# JAVA实用笔记大全

### 一mabits map方式返回值的内容

|  |
| --- |
| Xml层  <select id=*"selectSampleOperType"* resultType=*"map"*>  select \* from lis\_SampleOperType  </select> |
| Dao层  List<Map<String,String>> selectSampleOperType(); |
| Server层  List<Map<String,String>> selectSampleOperType(); |
| Serveripml层  @Override  **public** List<Map<String, String>> selectSampleOperType() {    **return** lisReqmainMapper.selectSampleOperType();  } |
| Controller层  //selectComDict  @ResponseBody  @RequestMapping(value="selectSampleOperType")  **public** Object selectSampleOperType() {  List<Map<String, String>> list = LisReqmainService.selectSampleOperType();  Map<String, Object> map = **new** HashMap<>();  map.put("code", 0);  map.put("msg", "");  map.put("data",list);  **return** map;  } |
| jsp层  ajax\_GET("/applyfor/selectComHospital",{},  **function**(data){  $.each(data.data,**function**(index,item){  $('#userSongJian').append(**new** Option(item.hosName,item.hosCode));//往下拉菜单里添加元素  })  form.render("select");//菜单渲染 把内容加载进去//成功调用后返回数据 \*/  },  **function**(s){//返回的错误信息  }) |
|  |

### 二 pdf打印获取其默认打印机

|  |
| --- |
| import javax.print.DocFlavor;  import javax.print.attribute.HashPrintRequestAttributeSet;  import javax.print.PrintService;  import javax.print.PrintServiceLookup;  import javax.print.attribute.PrintRequestAttributeSet;  PrintRequestAttributeSet pras = new HashPrintRequestAttributeSet();  DocFlavor flavor = DocFlavor.BYTE\_ARRAY.PNG;  //可用的打印机列表(字符串数组)  PrintService printService[] = PrintServiceLookup.lookupPrintServices(flavor, pras);  for(int i=0;i<printService.length;i++){          Printers[i]=printService[i].getName();  }  //当前默认打印机  PrintService PS = PrintServiceLookup.lookupDefaultPrintService(); PS.getName(); |

