iSignature_PDF_API

V6.0.0.598

中 间 件 技 术 白 皮 书

江西金格科技股份有限公司

地址： 江西南昌高新技术开发区火炬大街 579 号绿悦科技大厦 15 楼
邮编： 330096
网址： http://www.kinggrid.com
电话： *0791-82221588*
服务： *400-6776-800*

目录

1 引言

1.1. 模块概述

产品模块为可选件，按需选购提供。

1.1.1 文档生成

XML 模板生成 *PDF*

1.1.2 文档加工

文档合并、文档抽取、插入附件、生成书签

1.1.3 文档水印

文本水印、图片水印、文字跨页水印

1.1.4 防伪码

动态生成光学防伪水印

1.1.5 电子签章

普通印章、骑缝章（包含双侧对开）、删除印章、获取印章信息

1.1.6 数字签名

域数字签名（骑缝章、双侧对开）、不显示域数字签名、验证签名有效性、获取证书信息

1.1.7 文档管控

口令加密、口令解密、证书加密、文档安全性控制（打印、复制控制）、文档说明设置

1.2. 开发平台

开发平台：windows XP、Eclipse 3.4

编程语言：java

编译版本：JDK5

1.3. 运行环境

软件环境：JDK1.5 及以上的版本

操作系统：windows、linux、unix，支持 64 位操作系统

1.4. 性能并发

JDK 内存	文档大小/页数	支持的并发能力	备注
256M	*10M* 内/*600* 页内	*40-100*	指单文档的页数
512M	*10M* 内/*600* 页内	*70-150*	指单文档的页数

说明：

1. 并发能力与文档的大小和页数有关，单文档越大页数越多消耗的内存越多，并发能力就越小，反之亦然。
2. 如需要更优的性能（提高并发能力或响应时间），请联系金格科技，金格科技可根据客户项目实际应用场景，为客户提供一对一专项有偿服务。

2 接口说明

类名：

com.kinggrid.pdf. KGPdfHummer

说明:

接口详细说明里面红色的框内版本【*V4.0.0.208*】表示从这个版本开始支持这个接口。

2.1 接口列表

名称	描述	备注
createSignature	创建域数字签名及文档编辑实例	
createInstance	创建编辑文档实例	
setCertificate	设置证书	
setOcspValid	设置是否在线验证证书状态	【*6.0.0.546*】
setTSAClient	设置时间戳信息	
setPagen	设置需要操作的页码，*0* 表示所有页	
signature	域数字签名接口。	
addExecute	添加任务接口	
doSignature	执行任务，要数字签名。不需要执行 *close* 方法。	
doExecute	执行任务，不包含数字签名	
openPdfValidateSignature	打开 *PDF* 时验证数字签名	
generateAntiCopyImage	防复印水印	
generateOpticImage	生产光学防伪图片	【*V4.0.0.256*】
getSignatureCertificates	获取所有数字签名的公钥证书	
getSignatureSM2Details	获取域签名双证书模式中 *SM2* 签名信息	【*V5.2.0.468*】
verifySignatures	验证数字签名是否有效	
getMetaInfo	获取属性说明	
setMetaInfo	设置属性说明	
encryptPdf	设置安全性：口令加密、证书加密	

decryptPdf	口令解密	
getKGPdfHummerUtils	辅助类	
getNumberOfPages	获取 *PDF* 文档页数	
close	关闭实例，编辑 *PDF* 文档时必须关闭实例。	

2.2 接口详细说明

2.2.1 createSignature

1. ***public static KGPdfHummer createSignature(String fileName, byte[] ownerPassword, boolean partial, OutputStream os, File tmpDic, boolean append)***

功能说明：

创建域数字签名及文档编辑实例。

参数说明：

fileName：需要签名的文档路径。

ownerPassword：文档打开密码，没有传 *null*。

partial：需要时再加载文档内容。

os：签名后的文档。

tmpDic：签名时文档保存临时目录，*null* 时文档保存在内存中。

append：追加模式。

返回值：

KGPdfHummer 实例。

2. ***public static KGPdfHummer createSignature(InputStream in, byte[] ownerPassword, OutputStream os, File tmpDic, boolean append)***

功能说明：

创建域数字签名及文档编辑实例。

参数说明：

in：需要签名的文档流。

ownerPassword：文档打开密码，没有传时 *null*。

os：签名后的文档。

tmpDic：签名时文档保存临时目录，*null* 文档保存在内存中。

append：追加模式。

返回值：

KGPdfHummer 实例。

2.2.2 createInstance

1. ***public static KGPdfHummer createInstance(String fileName, byte[] ownerPassword, boolean partial, OutputStream os, boolean append)***

   功能说明：

   创建文档编辑实例。

   参数说明：

   filename：文档路径。

   ownerPassword：文档打开密码，没有传 *null*。

   partial：需要时再加载文档内容。

   os：签名后的文档。

   append：追加模式。

   返回值：

   KGPdfHummer 实例。

2. ***public static KGPdfHummer createInstance(InputStream in, byte[] ownerPassword, OutputStream os, boolean append)***

