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

Кафедра компьютерных технологий и программной инженерии

ОТЧЁТ
ЗАЩИЩЁН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

ассистент		Кочин Д. А.
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

Отчёт по лабораторной работе №1

по курсу: Технология разработки серверных информационных систем

СТУДЕНТ ГР. № 4932 (номер группы) _____ (подпись, дата) С. И. Коваленко (инициалы, фамилия)

Санкт-Петербург
2021

1. Цель работы:

Разработка простого серверного приложения J2EE с использованием сервлетов.

2. Вариант:

9. Складской учёт.

3. Описание:

Функциональные требования к системе складского учёта:

- Возможность добавлять новые помещения
- Возможность удалять помещения
- Возможность просмотра всех имеющихся помещений
- Возможность добавлять объекты в помещения
- Возможность удалять объекты из помещений
- Возможность просмотра всех предметов в помещении

4. Вывод:

В ходе выполнение лабораторной работы было создано приложение на языке kotlin с использованием технологии сервлетов. В качестве сервера использовано tomcat 9.0.53.

Приложение А

```
// src/main/kotlin/erroreStorage.kt
object erroreStorage {
    var lastError = ""

    /** сообщить о возникновении ошибки **/
    fun error(error: String) {
        println(error)
    }

    /** возвращает текст последней ошибки **/
    fun getError() = lastError
}

// src/main/kotlin/Obj.kt
data class Obj(val name: String, val count: Int)

// src/main/kotlin/objFactory.kt
fun objFactory(name: String, count: Int) =
    if ( name.isNotEmpty() && count >= 0 )
        Obj(name, count)
    else
        null

// src/main/kotlin/Room.kt
data class Room(val name: String, val id: Int) {
    val listObj = mutableListOf<Obj>()

    /** количество наименований объектов **/
    fun size() = listObj.size

    /** добавить объекты в комнату **/
    fun put(obj: Obj) {
        for (o in listObj)
            if (o.name == obj.name) {
                o.count + obj.count
                return
            }
        listObj.add(obj)
    }

    /** выгрузить объекты из комнаты **/
    fun get(name: String, count: Int) {
        for (o in listObj) {
            if ( o.name == name )
                if ( o.count < count )
                    println("Error insufficient quantity in stock")
```

```
            else
                o.count - count
        }
    }

}

// src/main/kotlin/Storage.kt
object Storage {
    val roomList = mutableListOf<Room>()

    /** количество комнат **/
    fun size() = roomList.size

    /** добавление комнаты на склад
     * true - успешное добавлени**/
    fun addRoom(name: String, id: Int) : Boolean {
        if (name.isNotEmpty() && id >= 0) {
            for (r in roomList)
                if (r.name == name || r.id == id) {
                    erroreStorage.error("Error room with such data already exists")
                    return false
                }
            roomList.add(Room(name, id))
            return true
        } else {
            erroreStorage.error("Error directional data about the room")
            return false
        }
    }

    /** удаление комнаты **/
    fun deleteRoom(id: Int): Boolean {
        for (i in 0 until roomList.size) {
            if ( roomList[i].id == id ) {
                roomList.removeAt(i)
                return true
            }
        }
        erroreStorage.error("Error this room not found")
        return false
    }

}

// src/main/kotlin/lab/example/lab1/addObj.kt
package lab.example.lab1
```

```
import objFactory
import javax.servlet.annotation.WebServlet
import javax.servlet.http.HttpServlet
import javax.servlet.http.HttpServletRequest
import javax.servlet.http.HttpServletResponse

@WebServlet(name = "addObj", value = ["/addObj"])
class AddObjServlet: HttpServlet() {
    public override fun doPost(request: HttpServletRequest, response: HttpServletResponse) {

        val objName = request.getParameter("objName")
        val id = request.getParameter("roomId").toInt()
        val objCount = request.getParameter("objCount").toInt()
        val obj = objFactory(objName, objCount)
        println("room=$id;objName=$objName;objCount=$objCount")
        if ( obj != null )
            for (r in Storage.roomList)
                if (r.id == id)
                    r.put(obj)
        response.sendRedirect("room?roomId=$id")
    }
    override fun destroy() {
    }
}

// src/main/kotlin/lab/example/lab1/AddRoomServlet.kt
package lab.example.lab1

import javax.servlet.annotation.WebServlet
import javax.servlet.http.HttpServlet
import javax.servlet.http.HttpServletRequest
import javax.servlet.http.HttpServletResponse

@WebServlet(name = "addRoom", value = ["/addRoom"])
class AddRoomServlet: HttpServlet() {
    public override fun doPost(request: HttpServletRequest, response: HttpServletResponse) {
        val name = request.getParameter("roomName")
        val id   = request.getParameter("roomId").toInt()
        if ( Storage.addRoom(name, id)) {
            println("Add new room\n\tname = $name\n\tid   = $id\n");
        }
        response.sendRedirect("storage")
    }

    override fun destroy() {
    }
}
```

```
// src/main/kotlin/lab/example/lab1/DeleteRoomServlet.kt
package lab.example.lab1

import javax.servlet.annotation.WebServlet
import javax.servlet.http.HttpServlet
import javax.servlet.http.HttpServletRequest
import javax.servlet.http.HttpServletResponse