### 三 pdf 合并在同一个目录然后前端调用打印功能

|  |
| --- |
| **package** com.rn.util;  **import** java.io.ByteArrayOutputStream;  **import** java.io.File;  **import** java.io.FileOutputStream;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** java.net.HttpURLConnection;  **import** java.net.URL;  **import** java.text.SimpleDateFormat;  **import** java.util.ArrayList;  **import** java.util.Calendar;  **import** java.util.Date;  **import** java.util.HashMap;  **import** java.util.Iterator;  **import** java.util.List;  **import** java.util.Map;  **import** java.util.Random;      **import** com.itextpdf.text.Document;  **import** com.itextpdf.text.DocumentException;  **import** com.itextpdf.text.pdf.PdfContentByte;  **import** com.itextpdf.text.pdf.PdfImportedPage;  **import** com.itextpdf.text.pdf.PdfReader;  **import** com.itextpdf.text.pdf.PdfWriter;      **public** **class** PDFHEBING {      **public** **static** **void** main(String[] args) **throws** IOException, DocumentException {  List<String> pdfs= **new** ArrayList<String>();  pdfs.add("E:/imagefile/pdfHB/one.pdf");  pdfs.add("E:/imagefile/pdfHB/two.pdf");  SimpleDateFormat formatter = **new** SimpleDateFormat("yyyy-MM-dd");  Date date = **new** Date();  String day = formatter.format(date);  String mulu = "E:/imagefile/"+"pdfHB/"+day;  Random r = **new** Random();  String mlName = "/hebing"+ r.nextInt(99999)+".pdf";  String pdfurl = "/pdfHB/"+day;  Map<String, String> maphb = *pdfHeBing*(pdfs,mulu,mlName);  maphb.put("pdfurl",pdfurl );  System.***out***.println("文件路径:"+maphb.get("dypdf"));  System.***out***.println("文件份数:"+maphb.get("count"));  System.***out***.println("文件的页数"+maphb.get("totalPages"));  System.***out***.println("合并好的pdf后半截的url:"+pdfurl);  }  /\*\*  \* 单个 + 单个的合并pdf  \* **@throws** IOException  \* **@throws** DocumentException  \*/  **public** **static** Map<String,String> pdfHeBing(List<String> pdfs,String mulu,String mlName) **throws** IOException, DocumentException {  Map<String,String> map = **new** HashMap<String, String>();  List<PdfReader> readers = **new** ArrayList<PdfReader>(); //要合并的pdf集合  **int** totalPages = 0; //总页数  **int** count = 0; //页数  File file = **new** File(mulu);  *checkDirExists*(file);  String dypdf = mulu+mlName;  FileOutputStream out = **new** FileOutputStream(dypdf); //要合并的目录  //pdf存入集合中  **for** (String pdf : pdfs) {  **int** exist = *isExist*(pdf); //判断文件是否存在 存在则合并  **if**(exist == 1) {  count++;  PdfReader reader = **new** PdfReader(pdf);  totalPages += reader.getNumberOfPages();  readers.add(reader);  }    }    Document document = **new** Document();  PdfWriter writer = PdfWriter.*getInstance*(document, out);    document.open();  PdfContentByte cb = writer.getDirectContent();    **int** pageOfCurrentReaderPDF = 0;  Iterator<PdfReader> iteratorPDFReader = readers.iterator();    **while** (iteratorPDFReader.hasNext()) {  PdfReader pdfReader = iteratorPDFReader.next();    **while** (pageOfCurrentReaderPDF < pdfReader.getNumberOfPages()) {  document.newPage();  pageOfCurrentReaderPDF++;  PdfImportedPage page = writer.getImportedPage(pdfReader,  pageOfCurrentReaderPDF);  cb.addTemplate(page, 0, 0);  }  pageOfCurrentReaderPDF = 0;  }  out.flush();  document.close();  out.close();  map.put("dypdf", dypdf); //pdf的完整路径  map.put("totalPages", totalPages+""); //总页数  map.put("count", count+""); //页数  **return** map;  }  /\*\*  \* 判断文件是否存在 这里判断pdf  \*/  **public** **static** **int** isExist(String pdf) {  File file = **new** File(pdf);  **if** (!file.exists()) {  System.***out***.println("文件不存在");  **return** 0;  }**else** {  System.***out***.println("文件存在");  **return** 1;  }  }  /\*\*  \* 判断文件夹是否存在  \* **@param** file  \*/  **public** **static** **void** checkDirExists(File file) {  **if** (file.exists()) {  **if** (file.isDirectory()) {  System.***out***.println("目录存在");  } **else** {  System.***out***.println("同名文件存在, 不能创建");  }  } **else** {  System.***out***.println("目录不存在，创建目录");  file.mkdir();  }    //删除前一天的目录  SimpleDateFormat format = **new** SimpleDateFormat("yyyy-MM-dd");  Calendar c = Calendar.*getInstance*();  c.setTime(**new** Date());  c.add(Calendar.