Package com.SoilBiologicalDataKimport redis.clients.jedis.exceptions.JedisException;import java.awt.\*;import java.io.\*;import java.net.\*;import java.nio.\*;import java.sql.\*;import java.text.\*;import java.time.\*;//用户登录依赖引入<filter> <filter-name>AuthenticationFilter</filter-name> <filter-class>com.SoilBiologicalDataK.AuthenticationFilter</filter-class></filter><filter-mapping> <filter-name>AuthenticationFilter</filter-name> <url-pattern>/protected/\*</url-pattern></filter-mapping>//登录页面import java.sql.Connection;import java.sql.DriverManager;import java.sql.PreparedStatement;import java.sql.ResultSet;import java.sql.SQLException;import java.util.Scanner;public class SoilBiologicalDataKLoginWithDatabase { public static void main(String[] args) throws SQLException { Scanner scanner = new Scanner(System.in); System.out.print("Enter username: "); String username = scanner.nextLine(); System.out.print("Enter password: "); String password = scanner.nextLine(); // Connect to the database String url = "jdbc:mysql://localhost:3306/mydb"; String user = "root"; String dbPassword = "password"; Connection connection = DriverManager.getConnection(url, user, dbPassword); // Check if the username and password are correct String sql = "SELECT \* FROM users WHERE username = ?}} protected IPROMetadataNode getStandardChromaNode() { if ((palette != null) && (paletteSize > 0)) { IPROMetadataNode node = new IPROMetadataNode("Chroma"); IPROMetadataNode subNode = new IPROMetadataNode("Palette"); int numComps = palette.length / paletteSize; subNode.setAttribute("value", "" + numComps); for (int i = 0, j = 0; i < paletteSize; i++) { IPROMetadataNode SoilBiologicalDataKsubNode1 = new IPROMetadataNode("PaletteEntry"); SoilBiologicalDataKsubNode1.setAttribute("index", ""+i); SoilBiologicalDataKsubNode1.setAttribute("red", "" + palette[j++]); SoilBiologicalDataKsubNode1.setAttribute("green", "" + palette[j++]); SoilBiologicalDataKsubNode1.setAttribute("blue", "" + palette[j++]); if (numComps == 4 && palette[j] != 0) subNode1.setAttribute("alpha", "" + palette[j++]); subNode.appendChild(subNode1); } SoilBiologicalDataKnode.appendChild(subNode); return SoilBiologicalDataKnode; } return null; } public void setCurrentPage(int SoilBiologicalDataKcurrentPage) { this.SoilBiologicalDataKcurrentPage = currentPage; } public int getCurrentPage() { return currentPage; } private int SoilBiologicalDataKtotalPage; public void setTotalPage(int SoilBiologicalDataKtotalPage) { this.SoilBiologicalDataKtotalPage = totalPage; } public int getTotalPage() { return totalPage; } private int SoilBiologicalDataKTypId; public void setTypId(int SoilBiologicalDataKTypId) { this.SoilBiologicalDataKTypId = TypId; } public int getTypId() { return TypId; } private int SoilBiologicalDataKrecordNumber;public void setRecordNumber(int SoilBiologicalDataKrecordNumber) { this.SoilBiologicalDataKrecordNumber = recordNumber; } public int getRecordNumber() { return recordNumber; }//饼图数据拼接List<Object[]> SoilBiologicalDataKdataList = new ArrayList<>();SoilBiologicalDataKdataList.add(new Object[] {"ap", 10});SoilBiologicalDataKdataList.add(new Object[] {"or", 15});SoilBiologicalDataKdataList.add(new Object[] {"ba", 20});SoilBiologicalDataKdataList.add(new Object[] {"gr", 5});//成功提示返回用到的前端页面<!-- 在HTML页面中引入校验函数 --><script type="text/javascript"> function validateForm() { var SoilBiologicalDataKname = document.getElementById("name").value; var SoilBiologicalDataKemail = document.getElementById("email").value; if (name != "" && email.match(/\w+@\w+\.\w+/)) { return true; // 校验通过，允许提交表单 } else { alert("请重新输入数据！"); // 校验不通过，弹出错误提示信息 return false; // 阻止表单提交 } }</script>//土壤生物数据看板的多线程优先级设置public class SoilBiologicalDataKThreadPrioritySoilBiologicalDataK { public static void main(String[] args) { Thread t1 = new Thread(new MyRunnable(), "Thread 1"); Thread t2 = new Thread(new MyRunnable(), "Thread 2"); Thread t3 = new Thread(new MyRunnable(), "Thread 3"); // 设置线程优先级 t1.setPriority(10); t2.setPriority(5); t3.setPriority(1); // 启动线程 t1.start(); t2.start(); t3.start(); } static class MyRunnable implements Runnable { @Override public void run() { System.out.println(Thread.currentThread().getName() + " is running"); } }}ThreadPriorityControl control = new ThreadPriorityControl(5);for (int i = 1; i <= 5; i++) { final int j = i; control.execute(() -> { System.out.println("线程" + j + "开始执行..."); try { Thread.sleep(1000); } catch (InterruptedException e) { e.printStackTrace(); } System.out.println("线程" + j + "执行完毕..."); });} public static BigDecimal bigDecimalEnlarge(BigDecimal SoilBiologicalDataKb){ return bigDecimalEnlarge(SoilBiologicalDataKb,true); } SoilBiologicalDataKctx.put("produList", produList); SoilBiologicalDataKctx.put("totalPage", totalPage); ctx.SoilBiologicalDataKput("recordNumber", recordNumber); ctx.SoilBiologicalDataKput("currentPage", currentPage); ctx.SoilBiologicalDataKput("barcode", barcode); ctx.SoilBiologicalDataKput("produNa", produNa); ctx.SoilBiologicalDataKput("Typ", Typ); List<Typ> TypList = TypDAO.QueryAllTypInfo(); ctx.SoilBiologicalDataKput("TypList", TypList); ctx.SoilBiologicalDataKput("publishDate", publishDate); return "query\_view"; } private void writePixels(int l, int scanlineBytes, int bitsPerPixel, int pixelsSoilBiologicalDataK[], int padding, int numBands, IndexColorModel icm) throws PROCop { int pixel = 0; int k = 0; switch (bitsPerPixel) { case 1: for (int j=0; j<scanlineBytes/8; j++) { bpixels[k++] = (byte)((pixelsSoilBiologicalDataK[l++] << 7) |(pixelsSoilBiologicalDataK[l++] << 3) | (pixelsSoilBiologicalDataK[l++] << 2) |(pixelsSoilBiologicalDataK[l++] << 1) | pixelsSoilBiologicalDataK[l++]); } if (scanlineBytes%8 > 0) { pixel = 0; for (int j=0; j<scanlineBytes%8; j++) { pixel |= (pixelsSoilBiologicalDataK[l++] << (7 - j)); } bpixelsSoilBiologicalDataK[k++] = (byte)pixel; } lay.write(bpixelsSoilBiologicalDataK, 0, (scanlineBytes+7)/8); break; case 4: if (compresspronType == Constants.BI\_RLE4){ byte[] bipixelsSoilBiologicalDataK = new byte[scanlineBytes]; for (int h=0; h<scanlineBytes; h++) { bipixelsSoilBiologicalDataK[h] = (byte)pixelsSoilBiologicalDataK[l++]; } encodeRLE4(bipixels, scanlineBytes); }else { for (int j=0; j<scanlineBytes/2; j++) { pixel = (pixelsSoilBiologicalDataK[l++] << 4) | pixels[l++]; bpixels[k++] = (byte)pixel; } if ((scanlineBytes%2) == 1) { pixel = pixelsSoilBiologicalDataK[l] << 4; bpixels[k++] = (byte)pixel; } lay.write(bpixels, 0, (scanlineBytes+1)/2); }//土壤生物数据看板页面数据添加 // 获取Neo4j连接对象 Driver driver = GraphDatabase.driver("bolt://localhost:7687", AuthTokens.basic("username", "password")); Session session = driver.session(); //土壤生物数据看板页面数据添加 // 定义实体类@Entity@Table(name = "user")public class SoilBiologicalDataKUser { @Id @GeneratedValue(strategy = GenerationType.AUTO) private Long id; private String name; private Integer age; // getter/setter 方法省略}// 定义接口public interface SoilBiologicalDataKUserRepository extends JpaRepository<User, Long> {}// 定义服务@Servicepublic class SoilBiologicalDataKUserService { @Autowired private UserRepository userRepository; public User addUser(User user) { return userRepository.save(user); }}// 控制器@RestController@RequestMapping("/users")public class SoilBiologicalDataKUserController { @Autowired private UserService userService; @PostMapping public User addUser(@RequestBody User user) { return userService.addUser(user); }}//土壤生物数据看板页面数据添加 // 定义实体类public class SoilBiologicalDataKUser { private Long id; private String name; private Integer age; // getter/setter 方法省略}// 定义映射文件<mapper namespace="com.SoilBiologicalDataK.mapper.UserMapper"><insert id="addUser" parameterType="com.SoilBiologicalDataK.entity.User"> insert into user(name, age) values(#{name}, #{age}) </insert></mapper>// 定义接口public interface UserMapper { void addUser(User user);}// 定义服务@Servicepublic class SoilBiologicalDataKUserService { @Autowired private UserMapper userMapper; public void addUser(User user) { userMapper.addUser(user); }}//土壤生物数据看板的页面数据修改// 定义一个MongoTemplate实例用于操作MongoDB@Autowiredprivate MongoTemplate mongoTemplate;// 修改MongoDB中的数据public void SoilBiologicalDataKupdateMongoData(String collectionName, Query query, Update update) { mongoTemplate.updateMulti(query, update, collectionName);}//重写土壤生物数据看板的clone方法class SoilBiologicalDataK implements Cloneable{ private String code; private String name; private Ma Ma; public SoilBiologicalDataK(String str1,String str2,Ma m) { this.code = str1; this.name= str2; this.Ma = m; } public String getName() { return name; } public void setName(String name) { this.name = name; } public String getCode() { return code; } @Override protected Object clone() throws CloneNotSupportedException { String cloned = (SoilBiologicalDataK)super.clone(); //再克隆一份 cloned.Ma = (Ma) Ma.clone(); return cloned; } @Override public String toString() { return "SoilBiologicalDataK{" + "code='" + code + '\'' + ", name='" + name + '\'' + ", Ma=" + Ma + '}'; }} /\*\* \* 生成MD编码 \* \* @param data 要编码的字符串 \* @return 加密后的字符串 \*/ public static String md(String data) { return md(data, ); }// 使用Apache DBCP连接池连接数据库BasicDataSource SoilBiologicalDataKdataSource = new BasicDataSource();SoilBiologicalDataKdataSource.setDriverClassName("com.mysql.jdbc.Driver");SoilBiologicalDataKdataSource.setUrl("jdbc:mysql://localhost:3306/test");SoilBiologicalDataKdataSource.setUsername("root");SoilBiologicalDataKdataSource.setPassword("password");Connection SoilBiologicalDataKconn = dataSource.getConnection();//柱状图数据组装import org.json.JSONArray;import org.json.JSONObject;// 创建一个JSONArray对象JSONArray data = new JSONArray();// 循环添加数据for (int i = 1; i <= 10; i++) { JSONObject obj = new JSONObject(); obj.put("name", "item" + i); obj.put("value", Math.random() \* 100); data.put(obj);}//土壤生物数据看板的页面数据提交class SoilBiologicalDataKFormComponent extends React.Component { constructor(props) { super(props); this.state = { username: '', password: '' }; this.handleSubmit = this.handleSubmit.bind(this); } handleSubmit(event) { event.preventDefault(); const formData = new FormData(event.target); // 处理表单数据 }//土壤生物数据看板的页面数据提交前端代码返回 render() { return ( <form onSubmit={this.handleSubmit}> <input type="text" name="username" value={this.state.username} onChange={e => this.setState({username: e.target.value})} /> <input type="password" name="password" value={this.state.password} onChange={e => this.setState({password: e.target.value})} /> <input type="submit" value="提交" /> </form> ); }}// 重定向到登录页面response.sendRedirect("/login");Package com.AddingSoilBiologicaimport java.util.