**jobconverter学习文档**

# 相关网站

官网：<https://github.com/sbraconnier/jodconverter>

官方文档：<https://github.com/sbraconnier/jodconverter/wiki>

Maven网址：<https://mvnrepository.com/artifact/org.jodconverter/jodconverter-local>

Openoffice: <https://www.openoffice.org>

libreoffice : <https://www.libreoffice.org/>

其他网站：

<https://blog.csdn.net/qingtian_1993/article/details/79901843>

<https://gitee.com/kekingcn/file-online-preview>

Linux 安装openOffice CentOS 7 安装openOffice :

<https://blog.csdn.net/qq_38082304/article/details/79401709>

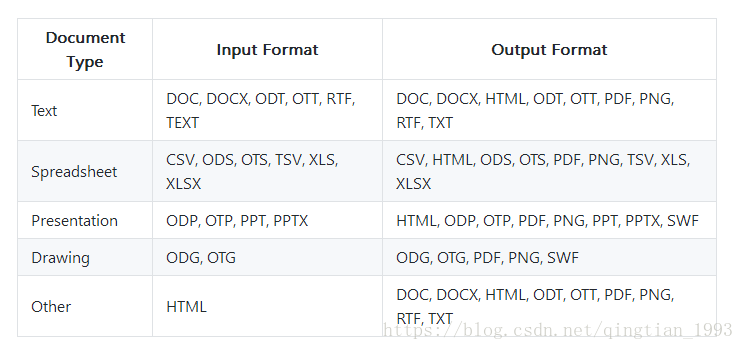
CentOS 下安装openOffice，并在Linux上实现office转PDF:

<https://blog.csdn.net/Douz_lungfish/article/details/83269967>

# 简介

JODConverter是一种Java OpenDocument转换器，能够转换不同格式的文档，它依赖于Apache OpenOffice或 LibreOffice ，它为OpenDocument和Microsoft Office提供了最好的免费导入/导出的过滤器。

JODConverter自动支持OpenOffice和LibreOffice提供的所有转换，如下：



JODConverter可以用在这几种地方：

作为一个Java类库，嵌入到你的Java应用程序中。

作为一个命令行工具，可以在你的脚本中调用。

作为一个简单的web应用，上传文档，选择转换的格式并下载转换后的版本。

# Maven Setup

添加下列的依赖到pom.xml文件中

<properties>

<jodconverter.version>4.2.0</jodconverter.version>

</properties>

<dependencies>

<dependency>

<groupId>org.jodconverter</groupId>

<artifactId>jodconverter-local</artifactId>

<version>${jodconverter.version}</version>

</dependency>

</dependencies>

Gradle Setup

添加以下信息到你的 build.gradle

ext {

jodconverterVersion = 4.2.0

}

compile "org.jodconverter:jodconverter-local:$jodconverterVersion"

当提到OpenOffice.org（简称为OOo），这通常可以理解为任何来自OOo的办公套件，比如Apache OpenOffice或 LibreOffice 。

# 环境要求

为了使用JODConverter你需要有以下配置：

Java环境（JDK1.7或更高版本）。

Apache OpenOffice或 LibreOffice ，最新且稳定的版本。

任何可以使用Java和Apache OpenOffice/LibreOffice的操作系统（Windows, Linux, Mac OS X, Solaris, FreeBSD, etc.），请参考你计划用的操作系统安装OOo发行版的系统需求：

Apache OpenOffice system requirements.

LibreOffice system requirements.

为了运行web项目，你需要一个servlet容器，比如Apache Tomcat。

# 配置信息

JODConverter所使用的时间值均为毫秒

使用LocalOfficeManager需要设置一些参数。使用JODConverter时，会采用默认配置。虽然这些配置信息不一定是最好的，但是他们更有可能被选中使用。

protNumbers 和 pipeNames

OOo 进程间的通信可以使用TCP sockets 或者 命名管道。命名管道具有不占用TCP端口的优势（存在安全隐患），并且可能会更快一些。然而命名管道需要由JVM加载本地库，这意味着必须在java.library.path中设置路径，这就是为什么pipeNames不是默认配置的原因了。由于平台的不同，添加到java/library.path的路径也不同，但它应该是包含了libjipe的OOo安装目录。

On Linux it’s e.g.: java -Djava.library.path=/opt/openoffice.org/ure/lib

On Windows it’s e.g.: java “-Djava.library.path=C:\Program Files (x86)\OpenOffice 5\program”

Default：TCP socket，on port 2002，

// This example will use 4 TCP ports, which will cause

// JODConverter to start 4 office processes when the

// OfficeManager will be started.