   功能说明：

   创建文档编辑实例。

   参数说明：

   in：文档流。

   ownerPassword：文档打开密码，没有传 *null*。

   os：签名后的文档。

   append：追加模式。

   返回值：

   KGPdfHummer 实例。

3. ***public static KGPdfHummer createInstance(String fileName, byte[] ownerPassword, boolean partial)***

   功能说明：

   创建读取文档信息实例。

   参数说明：

   filename：文档路径。

   ownerPassword：文档打开密码，没有传 *null*。

   partial：需要时再加载文档内容。

   返回值：

KGPdfHummer 实例。

4. ***public static KGPdfHummer createInstance(InputStream in, byte[] ownerPassword,)***

功能说明：

创建读取文档信息实例。

参数说明：

in：文档流。

ownerPassword：文档打开密码，没有传 *null*。

返回值：

KGPdfHummer 实例。

2.2.3 setCertificate

1. ***public void setCertificate(InputStream cert, String keystorePwd, String keyPwd)***

功能说明：

设置证书,证书库中只有一个证书*(*包含私钥：*PFX* 格式*)*。证书库密码和证书密码一般情况下是一样的。

参数说明：

cert ： 带私钥证书。

keystorePwd ： 证书库密码。

key ： 证书密码。

返回值：

void

2. ***public void setCertificate(Certificate cert, SignatureInter signatureInter)***

功能说明：

对接加密卡（*smart card*）、加密槽。

参数说明：

cert ： 带公钥证书。

signatureInter ： 加密卡、加密槽回调接口。

返回值：

Void

3. ***public void setCertificate (DigitalSignature digitalSignature)***

功能说明：

对接加密卡（*smart card*）、加密槽。

参数说明：

signatureInter ： 加密卡、加密槽回调接口。

返回值：

void

2.2.4 setOcspValid

1. ***public void setOcspValid (boolean ocspValid)***

功能说明：

设置是否在线验证证书状态。

参数说明：

ocspValid：验证

返回值：

void

2.2.5 setTSAClient

1. ***public void setTSAClient(String url, String username, String password)***

功能说明：

设置时间戳。

参数说明：

url ：时间戳地址，测试地址：*http://timestamp.comodoca.com/rfc3161*。

username ：用户名，不需要用户名和密码时传 *null*。

password ：密码

返回值：

void

2.2.6 setPagen

2. ***public void setPagen(int pagen)***

功能说明：

设置需要操作的页码。

参数说明：

pagen：页码，*0* 表示所有页

返回值：

void

2.2.7 signature

1. *public void signature(String fieldName, Rectangle rectangle, Image image)*

功能说明：

域数字签名。通过创建矩形区域确定位置。

参数说明：

fieldName ： 域名称，唯一性不能重复。

rectangle ：矩形区域。

image ：印章图片。

返回值：

void

2.2.8 addExecute

1. *public void addExecute(KGExecute execute)*

功能说明：

添加任务接口。

参数说明：

无。

返回值：

void

2.2.9 doSignature

1. *public void doSignature()*

功能说明：

执行任务，要有数字签名任务。

参数说明：

无。

返回值：

void

2.2.10 doExecute

1. *public void doExecute()*

功能说明：

执行任务，没有数字签名任务。

参数说明：

无。

返回值：

void

2.2.11 openPdfValidateSignature

1. ***public void openPdfValidateSignature()***

功能说明：

打开 *PDF* 时验证数字签名。

参数说明：

无。

返回值：

void

2.2.12 generateAntiCopyImage

1. ***public Image generateAntiCopyImage(String text, double waterMarkWidth, double watermarkHeight, int fontChroma, int rowChroma, int columnChroma, String fontName)***

功能说明：

防复印水印。

参数说明：

text：水印文字内容

waterMarkWidth：水印宽度

watermarkHeight：水印宽度

fontChroma：字浓度率，*0-60*

rowChroma：行浓度率，*0-10*

columnChroma：列（宽）浓度率，*0-10*

fontName：字体名称。

返回值：

Image：防复印水印

2.2.13 generateOpticImage

1. ***public Image generateOpticImage (String text, double width,double height, int accuracy)***

功能说明：

生产光学防伪图片。

参数说明：

text：文字内容

width：宽度

height：高度

accuracy：图片精度*(1-6*，通常设置为 *3)*

返回值：

Image：光学防伪图片

2. ***public Image generateOpticImage (String text, double width,double height, int accuracy, boolean vertical)***