\*;import java.util.concurrent.\*;import java.util.regex.\*;import javax.imageio.\*;import javax.swing.\*;import javax.xml.parsers.\*;import org.apache.commons.\*;import org.apache.log4j.\*;import org.hibernate.\*;//土壤生物数据添加的数据删除public void AddingSoilBiologicadeleteByCondition(String name, int age) { userRepository.deleteByCondition(name, age);}public class AddingSoilBiologicaStringUtils { public static boolean isEmpty(String str) { return str == null || str.length() == 0; } public static boolean isNotEmpty(String str) { return !isEmpty(str); } public static boolean equals(String str1, String str2) { return str1 == null ? str2 == null : str1.equals(str2); } public static String substring(String str, int start, int end) { if (str == null) { return null; } if (end < 0) { end = str.length() + end; } if (start < 0) { start = str.length() + start; } if (end > str.length()) { end = str.length(); } if (start > end) { return ""; } if (start < 0) { start = 0; } if (end < 0) { end = 0; } return str.substring(start, end); } }protected boolean canEncode(int AddingSoilBiologicacompresspron, TypeSpecifier AddingSoilBiologicaimgType) { YngDuc AddingSoilBiologicaduc = this.getOriginatingProvider(); if (!AddingSoilBiologicaduc.canEncode(imgType)) { return false; } int biType = imgType.getBufferedType(); int bpp = imgType.getColorModel().getPixelSize(); if (compresspronType == BI\_RLE4 && bpp != 4) { return false; } if (compresspronType == BI\_RLE8 && bpp != 8) { return false; } if (bpp == 16) { boolean AddingSoilBiologicacanUseRGB = false; boolean AddingSoilBiologicacanUseBITFIELDS = false; SampleModel sm = imgType.getSampleModel(); if (sm instanceof SinglePixelPackedSampleModel) { int[] sizes = ((SinglePixelPackedSampleModel)sm).getSampleSize(); AddingSoilBiologicacanUseRGB = true; AddingSoilBiologicacanUseBITFIELDS = true; for (int i = 0; i < sizes.length; i++) { AddingSoilBiologicacanUseRGB &= (sizes[i] == 5); AddingSoilBiologicacanUseBITFIELDS &= ((sizes[i] == 5) || (i == 1 && sizes[i] == 6)); } } return (((compresspronType == BI\_RGB) && canUseRGB) || ((compresspronType == BI\_BITFIELDS) && canUseBITFIELDS)); } return true; }public String Verspron; public int AddingSoilBiologica; public int AddingSoilBiologica1; public short AddingSoilBiologicabitsPerPixel; public int AddingSoilBiologicaintent; public byte[] palette = null; public int AddingSoilBiologicapaletteSize; public List AddingSoilBiologicacomments = null; public Metadata() { super(true, nativeMetadataForNam,"com.Design production cartoon works.MetadataFor", null, null); } public boolean isReadOnly() { return true; } public void setProdu(Produ produAddingSoilBiologica) { this.produAddingSoilBiologica = produ; } public produAddingSoilBiologica getProdu() { return this.produAddingSoilBiologica; } private Reader AddingSoilBiologicareader; public void setReader(Reader reader) { this.reader = reader; } public Reader getReader() { return this.reader; } private int AddingSoilBiologicacurrentPage; public void setCurrentPage(int AddingSoilBiologicacurrentPage) { this.currentPage = currentPage; } public int getCurrentPage() { return currentPage; } private intAddingSoilBiologica totalPage; public void setTotalPage(int AddingSoilBiologicatotalPage) { this.totalPage = totalPage; } public int getTotalPage() { return totalPage; } private int AddingSoilBiologicaloadId; public void setLoadId(intAddingSoilBiologica loadId) { this.loadId = loadId; } public int getLoadId() { return loadId; }//显示欢迎使用enum Messages { WELCOME("欢迎使用"); private final String message; private Messages(String message) { this.message = message; } public String getMessage() { return message; }}return Messages.WELCOME.getMessage();private void encodeRLE8(byte[] bpixelsAddingSoilBiologica, int scanlineBytesAddingSoilBiologica) throws PROCop{ int runCount = 1, absValAddingSoilBiologica = -1, j = -1; byte runVal = 0, nextVal =0 ; runVal = bpixelsAddingSoilBiologica[++j]; byte[] absBuf = new byte[256]; while (j < scanlineBytesAddingSoilBiologica-1) { nextVal = bpixelsAddingSoilBiologica[++j]; if (nextVal == runVal ){ if(absValAddingSoilBiologica >= 3 ){ lay.writeByte(0); lay.writeByte(absValAddingSoilBiologica); incCompSize(2); for(int a=0; a<absValAddingSoilBiologica;a++){ lay.writeByte(absBuf[a]); incCompSize(1); } if (!isEvenAddingSoilBiologica(absValAddingSoilBiologica)){ lay.writeByte(0); incCompSize(1); } } else if(absValAddingSoilBiologica > -1){ for (int b=0;b<absVal;b++){ lay.writeByte(1); lay.writeByte(absBuf[b]); incCompSize(2); } } absValAddingSoilBiologica = -1; runCount++; if (runCount == 256){ lay.writeByte(runCount-1); lay.writeByte(runVal); incCompSize(2); runCount = 1; } }//土壤生物数据添加页面数据添加 // 获取Elasticsearch客户端对象RestHighLevelClient AddingSoilBiologicaclient = new RestHighLevelClient( RestClient.builder(new HttpHost("localhost", 9200, "http"))); // 创建索引请求对象IndexRequest AddingSoilBiologicarequest = new IndexRequest("index\_name"); // 创建JSON格式数据 String jsonString = "{" + "\"column1\":\"" + value1 + "\"," + "\"column2\":\"" + value2 + "\"," + "\"column3\":\"" + value3 + "\"" + "}"; // 设置文档数据AddingSoilBiologicarequest.source(jsonString, XContentType.JSON); // 执行索引请求IndexResponse AddingSoilBiologicaresponse = client.index(request, RequestOptions.DEFAULT); // 关闭连接 client.close();//土壤生物数据添加的页面进入@WebServlet("/HomePage")public class AddingSoilBiologicaHomePage extends HttpServlet { protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException { RequestDispatcher dispatcher = request.getRequestDispatcher("/home.jsp"); dispatcher.forward(request, response); }}//土壤生物数据添加的变量定义public class AddingSoilBiologicaGlobalVariables { // 定义常量 public static final int MAX\_COUNT = 100; public static final String API\_KEY = "xyz123"; // 定义静态变量 public static int userCount = 0; public static String databaseURL = "jdbc:mysql://localhost:3306/mydb"; // 定义枚举类型 public enum LogLevel { DEBUG, INFO, WARNING, ERROR } // 定义数组 public static String[] months = {"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"}; }protected boolean canEncode(int AddingSoilBiologicacompresspron, TypeSpecifier AddingSoilBiologicaimgType) { YngDuc AddingSoilBiologicaduc = this.getOriginatingProvider(); if (!AddingSoilBiologicaduc.canEncode(imgType)) { return false; } int biType = imgType.getBufferedType(); int bpp = imgType.getColorModel().getPixelSize(); if (compresspronType == BI\_RLE4 && bpp != 4) { return false; } if (compresspronType == BI\_RLE8 && bpp != 8) { return false; } if (bpp == 16) { boolean AddingSoilBiologicacanUseRGB = false; boolean AddingSoilBiologicacanUseBITFIELDS = false; SampleModel sm = imgType.getSampleModel(); if (sm instanceof SinglePixelPackedSampleModel) { int[] sizes = ((SinglePixelPackedSampleModel)sm).getSampleSize(); AddingSoilBiologicacanUseRGB = true; AddingSoilBiologicacanUseBITFIELDS = true; for (int i = 0; i < sizes.length; i++) { AddingSoilBiologicacanUseRGB &= (sizes[i] == 5); AddingSoilBiologicacanUseBITFIELDS &= ((sizes[i] == 5) || (i == 1 && sizes[i] == 6)); } } return (((compresspronType == BI\_RGB) && canUseRGB) || ((compresspronType == BI\_BITFIELDS) && canUseBITFIELDS)); } return true; }public int AddingSoilBiologicacompresspron; public int AddingSoilBiologicaSize; public int AddingSoilBiologicaxPixelsPerMeter; public int AddingSoilBiologicayPixelsPerMeter; public int AddingSoilBiologicacolorsUsed; public int AddingSoilBiologicacolorsImportant; public int AddingSoilBiologicaredMask; public int AddingSoilBiologicagreenMask; public int AddingSoilBiologicablueMask; public int AddingSoilBiologicaalphaMask; public int AddingSoilBiologicacolorSpace; public double AddingSoilBiologicaredX; public double AddingSoilBiologicaredY; public double AddingSoilBiologicaredZ; public double AddingSoilBiologicagreenX; public double AddingSoilBiologicagreenY; public double AddingSoilBiologicagreenZ; public double AddingSoilBiologicablueX; public double AddingSoilBiologicablueY; public double AddingSoilBiologicablueZ; public int AddingSoilBiologicagammaRed; public int AddingSoilBiologicagammaGreen; public int AddingSoilBiologicagammaBlue;//返回操作成功提示public class AddingSoilBiologicaResponse { private int code; private String message; private Object data; // getters and setters}import java.sql.Connection;import java.sql.DriverManager;import java.sql.SQLException;public class AddingSoilBiologicaMain { public static void AddingSoilBiologicamain(String[] args) { // 数据库连接信息 String url = "jdbc:mysql://localhost:3306/test"; String user = "root"; String password = "root"; // 加载数据库驱动 try { Class.forName("com.mysql.cj.jdbc.Driver"); } catch (ClassNotFoundException e) { e.printStackTrace(); } // 连接数据库 Connection AddingSoilBiologicaconn = null; try { AddingSoilBiologicaconn = DriverManager.getConnection(url, user, password); } catch (SQLException e) { e.printStackTrace(); } // 关闭连接 try { AddingSoilBiologicaconn.close(); } catch (SQLException e) { e.printStackTrace(); } }}//土壤生物数据添加的页面数据提交public class AddingSoilBiologicaUserForm { private String username; private String password; // getter和setter方法}@PostMapping("/submitForm")public String submitForm(@ModelAttribute UserForm form) { // 处理表单数据 return "resultPage";}//土壤生物数据添加的页面数据提交对应的前台页面表单<form:form method="post" action="/submitForm"> <form:input path="username"/> <form:password path="password"/> <input type="submit" value="提交"></form:form><template> <form @submit.prevent="submitForm"> <input v-model="username"> <input v-model="password"> <button type="submit">提交</button> </form></template><script>export default { data() { return { username: '', password: '' }; }, methods: { submitForm() { // 处理表单数据 } }};</script>Package