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 (КАФЕДРА №43)

|  |  |  |
| --- | --- | --- |
| ОТЧЕТ ЗАЩИЩЕН С ОЦЕНКОЙ: |  |  |

ПРЕПОДАВАТЕЛЬ:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| канд. техн. наук, |  |  |  | Степанов П.А. |
| (должность, уч. степень, звание) |  | (подпись, дата) |  | (инициалы, фамилия) |

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1

«Разработка простого серверного приложения J2EE c использованием сервлетов»

ПО КУРСУ: «Технология разработки серверных информационных систем»

|  |  |  |  |
| --- | --- | --- | --- |
| РАБОТУ ВЫПОЛНИЛ СТУДЕНТ: | Z0431 | / | Пасюкова Е.А. |
|  | (номер группы) |  | (инициалы, фамилия) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | / |  | / | 08.05.2024 |
|  |  | (подпись студента) |  | (дата отчета) |

Санкт-Петербург 2024

**Задание**

Разработка простого серверного приложения J2EE c использованием сервлетов

Выполните следующие задачи.

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**

23.Учет содержимого холодильника (со сроками годности)

**Ход работы**

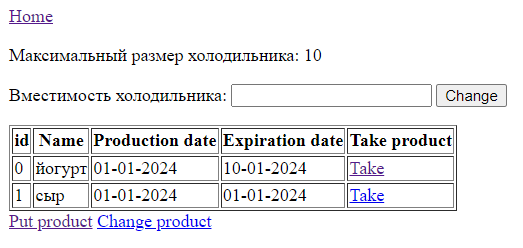
Репозиторий с проектом: <https://github.com/M9hvostik/TRSISLr1Var23>

Используемые технологии:

* JDK
* JSP
* Maven
* Spring
* NetBeans

В качестве основных двух сущностей выбраны “Fridge” и “Product”. Класс “Product” хранит минимальную необходимую информацию о продукте и реализует CRUD. Класс “Fridge” хранит в себе “List” – список продуктов, находящихся в холодильнике, и реализует CRUD для данного списка. Класс “Fridge” использует шаблон проектирования “Singleton”, в каждом сервлете используется указатель на экземпляр данного класса.

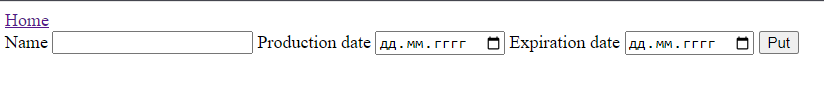
Реализованы 5 экранных форм приложения для изменения холодильника и его содержимого с помощью сервлетов и JSP:

1. Главная страница с просмотром содержимого  
   

Screenshot 1. Главная страница с просмотром содержимого.

Переход на страницу осуществляется с помощью адреса: <http://localhost:8080/fridge> .

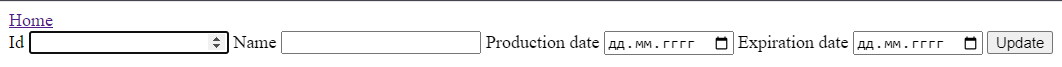
В сервлете “FridgeListServlet” методы doGet и doPost вызывают метод processRequest, который устанавливает необходимые атрибуты запроса для вывода информации и вызывает fridgeList.jsp.   
В данном случае ответ на запросы типа GET и POST идентичны, так как серверу не нужна дополнительная информация от клиента для вывода списка содержимого холодильника. Он просто перенаправляет запрос на страницу JSP.

1. Добавление продукта в холодильник  
   

Screenshot 2. Страница добавления продукта в холодильник.

Переход осуществляется по адресу: <http://localhost:8080/put>

В данном случае, так как цель – создать продукт, добавить данные, для этой задачи используется запрос POST и вставку данных сервлет “PutProductServlet” осуществляет с помощью метода doPost, метод doGet всего лишь перенаправляет на нужную страницу JSP с формой для заполнения данных о продукте put.jsp.

