Код программы

Класс Contoller:

package server;  
  
  
import com.google.gson.Gson;  
import inputExceptionsMongo.\*;  
import mongo.MongoWorkNote;  
import objects.Book;  
import objects.Note;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.PrintWriter;  
import java.util.ArrayList;  
import java.util.List;  
  
@WebServlet(name="server", displayName="Notification Servlet", urlPatterns = {"/server"})  
public class Controller extends HttpServlet {  
  
 private String getBody(HttpServletRequest req) throws IOException {  
  
 StringBuilder buffer = new StringBuilder();  
 BufferedReader reader = req.getReader();  
 String line;  
 while ((line = reader.readLine()) != null) {  
 buffer.append(line);  
 buffer.append('\0');  
 }  
 String data = buffer.toString();  
 return data;  
 }

private List<String> parseBodyToList(String string){  
  
 char[] stringChar=string.toCharArray();  
 List<String> list=new ArrayList<String>();  
 int i=0;  
 while(i<stringChar.length){  
 StringBuilder buffer=new StringBuilder();  
 while (isValid(stringChar[i])){  
 if(stringChar[i]=='%'){  
 buffer.append(' ');  
 i+=3;  
 continue;  
 }  
 buffer.append(stringChar[i]);  
 i++;  
 }  
 String string1=buffer.toString();  
 list.add(string1);  
 i++;  
 }  
 return list;  
 }  
  
 private String getFromBodyListByName(List<String> list, String name) throws ParameterException {  
  
 if(!list.contains(name)){  
 throw new ParameterException();  
 }  
 int index=0;  
 index=list.indexOf(name);  
 index++;  
 String result=list.get(index);  
 return result;  
 }  
  
 private boolean isValid(char symbol){  
 return ((symbol>='a'&&symbol<='z') || (symbol>='A'&&symbol<='Z') || (symbol>= '0'&&symbol<='9') || (symbol==' ') || (symbol=='%'));  
 }

@Override  
 protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 String header=req.getParameter(Note.*HEADER*);  
 String author=req.getParameter(Book.*AUTHOR*);  
 try {  
 MongoWorkNote mongo = new MongoWorkNote();  
 Note note = mongo.get(header);  
  
 String string = new Gson().toJson(note);  
 PrintWriter printWriter = resp.getWriter();  
 printWriter.write(string);  
 }catch (NoHeaderEcxeption | NoSuchElementInDBException ex) {  
 resp.sendError(HttpServletResponse.*SC\_BAD\_REQUEST*, ex.getMessage());  
 }  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
  
 String string=getBody(req);  
 List<String> list= parseBodyToList(string);   
 PrintWriter printWriter=resp.getWriter();  
 try {  
 String header = getFromBodyListByName(list, Note.*HEADER*);  
 String inf = getFromBodyListByName(list, Note.*INF*);  
 Note note = new Note(header, inf);  
  
 MongoWorkNote mongo = new MongoWorkNote();  
 mongo.add(note);  
 String j = new Gson().toJson(note);  
 resp.setStatus(HttpServletResponse.*SC\_OK*);  
 printWriter.write("added: "+j);  
 }catch (ParameterException | NoHeaderEcxeption | AlreadyHasHeaderException ex){  
 resp.sendError(HttpServletResponse.*SC\_BAD\_REQUEST*, ex.getMessage());  
 }  
 }  
  
 @Override  
 protected void doPut(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 String string=getBody(req);  
 List<String> list= parseBodyToList(string);   
 PrintWriter printWriter=resp.getWriter();  
 try {  
 String header = getFromBodyListByName(list, Note.*HEADER*);  
 String inf = getFromBodyListByName(list, Note.*INF*);  
 Note note=new Note(header, inf);  
  
 MongoWorkNote mongo=new MongoWorkNote();  
 mongo.update(note);  
  
 String j = new Gson().toJson(note);  
 resp.setStatus(HttpServletResponse.*SC\_OK*);  
 printWriter.write("updated: " +j);  
 }catch (ParameterException | NoSuchElementInDBException | NoHeaderEcxeption ex) {  
 resp.sendError(HttpServletResponse.*SC\_BAD\_REQUEST*, ex.getMessage());  
 }  
  