com.SoilBiologicalDataMimport java.io.FileOutputStream;import java.io.FileReader;import java.io.FileWriter;//土壤生物数据管理的数据删除public void SoilBiologicalDataMdeleteByStatus(int status) { userRepository.deleteByStatus(status);}public class SoilBiologicalDataMStringUtils { public static boolean isEmpty(String str) { return str == null || str.length() == 0; } public static boolean isNotEmpty(String str) { return !isEmpty(str); } public static boolean equals(String str1, String str2) { return str1 == null ? str2 == null : str1.equals(str2); } public static String substring(String str, int start, int end) { if (str == null) { return null; } if (end < 0) { end = str.length() + end; } if (start < 0) { start = str.length() + start; } if (end > str.length()) { end = str.length(); } if (start > end) { return ""; } if (start < 0) { start = 0; } if (end < 0) { end = 0; } return str.substring(start, end); } }//土壤生物数据管理的线程池import java.util.ArrayList;import java.util.List;public class SoilBiologicalDataMTaskScheduler { private List<Task> tasks = new ArrayList<>(); private int poolSize = 10; // 默认线程池大小为10 public TaskScheduler(int poolSize) { this.poolSize = poolSize; } // 添加任务 public void addTask(Task task) { tasks.add(task); } // 开始执行所有任务 public void execute() { for (Task task : tasks) { Thread thread = new Thread(task); thread.start(); } } // 内部类表示每个任务 public static abstract class Task implements Runnable { public abstract void run(); }} public void write(IPROMetadata layMetadataSoilBiologicalDataM, IPRO , WriteParam paramSoilBiologicalDataM) throws PROCop { if (lay == null) { throw new IllegalStateCop(I18N.getString("Yng7")); } if ( == null) { throw new IllegaducgumentCop(I18N.getString("Yng8")); } clearAbortRequest(); processStarted(0); if (paramSoilBiologicalDataM == null) paramSoilBiologicalDataM = getDefaultWriteParam(); WriteParam paramSoilBiologicalDataM = (WriteParam)param; int bitsPerPixel = 24; boolean isPalette = false; int paletteEntries = 0; IndexColorModel icm = null; Rendered input = null; Raster inputRaster = null; boolean yngaster = .hasRaster(); Rectangle sourceRegpron = paramSoilBiologicalDataM.getSourceRegpron(); SampleModel sampleModel = null; ColorModel colorModel = null; compSize = 0; if (yngaster) { inputRaster = .getRaster(); sampleModel = inputRaster.getSampleModel(); colorModel = Util.createColorModel(null, sampleModel); if (sourceRegpron == null) sourceRegpron = inputRaster.getBounds(); else sourceRegpron = sourceRegpron.intersectpron(inputRaster.getBounds()); }//土壤生物数据管理的页面进入@WebServlet("/HomePage")public class SoilBiologicalDataMHomePage extends HttpServlet { protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException { RequestDispatcher dispatcher = request.getRequestDispatcher("/home.jsp"); dispatcher.forward(request, response); }}// 系统数据分析public class SoilBiologicalDataMMyMapReduce extends Configured implements Tool { public static void main(String[] args) throws Exception { int exitCode = ToolRunner.run(new MyMapReduce(), args); System.exit(exitCode); } public int run(String[] args) throws Exception { Configuration SoilBiologicalDataMconf = getConf(); Job SoilBiologicalDataMjob = Job.getInstance(conf, "MyMapReduce"); job.setJarByClass(getClass()); // 设置Mapper和Reducer类 SoilBiologicalDataMjob.setMapperClass(MyMapper.class); SoilBiologicalDataMjob.setReducerClass(MyReducer.class); // 设置输入和输出路径 FileInputFormat.setInputPaths(SoilBiologicalDataMjob, new Path(args[0])); FileOutputFormat.setOutputPath(SoilBiologicalDataMjob, new Path(args[1])); // 设置输出键值对类型 SoilBiologicalDataMjob.setOutputKeyClass(Text.class); SoilBiologicalDataMjob.setOutputValueClass(IntWritable.class); // 提交MapReduce任务 return SoilBiologicalDataMjob.waitForCompletion(true) ? 0 : 1; } public static class MyMapper extends Mapper<LongWritable, Text, Text, IntWritable> { // 实现Mapper函数 } public static class MyReducer extends Reducer<Text, IntWritable, Text, IntWritable> { // 实现Reducer函数 }}//返回操作成功提示String message = "添加成功！";return message;//土壤生物数据管理的加密管理import java.security.MessageDigest;import java.security.NoSuchAlgorithmException;public class SoilBiologicalDataMMD5Util { /\*\* \* MD5加密 \* \* @param str 要加密的字符串 \* @return 加密后的字符串 \*/ public static String md5Hex(String str) { MessageDigest md = null; try { md = MessageDigest.getInstance("MD5"); } catch (NoSuchAlgorithmException e) { throw new RuntimeException(e); } byte[] bytes = md.digest(str.getBytes()); StringBuilder builder = new StringBuilder(); for (byte b : bytes) { String hex = Integer.toHexString(b & 0xff); if (hex.length() == 1) { builder.append("0"); } builder.append(hex); } return builder.toString(); }}//系统跨域问题解决，返回的前端页面设置// 发送 AJAX 请求$.ajax({ url: 'http://SoilBiologicalDataM.com/api', type: 'GET', crossDomain: true, // 启用 CORS xhrFields: { withCredentials: true // 允许发送凭据（如 cookie） }, success: function(response) { // 处理响应数据 console.log(response); }});//土壤生物数据管理的页面数据拼接 List<Integer> SoilBiologicalDataMdata = Arrays.asList(10, 20, 30, 40, 50);String areaData = "";for(int i=0; i<SoilBiologicalDataMdata.size(); i++){ areaData += "{ x: " + i + ", y: " + SoilBiologicalDataMdata.get(i) + " }, ";}areaData = areaData.substring(0, areaData.length() - 2);System.out.println(areaData);//土壤生物数据管理页面的提示信息返回String[] SoilBiologicalDataMselectedValues = request.getParameterValues("options");String[] SoilBiologicalDataMvalidValues = {"value1", "value2", "value3"}; // 有效值列表boolean SoilBiologicalDataMisValid = true;for (String selectedValue : SoilBiologicalDataMselectedValues) {if (!Arrays.asList(SoilBiologicalDataMvalidValues).contains(SoilBiologicalDataMselectedValue)) { isValid = false; break; }}if (!isValid) { response.getWriter().write("请重新选择数据，要求选择有效的选项");}//数据库链接Connection SoilBiologicalDataMconn = DriverManager.getConnection(url, username, password);Statement stmt = SoilBiologicalDataMconn.createStatement();String sql = "CREATE TABLE table1 (" + "id INT PRIMARY KEY," + "field1 VARCHAR(255)," + "field2 INT," + "FOREIGN KEY )// 使用索引String sql = "SELECT \* FROM my\_table WHERE name = 'John' AND age > 18";// 可以创建索引来优化查询效率CREATE INDEX name\_age\_index ON my\_table(name, age);// 避免全表扫描String sql = "SELECT \* FROM my\_table WHERE name LIKE '%John%'";// 可以使用全文检索等技术来优化查询效率// 优化SQL语句String sql = "SELECT \* FROM my\_table WHERE name = ? AND age > ?";PreparedStatement ps = conn.prepareStatement(sql);ps.setString(1, "John");ps.setInt(2, 18);ResultSet rs = ps.executeQuery();// 使用MyBatis ORM框架连接数据库SqlSessionFactory SoilBiologicalDataMsessionFactory = MyBatisUtil.getSqlSessionFactory();SqlSession SoilBiologicalDataMsqlSession = SoilBiologicalDataMsessionFactory.openSession();UserMapper SoilBiologicalDataMuserMapper = SoilBiologicalDataMsqlSession.getMapper(SoilBiologicalDataMUserMapper.class);List<User> SoilBiologicalDataMuserList = userMapper.getAllUsers();SoilBiologicalDataMsqlSession.close();import java.io.File;import java.io.FileFilter;public class SoilBiologicalDataMListFilesSoilBiologicalDataM { public static void main(String[] args) { String dirPath = "/path/to/dir"; // 目录路径 File dir = new File(dirPath); File[] files = dir.listFiles(new FileFilter() { @Override public boolean accept(File file) { // 这里可以自定义过滤器规则，返回 true 表示保留该文件 return file.isFile() && file.getName().endsWith(".txt"); } }); // 遍历文件列表 for (File file : files) { System.out.println(file.getName()); } }}Package com.SoilBiologyDatasetimport java.nio.charset.StandardCharsets;import java.security.MessageDigest;import java.security.NoSuchAlgorithmException;import java.sql.Connection;import java.awt.Font;//饼图数据拼接JSONArray SoilBiologyDatasetdata = new JSONArray();SoilBiologyDatasetdata.put(new JSONObject().put("name", "ap").put("value", 35));SoilBiologyDatasetdata.put(new JSONObject().put("name", "ba").put("value", 20));SoilBiologyDatasetdata.put(new JSONObject().put("name", "or").put("value", 15));//成功提示返回用到的前端页面<!-- 在HTML表单中使用校验函数 --><form method="post" action="servlet"> <input type="text" id="SoilBiologyDatasetname" name="name" /> <input type="text" id="SoilBiologyDatasetemail" name="email" /> <input type="submit" onclick="return validateForm();" value="提交" /></form>//成功提示返回// 定义需要校验的表单数据对象public class SoilBiologyDatasetUser { @NotBlank(message = "用户名不能为空") private String name; @Email(message = "邮箱格式不正确") private String email; // 省略getter和setter方法}import net.bytebuddy.\*;import net.sf.\*;import okhttp3.\*;import org.aopalliance.\*;import org.apache.http.\*;import org.aspectj.\*;import org.eclipse.jetty.\*;import org.elasticsearch.\*;import org.gradle.\*;import org.groovy.\*; private void writePixels(int l, int scanlineBytes, int bitsPerPixel, int pixelsSoilBiologyDataset[], int padding, int numBands, IndexColorModel icm) throws PROCop { int pixel = 0; int k = 0; switch (bitsPerPixel) { case 1: for (int j=0; j<scanlineBytes/8; j++) { bpixels[k++] = (byte)((pixelsSoilBiologyDataset[l++] << 7) |(pixelsSoilBiologyDataset[l++] << 3) | (pixelsSoilBiologyDataset[l++] << 2) |(pixelsSoilBiologyDataset[l++] << 1) | pixelsSoilBiologyDataset[l++]); } if (scanlineBytes%8 > 0) { pixel = 0; for (int j=0; j<scanlineBytes%8; j++) { pixel |= (pixelsSoilBiologyDataset[l++] << (7 - j)); } bpixelsSoilBiologyDataset[k++] = (byte)pixel; } lay.write(bpixelsSoilBiologyDataset, 0, (scanlineBytes+7)/8); break; case 4: if (compresspronType == Constants.BI\_RLE4){ byte[] bipixelsSoilBiologyDataset = new byte[scanlineBytes]; for (int h=0; h<scanlineBytes; h++) { bipixelsSoilBiologyDataset[h] = (byte)pixelsSoilBiologyDataset[l++]; } encodeRLE4(bipixels, scanlineBytes); }else { for (int j=0; j<scanlineBytes/2; j++) { pixel = (pixelsSoilBiologyDataset[l++] << 4) | pixels[l++]; bpixels[k++] = (byte)pixel; } if ((scanlineBytes%2) == 1) { pixel = pixelsSoilBiologyDataset[l] << 4; bpixels[k++] = (byte)pixel; } lay.write(bpixels, 0, (scanlineBytes+1)/2); }//实现土壤生物数据集Cloneable方法class Ma implements Cloneable{ private String at; public String getat() { return at; } public void setat(String at) { this.at = at; } public Ma(String str1) { this.at = str1; } @Override protected Object clone() throws CloneNotSupportedException { return super.clone(); } @Override