功能说明：

生产光学防伪图片。

参数说明：

text：文字内容

width：宽度

height：高度

accuracy：图片精度*(1-6*，通常设置为 *3)*

vertical：自定义文字是否垂直时可见

返回值：

Image：光学防伪图片

2.2.14 getSignatureCertificates

1. ***public List<Certificate> getSignatureCertificates()***

功能说明：

获取所有数字签名的公钥证书。

参数说明：

无

返回值：

List<Certificate> ：证书列表

2.2.15 getSignatureSM2Details

2. ***public List<Map<String, String>> getSignatureSM2Details()***

功能说明：

获取域签名双证书模式中 *SM2* 签名信息。

参数说明：

无

返回值：

List<Map<String, String>> ：*SM2* 签名详细信息

2.2.16 verifySignatures

1. ***public String verifySignatures()***

功能说明：

验证数字签名是否有效。

参数说明：

无

返回值：

-1：文档不存在数字签名。

0：至少有一个签名是无效的。

1：所有签名有效。

2：所有签名有效，最后一次签名后追加了内容。

2.2.17 getMetaInfo

1. ***public Map getMetaInfo()***

功能说明：

获取属性说明。

参数说明：

无

返回值：

Map

map 关键字：

关键字	说明

Title	标题
Author	作者
Subject	主题
Keywords	关键字
Creator	创建程序

2.2.18 setMetaInfo

1. ***public void setMetaInfo(Map map)***

功能说明：

设置属性说明。

参数说明：

map：属性说明

map 关键字：

关键字	说明
Title	标题
Author	作者
Subject	主题
Keywords	关键字
Creator	创建程序

返回值：

void

2.2.19 encryptPdf

1. ***public void encryptPdf(String user, String owner, int permission)***

功能说明：

口令加密。

参数说明：

user：打开 **PDF** 文档时的密码

owner：修改权限设置的密码

permission：权限控制值，文档不允许任何操作值为：**0**。

permission 值：

PdfWriter.ALLOW_PRINTING（打印）

PdfWriter.ALLOW_MODIFY_CONTENTS（更改文档）

PdfWriter.ALLOW_COPY（内容复制或提取、提取内容用于辅助工具）

PdfWriter.ALLOW_MODIFY_ANNOTATIONS（注释）

PdfWriter.ALLOW_FILL_IN（填写表单域）

PdfWriter.ALLOW_SCREENREADERS（启用内容辅助工具）

*PdfWriter.ALLOW_ASSEMBLY(*文档组合*)*

PdfWriter.ALLOW_DEGRADED_PRINTING（弱打印，打印效果不佳）

也可以组合使用：

PdfWriter.ALLOW_DEGRADED_PRINTING | PdfWriter.ALLOW_DEGRADED_PRINTING

返回值：

void

2. ***public void encryptPdf(Certificate certificate, int[] permission)***

功能说明：

证书加密，需要导入相应私钥证书到 **PC** 机（电脑），才能打开 **PDF** 文档。

参数说明：

certificate：公钥证书

permission：权限控制值，文档不允许任何操作值为：**new int[]{0}**

permission 数组的值：

PdfWriter.ALLOW_PRINTING

PdfWriter.ALLOW_MODIFY_CONTENTS

PdfWriter.ALLOW_COPY

PdfWriter.ALLOW_MODIFY_ANNOTATIONS

PdfWriter.ALLOW_FILL_IN

PdfWriter.ALLOW_SCREENREADERS

PdfWriter.ALLOW_ASSEMBLY

PdfWriter.ALLOW_DEGRADED_PRINTING

返回值： *void*

2.2.20 decryptPdf

1. ***public void decryptPdf (OutputStream fos)***

功能说明：

口令解密。

参数说明：

os：解密后的文档

返回值：

void

2.2.21 getKGPdfHummerUtils

1. ***public KGPdfHummerUtils getKGPdfHummerUtils***

功能说明：

辅助类。

参数说明：

无

返回值：

`KGPdfHummerUtils`

2.2.22 getNumberOfPages

1. ***public int getNumberOfPages***

功能说明：

获取 **`PDF`** 文档页数。

参数说明：

无

返回值：

int

2.2.23 close

1. ***public void close()***

功能说明：

关闭实例。

参数说明：

无

返回值：

void

3 任务接口说明

3.1 数字签名接口

3.1.1 添加数字签名

类名：*com.kinggrid.pdf.executes. PdfSignature4KG*

构造方法：

1. ***PdfSignature4KG(String keyPath, int sealNumb, String sealPwd)***

   .key 文件对接

2. ***PdfSignature4KG(String url,KGServerTypeEnum type, String keySN, String password,String signName)***

   签章服务器对接

   url：签章服务器地址 *http://127.0.0.1:8089/iSignatureServer/OfficeServer.jsp*

   type：*KGServerTypeEnum.NET* 网络版

   KGServerTypeEnum.CA0

   KGServerTypeEnum.CA2

   KGServerTypeEnum.CA3

   KGServerTypeEnum.AUTO 【*V4.0.0.254*】自动配置版本，*V7* 及 *V8* 的签章服务器都支持。

   keySN：密钥盘序列号或者用户编码，通过 *setQueryBy* 接口确定获取印章方式。

   password：用户密码，*CA0* 版才会验证密码。

   signName：印章名称。

3. ***PdfSignature4KG(String url,KGServerTypeEnum type, String keySN, String password,int index)***