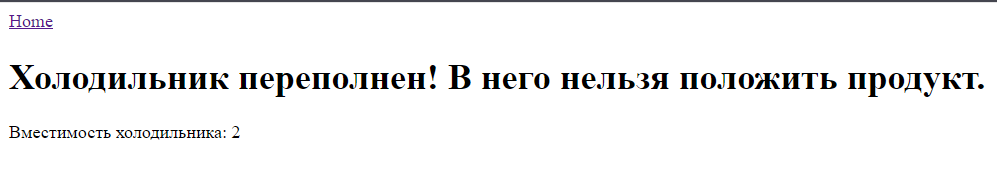
1. Страница с изменением данных продукта  
   

Screenshot 5. Страница с изменением данных продукта

Переход осуществляется по адресу: <http://localhost:8080/changeProduct>.

Задача изменить данные, следовательно используется запрос POST. Изменение данных сервлет “ChangeProductServlet” осуществляет с помощью метода doPost, метод doGet всего лишь перенаправляет на нужную страницу JSP с формой для заполнения данных о продукте changeProduct.jsp.

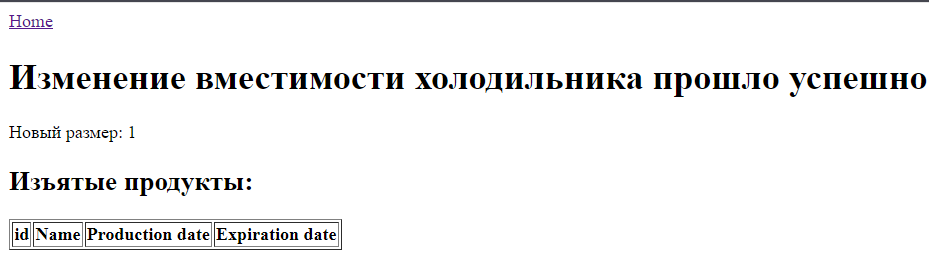
1. Страница с ошибкой в случае, если холодильник переполнен



Screenshot 4. Страница с ошибкой в случае, если холодильник переполнен.

На данную страницу можно попасть при попытке положить продукт в переполненный холодильник на странице <http://localhost:8080/put>.

Сервелет “PutProductServlet” перед вставкой продукта проверяет наполненность холодильника, в случае если места в холодильнике нет, вместо вставки он перенаправляет на страницу JSP с данным уведомлением errorFridgeIsFull.jsp(следовало бы такую проверку реализовать с помощью фильтров).

1. Страница с уведомлением об успешном изменении размера холодильника  
   

Screenshot 3. Страница с уведомлением об успешном изменении холодильника.

Переход осуществляется по адресу: <http://localhost:8080/changeMaxSizeFridge>.

На данную страницу можно попасть при изменении размера холодильника на главной странице. В случае, если продуктов в холодильнике больше, чем его вместимость, то продукты изымаются. Пользователь несёт ответственность за сохранность данных об извлеченных продуктах, а также за корректность введённых данных (полагаю корректность ввода стоит проверять в фильтрах).

Сервлет “ChangeMaxSizeFridgeServlet” осуществляет изменение размера холодильника в методе processRequest, который вызывается в doGet и doPost методах. Да, в данном случае мы передаем информацию о новом размере холодильника на сервер, но в связи с тем, что размер холодильника без проблем помещается в URI в Get запросе, и данная операция, согласно заданию, не предусматривает наличие какой-либо защищённости информации о размере холодильника, мы можем использовать и GET запрос, и POST одинаково.

Стоит также упомянуть сервлет “TakeProductServlet”. Он вызывается при нажатии на ссылку “Take” в холодильнике: [http://localhost:8080/take.](http://localhost:8080/taket) Данный сервлет удаляет продукт из холодильника в методе doGet. Запрос GET используется для удобства передачи данных между страницами, URI для каждого продукта генерируется в таблице холодильника в fridgeList.jsp.