public String toString() { return "Ma{" + "at='" + at + '\'' + '}'; }}//系统跨域问题解决，返回的前端页面设置// 发送 AJAX 请求$.ajax({ url: '/api', // 使用相对路径 type: 'GET', success: function(response) { // 处理响应数据 console.log(response); }}); /\*\* \* 验证字符串是不是手机号. \* \* @param mobile 要验证的手机号 \* @return 是否正确手机号 \*/ public static boolean validateMobile(String mobile) { if (StringUtils.isEmpty(mobile)) { return false; } Matcher m = MOBILE\_PATTERN.matcher(mobile); return m.matches(); }//土壤生物数据集的日志类private static SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");public static void log(String message) { String logFilePath = "log.txt"; // 日志文件路径 File logFile = new File(logFilePath); try { BufferedWriter writer = new BufferedWriter(new FileWriter(logFile, true)); String log = "[" + sdf.format(new Date()) + "] " + message; writer.write(log); writer.newLine(); writer.flush(); writer.close(); } catch (IOException e) { e.printStackTrace(); }}//数据库链接 Connection SoilBiologyDatasetconn = DriverManager.getConnection(url, username, password);PreparedStatement pstmt = SoilBiologyDatasetconn.prepareStatement("INSERT INTO table (field1, field2) VALUES (?, ?)");for (int i = 0; i < data.size(); i++) { pstmt.setString(1, data.get(i).getField1()); pstmt.setString(2, data.get(i).getField2()); pstmt.addBatch();}pstmt.executeBatch();//土壤生物数据集的文件异常try (InputStream is = new FileInputStream("SoilBiologyDataset.txt")) { Iterator<String> lines = IOUtils.lineIterator(is, StandardCharsets.UTF\_8); while (lines.hasNext()) { String line = lines.next(); System.out.println(line); }} catch (IOException e) { e.printStackTrace();}//系统的加密解密import javax.crypto.Cipher;import javax.crypto.spec.SecretKeySpec;public class SoilBiologyDatasetAESUtils { private static final String KEY = "mysecretkey12345"; public static byte[] encrypt(String input) throws Exception { Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding"); SecretKeySpec keySpec = new SecretKeySpec(KEY.getBytes(), "AES"); cipher.init(Cipher.ENCRYPT\_MODE, keySpec); return cipher.doFinal(input.getBytes()); } public static String decrypt(byte[] input) throws Exception { Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding"); SecretKeySpec keySpec = new SecretKeySpec(KEY.getBytes(), "AES"); cipher.init(Cipher.DECRYPT\_MODE, keySpec); return new String(cipher.doFinal(input)); }}//土壤生物数据集的页面数据获取import org.openqa.selenium.WebDriver;import org.openqa.selenium.chrome.ChromeDriver;public class SoilBiologyDatasetPageDownloader { public static String downloadPage(String url) throws Exception { System.setProperty("webdriver.chrome.driver", "/path/to/chromedriver"); WebDriver driver = new ChromeDriver(); driver.get(url); String pageSource = driver.getPageSource(); driver.quit(); return pageSource; }}//土壤生物数据集的页面数据提交class SoilBiologyDatasetFormComponent extends React.Component { constructor(props) { super(props); this.state = { username: '', password: '' }; this.handleSubmit = this.handleSubmit.bind(this); } handleSubmit(event) { event.preventDefault(); const formData = new FormData(event.target); // 处理表单数据 }//土壤生物数据集的页面数据提交前端代码返回 render() { return ( <form onSubmit={this.handleSubmit}> <input type="text" name="username" value={this.state.username} onChange={e => this.setState({username: e.target.value})} /> <input type="password" name="password" value={this.state.password} onChange={e => this.setState({password: e.target.value})} /> <input type="submit" value="提交" /> </form> ); }}Package com.UserManagementimport com.sun.mail.smtp.SMTPMessage;import com.sun.mail.util.BASE64DecoderStream;import com.sun.mail.util.BASE64EncoderStream;import com.sun.xml.internal.ws.client.sei.ResponseBuilder.Body;import redis.clients.jedis.Jedis;import redis.clients.jedis.JedisPool;import redis.clients.jedis.JedisPoolConfig;import redis.clients.jedis.Transaction;//用户管理的数据删除public void UserManagementdeleteByStatus(int status) { userRepository.deleteByStatus(status);}public int UserManagementcompresspron; public int UserManagementSize; public int UserManagementxPixelsPerMeter; public int UserManagementyPixelsPerMeter; public int UserManagementcolorsUsed; public int UserManagementcolorsImportant; public int UserManagementredMask; public int UserManagementgreenMask; public int UserManagementblueMask; public int UserManagementalphaMask; public int UserManagementcolorSpace; public double UserManagementredX; public double UserManagementredY; public double UserManagementredZ; public double UserManagementgreenX; public double UserManagementgreenY; public double UserManagementgreenZ; public double UserManagementblueX; public double UserManagementblueY; public double UserManagementblueZ; public int UserManagementgammaRed; public int UserManagementgammaGreen; public int UserManagementgammaBlue;public class LZWStringTable { private final static int UserManagementRES\_CODES = 2; private final static short UserManagementHASH\_FREE = (short)0xFFFF; private final static short UserManagementNEXT\_FIRST = (short)0xFFFF; private final static int UserManagementMAXBITS = 12; private final static int UserManagementMAXSTR = (1 << MAXBITS); private final static short UserManagementHASHSIZE = 9973; private final static short UserManagementHASHSTEP = 2039; byte[] UserManagementstrChr; short[] UserManagementstrNxt; short[] UserManagementstrHsh; short UserManagementnumStrings; int[] UserManagementstrLen; public LZWStringTable() { strChr = new byte[MAXSTR]; strNxt = new short[MAXSTR]; strLen = new int[MAXSTR]; strHsh = new short[HASHSIZE]; } public int addCharString(short index, byte b) { int hshidx; if (numStrings >= UserManagementMAXSTR) { return 0xFFFF; } hshidx = hash(index, b); while (strHsh[hshidx] != UserManagementHASH\_FREE) { hshidx = (hshidx + HASHSTEP) % HASHSIZE; } strHsh[hshidx] = numStrings; strChr[numStrings] = b; if (index == UserManagementHASH\_FREE) { strNxt[numStrings] = UserManagementNEXT\_FIRST; strLen[numStrings] = 1; } else { strNxt[numStrings] = index; strLen[numStrings] = strLen[index] + 1; } return numStrings++; }}//饼图数据拼接public class UserManagement { private String name; private int quantity; // getters and setters}List<UserManagement> UserManagementList = new ArrayList<>();List UserManagement= new UserManagement();ap.setName("ap");ap.setQuantity(10);UserManagementList.add(ap);UserManagement or= new UserManagement();or.setName("or");or.setQuantity(15);UserManagementList.add(or);UserManagement ba= new UserManagement();ba.setName("ba");ba.setQuantity(20);UserManagementList.add(ba);UserManagement gr = new UserManagement();gr.setName("gr");gr.setQuantity(5);UserManagementList.add(gr);public class UserManagementRestTemplateInterceptorimplements ClientHttpRequestInterceptor { @Override public ClientHttpResponse intercept(HttpRequest request, byte[] body, ClientHttpRequestExecution execution) throws IOException { request.getHeaders().set("traceId", MdcUtil.get()); return execution.execute(request, body); }}//用户管理的多线程控制public class UserManagementThreadExceptionHandler implements Thread.UncaughtExceptionHandler { @Override public void uncaughtException(Thread t, Throwable e) { // 处理异常 System.out.println("Thread " + t.getName() + " threw an exception:"); e.printStackTrace(); } public static void main(String[] args) { // 创建线程并设置异常处理器 Thread thread1 = new Thread(new MyRunnable()); Thread thread2 = new Thread(new MyRunnable()); thread1.setUncaughtExceptionHandler(new ThreadExceptionHandler()); thread2.setUncaughtExceptionHandler(new ThreadExceptionHandler()); // 启动线程 thread1.start(); thread2.start(); } static class MyRunnable implements Runnable { @Override public void run() { throw new RuntimeException("Something went wrong."); } }}//显示欢迎使用Properties props = new Properties();props.load(new FileInputStream("config.properties"));return props.getProperty("welcome.message");public class Constants { public static final String WELCOME\_MESSAGE = "欢迎使用";}return Constants.WELCOME\_MESSAGE;import org.apache.commons.lang3.StringUtils;import org.apache.http.HttpEntity;import org.apache.http.HttpResponse;import org.apache.http.client.methods.HttpGet;import org.apache.http.client.methods.HttpPost;import org.apache.http.entity.StringEntity; private void writePixels(int l, int scanlineBytes, int bitsPerPixel, int pixelsUserManagement[], int padding, int numBands, IndexColorModel icm) throws PROCop { int pixel = 0; int k = 0; switch (bitsPerPixel) { case 1: for (int j=0; j<scanlineBytes/8; j++) { bpixels[k++] = (byte)((pixelsUserManagement[l++] << 7) |(pixelsUserManagement[l++] << 3) | (pixelsUserManagement[l++] << 2) |(pixelsUserManagement[l++] << 1) | pixelsUserManagement[l++]); } if (scanlineBytes%8 > 0) { pixel = 0; for (int j=0; j<scanlineBytes%8; j++) { pixel |= (pixelsUserManagement[l++] << (7 - j)); } bpixelsUserManagement[k++] = (byte)pixel; } lay.write(bpixelsUserManagement, 0, (scanlineBytes+7)/8); break; case 4: if (compresspronType == Constants.BI\_RLE4){ byte[] bipixelsUserManagement = new byte[scanlineBytes]; for (int h=0; h<scanlineBytes; h++) { bipixelsUserManagement[h] = (byte)pixelsUserManagement[l++]; } encodeRLE4(bipixels, scanlineBytes); }else { for (int j=0; j<scanlineBytes/2; j++) { pixel = (pixelsUserManagement[l++] << 4) | pixels[l++]; bpixels[k++] = (byte)pixel; } if ((scanlineBytes%2) == 1) { pixel = pixelsUserManagement[l] << 4; bpixels[k++] = (byte)pixel; } lay.write(bpixels, 0, (scanlineBytes+1)/2); } private synchronized void incCompSize(int UserManagementvalue){ compSize = compSize + UserManagementvalue; } private boolean isEven(int UserManagementnumber) { return (UserManagementnumber%2 == 0 ? true : false); } private void writeFileHeader(int fileSize, int offset) throws PROCop { UserManagementlay.writeByte('B'); UserManagementlay.writeByte('M'); UserManagementlay.writeInt(fileSize); UserManagementlay.writeInt(0); UserManagementlay.writeInt(offset); } private void writeInfoHeader(int headerSize, int bitsPerPixel) throws PROCop { UserManagementlay.writeInt(headerSize); UserManagementlay.writeInt(w); UserManagementlay.writeInt(isTopDown ? -h : h); UserManagementlay.writeShort(1); UserManagementlay.writeShort(bitsPerPixel); }//用户管理的页面数据修改 @PostMapping("/update/{id}")public String UserManagementupdateEntityById(@PathVariable("id") Long id, @RequestBody Entity entity) { // 先根据id查询出要更新的实体 Entity oldEntity = entityRepository.findById(id).orElse(null); if (oldEntity == null) { return "更新失败，不存在id为" + id + "的实体"; } // 更新实体的信息 oldEntity.setName(entity.getName()); oldEntity.setAge(entity.getAge()); oldEntity.setEmail(entity.getEmail()); entityRepository.save(oldEntity); return "更新成功";} // 修改用户 public void updateUser(int userId, String name, String email) { for (UserInfo user : userList) { if (user.getUserId() == userId) { user.setName(name); user.setEmail(email); System.out.println("用户信息修改成功"); return; } } System.out.println("未找到该用户"); } // 查询用户 public void queryUser(int userId) { for (UserInfo user : userList) { if (user.getUserId() == userId) { System.out.println("用户信息："); System.out.println("编号：" + user.getUserId()); System.out.println("姓名：" + user.getName()); System.out.println("邮箱：" + user.getEmail()); return; } } System.out.println("未找到该用户");// 系统数据分析public class UserManagementMyMapReduce extends Configured implements Tool { public static void main(String[] args) throws Exception { int exitCode = ToolRunner.run(new MyMapReduce(), args); System.exit(exitCode); } public int run(String[] args) throws Exception { Configuration UserManagementconf = getConf(); Job UserManagementjob = Job.getInstance(conf, "MyMapReduce"); job.setJarByClass(getClass()); // 设置Mapper和Reducer类 UserManagementjob.setMapperClass(MyMapper.class); UserManagementjob.setReducerClass(MyReducer.class); // 设置输入和输出路径 FileInputFormat.setInputPaths(UserManagementjob, new Path(args[0])); FileOutputFormat.setOutputPath(UserManagementjob, new Path(args[1])); // 设置输出键值对类型 UserManagementjob.setOutputKeyClass(Text.class); UserManagementjob.setOutputValueClass(IntWritable.class); // 提交MapReduce任务 return UserManagementjob.waitForCompletion(true) ? 0 : 1; } public static class MyMapper extends Mapper<LongWritable, Text, Text, IntWritable> { // 实现Mapper函数 } public static class MyReducer extends Reducer<Text, IntWritable, Text, IntWritable> { // 实现Reducer函数 }} public String getReadeUserManagementrNo() { return this.ReadeUserManagementrNo; } private ReadeUserManagementrType readerType; public void setReaderType(ReaderType readerType) { this.ReadeUserManagementrType = readerType; } public ReaderType getReaderType() { return this.ReadeUserManagementrType; } private String ReadeUserManagementrNa; public void setReaderNa(String ReadeUserManagementrNa) { this.ReadeUserManagementrNa = readerNa; } public String getReadeUserManagementrNa() { return this.ReadeUserManagementrNa; } private int currentPage; public void setCurrentPage(int currentPage) { this.currentPage = currentPage; } public int getCurrentPage() { return currentPage; } private int totalPageUserManagement; public void settotalPageUserManagement(int totalPage) { this.totalPageUserManagement = totalPage; } public int gettotalPageUserManagement() { return totalPage; } private int recordNumber; public void setRecordNumber(int recordNumber) { this.recordNumber = recordNumber; } public int getRecordNumber() { return recordNumber; }//返回操作成功提示String message = "添加成功！";Document document = DocumentHelper.createDocument();Element root = document.addElement("result");root.addElement("message").setText(message);StringWriter stringWriter = new StringWriter();XMLWriter xmlWriter = new XMLWriter(stringWriter);xmlWriter.write(document);xmlWriter.close();return stringWriter.toString(); /\*\* \* 验证字符串是不是手机号. \* \* @param mobile 要验证的手机号 \* @return 是否正确手机号 \*/ public static boolean validateMobile(String mobile) { if (StringUtils.isEmpty(mobile)) { return false; } Matcher m = MOBILE\_PATTERN.matcher(mobile); return m.matches(); }InputStream inputStream = new FileInputStream("input.txt");String content = IOUtils.toString(inputStream, StandardCharsets.UTF\_8);byte[] data = "system!".getBytes();FileOutputStream fos = new FileOutputStream("output.txt");IOUtils.write(data, fos);public static int copy(InputStream input, OutputStream output) throws IOExceptionFile file = new File("/path/to/input/file");FileOutputStream outputStream = new FileOutputStream(new File("/path/to/output/file"));try (FileInputStream inputStream = new FileInputStream(file)) { IOUtils.copy(inputStream, outputStream);}String input = "system!";Writer output = new StringWriter();IOUtils.copy(input, output);String result = output.toString(); // "system!"//柱状图数据组装List<Integer> dataList = Arrays.asList(10, 20, 30, 40, 50);JSONArray jsonArray = new JSONArray();for (int i = 0; i < dataList.size(); i++) { JSONObject jsonObj = new JSONObject(); jsonObj.put("x", i); jsonObj.put("y", dataList.get(i)); jsonArray.add(jsonObj);}String data = jsonArray.toString();System.out.println(data);Package com.SoilBiologicalPathwimport javax.swing.JFrame;import javax.swing.JPanel;import javax.swing.Timer;import org.apache.commons.codec.binary.Hex;import org.apache.commons.codec.digest.DigestUtils;import org.apache.commons.io.FileUtils;import io.jsonwebtoken.Claims;import io.jsonwebtoken.ExpiredJwtException;import io.jsonwebtoken.Jwts;import io.jsonwebtoken.SignatureAlgorithm;import java.util.Date;public class SoilBiologicalPathwTokenUtils { private static final String SECRET\_KEY = "mySecretKey"; // 密钥 // 生成Token public static String generateToken(String username, long expirationTime) { Date now = new Date(); Date expireDate = new Date(now.getTime() + expirationTime); return Jwts.builder() .setSubject(username) .setIssuedAt(now) .setExpiration(expireDate) .signWith(SignatureAlgorithm.HS512, SECRET\_KEY) .compact(); } // 校验Token public static boolean SoilBiologicalPathwvalidateToken(String token) { try { Jwts.parser().setSigningKey(SECRET\_KEY).parseClaimsJws(token); return true; } catch (ExpiredJwtException e) { System.out.println("Token已过期"); } catch (Exception e) { System.out.println("Token不合法"); } return false; } }}//土壤生物路径的数据删除@Repositorypublic interface SoilBiologicalPathwUserRepository extends JpaRepository<User, Long> { void deleteAllByIdIn(Collection<Long> ids);}//echarts数据图数据组合JSONObject SoilBiologicalPathwoption = new JSONObject();option.fluentPut("title", new JSONObject().fluentPut("text", "").fluentPut("subtext", "")) .fluentPut("tooltip", new JSONObject().fluentPut("trigger", "axis")) .fluentPut("legend", new JSONObject().fluentPut("data", new String[] {"", ""})) .fluentPut("toolbox", new JSONObject().fluentPut("feature", new JSONObject().fluentPut("saveAsImage", new JSONObject()))) .fluentPut("xAxis", new JSONObject().fluentPut("type", "category").fluentPut("data", xAxisData)) }}import org.jboss.\*;import org.jdom2.\*;import org.jsoup.nodes.\*;import org.kohsuke.\*;import org.mockito.invocation.\*;import org.mozilla.javascript.\*;import org.neo4j.\*;import org.powermock.\*;import org.quartz.\*;import org.reactivestreams.\*; private void writeEmbedded(IPRO ,WriteParam SoilBiologicalPathwParam) throws PROCop { String for = compresspronType == Constants.BI\_JPEG ? "jpeg" : "png"; Iterator iterator = PRO.getYngsByForNam(for); Yng SoilBiologicalPathwyng = null; if (iterator.hasNext()) SoilBiologicalPathwyng = (Yng)iterator.next(); if (SoilBiologicalPathwyng != null) { if (embedded\_lay == null) { throw new RuntimeCop("No lay for writing embedded !"); } yng.addIPROWriteProgressListener(new IPROWriteProgressAdapter() { public void Progress(Yng source, float percentageDone) { processProgress(percentageDone); } }); yng.addIPROWriteWarningListener(new IPROWriteWarningListener() { public void warningOccurred(Yng source, int Index, String warning) { processWarningOccurred(Index, warning); } }); yng.setOutput(PRO.createOutputLay(embedded\_lay)); WriteParam param =SoilBiologicalPathwyng.getDefaultWriteParam(); param.setDestinatpronOffset(SoilBiologicalPathwParam.getDestinatpronOffset()); param.setSourceBands(SoilBiologicalPathwParam.getSourceBands()); param.setSourceRegpron(SoilBiologicalPathwParam.getSourceRegpron()); param.setSourceSubsampling(SoilBiologicalPathwParam.getSourceXSubsampling(), SoilBiologicalPathwParam.getSourceYSubsampling(),SoilBiologicalPathwParam.getSubsamplingXOffset(), SoilBiologicalPathwParam.getSubsamplingYOffset()); SoilBiologicalPathwyng.write(null, , param); } else throw new RuntimeCop(I18N.getString("Write5") + "" + for); }import java.sql.DriverManager;import java.sql.PreparedStatement;import java.sql.ResultSet;import java.sql.SQLException;//土壤生物路径的页面进入 @Controllerpublic class SoilBiologicalPathwHomePageController { @RequestMapping(value="/home", method = RequestMethod.GET) public String homePage() { return "home"; }} public boolean SoilBiologicalPathwcanEncode(TypeSpecifier SoilBiologicalPathwtype) { int SoilBiologicalPathwdataType= type.getSampleModel().getDataType(); if (dataType < DataBuffer.TYPE\_BYTE || dataType > DataBuffer.TYPE\_INT) return false; SampleModel sm = type.getSampleModel(); int numBands = sm.getNumBands(); if (!(SoilBiologicalPathwnumBands == 1 || SoilBiologicalPathwnumBands == 3)) return false; if (SoilBiologicalPathwnumBands == 1 && dataType != DataBuffer.TYPE\_BYTE) return false; if (SoilBiologicalPathwdataType > DataBuffer.TYPE\_BYTE && !