходе лабораторной работы были получены основные навыки разработки простых серверных приложений с помощью фреймворка Spring, изучены основы взаимодействия клиент-сервер посредством http запросов, освоен язык разметки html и проведена работа с созданием и применением базы данных.

# Приложение 1.

EstateController.java

```
package com.example.serv1nazarov.controllers;

import org.springframework.ui.Model;
import com.example.serv1nazarov.models.Estate;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.List;

@Controller
public class EstateController {

    private final static ArrayList<Estate> estateList = new ArrayList(List.of(
        (new Estate(1, "First apartments", "Saint Petersburg", "Nevsky Prospect, 3", 28.0, 27000)),
        (new Estate(2, "Second apartments", "Saint Petersburg", "Nevsky Prospect, 27", 31.8, 31000)),
        (new Estate(3, "Third apartments", "Saint Petersburg", "Bolshoy prospect, 73", 70.5, 72999)),
        (new Estate(4, "Not apartments", "Moscow", "Lenin Street, 78", 25.0, 55000)),
        (new Estate(5, "Good apartments", "Vorkuta", "Peace Street, 27", 69.0, 26500))
    ));
    private static Integer ID = estateList.size() + 1;

    @GetMapping("estates")
    String getEstates(Model model) {
        model.addAttribute("estates", estateList);
        return "estates";
    }

    @GetMapping("estates/add")
    String getEstatesAdd() {
        return "add";
    }

    @PostMapping("estates/add")
    String addEstate(@RequestParam String name,
        @RequestParam String city,
        @RequestParam String address,
        @RequestParam Double area,
        @RequestParam Integer cost) {

        if (name.isBlank() ||
            address.isBlank() ||
            area == null ||
            city.isBlank() ||
            cost == null) {

            return "redirect:/error";
        }
        estateList.add(new Estate(ID++, name, city, address, area, cost));
        return "redirect:/estates";
    }
}
```

```

    }

    @PostMapping("estates/remove/{id}")
    String removeEstate(@PathVariable Integer id) {
        estateList.removeIf(estate -> id.equals(estate.getId()));
        return "redirect:/estates";
    }
}

```

## MainController.java

```

package com.example.serv1nazarov.controllers;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class MainController {

    @GetMapping("/")
    String getHome() { return "home"; }

    @GetMapping("/error")
    String getError() { return "error"; }
}

```

## Estate.java

```

package com.example.serv1nazarov.models;

public class Estate {

    private final Integer id;
    private final String name;
    private final String city;
    private final String address;
    private final Double area;
    private final Integer cost;

    public Estate(
        Integer id,
        String name,
        String city,
        String address,
        Double area,
        Integer cost) {

        this.id = id;
        this.name = name;
        this.city = city;
        this.address = address;
        this.area = area;
    }
}

```



```

        this.cost = cost;
    }

    public Integer getId() { return id; }
    public String getName() { return name; }
    public String getCity() { return city; }
    public String getAddress() { return address; }
    public Integer getCost() { return cost; }
    public Double getArea() { return area; }
}

```

#### Serv1NazarovApplication.java

```

package com.example.serv1nazarov;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Serv1NazarovApplication {

    public static void main(String[] args) {
        SpringApplication.run(Serv1NazarovApplication.class, args);
    }

}

```

#### add.html

```

<!DOCTYPE html>
<html lang="en">
<head>

    <meta charset="UTF-8">
    <title>Creating estate</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-F3w7mX95PdgyTmZZMECAngseQB83DfGTowi0iMjiWaeVhAn4FJkqJByhZMI3AhiU"
crossorigin="anonymous">

    <style>
        input {
            margin-bottom: 5px;
        }
    </style>

</head>
<body>

<header th:insert="header :: header"></header>

<div class="container mt-5">
    <h2 align="middle">Create your own estate announcement</h2>
    <form method="post">
        <br>
        <input type="text" name="name" placeholder="name" class="form-control" aria-label="name">
        <input type="text" name="city" placeholder="city" class="form-control" aria-label="city">

```

```

        <input type="text" name="address" placeholder="address" class="form-control" aria-label="address">
        <input type="number" name="area" placeholder="area" class="form-control" aria-label="area" min="0"
step="0.1">
        <input type="number" name="cost" placeholder="cost" class="form-control" aria-label="cost" min="0">

        <button type="submit" class="btn btn-success btn-lg btn-block">create</button>
    </form>
</div>

</body>
</html>

```

error.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-F3w7mX95PdgyTmZZMECAngseQB83DfGTowi0iMjiWaeVhAn4FJkqJByhZMI3AhiU"
crossorigin="anonymous">
    <title>Error</title>

    <style>
        h1, h2 {
            font-family: Calibri, serif;
        }
    </style>

</head>
<body>

<header th:insert="header :: header"></header>
<div class="container mt-5">
    <h1 align="middle">This is error page</h1>
    <h2 align="middle">If you are here</h2>
    <h2 align="middle">You did something wrong ͇͇(͇)͇͇</h2>
</div>

</body>
</html>

```

estates.html

```

<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Estates list</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-F3w7mX95PdgyTmZZMECAngseQB83DfGTowi0iMjiWaeVhAn4FJkqJByhZMI3AhiU"
crossorigin="anonymous">

    <style>

        .cost {
            float: right;

```

```

    }
    p, strong {
        font-family: Calibri, serif;
        font-size: 20px;
    }

</style>
</head>
<body>

<header th:insert="header :: header"></header>
<div class="container mt-5">
    <div th:if="{!estates.isEmpty()}">
        <h1 align="middle">List of estates:</h1>
        <div th:each="el : ${estates}" class="alert alert-success mt-2">
            <p class="cost" th:utext="{el.cost} + 'P'"></p>
            <strong th:utext="{el.name}"></strong>
            <p th:utext="{el.city} + ', ' + {el.address}"></p>
            <p th:utext="{el.area} + 'm²'"></p>
            <form th:action="{estates/remove/' + {el.id}" method="post">
                <button th:href="{estates/remove/' + {el.id}" type="submit" class="btn btn-outline-danger btn-circle btn-
lg"><i class="bi bi-trash"></i></button>
            </form>
        </div>
    </div>
    <div th:unless="{!estates.isEmpty()}">
        <h1>List of estates is empty</h1>
    </div>
</div>

</body>
</html>

```

header.html

```

<div th:fragment="header">
    <div>
        <nav class="navbar navbar-expand-lg navbar-light bg-light">
            <a class="navbar-brand" href="/" style="margin-left: 25px">Home</a>
            <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-
controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav">
                    <li class="nav-item">
                        <a class="nav-link" href="/estates">Estates</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/estates/add">Create announcement</a>
                    </li>
                </ul>
            </div>
        </nav>
    </div>
</div>

```

home.html

```
<!DOCTYPE html>
<html lang="en">
<head>

  <meta charset="UTF-8">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-F3w7mX95PdgyTmZZMECAngseQB83DfGTowi0iMjiWaeVhAn4FJkqJByhZMI3AhiU"
crossorigin="anonymous">

  <title>Home</title>

  <style>
    h1, h2 {
      font-family: Calibri, serif;
    }
  </style>

</head>
<body>
<header th:insert="header :: header"></header>

  <div class="container mt-5">
    <h1 align="middle">Сайт сдачи недвижимости в аренду</h1>
  </div>

</body>
</html>
```