**Листинги**

1. pom.xml

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.mycompany</groupId>

<artifactId>filter-servlet-example</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>war</packaging>

<url>https://maven.apache.org</url>

<name>filter and servlet example</name>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.7.1</version>

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-jasper</artifactId>

<!-- <version>10.1.18</version>-->

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

</dependency>

</dependencies>

<build>

<finalName>lab1</finalName>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<configuration>

<addResources>true</addResources>

</configuration>

<executions>

<execution>

<goals>

<goal>repackage</goal>

</goals>

<configuration>

<classifier>executable</classifier>

</configuration>

</execution>

</executions>

</plugin>

<plugin>

<artifactId>maven-resources-plugin</artifactId>

<version>2.6</version>

<executions>

<execution>

<id>copy-resources</id>

<phase>validate</phase>

<goals>

<goal>copy-resources</goal>

</goals>

<configuration>

<outputDirectory>${basedir}/target/classes/static</outputDirectory>

<resources>

<resource>

<directory>src/main/webapp</directory>

<filtering>true</filtering>

</resource>

</resources>

</configuration>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

1. Application.java

package Lr1;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.builder.SpringApplicationBuilder;

import org.springframework.boot.web.servlet.ServletComponentScan;

import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

@ServletComponentScan

@SpringBootApplication

public class Application extends SpringBootServletInitializer {

public static void main(String[] args) {

SpringApplication.run(applicationClass, args);

}

@Override

protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {

return application.sources(applicationClass);

}

private static final Class<Application> applicationClass = Application.class;

}

1. Product.java

package Lr1.model;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

/\*\*

\*

\* @author 0

\*/

public class Product {

private String name;

private Date productionDate;

private Date expirationDate;

private final String formatDate = "yyyy-mm-dd";

public Product(String newName, String newProductionDate, String newExpirationDate) throws ParseException

{

name = newName;

productionDate = new SimpleDateFormat(formatDate).parse(newProductionDate);

expirationDate = new SimpleDateFormat(formatDate).parse(newExpirationDate);

}

public void setName(final String newName)

{

name = newName;

}

public void setProductionDate(final Date newProdDate)

{

productionDate = newProdDate;

}

public void setProductionDate(final String newProdDate) throws ParseException

{

productionDate = new SimpleDateFormat(formatDate).parse(newProdDate);

}

public void setExpirationDate(final Date newExpDate)

{

expirationDate = newExpDate;

}

public void setExpirationDate(final String newExpDate) throws ParseException

{

expirationDate = new SimpleDateFormat(formatDate).parse(newExpDate);

}

public String getName()

{

return name;

}

public Date getProductionDate()

{

return productionDate;

}

public String getStringProductionDate()

{

return new SimpleDateFormat(formatDate).format(productionDate);

}

public Date getExpirationDate()

{

return expirationDate;

}

public String getStringExpirationDate()

{

return new SimpleDateFormat(formatDate).format(expirationDate);

}

}

1. Fridge.iava

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

\*/

package Lr1.model;

import java.text.ParseException;

import java.util.ArrayList;

import java.util.List;

/\*\*

\*

\* @author 0

\*/

public class Fridge {

private static Fridge instance;

private int maxSizeFridge = 10;

private List<Product> fridge = new ArrayList<>(maxSizeFridge);

private Fridge() {};

public static Fridge getInstance()

{

if (instance == null) {

instance = new Fridge();

}

return instance;

}

public List<Product> setMaxSizeFridge(int size)

{

maxSizeFridge = size;

int currentSize = fridge.size();

List<Product> removedProducts = new ArrayList<>();

while(maxSizeFridge<currentSize)

{

removedProducts.add(fridge.remove(currentSize-1));

currentSize-=1;

}

return removedProducts;

}

public boolean putProduct(Product product)

{

if (fridge.size() < maxSizeFridge)

{

fridge.add(product);

return true;

}

return false;