 }  
  
 @Override  
 protected void doDelete(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 String string=getBody(req);  
 List<String> list= parseBodyToList(string);   
 PrintWriter printWriter=resp.getWriter();  
 try{  
 String header = getFromBodyListByName(list, Note.*HEADER*);  
 Note note=new Note(header, Note.*INF*);  
  
 MongoWorkNote mongo=new MongoWorkNote();  
 mongo.delete(note);  
  
 String j = new Gson().toJson(note);  
 resp.setStatus(HttpServletResponse.*SC\_OK*);  
 printWriter.write("deleted: " +j);  
 } catch (ParameterException | NoSuchElementInDBException | NoHeaderEcxeption ex) {  
 resp.sendError(HttpServletResponse.*SC\_BAD\_REQUEST*, ex.getMessage());  
 }  
 }  
}

Класс Note:

package objects;  
  
public class Note {  
  
 public final static String *HEADER*="header";  
 public final static String *INF*="inf";  
 private String header;  
 private String inf;  
  
 public Note(String header, String inf) {  
 this.header = header;  
 this.inf = inf;  
 }  
  
 public String getHeader() {  
 return header;  
 }  
  
 public void setHeader(String header) {  
 this.header = header;  
 }  
  
 public String getInf() {  
 return inf;  
 }  
  
 public void setInf(String inf) {  
 this.inf = inf;  
 }  
  
}

Класс MongoWorkNote:

package mongo;  
  
import com.google.gson.Gson;  
import com.mongodb.BasicDBObject;  
import com.mongodb.DB;  
import com.mongodb.DBCollection;  
import com.mongodb.DBObject;  
import com.mongodb.Mongo;  
import com.mongodb.util.JSON;  
import inputExceptionsMongo.\*;  
import objects.Note;  
  
import java.net.UnknownHostException;  
import java.util.List;  
import java.util.Map;  
  
  
public class MongoWorkNote {  
  
 private Mongo mongo;  
 private DB db;  
 private DBCollection collection;  
  
 public MongoWorkNote() throws UnknownHostException {  
 mongo=new Mongo("localhost", 27017);  
 db= mongo.getDB("server");  
 collection=db.getCollection("articles");  
 }  
  
 public String getByHeader(String searchHeader) throws NoHeaderEcxeption, NoSuchElementInDBException {  
 if (searchHeader == null || searchHeader.isEmpty()) {  
 throw new NoHeaderEcxeption();  
 }  
 String result = null;  
 BasicDBObject query = new BasicDBObject();  
  
 query.put(Note.*HEADER*, searchHeader);  
 DBObject findElement = collection.findOne(query);  
 if (findElement == null) {  
 throw new NoSuchElementInDBException();  
 }  
 result = String.*valueOf*(findElement.get(Note.*INF*));  
  
 return result;  
 }

public void add(Note note) throws AlreadyHasHeaderException, NoHeaderEcxeption {  
  
 try{  
 String string=getByHeader(note.getHeader());  
 if(string!=null) {  
 throw new AlreadyHasHeaderException();  
 }  
 }catch (NoSuchElementInDBException e) {  
 }  
 String jSONstring=new Gson().toJson(note);  
 BasicDBObject dbObject= (BasicDBObject) JSON.*parse*(jSONstring);  
 collection.insert(dbObject);  
 }  
  
 public Note get(String searchHeader) throws NoHeaderEcxeption, NoSuchElementInDBException {  
  
 return new Note(searchHeader, getByHeader(searchHeader));  
 }  
  
 public void update(Note note) throws NoHeaderEcxeption, NoSuchElementInDBException {  
  
 String jSONstring=new Gson().toJson(note);  
 BasicDBObject newDBObject=(BasicDBObject) JSON.*parse*(jSONstring);  
 BasicDBObject oldDBObject=new BasicDBObject().append(Note.*INF*, getByHeader(note.getHeader()));  
 collection.update(oldDBObject, newDBObject);  
 }  
  
 public void delete(Note note) throws NoHeaderEcxeption, NoSuchElementInDBException {  
  
 note.setInf(getByHeader(note.getHeader()));  
 String jSONstring=new Gson().toJson(note);  
 BasicDBObject query=(BasicDBObject) JSON.*parse*(jSONstring);  
 collection.remove(query);  
 }  
}