   签章服务器对接

   url：签章服务器地址 *http://127.0.0.1:8089/iSignatureServer/OfficeServer.jsp*

   type：*KGServerTypeEnum.NET* 网络版

   KGServerTypeEnum.CA0

   KGServerTypeEnum.CA2

   KGServerTypeEnum.CA3

   KGServerTypeEnum.AUTO 【*V4.0.0.254*】自动配置版本，*V7* 及 *V8* 的签章服务器都支持。

keySN：密钥盘序列号或者用户编码，通过 *setQueryBy* 接口确定获取印章方式。

password：用户密码，*CA0* 版才会验证密码。

index：当前用户印章序号，从 *0* 开始。

属性：

```
/**印章跟随文档旋转,默认不跟随自动翻转*/
private boolean rotation;
/**骑缝章，只对第一个印模做数字签名，其他以水印的方式添加 */
private boolean qfzOnlyFirstIsSig;
```
【*V4.0.0.240*】

方法：

1. ***public void setText(String text)***

   文本定位，所有匹配的文本都会加盖印章。

   注意：当文本中包含空格且定位不到时，需要把空格去掉。在直接拷贝 *PDF* 文档中的文本，由于字体的因素，文本会包含空格，实际上是没有空格的，所以会出现定位不到的情况。

2. ***public void setText(String text, boolean onceFindReturn)***

   文本定位

   text *:* 需要定位的文本内容

   onceFindReturn *:*

   true : 只在第一个匹配的文本上加盖印章

   false : 所有匹配的文本都加盖印章

3. ***public void setTextOffset(float offsetX, float offsetY)***

   文本定位时，设置相对文本位置的偏移量，*PDF* 左下角为坐标原点。

   offsetX *: X* 轴偏移量

   offsetY *: Y* 轴偏移量

4. ***public void setXY(float x, float y)***

   坐标定位

5. ***public void setXY(XYType xyType,float x, float y)*** 【*V5.1.0.392*】

   坐标定位，以 *xyType* 为坐标原点，内容显示区为第一象限。

   xyType : 左下角 *XYType .LOWER_LEFT*

   右下角 *XYType.LOWER_RIGHT*

   右上角 *XYType .UPPER_RIGHT*

   左上角 *XYType .TOP_LEFT*

6. ***public void qfz(int number,KGQfzModeEnum qfzMode,int yDistanceOrigin)***

   骑缝章

   number : *number* 页均分一个章，循环分配。模（余数：*mod*）等于 *1*，后面

number+1 页均分一个章；否则 *mod* 页均分一个章。*2<=number<=32*。

qfzMode: *KGQfzModeEnum.ALLPAGE* 所有页

KGQfzModeEnum.ODDPAGE 奇数页

KGQfzModeEnum.EVENPAGE 偶数页

yDistanceOrigin: 印章距离底部的高度

7. ***public void qfzBilateralOff(int yDistanceOrigin)***

骑缝章，双侧对开

8. ***public void setStartPage (int startPage)***

骑缝章，设置盖章的开始页

9. ***public void setEndPage (int endPage)***

骑缝章，设置盖章的结束页

10. ***public void addImgText(KGTextInfo textInfo)***

印章上添加文本信息，可连续调用该方法。

11. ***public void notInvisible()***

签名域不可见。

12. ***public void forceValidatePwd()*** ***【V4.0.0.268】***

与签章服务器对接获取印章，设置强制验证用户密码。默认只验证 *0* 版用户密码。

13. ***public void setCertificationLevel(int certificationLevel)【V4.0.0.268】***

设置数字签名安全等级

certificationLevel：*0 Approval signature*

1 Author signature, no changes allowed

2 Author signature, form filling allowed

3 Author signature, form filling and annotations allowed

默认值为 *0*。

14. ***public void setQueryBy(KGQueryByEnum queryBy)【V4.0.0.292】***

设置通过“密钥盘序列号”或者“用户编码”获取印章

queryBy：密钥盘序列号（*KGQueryByEnum.KEYSN*）

用户编码（*KGQueryByEnum.USERCODE*），签章服务器需要 *768* 及以上版本支持

15. ***public void setBlendMode(PdfName blendMode)【V5.1.0.410】***

设置印章图片显示的复合模式。

blendMode：

PdfGState.BM_NORMAL：印章遮住文字，印章图片底色必须透明。

PdfGState.BM_MULTIPLY：印章不遮住文字。默认值

3.1.2 删除数字签名

1. **类名**：*com.kinggrid.pdf.executes. DeleteSignature*

   属性：无

   说明：删除全部数字签名。

2. **类名**：*com.kinggrid.pdf.executes. DeleteSignature*

   方法：*public void deleteLastOne(String keysn)*

   *keysn:*用户 *key* 序列号

   说明：删除最后一个印章，传入 *keysn* 参数，若最后一个印章中的 *keysn* 与传入的参数相匹配则删除。

3.1.3 数字证书信息

参考 *KGPdfHummer* 的 *getSignatureCertificates* 接口。

3.2 电子签章接口

3.2.1 添加电子签章

类名：*com.kinggrid.pdf.executes. PdfElectronicSeal4KG*

构造方法：

1. ***PdfElectronicSeal4KG (String keyPath, int sealNumb, String sealPwd)***

   .key 文件对接

2. ***PdfElectronicSeal4KG (String url,KGServerTypeEnum type, String keySN, String password,String signName)***