}

public boolean putProduct(String name, String productionDate, String expirationDate) throws ParseException

{

if (fridge.size() < maxSizeFridge)

{

fridge.add(new Product(name,productionDate, expirationDate));

return true;

}

return false;

}

public Product takeProduct(int index)

{

return fridge.remove(index);

}

public Product takeProduct(Product product)

{

int index = fridge.indexOf(product);

if(index == -1)

return null;

return fridge.remove(index);

}

public Product takeProduct(String name, String productionDate, String expirationDate) throws ParseException

{

Product product = new Product(name, productionDate, expirationDate);

int index = fridge.indexOf(product);

if(index == -1)

return null;

return fridge.remove(index);

}

public final List<Product> getFridge()

{

return fridge;

}

public int getMaxSizeFridge()

{

return maxSizeFridge;

}

public void updateProduct(int index, String name,String productionDate, String expirationDate ) throws ParseException

{

Product product = fridge.get(index);

product.setName(name);

product.setProductionDate(productionDate);

product.setExpirationDate(expirationDate);

}

}

1. FridgeListServlet.java

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

\*/

package Lr1.servlets;

import Lr1.model.Fridge;

import Lr1.model.Product;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\*

\* @author 0

\*/

@WebServlet(name = "FridgeListServlet", urlPatterns = {"/fridge"})

public class FridgeListServlet extends HttpServlet {

private Fridge fridgeObj = Fridge.getInstance();

/\*\*

\* Processes requests for both HTTP <code>GET</code> and <code>POST</code>

\* methods.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException

{

response.setContentType("text/html");

request.setAttribute("maxSize", fridgeObj.getMaxSizeFridge());

request.setAttribute("fridge", fridgeObj.getFridge());

request.getRequestDispatcher("fridgeList.jsp").forward(request, response);

}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/\*\*

\* Handles the HTTP <code>GET</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Handles the HTTP <code>POST</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

request.getRequestDispatcher("fridgeList.jsp").forward(request, response);

}

}

1. PutProductServlet.java

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/JSP\_Servlet/Servlet.java to edit this template

\*/

package Lr1.servlets;

import Lr1.model.Fridge;

import Lr1.model.Product;

import java.io.IOException;

import java.io.PrintWriter;

import java.text.ParseException;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\*

\* @author 0

\*/

@WebServlet(name = "PutProductServlet", urlPatterns = {"/put"})

public class PutProductServlet extends HttpServlet {

private Fridge fridgeObj = Fridge.getInstance();

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/\*\*

\* Handles the HTTP <code>GET</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

request.getRequestDispatcher("put.jsp").forward(request, response);

}

/\*\*

\* Handles the HTTP <code>POST</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

String name = request.getParameter("name");

String productionDate = request.getParameter("productionDate");

String expirationDate = request.getParameter("expirationDate");

Product product = null;

try {

product = new Product(name,productionDate, expirationDate);

} catch (ParseException ex) {

Logger.getLogger(PutProductServlet.class.getName()).log(Level.SEVERE, null, ex);

}

if(fridgeObj.putProduct(product))

{

request.setAttribute("fridge", fridgeObj.getFridge());

response.sendRedirect("/fridge");

}

else

{

request.setAttribute("maxSize", fridgeObj.getMaxSizeFridge());

request.getRequestDispatcher("errorFridgeIsFull.jsp").forward(request, response);

}

}

/\*\*

\* Returns a short description of the servlet.

\*

\* @return a String containing servlet description

\*/

@Override

public String getServletInfo() {

return "Short description";

}// </editor-fold>

}

1. TakeProductServlet.java

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/JSP\_Servlet/Servlet.java to edit this template

\*/

package Lr1.servlets;

import Lr1.model.Fridge;

import Lr1.model.Product;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\*

\* @author 0

\*/

@WebServlet(name = "TakeProductServlet", urlPatterns = {"/take"})

public class TakeProductServlet extends HttpServlet {

private Fridge fridgeObj = Fridge.getInstance();

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/\*\*

\* Handles the HTTP <code>GET</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

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

int id = Integer.parseInt(idStr);

Product product = fridgeObj.takeProduct(id);

request.setAttribute("fridge", fridgeObj.getFridge());

response.sendRedirect("/fridge");

}

/\*\*

\* Handles the HTTP <code>POST</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

}

/\*\*

\* Returns a short description of the servlet.