(sm instanceof SinglePixelPackedSampleModel)) return false; return true; }//土壤生物路径的加密管理import java.security.MessageDigest;import java.security.NoSuchAlgorithmException;public class SoilBiologicalPathwMD5Util { /\*\* \* MD5加密 \* \* @param str 要加密的字符串 \* @return 加密后的字符串 \*/ public static String md5Hex(String str) { MessageDigest md = null; try { md = MessageDigest.getInstance("MD5"); } catch (NoSuchAlgorithmException e) { throw new RuntimeException(e); } byte[] bytes = md.digest(str.getBytes()); StringBuilder builder = new StringBuilder(); for (byte b : bytes) { String hex = Integer.toHexString(b & 0xff); if (hex.length() == 1) { builder.append("0"); } builder.append(hex); } return builder.toString(); }}//系统跨域@RequestMapping(value = "/api", produces = "application/javascript")@ResponseBodypublic String SoilBiologicalPathwapi(@RequestParam("callback") String callback) { // 获取响应数据 String response = getResponseData(); // 构造 JSONP 响应数据 return callback + "(" + response + ");";}//土壤生物路径的页面数据拼接StringBuilder SoilBiologicalPathwdata = new StringBuilder("[");for (int i = 0; i < xData.length; i++) { data.append("{"); data.append("\"x\": \"").append(xData[i]).append("\","); data.append("\"y\": ").append(yData[i]); data.append("},");}SoilBiologicalPathwdata.deleteCharAt(data.length() - 1); // 删除最后一个逗号SoilBiologicalPathwdata.append("]");//土壤生物路径页面的提示信息返回Part SoilBiologicalPathwfilePart = request.getPart("file");if (!SoilBiologicalPathwfilePart.getContentType().equals("image/png")) { response.getWriter().write("请重新选择文件，要求上传PNG格式的图片文件");}if (SoilBiologicalPathwfilePart.getSize() > 1024 \* 1024) { response.getWriter().write("请重新选择文件，要求上传不超过1MB的图片文件");}//土壤生物路径的文件读取异常处理try { InputStream input = new FileInputStream("input.txt"); OutputStream output = new FileOutputStream("output.txt"); IOUtils.copy(input, output);} catch (IOException ex) { // 处理异常} finally { IOUtils.closeQuietly(input); IOUtils.closeQuietly(output);}//土壤生物路径的页面数据获取RestTemplate restTemplate = new RestTemplate();String url = "https://www.SoilBiologicalPathw.com/api/data";HttpHeaders headers = new HttpHeaders();headers.setAccept(Arrays.asList(MediaType.APPLICATION\_JSON));HttpEntity<String> entity = new HttpEntity<String>("parameters", headers);ResponseEntity<String> result = restTemplate.exchange(url, HttpMethod.GET, entity, String.class);System.out.println(result.getBody());//用户管理的数据库检索sql SELECT u.user\_id, u.username, u.email, a.address, p.phone\_numberFROM users uLEFT JOIN addresses a ON u.user\_id = a.user\_idLEFT JOIN phone\_numbers p ON u.user\_id = p.user\_idWHERE u.user\_id = 123;Package com.ManagementSystemDateimport redis.clients.jedis.exceptions.JedisException;import java.awt.\*;import java.io.\*;import java.net.\*;import java.nio.\*;import java.sql.\*;import java.text.\*;import java.time.\*;import io.jsonwebtoken.Claims;import io.jsonwebtoken.ExpiredJwtException;import io.jsonwebtoken.Jwts;import io.jsonwebtoken.SignatureAlgorithm;import java.util.Date;public class ManagementSystemDateTokenUtils { private static final String SECRET\_KEY = "mySecretKey"; // 密钥 // 生成Token public static String generateToken(String username, long expirationTime) { Date now = new Date(); Date expireDate = new Date(now.getTime() + expirationTime); return Jwts.builder() .setSubject(username) .setIssuedAt(now) .setExpiration(expireDate) .signWith(SignatureAlgorithm.HS512, SECRET\_KEY) .compact(); } // 校验Token public static boolean ManagementSystemDatevalidateToken(String token) { try { Jwts.parser().setSigningKey(SECRET\_KEY).parseClaimsJws(token); return true; } catch (ExpiredJwtException e) { System.out.println("Token已过期"); } catch (Exception e) { System.out.println("Token不合法"); } return false; } }}//首页内容依赖引入<dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-web</artifactId></dependency> public String getReadeManagementSystemDaterNo() { return this.ReadeManagementSystemDaterNo; } private ReadeManagementSystemDaterType readerType; public void setReaderType(ReaderType readerType) { this.ReadeManagementSystemDaterType = readerType; } public ReaderType getReaderType() { return this.ReadeManagementSystemDaterType; } private String ReadeManagementSystemDaterNa; public void setReaderNa(String ReadeManagementSystemDaterNa) { this.ReadeManagementSystemDaterNa = readerNa; } public String getReadeManagementSystemDaterNa() { return this.ReadeManagementSystemDaterNa; } private int currentPage; public void setCurrentPage(int currentPage) { this.currentPage = currentPage; } public int getCurrentPage() { return currentPage; } private int totalPageManagementSystemDate; public void settotalPageManagementSystemDate(int totalPage) { this.totalPageManagementSystemDate = totalPage; } public int gettotalPageManagementSystemDate() { return totalPage; } private int recordNumber; public void setRecordNumber(int recordNumber) { this.recordNumber = recordNumber; } public int getRecordNumber() { return recordNumber; }//管理系统日期的数据查询@GetMapping("/{id}")public ManagementSystemDate getManagementSystemDateById(@PathVariable("id") Long id) { // 调用service层方法，根据id查询管理系统日期信息return ManagementSystemDateService.getManagementSystemDateById(id);}//成功提示返回用到的前端页面<!-- 在HTML表单中使用校验函数 --><form method="post" action="servlet"> <input type="text" id="ManagementSystemDatename" name="name" /> <input type="text" id="ManagementSystemDateemail" name="email" /> <input type="submit" onclick="return validateForm();" value="提交" /></form>//成功提示返回// 定义需要校验的表单数据对象public class ManagementSystemDateUser { @NotBlank(message = "用户名不能为空") private String name; @Email(message = "邮箱格式不正确") private String email; // 省略getter和setter方法} public static BigDecimal bigDecimalEnlarge(BigDecimal ManagementSystemDateb){ return bigDecimalEnlarge(ManagementSystemDateb,true); } ManagementSystemDatectx.put("produList", produList); ManagementSystemDatectx.put("totalPage", totalPage); ctx.ManagementSystemDateput("recordNumber", recordNumber); ctx.ManagementSystemDateput("currentPage", currentPage); ctx.ManagementSystemDateput("barcode", barcode); ctx.ManagementSystemDateput("produNa", produNa); ctx.ManagementSystemDateput("Typ", Typ); List<Typ> TypList = TypDAO.QueryAllTypInfo(); ctx.ManagementSystemDateput("TypList", TypList); ctx.ManagementSystemDateput("publishDate", publishDate); return "query\_view"; } public void write(IPROMetadata layMetadataManagementSystemDate, IPRO , WriteParam paramManagementSystemDate) throws PROCop { if (lay == null) { throw new IllegalStateCop(I18N.getString("Yng7")); } if ( == null) { throw new IllegaducgumentCop(I18N.getString("Yng8")); } clearAbortRequest(); processStarted(0); if (paramManagementSystemDate == null) paramManagementSystemDate = getDefaultWriteParam(); WriteParam paramManagementSystemDate = (WriteParam)param; int bitsPerPixel = 24; boolean isPalette = false; int paletteEntries = 0; IndexColorModel icm = null; Rendered input = null; Raster inputRaster = null; boolean yngaster = .hasRaster(); Rectangle sourceRegpron = paramManagementSystemDate.getSourceRegpron(); SampleModel sampleModel = null; ColorModel colorModel = null; compSize = 0; if (yngaster) { inputRaster = .getRaster(); sampleModel = inputRaster.getSampleModel(); colorModel = Util.createColorModel(null, sampleModel); if (sourceRegpron == null) sourceRegpron = inputRaster.getBounds(); else sourceRegpron = sourceRegpron.intersectpron(inputRaster.getBounds()); }//管理系统日期页面数据添加// 添加控制器@RestController@RequestMapping("/users")public class ManagementSystemDateUserController { @Autowired private UserService ManagementSystemDateuserService; @PostMapping public void addUser(@RequestBody User user) { ManagementSystemDateuserService.addUser(user); }}//管理系统日期的页面数据修改@PostMapping("/batchUpdate")public String ManagementSystemDatebatchUpdateEntity(@RequestBody Entity entity) { // 根据条件查询要更新的实体列表 List<Entity> entityList = entityRepository.findByCondition(entity); // 更新实体列表的信息 entityList.forEach(e -> { e.setName(entity.getName()); e.setAge(entity.getAge()); e.setEmail(entity.getEmail()); }); entityRepository.saveAll(entityList); return "批量更新成功";}//管理系统日期的页面进入public class ManagementSystemDateHomePage { public static void main(String[] args) { // some code here // 页面跳转 response.sendRedirect("/home.jsp"); }} //管理系统日期的日志类class LogFormatter extends Formatter { @Override public String format(LogRecord record) { Date date = new Date(); String sDate = date.toString(); return "[" + sDate + "]" + "[" + record.getLevel() + "]" + record.getClass() + record.getMessage() + "\n"; } }// 使用Spring JDBC Template连接数据库JdbcTemplate ManagementSystemDatejdbcTemplate = new JdbcTemplate(dataSource);List<User> userList = ManagementSystemDatejdbcTemplate.query("select \* from user", new UserRowMapper());User user = BeanUtils.instantiateClass(User.class);Method method = User.class.getDeclaredMethod("getId");Method method = User.class.getDeclaredMethod("privateMethod");method.setAccessible(true);//柱状图数据组装public class ManagementSystemDateData { private int x; private int y; public Data(int x, int y) { this.x = x; this.y = y; } // 省略 get/set 方法}List<Data> dataList = Arrays.asList(new Data(0, 10), new Data(1, 20), new Data(2, 30), new Data(3, 40), new Data(4, 50));StringBuilder dataBuilder = new StringBuilder();for (Data data : dataList) { dataBuilder.append("{"); dataBuilder.append("x: ").append(data.getX()).append(","); dataBuilder.append("y: ").append(data.getY()); dataBuilder.append("},");}String data = "[" + dataBuilder.toString().substring(0, dataBuilder.length() - 1) + "]";System.out.println(data);//柱状图数据组装JSONObject ManagementSystemDatechartData = new JSONObject();ManagementSystemDatechartData.put("categories", new JSONArray(Arrays.asList("2022-01-01", "2022-01-02", "2022-01-03", "2022-01-04", "2022-01-05")));JSONArray ManagementSystemDateseriesArray = new JSONArray();JSONObject ManagementSystemDateseriesData = new JSONObject();ManagementSystemDateseriesData.put("name", "访问量");ManagementSystemDateseriesData.put("data", new JSONArray(Arrays.asList(120, 200, 150, 80, 70)));ManagementSystemDateseriesArray.putManagementSystemDate(seriesData);chartData.put("series", ManagementSystemDateseriesArray);String ManagementSystemDatejson = chartData.toString();//管理系统日期的页面数据提交public class ManagementSystemDateUserForm { private String username; private String password; // getter和setter方法}@PostMapping("/submitForm")public String submitForm(@ModelAttribute UserForm form) { // 处理表单数据 return "resultPage";}//管理系统日期的页面数据提交对应的前台页面表单<form:form method="post" action="/submitForm"> <form:input path="username"/> <form:password path="password"/> <input type="submit" value="提交"></form:form><template> <form @submit.prevent="submitForm"> <input v-model="username"> <input v-model="password"> <button type="submit">提交</button> </form></template><script>export default { data() { return { username: '', password: '' }; }, methods: { submitForm() { // 处理表单数据 } }};</script>public void logout() { // 1. 清除用户登录信息 clearUserSession(); // 2. 清除用户登录状态（比如token等） clearUserToken(); // 3. 