   签章服务器对接

   url：签章服务器地址 *http://127.0.0.1:8089/iSignatureServer/OfficeServer.jsp*

   type：*KGServerTypeEnum.NET* 网络版

   KGServerTypeEnum.CA0

   KGServerTypeEnum.CA2

   KGServerTypeEnum.CA3

   KGServerTypeEnum.AUTO 【*V4.0.0.254*】自动配置版本，*V7* 及 *V8* 的签章服务器都支持。

   keySN：密钥盘序列号或者用户编码，通过 *setQueryBy* 接口确定获取印章方式。

   password：用户密码，*CA0* 版才会验证密码。

signName：印章名称。

3. ***PdfElectronicSeal4KG (String url,KGServerTypeEnum type, String keySN, String password,int index)***

   签章服务器对接

   url：签章服务器地址 *http://127.0.0.1:8089/iSignatureServer/OfficeServer.jsp*

   type：*KGServerTypeEnum.NET* 网络版

   KGServerTypeEnum.CA0

   KGServerTypeEnum.CA2

   KGServerTypeEnum.CA3

   KGServerTypeEnum.AUTO 【*V4.0.0.254*】自动配置版本，*V7* 及 *V8* 的签章服务器都支持。

   keySN：密钥盘序列号或者用户编码，通过 *setQueryBy* 接口确定获取印章方式。

   password：用户密码，*CA0* 版才会验证密码。

   index：当前印章序号，从 *0* 开始。

属性：

```
/**印章跟随文档旋转,默认不跟随自动翻转*/
private boolean rotation;
```

方法：

1. ***public void setText(String text)***

   文本定位

   注意：当文本中包含空格且定位不到时，需要把空格去掉。在直接拷贝 *PDF* 文档中的文本，由于字体的因素，文本会包含空格，实际上是没有空格的，所以会出现定位不到的情况。

2. ***public void setText(String text, boolean onceFindReturn)***

   文本定位

   text *:* 需要定位的文本内容

   onceFindReturn *:*

   true *:* 只在第一个匹配的文本上加盖印章

   false *:* 所有匹配的文本都加盖印章

3. ***public void setTextOffset(float offsetX, float offsetY)***

   文本定位时，设置相对文本位置的偏移量，*PDF* 左下角为坐标原点。

   offsetX *: X* 轴偏移量

   offsetY *: Y* 轴偏移量

4. ***public void setXY(float x, float y)***

坐标定位

5. ***public void setXY(XYType xyType,float x, float y)*** **【*V5.1.0.392*】**

   坐标定位，以 *xyType* 为坐标原点，内容显示区为第一象限

   xyType : 左上角 *XYType .LOWER_LEFT*

   右下角 *XYType.LOWER_RIGHT*

   右上角 *XYType .UPPER_RIGHT*

   左上角 *XYType .TOP_LEFT*

6. ***public void qfz(int number,KGQfzModeEnum qfzMode,int yDistanceOrigin)***

   骑缝章

   number: *number* 页均分一个章，循环分配。模（余数：*mod*）等于 *1*，后面 *number+1* 页均分一个章；否则 *mod* 页均分一个章。*2<=number<=32*。

   qfzMode: *KGQfzModeEnum.ALLPAGE* 所有页

   KGQfzModeEnum.ODDPAGE 奇数页

   KGQfzModeEnum.EVENPAGE 偶数页

   yDistanceOrigin: 印章距离底部的高度

7. ***public void qfzBilateralOff(int yDistanceOrigin)***

   骑缝章，双侧对开。

8. ***public void setStartPage (int startPage)***

   骑缝章，设置盖章的开始页。

9. ***public void setEndPage (int endPage)***

   骑缝章，设置盖章的结束页。

10. ***public void addImgText(KGTextInfo textInfo)***

    印章上添加文本信息，可连续调用该方法。

11. ***public void saveLog(String documentID,String documentName,String logMemo, String extparam1,String extparam2)***