OfficeManager officeManager =

LocalOfficeManager.builder()

.portNumbers(2002, 2003, 2004, 2005)

.build();

officeHome

该属性应该设置为OOo的安装目录。若没有配置，则创建OfficeManager时会自动检测，从LibreOffice（优先于OpenOffice）的最新版本开始。

// This example will force JODConverter to use the OpenOffice 4

// installation that can be found using the specified path.

OfficeManager officeManager =

LocalOfficeManager.builder()

.officeHome("D:\\Program Files (x86)\\OpenOffice 4")

.build();

processManager

JODConverter开始处理一个office进程时，就需要使用到进程管理器。当它开始进行这项工作，就必须要检索该进程的PID，以便在需要时能够kill it。默认情况下，JODConverter会根据JODConverter运行的操作系统来选择最佳的进程管理器。但是，在classpath中发现的，且继承了ProcessManager接口的进程管理器均可以被使用。

// This example will create an instance of the com.example.foo.CustomProcessManager

// class that will be used by the created OfficeManager.

OfficeManager officeManager =

LocalOfficeManager.builder()

.processManager("com.example.foo.CustomProcessManager")

.build();

workingDir

该属性用来设置office临时文件配置目录。每个office进程启动时，一个文件配置目录将会被创建。当使用InputStream/OutputStream来转换时，这个目录也会被创建。默认由指定的路径为java.io.tmpdir

templateProfileDir

为了避免进程被中断或者用户使用了另一个OOo实例，LocalOfficeManager会为OOo进程创建一个临时配置文件目录。使用这个属性，你可以提供一个包含个性化设置的临时配置文件目录。OfficeManager会将以其为模板，来生成临时配置文件目录。所以当我们创建新的OOo实例时，都会使用相同的配置。默认情况虾，这个临时的配置文件由OOo使用默认配置来创建，并且其依赖于 -nofirststartwizard这个命令选项。

killExistingProcess

该属性能够指定，当一个包含相同connection string的office进程启动，是否杀死一个已经存在的office进程。默认为true.

processTimeout

当尝试调用一个office进程时（开始/中止），该属性可以设置超时时间，单位为毫秒。默认为120000（2 minutes）

processRetryInterval

每当尝试调用一个office进程时（开始/中止）的间隙，可用该属性设置延迟，单位为毫秒。默认为250（0.25 seconds）

taskExecutionTimeout

该属性设置执行一个任务的最大时间，若超过这个时间任务仍未执行，则当前任务被中止且执行下个任务。默认为12000（两分钟）

maxTasksPerProcess

该属性设置一个office进程在重启之前所能执行的最大任务数。默认为200个。

disableOpengl

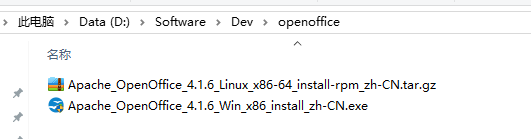
当启动一个新的office进程时（在LibreOffice的情况下），该属性能够指定是否禁止OpenGL。如果OpenGL已经根据office进程使用的用户配置禁用，那么将不会执行任何操作。如果该属性改变，那么office进程必须重启。如果LO进程奔溃，那么你可以尝试测试该属性。默认为false

taskQueueTimeout

该属性用来设置一个任务在转换队列中的最大生存时间。如果等待时间超过最大生存时间或者有OfficeException异常抛出，则任务将会从队列中移除。默认为30000（30 seconds）

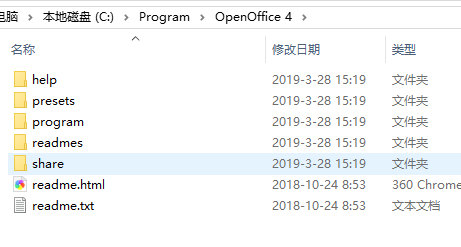
# OpenOffice安装

下载地址：<https://www.openoffice.org/zh-cn/download/>



## window

下载后，下一步下一步安装即可



启动：

命令行定位到program文件夹下

cd C:\Program\OpenOffice 4\program

soffice -headless -accept="socket,host=127.0.0.1,port=8100;urp;" -nofirststartwizard &