***DATE***, -1);  Date start = c.getTime();  String qyt= format.format(start);//前一天  System.***out***.println(qyt);  }  // 原始的备份  // /\*\*  // \* 单个 + 单个的合并pdf  // \*/  // public static void main(String[] args) throws IOException, DocumentException {  //  // PdfReader reader1 = new PdfReader("F:/test/one.pdf");  // PdfReader reader2 = new PdfReader("F:/test/two.pdf");  //  // FileOutputStream out = new FileOutputStream("F:/test/合并.pdf");  //  // Document document = new Document();  // PdfWriter writer = PdfWriter.getInstance(document, out);  //  // document.open();  // PdfContentByte cb = writer.getDirectContent();  //  // int totalPages = 0;  // totalPages += reader1.getNumberOfPages();  // totalPages += reader2.getNumberOfPages();  //  // List<PdfReader> readers = new ArrayList<PdfReader>();  // readers.add(reader1);  // readers.add(reader2);  //  // int pageOfCurrentReaderPDF = 0;  // Iterator<PdfReader> iteratorPDFReader = readers.iterator();  //  // while (iteratorPDFReader.hasNext()) {  // PdfReader pdfReader = iteratorPDFReader.next();  //  // while (pageOfCurrentReaderPDF < pdfReader.getNumberOfPages()) {  // document.newPage();  // pageOfCurrentReaderPDF++;  // PdfImportedPage page = writer.getImportedPage(pdfReader,  // pageOfCurrentReaderPDF);  // cb.addTemplate(page, 0, 0);  // }  // pageOfCurrentReaderPDF = 0;  // }  // out.flush();  // document.close();  // out.close();  // return "合并成功";  // }  /\*\*  \* 根据目录合并所有pdf  \* **@param** args  \* **@throws** IOException  \* **@throws** DocumentException  \*/  // public static void main(String[] args) throws Exception {  // PDFMergerUtility mergePdf = new PDFMergerUtility();  // String folder = "F:/test";  // System.out.println(folder);  // String destinationFileName = "F:/test/合成.pdf";  //  // String[] filesInFolder = getFiles(folder);  // for (int i = 0; i < filesInFolder.length; i++) {  // mergePdf.addSource(folder + File.separator + filesInFolder[i]);  // mergePdf.setDestinationFileName(destinationFileName);  // mergePdf.mergeDocuments();  // }  // System.out.print("合并完成");  // }  /\*\* 得到目录吓得..  \* **@param** args  \*/  **private** **static** String[] getFiles(String folder) **throws** IOException  {  File \_folder = **new** File(folder);  String[] filesInFolder;    **if**(\_folder.isDirectory())  {  filesInFolder = \_folder.list();  **return** filesInFolder;  }  **else**  {  **throw** **new** IOException("Path is not a directory");  }  }  /\*\*  \* 下方 林  \*/  /\*\*  \* 从网络Url中下载文件  \* **@param** urlStr  \* **@throws** IOException  \*/  **public** **static** **byte**[] downLoadByUrl(String urlStr) **throws** IOException {  URL url = **new** URL(urlStr);  HttpURLConnection conn = (HttpURLConnection)url.openConnection();  //设置超时间为3秒  conn.setConnectTimeout(5\*1000);  //防止屏蔽程序抓取而返回403错误  conn.setRequestProperty("User-Agent", "Mozilla/4.0 (compatible; MSIE 5.0; Windows NT; DigExt)");  //得到输入流  InputStream inputStream = conn.getInputStream();  //获取自己数组  **byte**[] getData = *readInputStream*(inputStream);  **if**(inputStream!=**null**){  inputStream.close();  }  **return** getData;  }    /\*\*  \* 从输入流中获取字节数组  \* **@param** inputStream  \* **@return**  \* **@throws** IOException  \*/  **public** **static** **byte**[] readInputStream(InputStream inputStream) **throws** IOException {  **byte**[] buffer = **new** **byte**[1024];  **int** len = 0;  ByteArrayOutputStream bos = **new** ByteArrayOutputStream();  **while**((len = inputStream.read(buffer)) != -1) {  bos.write(buffer, 0, len);  }  bos.close();  **return** bos.toByteArray();  } } |

### 四 java获取日期 当前的 前一天的 前一个月的 前一年的

|  |
| --- |
| SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");  Calendar c = Calendar.getInstance();  c.setTime(new Date());  Date end = c.getTime();  String dqrq= format.format(end);//当前日期    c.add(Calendar.DATE, -1);  Date start = c.getTime();  String qyt= format.format(start);//前一天    c.add(Calendar.MONTH, -1);  Date start = c.getTime();  String startDay = format.format(start);//前一月    c.add(Calendar.YEAR, -1);  Date start = c.getTime();  String startDay = format.format(start);//前一年 |