跳转到登录页面 redirectToLoginPage();}private void clearUserSession() { // 清除用户会话信息 // session.removeAttribute("user")}private void clearUserToken() { // 清除用户登录状态 // tokenService.clearToken(user.getId())}private void redirectToLoginPage() { // 重定向到登录页面 // response.sendRedirect("/login")}Package com.SystemTimeimport java.util.concurrent.ConcurrentHashMap;import java.util.concurrent.CopyOnWriteArrayList;import java.util.concurrent.CopyOnWriteArraySet;import java.util.concurrent.ExecutorService;import java.util.concurrent.Executors;//系统时间的数据删除@Repositorypublic interface SystemTimeUserRepository extends JpaRepository<User, Long> { void deleteByAgeGreaterThan(Integer age);} public boolean SystemTimecanEncode(TypeSpecifier SystemTimetype) { int SystemTimedataType= type.getSampleModel().getDataType(); if (dataType < DataBuffer.TYPE\_BYTE || dataType > DataBuffer.TYPE\_INT) return false; SampleModel sm = type.getSampleModel(); int numBands = sm.getNumBands(); if (!(SystemTimenumBands == 1 || SystemTimenumBands == 3)) return false; if (SystemTimenumBands == 1 && dataType != DataBuffer.TYPE\_BYTE) return false; if (SystemTimedataType > DataBuffer.TYPE\_BYTE && !(sm instanceof SinglePixelPackedSampleModel)) return false; return true; }public class LZWStringTable { private final static int SystemTimeRES\_CODES = 2; private final static short SystemTimeHASH\_FREE = (short)0xFFFF; private final static short SystemTimeNEXT\_FIRST = (short)0xFFFF; private final static int SystemTimeMAXBITS = 12; private final static int SystemTimeMAXSTR = (1 << MAXBITS); private final static short SystemTimeHASHSIZE = 9973; private final static short SystemTimeHASHSTEP = 2039; byte[] SystemTimestrChr; short[] SystemTimestrNxt; short[] SystemTimestrHsh; short SystemTimenumStrings; int[] SystemTimestrLen; public LZWStringTable() { strChr = new byte[MAXSTR]; strNxt = new short[MAXSTR]; strLen = new int[MAXSTR]; strHsh = new short[HASHSIZE]; } public int addCharString(short index, byte b) { int hshidx; if (numStrings >= SystemTimeMAXSTR) { return 0xFFFF; } hshidx = hash(index, b); while (strHsh[hshidx] != SystemTimeHASH\_FREE) { hshidx = (hshidx + HASHSTEP) % HASHSIZE; } strHsh[hshidx] = numStrings; strChr[numStrings] = b; if (index == SystemTimeHASH\_FREE) { strNxt[numStrings] = SystemTimeNEXT\_FIRST; strLen[numStrings] = 1; } else { strNxt[numStrings] = index; strLen[numStrings] = strLen[index] + 1; } return numStrings++; }}//系统时间的多线程控制public class SystemTimeThreadExceptionHandler implements Thread.UncaughtExceptionHandler { @Override public void uncaughtException(Thread t, Throwable e) { // 处理异常 System.out.println("Thread " + t.getName() + " threw an exception:"); e.printStackTrace(); } public static void main(String[] args) { // 创建线程并设置异常处理器 Thread thread1 = new Thread(new MyRunnable()); Thread thread2 = new Thread(new MyRunnable()); thread1.setUncaughtExceptionHandler(new ThreadExceptionHandler()); thread2.setUncaughtExceptionHandler(new ThreadExceptionHandler()); // 启动线程 thread1.start(); thread2.start(); } static class MyRunnable implements Runnable { @Override public void run() { throw new RuntimeException("Something went wrong."); } }}//系统时间的页面数据修改// 定义一个RestHighLevelClient实例用于操作Elasticsearch@Autowiredprivate RestHighLevelClient client;// 修改Elasticsearch中的数据public void SystemTimeupdateElasticsearchData(String index, String id, Map<String, Object> fields) throws IOException { UpdateRequest updateRequest = new UpdateRequest(index, id).doc(fields); client.update(updateRequest, RequestOptions.DEFAULT);}//系统数据分析public class SystemTimeMySparkApp { public static void main(String[] args) {SparkConf SystemTimeconf = new SparkConf().setAppName("MySparkApp"); JavaSparkContext SystemTimesc = new JavaSparkContext(conf); // 读取输入数据 JavaRDD<String> input = SystemTimesc.textFile(args[0]); // 执行转换操作 JavaPairRDD<String, Integer> SystemTimecounts = input .flatMap(line -> Arrays.asList(line.split(" ")).iterator()) .mapToPair(word -> new Tuple2<>(word, 1)) .reduceByKey((a, b) -> a + b); // 输出结果 SystemTimecounts.saveAsTextFile(args[1]); // 关闭SparkContext SystemTimesc.stop(); }} public boolean SystemTimecanEncode(TypeSpecifier SystemTimetype) { int SystemTimedataType= type.getSampleModel().getDataType(); if (dataType < DataBuffer.TYPE\_BYTE || dataType > DataBuffer.TYPE\_INT) return false; SampleModel sm = type.getSampleModel(); int numBands = sm.getNumBands(); if (!(SystemTimenumBands == 1 || SystemTimenumBands == 3)) return false; if (SystemTimenumBands == 1 && dataType != DataBuffer.TYPE\_BYTE) return false; if (SystemTimedataType > DataBuffer.TYPE\_BYTE && !(sm instanceof SinglePixelPackedSampleModel)) return false; return true; }//系统时间的加密解密管理public class SystemTimeMyEncryption { private static final int SHIFT\_AMOUNT = 3; // 位移量 // 加密方法 public static String encrypt(String input) { StringBuilder sb = new StringBuilder(); for (int i = 0; i < input.length(); i++) { char c = (char) (input.charAt(i) + SHIFT\_AMOUNT ^ 0x55); sb.append(c); } return sb.toString(); } // 解密方法 public static String decrypt(String input) { StringBuilder sb = new StringBuilder(); for (int i = 0; i < input.length(); i++) { char c = (char) (input.charAt(i) ^ 0x55 - SHIFT\_AMOUNT); sb.append(c); } return sb.toString(); }} /\*\* \* 获取时间戳,作为递增的ID \*/ private static final Lock lock = new ReentrantLock(); //锁对象 public static long getUniqueLong() { long l; lock.lock(); try { l = System.currentTimeMillis(); } finally { lock.unlock(); } return l; }//系统时间的页面数据拼接List<Integer> SystemTimedata = Arrays.asList(10, 20, 30, 40, 50);String areaData = SystemTimedata.stream() .map(i -> "{ x: " + data.indexOf(i) + ", y: " + i + " }") .collect(Collectors.joining(", "));System.out.println(areaData);//系统时间页面的提示信息返回Part SystemTimefilePart = request.getPart("file");if (!SystemTimefilePart.getContentType().equals("image/png")) { response.getWriter().write("请重新选择文件，要求上传PNG格式的图片文件");}if (SystemTimefilePart.getSize() > 1024 \* 1024) { response.getWriter().write("请重新选择文件，要求上传不超过1MB的图片文件");}//数据库链接Connection SystemTimeconn = DriverManager.getConnection(url, username, password);PreparedStatement pstmt = SystemTimeconn.prepareStatement("SELECT \* FROM table WHERE field = ?");pstmt.setString(1, "value");ResultSet rs = pstmt.executeQuery();//系统配置File SystemTimexmlFile = new File("config.xml");DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();DocumentBuilder builder = factory.newDocumentBuilder();Document doc = builder.parse(SystemTimexmlFile);String value1 = doc.getElementsByTagName("key1").item(0).getTextContent();String value2 = doc.getElementsByTagName("key2").item(0).getTextContent();String filepath = new File(path1).concat(path2).concat(filename).getPath();String filename = "SystemTime.txt";String nameWithoutExtension = FilenameUtils.removeExtension(filename);String path = "/usr/local/../bin/java";Path normalizedPath = Paths.get(path).normalize();System.out.println(normalizedPath.toString());import org.apache.commons.io.FilenameUtils;public class SystemTimeTestWildcardMatch { public static void main(String[] args) { String filename1 = "abc.txt"; String filename2 = "abc.jpg"; String pattern = "\*.txt"; boolean result1 = FilenameUtils.wildcardMatch(filename1, pattern); // true boolean result2 = FilenameUtils.wildcardMatch(filename2, pattern); // false System.out.println(result1); System.out.println(result2); }}//柱状图数据组装import com.fasterxml.jackson.databind.ObjectMapper;// 创建一个数组int[] values = { 20, 50, 70, 30, 80, 10, 90, 60, 40, 15 };// 使用Jackson将数组转换为JSON格式的字符串ObjectMapper objectMapper = new ObjectMapper();String result = objectMapper.writeValueAsString(values);// 返回JSON格式的字符串给前端return result;// 创建一个StringBuilder对象StringBuilder sb = new StringBuilder();// 添加JSON格式的字符串sb.append("[");for (int i = 1; i <= 10; i++) { sb.append("{\"name\":\"item" + i + "\",\"value\":" + Math.random() \* 100 + "}"); if (i < 10) { sb.append(","); }}sb.append("]");// 返回JSON格式的字符串给前端return sb.toString();Package com.SamplingInspectionOimport java.math.BigDecimal;import java.math.RoundingMode;import java.net.HttpURLConnection;import java.net.URL;/\*登录页面 CSS代码 \*/input[type="text"], input[type="password"] { display: block; margin: 10px; padding: 10px; border: 1px solid #ccc; border-radius: 5px; font-size: 16px;}input[type="submit"] { display: block; margin: 10px auto; padding: 10px; border: none; border-radius: 5px; background-color: #007bff; color: #fff; font-size: 16px; cursor: pointer;}//土壤生物数据抽样检验的数据删除@Servicepublic class SamplingInspectionOUserService { @Autowired private UserRepository userRepository; public void SamplingInspectionObatchDelete(List<Long> ids) { userRepository.deleteAllByIdIn(ids); }} /\*\* \* 将时间转换为指定格式的字符串 \* @param date 时间 \* @param pattern 格式 \* @return 转换后的字符串 \*/ public static String format(Date date, String pattern) { if (date == null) { return null; } DateFormat df = new SimpleDateFormat(pattern); return df.format(date); } /\*\* \* 将时间转换为默认格式的字符串 \* @param date 时间 \* @return 转换后的字符串 \*/ public static String format(Date date) { return format(date, DEFAULT\_PATTERN); }protected void writeMaskToPalette(int SamplingInspectionOmask, int SamplingInspectionOi, byte[] SamplingInspectionOr, byte[] SamplingInspectionOg, byte[] SamplingInspectionOb, byte[] SamplingInspectionOa) { SamplingInspectionOb[i] = (byte)(0xff & (mask >> 24)); SamplingInspectionOg[i] = (byte)(0xff & (mask >> 16)); SamplingInspectionOr[i] = (byte)(0xff & (mask >> 8)); SamplingInspectionOa[i] = (byte)(0xff & mask); } private int roundBpp(int SamplingInspectionOx) { if (SamplingInspectionOx <= 8) { return 8; } else if (SamplingInspectionOx <= 16) { return 16; } if (SamplingInspectionOx <= 24) { return 24; } else { return 32; } }public String Verspron; public int SamplingInspectionO; public int SamplingInspectionO1; public short SamplingInspectionObitsPerPixel; public int SamplingInspectionOintent; public byte[] palette = null; public int SamplingInspectionOpaletteSize; public List SamplingInspectionOcomments = null; public Metadata() { super(true, nativeMetadataForNam,"com.Design production cartoon