\*

\* @return a String containing servlet description

\*/

@Override

public String getServletInfo() {

return "Short description";

}// </editor-fold>

}

1. ChangeProductServlet.java

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/JSP\_Servlet/Servlet.java to edit this template

\*/

package Lr1.servlets;

import Lr1.model.Fridge;

import java.io.IOException;

import java.io.PrintWriter;

import java.text.ParseException;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\*

\* @author 0

\*/

@WebServlet(name = "ChangeProductServlet", urlPatterns = {"/changeProduct"})

public class ChangeProductServlet extends HttpServlet {

private Fridge fridgeObj = Fridge.getInstance();

/\*\*

\* Processes requests for both HTTP <code>GET</code> and <code>POST</code>

\* methods.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/\*\*

\* Handles the HTTP <code>GET</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

request.getRequestDispatcher("changeProduct.jsp").forward(request, response);

}

/\*\*

\* Handles the HTTP <code>POST</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

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

int id = Integer.parseInt(idStr);

String name = request.getParameter("name");

String productionDate = request.getParameter("productionDate");

String expirationDate = request.getParameter("expirationDate");

try {

fridgeObj.updateProduct(id, name, productionDate, expirationDate);

} catch (ParseException ex) {

Logger.getLogger(ChangeProductServlet.class.getName()).log(Level.SEVERE, null, ex);

}

request.setAttribute("fridge", fridgeObj.getFridge());

response.sendRedirect("/fridge");

}

/\*\*

\* Returns a short description of the servlet.

\*

\* @return a String containing servlet description

\*/

@Override

public String getServletInfo() {

return "Short description";

}// </editor-fold>

}

1. ChangeMaxSizeFridgeServlet.java

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/JSP\_Servlet/Servlet.java to edit this template

\*/

package Lr1.servlets;

import Lr1.model.Fridge;

import Lr1.model.Product;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\*

\* @author 0

\*/

@WebServlet(name = "ChangeMaxSizeFridgeServlet", urlPatterns = {"/changeMaxSizeFridge"})

public class ChangeMaxSizeFridgeServlet extends HttpServlet {

private Fridge fridgeObj = Fridge.getInstance();

/\*\*

\* Processes requests for both HTTP <code>GET</code> and <code>POST</code>

\* methods.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

String maxSizeStr = request.getParameter("maxSize");

int maxSize = Integer.parseInt(maxSizeStr);

response.setContentType("text/html;charset=UTF-8");

List<Product> removedProducts = fridgeObj.setMaxSizeFridge(maxSize);

request.setAttribute("maxSize", fridgeObj.getMaxSizeFridge());

request.setAttribute("removedProduct", removedProducts);

request.getRequestDispatcher("changeMaxSizeSuccessfully.jsp").forward(request, response);

}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/\*\*

\* Handles the HTTP <code>GET</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Handles the HTTP <code>POST</code> method.

\*

\* @param request servlet request

\* @param response servlet response

\* @throws ServletException if a servlet-specific error occurs

\* @throws IOException if an I/O error occurs

\*/

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Returns a short description of the servlet.