    记录日志，必须是印章来着签章服务器，才能成功保存日志。

    documentID：文档编码

    documentName：文档名称

    logMemo：日志描述

    extparam1：扩展项 *1*

    extparam2：扩展项 *2*

12. ***public void forceValidatePwd()*** **【*V4.0.0.268*】**

    与签章服务器对接获取印章，设置强制验证用户密码。默认只验证 *0* 版用户密码。

13. ***public void setQueryBy(KGQueryByEnum queryBy)*** **【*V4.0.0.292*】**

    设置通过“密钥盘序列号”或者“用户编码”获取印章

queryBy：密钥盘序列号（*KGQueryByEnum.KEYSN*）；

用户编码（*KGQueryByEnum.USERCODE*），签章服务器需要 *768* 及以上版本支持

14. ***public void setProtectDoc(String protectDoc)*** **【*V4.0.0.296*】**

设置电子签章保护模式

protectDoc：*1*（保护文档），*0*（不保护文档）

15. ***public void onlyShowInIWebPDF ()***

设置电子签章只在 *iWebPDF* 可见，其他 *PDF* 阅读器不可见。

16. ***public void setBlendMode(PdfName blendMode)*** **【*V5.1.0.410*】**

设置印章图片显示的复合模式。

blendMode：

PdfGState.BM_NORMAL：印章遮住文字，印章图片底色必须透明。

PdfGState.BM_MULTIPLY：印章不遮住文字。默认值

17. ***public void setSealType(KGSealTypeEnum sealType)*** **【*V5.2.0.500*】**

设置签章类型。

sealType:

KGSealTypeEnum.GENERAL:普通签章类型

KGSealTypeEnum.GM:国密签章类型

3.2.2 添加电子签章*(*对接公安版服务器*)*

类名：*com.kinggrid.pdf.executes. PdfElectronicSeal4GA*

构造方法：

1. ***PdfElectronicSeal4GA(String url, String keySN, String password,String signName)***

签章服务器对接

url：签章服务器地址 *http://127.0.0.1:8089/iSignatureServer/OfficeServer.jsp*

keySN：密钥盘序列号

password：用户密码

signName：印章名称。

3.2.3 添加电子签章*(*对接应用系统加盖政务印章*)*

类名：*com.kinggrid.pdf.executes. PdfElectronicSeal4GB* **【*V5.2.0.526*】**

构造方法：

2. ***PdfElectronicSeal4GB (String url, String appid, String esid)***

应用系统对接

url：应用系统部署地址 *http://192.168.0.184:8081/iSignatureStamp*

salt: 盐值

appid: 授权码

esid：印章编码

3.2.4 电子签章信息

类名：*com.kinggrid.pdf.executes. PdfElectronicSealDetails*

属性：无

方法：

1. ***public List<Signinfo> getSeals()***

   印章信息列表

说明：无

3.2.5 电子签章脱保

类名：*com.kinggrid.pdf.executes.DeviateProtectElectronicSeal*【*V5.2.0.528*】

属性：无

说明：无

3.2.6 删除电子签章

类名：*com.kinggrid.pdf.executes. DeleteElectronicSeal*

属性：

```
/**删除最后一个印章*/
private boolean deleteLastOneSeal;
```

说明：无

3. 3 水印接口

3.3.1 添加水印

类名： *com.kinggrid.pdf.executes.PdfWatermark*

属性：

```
/** 水印标识*/
private String warterMarkerName = "KINGGRID";
/**文字水印*/
private String text = "江西金格科技股份有限公司";
/** 图片水印*/
private Image image;
/** 文字水印字体*/
private BaseFont baseFont;
/** 文字水印 字体大小*/
private float fontSize = 40;
/** 文字水印选择角度*/
private float rotation = 45;
/** 文字水印颜色*/
private BaseColor color = BaseColor.RED;
/**文字水印居中方式*/
private int alignment = Element.ALIGN_CENTER;
/** 水印X坐标*/
private float x;
/** 水印Y坐标*/
private float y;
/** 水印透明度*/
private float fillOpacity = 0.5F;
/** 是否开启跨页水印 */
private boolean crossPage = false;【V4.0.0.240】
/** 有跨页时，是否隐藏首页向上（尾页向下）跨页的水印。仅适用于每页单个水印的情形 */
private boolean hideFirstAndLast = true;【V4.0.0.240】
```

方法：

1. ***public void setXYDistance(float xDistance, float yDistance, float marginLeft, float marginTop)***【*V4.0.0.240*】

   通过指定 *x*、*y* 轴上相邻水印距离添加文本水印需要的参数设置

   xDistance：*x* 轴上相邻水印的距离

   yDistance：*y* 轴上相邻水印的距离

   marginLeft：起始点到页面左边框的距离

marginTop：水印起始点到页面上边框的距离

2. ***public void setDensity(float density)***【*V4.0.0.240*】

密度水印。相邻水印在*x*、*y*轴上的距离相等。

density：*0~1*

3.3.2 删除水印

类名：*com.kinggrid.pdf.executes. DeleteWatermark*

属性：

```
/** 水印标识*/
private String watermarkName;
```

3. 4 文本域接口

3.4.1 填充文本域

类名：*com.kinggrid.pdf.executes.fields. PdfFieldsFillText*【*V4.0.0.246*】

构造方法：

1. ***PdfFieldsFillText(String fields)***

XML 字符串填充 *PDF* 文本域。

2. ***PdfFieldsFillText(Map<String,String> fields)***

键值对填充 *PDF* 文本域。

属性：

```
/**填充文本域强制使用STSong-Light字体*/
private boolean fore = true; 【V4.0.0.248】
/**字体*/
private BaseFont font; 【V4.0.0.248】
```