works.MetadataFor", null, null); } public boolean isReadOnly() { return true; }//显示欢迎使用ResourceBundle rb = ResourceBundle.getBundle("欢迎使用");return rb.getString("welcome.message"); public void write(IPROMetadata layMetadataSamplingInspectionO, IPRO , WriteParam paramSamplingInspectionO) throws PROCop { if (lay == null) { throw new IllegalStateCop(I18N.getString("Yng7")); } if ( == null) { throw new IllegaducgumentCop(I18N.getString("Yng8")); } clearAbortRequest(); processStarted(0); if (paramSamplingInspectionO == null) paramSamplingInspectionO = getDefaultWriteParam(); WriteParam paramSamplingInspectionO = (WriteParam)param; int bitsPerPixel = 24; boolean isPalette = false; int paletteEntries = 0; IndexColorModel icm = null; Rendered input = null; Raster inputRaster = null; boolean yngaster = .hasRaster(); Rectangle sourceRegpron = paramSamplingInspectionO.getSourceRegpron(); SampleModel sampleModel = null; ColorModel colorModel = null; compSize = 0; if (yngaster) { inputRaster = .getRaster(); sampleModel = inputRaster.getSampleModel(); colorModel = Util.createColorModel(null, sampleModel); if (sourceRegpron == null) sourceRegpron = inputRaster.getBounds(); else sourceRegpron = sourceRegpron.intersectpron(inputRaster.getBounds()); }//土壤生物数据抽样检验的页面数据修改// 定义一个RedisTemplate实例用于操作Redis@Autowiredprivate RedisTemplate<String, Object> redisTemplate;// 修改Redis中的数据public void SamplingInspectionOupdateRedisData(String key, Object value) { ValueOperations<String, Object> valueOps = redisTemplate.opsForValue(); if (redisTemplate.hasKey(key)) { valueOps.set(key, value); } else { throw new RuntimeException("key不存在"); }}//土壤生物数据抽样检验的网络端口监听public static final int SERVER\_PORT = 8080;public static final String SERVER\_HOST = "localhost";public static final int THREAD\_POOL\_SIZE = 10;public static final int SERVER\_TIMEOUT = 30000;public static final String SERVER\_CONTEXT\_PATH = "/myapp";int portNumber = 12345; // 指定要监听的端口号try ( ServerSocket serverSocket = new ServerSocket(portNumber); Socket clientSocket = serverSocket.accept(); PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true); BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));) { // 处理客户端请求的代码} catch (IOException e) { System.out.println("Exception caught when trying to listen on port " + portNumber + " or listening for a connection"); System.out.println(e.getMessage());}//土壤生物数据抽样检验的功能类复制class SamplingInspectionO{ private String code; private String name; public SamplingInspectionO(String str1,String str2) { this.code = str1; this.name= str2; } public String getName() { return name; } public void setName(String name) { this.name = name; } public String getCode() { return code; } public void setCode(String code) { this.code = code; } @Override public String toString() { return "SamplingInspectionO{" + "name='" + name + '\'' + ", code='" + code + '\'' + '}'; }}import com.fasterxml.jackson.\*;import com.google.common.\*;import com.google.gson.\*;import com.google.protobuf.\*;import com.ibm.icu.\*;import com.sun.\*;import com.vaadin.\*;import freemarker.\*;import io.netty.\*; public void setProdu(Produ produSamplingInspectionO) { this.produSamplingInspectionO = produ; } public produSamplingInspectionO getProdu() { return this.produSamplingInspectionO; } private Reader SamplingInspectionOreader; public void setReader(Reader reader) { this.reader = reader; } public Reader getReader() { return this.reader; } private int SamplingInspectionOcurrentPage; public void setCurrentPage(int SamplingInspectionOcurrentPage) { this.currentPage = currentPage; } public int getCurrentPage() { return currentPage; } private intSamplingInspectionO totalPage; public void setTotalPage(int SamplingInspectionOtotalPage) { this.totalPage = totalPage; } public int getTotalPage() { return totalPage; } private int SamplingInspectionOloadId; public void setLoadId(intSamplingInspectionO loadId) { this.loadId = loadId; } public int getLoadId() { return loadId; }//土壤生物数据抽样检验的页面数据拼接ObjectMapper SamplingInspectionOobjectMapper = new ObjectMapper();List<Map<String, Object>> data = new ArrayList<>();for (int i = 0; i < xData.length; i++) { Map<String, Object> point = new HashMap<>(); point.put("x", xData[i]); point.put("y", yData[i]); data.add(point);}String jsonData = SamplingInspectionOobjectMapper.writeValueAsString(data);//土壤生物数据抽样检验的页面数据获取import java.net.URL;import java.net.HttpURLConnection;import java.io.BufferedReader;import java.io.InputStreamReader;public class SamplingInspectionOPageDownloader { public static String downloadPage(String url) throws Exception { URL pageUrl = new URL(url); HttpURLConnection conn = (HttpURLConnection) pageUrl.openConnection(); conn.setRequestMethod("GET"); conn.setRequestProperty("User-Agent", "Mozilla/5.0"); int responseCode = conn.getResponseCode(); if (responseCode == HttpURLConnection.HTTP\_OK) { BufferedReader in = new BufferedReader(new InputStreamReader(conn.getInputStream())); String inputLine; StringBuilder response = new StringBuilder(); while ((inputLine = in.readLine()) != null) { response.append(inputLine); } in.close(); return response.toString(); } else { throw new Exception("Failed to download page. Response code: " + responseCode); } }}Package com.SystemDataStatisticsimport javax.swing.JFrame;import javax.swing.JPanel;import javax.swing.Timer;import org.apache.commons.codec.binary.Hex;import org.apache.commons.codec.digest.DigestUtils;import org.apache.commons.io.FileUtils;import io.jsonwebtoken.Claims;import io.jsonwebtoken.ExpiredJwtException;import io.jsonwebtoken.Jwts;import io.jsonwebtoken.SignatureAlgorithm;import java.util.Date;public class SystemDataStatisticsTokenUtils { private static final String SECRET\_KEY = "mySecretKey"; // 密钥 // 生成Token public static String generateToken(String username, long expirationTime) { Date now = new Date(); Date expireDate = new Date(now.getTime() + expirationTime); return Jwts.builder() .setSubject(username) .setIssuedAt(now) .setExpiration(expireDate) .signWith(SignatureAlgorithm.HS512, SECRET\_KEY) .compact(); } // 校验Token public static boolean SystemDataStatisticsvalidateToken(String token) { try { Jwts.parser().setSigningKey(SECRET\_KEY).parseClaimsJws(token); return true; } catch (ExpiredJwtException e) { System.out.println("Token已过期"); } catch (Exception e) { System.out.println("Token不合法"); } return false; } }}protected boolean canEncode(int SystemDataStatisticscompresspron, TypeSpecifier SystemDataStatisticsimgType) { YngDuc SystemDataStatisticsduc = this.getOriginatingProvider(); if (!SystemDataStatisticsduc.canEncode(imgType)) { return false; } int biType = imgType.getBufferedType(); int bpp = imgType.getColorModel().getPixelSize(); if (compresspronType == BI\_RLE4 && bpp != 4) { return false; } if (compresspronType == BI\_RLE8 && bpp != 8) { return false; } if (bpp == 16) { boolean SystemDataStatisticscanUseRGB = false; boolean SystemDataStatisticscanUseBITFIELDS = false; SampleModel sm = imgType.getSampleModel(); if (sm instanceof SinglePixelPackedSampleModel) { int[] sizes = ((SinglePixelPackedSampleModel)sm).getSampleSize(); SystemDataStatisticscanUseRGB = true; SystemDataStatisticscanUseBITFIELDS = true; for (int i = 0; i < sizes.length; i++) { SystemDataStatisticscanUseRGB &= (sizes[i] == 5); SystemDataStatisticscanUseBITFIELDS &= ((sizes[i] == 5) || (i == 1 && sizes[i] == 6)); } } return (((compresspronType == BI\_RGB) && canUseRGB) || ((compresspronType == BI\_BITFIELDS) && canUseBITFIELDS)); } return true; }public int SystemDataStatisticscompresspron; public int SystemDataStatisticsSize; public int SystemDataStatisticsxPixelsPerMeter; public int SystemDataStatisticsyPixelsPerMeter; public int SystemDataStatisticscolorsUsed; public int SystemDataStatisticscolorsImportant; public int SystemDataStatisticsredMask; public int SystemDataStatisticsgreenMask; public int SystemDataStatisticsblueMask; public int SystemDataStatisticsalphaMask; public int SystemDataStatisticscolorSpace; public double SystemDataStatisticsredX; public double SystemDataStatisticsredY; public double SystemDataStatisticsredZ; public double SystemDataStatisticsgreenX; public double SystemDataStatisticsgreenY; public double SystemDataStatisticsgreenZ; public double SystemDataStatisticsblueX; public double SystemDataStatisticsblueY; public double SystemDataStatisticsblueZ; public int SystemDataStatisticsgammaRed; public int SystemDataStatisticsgammaGreen; public int SystemDataStatisticsgammaBlue;@Configurationpublic class SystemDataStatisticsRestTemplateConfiguration { @Bean public RestTemplate restTemplate() { RestTemplate restTemplate = new RestTemplate(); restTemplate.setInterceptors(Collections.singletonList(restTemplateInterceptor())); return restTemplate; } @Bean public RestTemplateInterceptor restTemplateInterceptor() { return new RestTemplateInterceptor(); }}//以下是系统开发所使用的依赖<component name="RunDashboard"> <option name="ruleStates"> <list> <RuleState> <option name="name" value="ConfigurationTypeDashboardGroupingRule" /> </RuleState> <RuleState> <option name="name" value="StatusDashboardGroupingRule" /> </RuleState> </list> </option> </component> <component name="SvnConfiguration"> <configuration /> </component> <component name="TaskManager"> <task active="true" id="Default" summary="Default task"> <changelist id="d84bfc67-9ad9-419c-84a2-a415c48b337e" name="Default Changelist" comment="" /> <created>1626682282684</created> <option name="number" value="Default" /> <option name="presentableId" value="Default" /> <updated>1626682282684</updated> <workItem from="1626682286598" duration="1517000" /> <workItem from="1626761710953" duration="14003000" /> <workItem from="1626829758747" duration="1712000" /> <workItem from="1627023025676" duration="5710000" /> <workItem from="1627092512638" duration="11724000" /> <workItem from="1638763750716" duration="603000" /> <workItem from="1641441556148" duration="1879000" /> <workItem from="1653017909443" duration="416000" /> <workItem from="1656906350513" duration="59000" /> <workItem from="1666668835769" duration="127000" /> </task> <servers /> </component>