\*

\* @return a String containing servlet description

\*/

@Override

public String getServletInfo() {

return "Short description";

}// </editor-fold>

}

1. fridgeList.jsp

<%@ page contentType="text/html;charset=UTF-8" %>

<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix = "c" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>

<html>

<head>

<title>Title</title>

</head>

<body>

<header><a href="${pageContext.request.contextPath}/fridge">Home</a></header>

<p>

Максимальный размер холодильника: ${maxSize}

<form method="post" action="${pageContext.request.contextPath}/changeMaxSizeFridge">

<label for="maxSize"> Вместимость холодильника:

<input type="number" id="maxSize" name="maxSize">

</label>

<input type="submit" value="Change">

</form>

</p>

<table border="1">

<tr>

<th>id</th>

<th>Name</th>

<th>Production date</th>

<th>Expiration date</th>

<th>Take product</th>

</tr>

<c:forEach items="${fridge}" var="product" varStatus="tierOneCount">

<tr>

<td>${tierOneCount.index}</td>

<td>${product.name}</td>

<td><fmt:formatDate value="${product.productionDate}" pattern="dd-MM-yyyy" /></td>

<td><fmt:formatDate value="${product.expirationDate}" pattern="dd-MM-yyyy" /></td>

<td><a href="${pageContext.servletContext.contextPath}/take?productId=${tierOneCount.index}">Take</a></td>

</tr>

</c:forEach>

</table>

<a href="${pageContext.request.contextPath}/put">Put product</a>

<a href="${pageContext.servletContext.contextPath}/changeProduct">Change product</a>

</body>

</html>

1. put.jsp

<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<html>

<head>

<title>Put</title>

</head>

<body>

<header><a href="${pageContext.request.contextPath}/fridge">Home</a></header>

<form method="post" action="${pageContext.request.contextPath}/put">

<label for="name"> Name

<input type="text" id="name" name="name">

</label>

<label for="productionDate"> Production date

<input type="date" id="productionDate" name="productionDate">

</label>

<label for="expirationDate"> Expiration date

<input type="date" id="expirationDate" name="expirationDate">

</label>

<input type="submit" value="Put">

</form>

</body>

</html>

1. changeProduct.jsp

<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<html>

<head>

<title>Change</title>

</head>

<body>

<header><a href="${pageContext.request.contextPath}/fridge">Home</a></header>

<form method="post" action="${pageContext.request.contextPath}/changeProduct">

<label for="productId"> Id

<input type="number" id="productId" name="productId">

</label>

<label for="name"> Name

<input type="text" id="name" name="name">

</label>

<label for="productionDate"> Production date

<input type="date" id="productionDate" name="productionDate">

</label>

<label for="expirationDate"> Expiration date

<input type="date" id="expirationDate" name="expirationDate">

</label>

<input type="submit" value="Update">

</form>

</body>

</html>

1. changeMaxSizeSuccessfully.jsp

<%--

Document : changeMaxSizeSuccessfully

Created on : 6 мая 2024 г., 19:57:57

Author : 0

--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix = "c" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Change max size Successfully</title>

</head>

<body>

<header><a href="${pageContext.request.contextPath}/fridge">Home</a></header>

<h1>Изменение вместимости холодильника прошло успешно</h1>

<p>Новый размер: ${maxSize}</p>

<h2>Изъятые продукты:</h2>

<table border="1">

<tr>

<th>id</th>

<th>Name</th>

<th>Production date</th>

<th>Expiration date</th>

</tr>

<c:forEach items="${removedProduct}" var="product" varStatus="tierOneCount">

<tr>

<td>${tierOneCount.index}</td>

<td>${product.name}</td>

<td><fmt:formatDate value="${product.productionDate}" pattern="dd-MM-yyyy" /></td>

<td><fmt:formatDate value="${product.expirationDate}" pattern="dd-MM-yyyy" /></td>

</tr>

</c:forEach>

</table>

</body>

</html>

1. errorFridgeIsFull.jsp

<%--

Document : errorFridgeIsFull

Created on : 6 мая 2024 г., 19:39:38

Author : 0

--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>ERROR Fridge is full</title>

</head>

<body>

<header><a href="${pageContext.request.contextPath}/fridge">Home</a></header>

<h1>Холодильник переполнен! В него нельзя положить продукт.</h1>

<p>Вместимость холодильника: ${maxSize}</p>

</body>

</html>