查看端口号

 netstat -ano|findstr "8100"

## centos7

1. 下载openOffice的安装包

[下载Apache\_OpenOffice\_4.1.6\_Linux\_x86-64\_install-rpm\_zh-CN.tar.gz](https://blog.csdn.net/qq_38082304/article/details/Apache_OpenOffice_4.1.4_Linux_x86-64_install-rpm_zh-CN.tar.gz)

2.将openOffice解压出来

tar -xzvf  Apache\_OpenOffice\_4.1.6\_Linux\_x86-64\_install-rpm\_zh-CN.tar.gz

3. 解压后会有一个文件夹生成  zh-CN 然后进入该文件夹的 RPMS 然后执行

cd zh-CN/ RPMS

yum localinstall \*.rpm

4.首先进入desktop-integration目录下，然后运行yum localinstall openoffice4.1.4-redhat-menus-4.1.4-9788.noarch.rpm

 cd desktop-integration/

yum localinstall openoffice4.1.6-redhat-menus-4.1.6-9788.noarch.rpm

5.启动动openoffice服务：cd /opt/openoffice4/program，执行：soffice -headless -accept="socket,host=127.0.0.1,port=8100;urp;"  -nofirststartwizard &

    错误解决办法：

     1）yum install libXext.x86\_64：libXext.so No such file or directory错误

     2）yum groupinstall "X Window System" ：no suitable windowing system found错误

6.服务启动验证：

  netstat  -nlp | grep 8100  //linux

  错误解决办法：yum install net-tools ：netstat找不到

# Java调用

在你的Java程序中使用JODConverter是非常简单的。接下来的例子展示了执行WORD文档转换成PDF主要代码：

File inputFile = **new** File("D:/tmp/1.docx");

File outputFile = **new** File("D:/tmp/1.pdf");

LocalOfficeManager officeManager = LocalOfficeManager.*builder*().portNumbers(8100).officeHome("C:\\Program\\OpenOffice 4").build();

**try** {

officeManager.start();

LocalConverter.*builder*()

.officeManager(officeManager)

.build()

.convert(inputFile)

.to(outputFile)

.execute();

} **catch** (OfficeException e) {

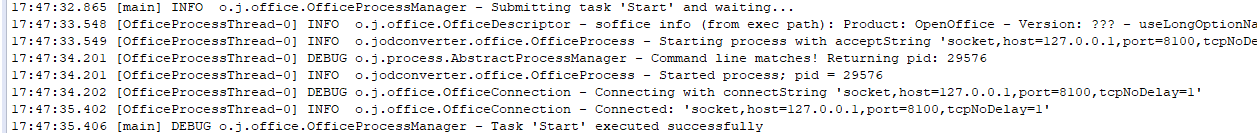
e.printStackTrace();

}**finally** {

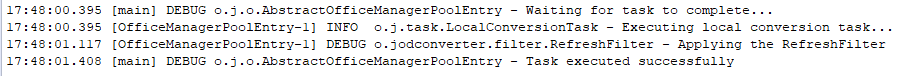
OfficeUtils.*stopQuietly*(officeManager);

}

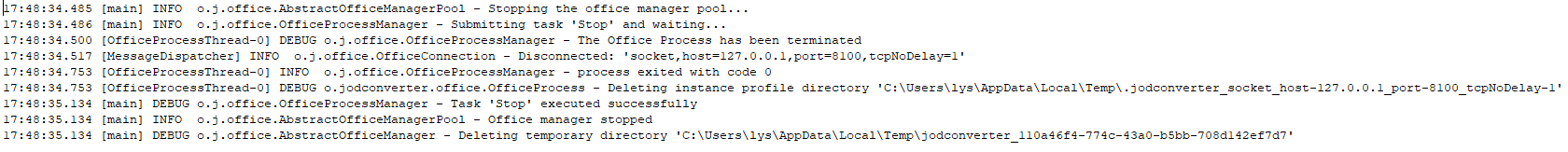
第一阶段 开启officeManager



第二阶段 转化



第三节点 关闭officeManager



参考：

<https://github.com/sbraconnier/jodconverter/blob/master/jodconverter-local/src/test/java/org/jodconverter/LocalConverterTest.java>