3.4.2 提取域内容

类名：*com.kinggrid.pdf.executes.fields. PdfFieldsExtractText*【*V4.0.0.246*】

属性：

无

方法：

1. ***public String getFieldsContent()***

*XML*的形式返回文本域的内容。

2．*public Map<String,String> getFields()*

*MAP*的形式返回文本域内容。

4 任务接口实例

4.1 数字签名实例

4.1.1 添加数字签名

```
public void digitalSignatures(){
    KGPdfHummer hummer = null;
    FileInputStream cert = null;
    FileOutputStream    fileOutputStream = null;
    try {
        cert = new FileInputStream("D:/tmp/sign.pfx");
        fileOutputStream = new FileOutputStream("D:/tmp/2.pdf");
        hummer = KGPdfHummer.createSignature("D:/tmp/1.pdf", null,
                true, fileOutputStream, new File("D:/tmp/"),true);
        hummer.setCertificate(cert, "123456", "123456");

        PdfSignature4KG pdfSignature4KG = new PdfSignature4KG(
                "d:/tmp/iSignature.key",1,"123456");
        pdfSignature4KG.setText("金格科技");

        hummer.setPdfSignature(pdfSignature4KG);
        hummer.doSignature();
    } catch (Exception e) {
        e.printStackTrace();
    }   finally {
        close(cert);
        close(fileOutputStream);
        if(hummer != null) hummer.close();
    }
}
```

4.1.2 删除数字签名

```
public void testDelDigitalSignatures(){
    KGPdfHummer hummer = null;
    FileOutputStream    fileOutputStream = null;
    try {
        fileOutputStream = new FileOutputStream("D:/tmp/2.pdf");
        hummer = KGPdfHummer.createInstance("D:/tmp/1.pdf", null,
                true, fileOutputStream, true);

        DeleteSignature deleteSignature = new DeleteSignature();
        hummer.addExecute(deleteSignature);

        hummer. doExecute ();
    } catch (Exception e) {
        e.printStackTrace();
    }   finally {
        close(fileOutputStream);
        if(hummer != null) hummer.close();
    }
}
```

4.1.3 数字证书信息

```
public void testDigitalcertificates(){
    KGPdfHummer hummer = null;
    try {
        hummer = KGPdfHummer.createInstance("D:/tmp/1.pdf", null, true);
        List<Certificate> certificates =hummer.getSignatureCertificates();
        for(Certificate certificate : certificates){
            X509Certificate x509Certificate = (X509Certificate)certificate;
            System.out.println(x509Certificate.getSubjectDN().getName());
            System.out.println(x509Certificate.getIssuerDN().getName());
            System.out.println(x509Certificate.getSigAlgName());
            System.out.println(x509Certificate.getVersion());
```

```
                System.out.println(x509Certificate.getType());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }   finally {
            if(hummer != null) hummer.close();
        }
    }
```

4.1.4 验证数字签名

```
    public void testVerifyDigitalSignatures(){
        KGPdfHummer hummer = null;
        try {
            hummer = KGPdfHummer.createInstance("D:/tmp/1.pdf", null, true);
            /*  -1：文档不存在数字签名。
                0：至少有一个签名是无效的。
                1：所有签名有效。
                2：所有签名有效，最后一次签名后追加了内容。
             */
            System.out.println(hummer.verifySignatures());
        } catch (Exception e) {
            e.printStackTrace();
        }   finally {
            if(hummer != null) hummer.close();
        }
    }
```