@WebServlet(name = "deleteRoom", value = ["/deleteRoom"])
class DeleteRoomServlet: HttpServlet() {
    public override fun doPost(request: HttpServletRequest, response: HttpServletResponse) {
        val id = request.getParameter("roomId").toInt()
        println("del $$id")
        if ( Storage.deleteRoom(id))
            println("Delete room id = $id\n")
        val out = response.writer
        out.println("""<html><body>
            |<a> Room [$id] was deleted</a>
            |</dr>
            |<a href="storage"> go back</a>
            |</body></html>""".trimMargin())
    }

    override fun destroy() {
    }
}

// src/main/kotlin/lab/example/lab1/GetObjServlet.kt
package lab.example.lab1

import javax.servlet.annotation.WebServlet
import javax.servlet.http.HttpServlet
import javax.servlet.http.HttpServletRequest
import javax.servlet.http.HttpServletResponse

@WebServlet(name = "getObj", value = ["/getObj"])
class GetObjServlet: HttpServlet() {
    public override fun doPost(request: HttpServletRequest, response: HttpServletResponse) {
        val id = request.getParameter("roomId").toInt()
        val objName = request.getParameter("objName")
        val objCount = request.getParameter("objCount").toInt()
        println("room=$id;objName=$objName;objCount=$objCount")
        for (r in Storage.roomList)
            if (r.id == id)
                r.get(objName, objCount)
        response.sendRedirect("room?roomId=$id")
    }
```

```
    override fun destroy() {
    }
}

// src/main/kotlin/lab/example/lab1/RoomServlet.kt
package lab.example.lab1

import javax.servlet.annotation.WebServlet
import javax.servlet.http.HttpServlet
import javax.servlet.http.HttpServletRequest
import javax.servlet.http.HttpServletResponse

@WebServlet(name = "room", value = ["/room"])
class RoomServlet: HttpServlet() {
    public override fun doGet(request: HttpServletRequest, response: HttpServletResponse) {
        val id  = request.getParameter("roomId").toInt()
        val out = response.writer
        println("print room id = $id\n")
        out.println("<html><body>")
            for (r in Storage.roomList)
                if ( r.id == id )
                    if ( r.size() != 0 )
                        r.listObj.forEach {
                            out.println("<a> ${it.name}    ${it.count}</a> </br>")
                        }
                    else
                        out.println("<h1>This room is empty </h1>")
        out.println("""
            |<h3>Put Object </h3>
            |<form name="loginForm" method="post" action="addObj?roomId=$id">
                    |Room Id: <input type="text" name="roomId"/> <br/>
                    |Object name: <input type="text" name="objName"/> <br/>
                    |Object count: <input type="text" name="objCount"/> <br/>
                    |<input type="submit" value="Put" />
            |</form>
            |<br/>""".trimMargin())
        out.println("""
            |<h3>Get Room </h3>
            |<form name="loginForm" method="post" action="getObj">
                    |Room Id: <input type="text" name="roomId"/> <br/>
                    |Object Name: <input type="text" name="objName"/> <br/>
                    |Object count: <input type="text" name="objCount"/> <br/>
                    |<input type="submit" value="Get" />
            |</form>
            |<br/>""".trimMargin())
        out.println("""|</dr>
            |<a href="storage"> go back</a>
            |</body></html>""".trimMargin())
```

```
    }

    override fun destroy() {
    }
}

// src/main/kotlin/lab/example/lab1/StorageServlet.kt
package lab.example.lab1

import javax.servlet.http.*
import javax.servlet.annotation.*

@WebServlet(name = "storage", value = ["/storage"])
class StorageServlet: HttpServlet() {

    override fun init() {
    }

    public override fun doGet(request: HttpServletRequest, response: HttpServletResponse) {
        response.contentType = "text/html"
        val out = response.writer

        if (Storage.size() != 0) {
            out.println("<h5> Room list</h5>")
            Storage.roomList.forEach {
                out.println("<a>${it.name}    ${it.id}</a> </br>")
            }
        }

        out.println(
            """
            |<h1>Add Room </h1>
            |<form name="loginForm" method="post" action="addRoom">
                    |Room name: <input type="text" name="roomName"/> <br/>
                    |Room id: <input type="text" name="roomId"/> <br/>
                    |<input type="submit" value="Add" />
            |</form>
            |<br/>""".trimMargin()
        )

        out.println(
            """
            |<h1>Delete Room </h1>
            |<form name="loginForm" method="post" action="deleteRoom">
                    |Room Id: <input type="text" name="roomId"/> <br/>
                    |<input type="submit" value="Delete" />
            |</form>
            |<br/>""".trimMargin()
```

```
        )

        out.println(
            """
            |<h1>Open Room </h1>
            |<form name="loginForm" method="get" action="room">
                  |RoomId: <input type="text" name="roomId"/> <br/>
                  |<input type="submit" value="Open" />
            |</form>
            |<br/>""".trimMargin()
        )

        out.println("</body></html>")
    }

    override fun destroy() {
    }
}
```