如果想转换成其他格式，可以更改文件名，转换的格式将根据文件的扩展名来决定。比如，将EXCEL文件转换成OpenDocument Spreadsheet格式：

File inputFile = **new** File("D:/tmp/1.docx");

File outputFile = **new** File("D:/tmp/1.txt");

LocalOfficeManager officeManager = LocalOfficeManager.*builder*().portNumbers(8100).officeHome("C:\\Program\\OpenOffice 4").build();

**try** {

officeManager.start();

LocalConverter.*builder*()

.officeManager(officeManager)

.build()

.convert(inputFile)

.to(outputFile)

.execute();

} **catch** (OfficeException e) {

e.printStackTrace();

}**finally** {

OfficeUtils.*stopQuietly*(officeManager);

}

如果你想使用IO流来替代文件，这个完全没有问题！你仅需要通知JODConverter输入流和输出流的文件格式即可。比如，将EXCEL文件转换成OpenDocument Spreadsheet格式：

File inputFile = **new** File("D:/tmp/1.docx");

File outputFile = **new** File("D:/tmp/1.pdf");

**try** {

LocalOfficeManager officeManager = LocalOfficeManager.*builder*().portNumbers(8100).officeHome("C:\\Program\\OpenOffice 4").build();

InputStream inputStream = **new** FileInputStream(inputFile);

OutputStream outputStream = **new** FileOutputStream(outputFile);

LocalConverter.*builder*()

.officeManager(officeManager)

.build()

.convert(inputStream)

.as(DefaultDocumentFormatRegistry.***DOCX***)

.to(outputStream)

.as(DefaultDocumentFormatRegistry.***PDF***)

.execute();

} **catch** (Exception e) {

e.printStackTrace();

}**finally** {

OfficeUtils.*stopQuietly*(*officeManager*);

}

十分简单，是吧？以上的例子已经能够满足绝大部分开发所需了。

从功能上来说，每开始一个转换，就创建一个新的连接，这是可以接受的。但是从性能上来说，这并不是最好的选择。如果你想把JODConverter整合进你的web应用中，那么你应该初始化一个单例的OfficeManager实例，当app启动时就初始化该实例，app关闭时则停止该实例。

# web应用

最重要的理念便是，在启动web应用时（使用servlet监听器，Spring上下文或者你选择的其他等效的框架），应该使用一个单例的OfficeManager实例，停止该实例当web应用停止时，并且让所有请求共享该实例。让OfficeManager管理多线程。

# 使用过滤器

在转换一个文档时，JODConverter允许你将加载的资源文档转换成目标文档前修改它。原文件不会被修改，只会修改加载的。当然，你所能做的修改是基于OOo能做的。使用JODConverter进行转换和使用OOo执行修改操作是一致的。

这是一个PageSelectorFilter实例，这个例子只转换了给定源文档的第二页。

final File inputFile = new File("document.rtf");

final File outputFile = new File("document.html");

final PageSelectorFilter selectorFilter = new PageSelectorFilter(2);

LocalConverter

.builder()

.filterChain(selectorFilter)

.build()

.convert(inputFile)

.to(outputFile)

.execute();

在每次转换中，你可以使用一个以上的过滤器。而且，当目标文件不是PDF时才需要使用这样的过滤器。事实上，将文件转换为pdf，最好使用自定义存储属性：

File inputFile = new File("document.rtf");

File outputFile = new File("document.pdf");

Map<String, Object> filterData = new HashMap<>();

filterData.put("PageRange", "2");

Map<String, Object> customProperties = new HashMap<>();

customProperties.put("FilterData", filterData);

LocalConverter

.builder()

.storeProperties(customProperties)

.build()

.convert(inputFile)

.to(outputFile)

.execute();

JODConverter提供了一些过滤器，但是你可以自己实现任何你需要的过滤器（你的过滤器必须继承Filter接口，并负责在过滤器链中调用下一个过滤器）。

# 完整例子

Pom.xml

<dependency>

<groupId>org.jodconverter</groupId>

<artifactId>jodconverter-local</artifactId>

<version>4.2.2</version>

</dependency>

**import** java.io.File;

**import** java.io.InputStream;

**import** java.io.OutputStream;

**import** java.util.Properties;

**import** org.jodconverter.LocalConverter;

**import** org.jodconverter.document.DocumentFormat;

**import** org.jodconverter.office.LocalOfficeManager;

**import** org.jodconverter.office.OfficeException;

**import** org.jodconverter.office.OfficeUtils;

/\*\*

\* jobconverter工具类

\* **@description**:

\* **@createTime**: 2019年3月29日下午2:46:51

\* **@author**：lin.yisong

\* **@version**：1.0

\*/

**public** **class** JobconverterUtil {

**private** **static** LocalOfficeManager *officeManager* = **null**;

**static** Properties *applicationPro* = **new** Properties();

/\*\*

\* 取得OfficeManager，做成单例目的：如果是linux系统部分服务器，连接会占用大量时间，如果是web系统，连接一次后续可直接使用该连接

\* **@createTime**: 2019年3月29日 下午2:47:40

\* **@author**: lin.yisong

\* **@return**

\*/

**public** **static** LocalOfficeManager getOfficeManager() {

**if** (*officeManager* == **null**) {

**try** {

*applicationPro*.load(JobconverterUtil.**class**.getClassLoader().getResourceAsStream("application.properties"));

Properties prop = System.*getProperties*();

String os = prop.getProperty("os.name");

String officeHome = **null**;

**if** (os != **null** && os.toLowerCase().indexOf("linux") > -1) {

officeHome = *applicationPro*.get("officeHome.linux").toString();

}**else**{

officeHome = *applicationPro*.get("officeHome.window").toString();

}

*officeManager* = LocalOfficeManager.*builder*().portNumbers(8100).officeHome(officeHome).build();

*officeManager*.start();

} **catch** (Exception e) {

e.printStackTrace();

}

}

**return** *officeManager*;

}

/\*\*

\* 将文件inputFile转化成文件outputFile

\* **@createTime**: 2019年3月29日 下午2:49:46

\* **@author**: lin.yisong

\* **@param** inputFile

\* **@param** outputFile

\* **@throws** OfficeException

\*/

**public** **static** **void** convert(File inputFile,File outputFile) **throws** OfficeException{

LocalConverter.*builder*()

.officeManager(*getOfficeManager*())

.build()

.convert(inputFile)

.to(outputFile)

.execute();

}

/\*\*

\* 将文件路径inputFilePath下的文件转化成文件路径outputFilePath下的文件

\* **@createTime**: 2019年3月29日 下午2:50:29

\* **@author**: lin.yisong

\* **@param** inputFilePath

\* **@param** outputFilePath

\* **@throws** OfficeException

\*/

**public** **static** **void** convert(String inputFilePath,String outputFilePath) **throws** OfficeException{

*convert*(**new** File(inputFilePath), **new** File(outputFilePath));

}

/\*\*

\* 使用IO流的方式转化

\* **@createTime**: 2019年3月29日 下午2:51:10

\* **@author**: lin.yisong

\* **@param** inputStream

\* **@param** inputType

\* **@param** outputStream

\* **@param** outType

\* **@throws** OfficeException

\*/

**public** **static** **void** convert(InputStream inputStream,DocumentFormat inputType ,OutputStream outputStream,DocumentFormat outType) **throws** OfficeException{

LocalConverter.*builder*()

.officeManager(*getOfficeManager*())

.build()

.convert(inputStream)

.as(inputType)

.to(outputStream)

.as(outType)

.execute();

}

**public** **static** **void** main(String[] args) {

**if**(args==**null**||args.length<2){

System.***out***.println("请输入参数");

}

String inputFilePath = args[0];

String outputFilePath = args[1];

// File inputFile = new File("D:/tmp/1.docx");

// File outputFile = new File("D:/tmp/1.pdf");

**try** {

JobconverterUtil.*convert*(inputFilePath, outputFilePath);

// InputStream inputStream = new FileInputStream(inputFile);

// OutputStream outputStream = new FileOutputStream(outputFile);

// JobconverterUtil.convert(inputStream,DefaultDocumentFormatRegistry.DOCX, outputStream,DefaultDocumentFormatRegistry.PDF);

} **catch** (Exception e) {

e.printStackTrace();

}**finally** {

OfficeUtils.*stopQuietly*(*officeManager*);

}

}

}

application.properties

officeHome.window=C:/Program/OpenOffice 4

officeHome.linux=/opt/openoffice4/