4.1.5 加密机加密卡

```
    public void testDigitalSignaturesSmartCard(){
        KGPdfHummer hummer = null;
        FileInputStream cert = null;
        FileOutputStream    fileOutputStream = null;
        try{
            fileOutputStream = new FileOutputStream("D:/tmp/2.pdf");
```

```
cert = new FileInputStream("D:/tmp/sign.pfx");
byte[] certb = new byte[cert.available()];
cert.read(certb);

hummer = KGPdfHummer.createSignature("D:/tmp/1.pdf", null, true,
        fileOutputStream, new File("D:/tmp/"),true);

/*------------------------------第一步：设置证书开始--------------------------------*/
//模拟加密机、加密槽
Security.addProvider(new BouncyCastleProvider());
KeyStore ks = KeyStore.getInstance("PKCS12", "BC");
ks.load(new ByteArrayInputStream(certb), "123456".toCharArray());
String alias = (String)ks.aliases().nextElement();
PrivateKey privateKey = (PrivateKey)ks.getKey(alias,"123456".toCharArray());
final Signature signature = Signature.getInstance("SHA1withRSA");
signature.initSign(privateKey);

X509Certificate x509Certificate = (X509Certificate) ks.getCertificate(alias);
// 公钥证书
/*CertificateFactory factory = CertificateFactory.getInstance("X.509");
X509Certificate x509Certificate = (X509Certificate)factory.generateCertificate(
        new FileInputStream("resources/sign.cer"));*/

hummer.setCertificate(x509Certificate, new SignatureInter(){

    public String getEncryptionAlgorithm() {
        return "RSA";
    }

    public String getHashAlgorithm() {
        return "SHA-1";
    }

    public byte[] sign(byte[] message)
            throws GeneralSecurityException {
```

```
                signature.update(message);
                return signature.sign();
            }

        });
        /*-------------------------------第一步：设置证书结束---------------------------------*/

        PdfSignature4KG pdfSignature4KG = new PdfSignature4KG(
                "d:/tmp/iSignature.key",1,"123456");
        pdfSignature4KG.setText("金格科技");

        hummer.setPdfSignature(pdfSignature4KG);
        hummer.doSignature();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        close(fileOutputStream);
        close(cert);
        if(hummer != null) hummer.close();
    }
}
```

4.1.6 骑缝章及双侧

```
public void testDigitalSignaturesQfz(){
    KGPdfHummer hummer = null;
    FileOutputStream    fileOutputStream = null;
    FileInputStream cert = null;
    try {
        fileOutputStream = new FileOutputStream("D:/tmp/2.pdf");
        cert = new FileInputStream("D:/tmp/sign.pfx");
        hummer = KGPdfHummer.createSignature("D:/tmp/11.pdf", null,
                true, fileOutputStream,new File("D:/tmp"), true);
        hummer.setCertificate(cert, "123456", "123456");

        PdfSignature4KG pdfSignature4KG = new PdfSignature4KG(
```

```
                    "d:/tmp/iSignature.key",0,"123456");
            pdfSignature4KG.setStartPage(2);
            pdfSignature4KG.setEndPage(5);
            pdfSignature4KG.qfz(5, KGQfzModeEnum.ALLPAGE, 100);
            //双侧对开
            //pdfSignature4KG.qfzBilateralOff(200);

            hummer.setPdfSignature(pdfSignature4KG);
            hummer.doSignature();
        } catch (Exception e) {
            e.printStackTrace();
        }   finally {
            close(cert);
            close(fileOutputStream);
            if(hummer != null) hummer.close();
        }
    }
```

4.1.7 签名域不可见

```
    public void testDigitalSignaturesNotinvisible(){
        KGPdfHummer hummer = null;
        FileInputStream cert = null;
        FileOutputStream    fileOutputStream = null;
        try {
            cert = new FileInputStream("D:/tmp/sign.pfx");
            fileOutputStream = new FileOutputStream("D:/tmp/2.pdf");
            hummer = KGPdfHummer.createSignature("D:/tmp/1.pdf", null,
                    true, fileOutputStream, new File("D:/tmp/"),true);

            hummer.setCertificate(cert, "123456", "123456");
            hummer.getPdfSignature().notInvisible();

            hummer.doSignature();
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }   finally {
            close(cert);
            close(fileOutputStream);
            if(hummer != null) hummer.close();
        }
    }
```

4.1.8 KEY 数字签名

第一步：

```
    public void testSignByKey(){
        KGPdfHummer hummer = null;
        FileOutputStream    fileOutputStream = null;
        try{
            fileOutputStream = new FileOutputStream("D:/tmp/2.pdf");
            hummer = KGPdfHummer.createSignature("D:/tmp/1.pdf", null, true,
                    fileOutputStream, new File("D:/tmp/"),true);

            // 公钥证书
            CertificateFactory factory = CertificateFactory.getInstance("X.509");
            X509Certificate x509Certificate = (X509Certificate)factory.generateCertificate(
                    new FileInputStream("d:/tmp/123456.cer"));

            DigitalSignatureByKey digitalSignatureByKey = new DigitalSignatureByKey();
            hummer.setCertificate(x509Certificate, digitalSignatureByKey);

            PdfSignature4KG pdfSignature4KG = new PdfSignature4KG(
                    "d:/tmp/iSignature.key",1,"123456");
            pdfSignature4KG.setXY(200F,200F);

            hummer.setPdfSignature(pdfSignature4KG);
            hummer.doSignature();

            // PDF 文档的摘要（16 进制），U 盾对摘要做签字
            System.out.println("hash = " + digitalSignatureByKey.getHash());
            // 签名值的起始位置
```

```
            System.out.println(digitalSignatureByKey.getContentsPostion());

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            close(fileOutputStream);
            if(hummer != null) hummer.close();
        }
    }
```

第二步：

KG_Crypt_COM_API 控件调用 *KEY* 的接口对 *PDF* 文档的摘要进行签名，并记录签名值。

第三步：

```
public void testSignfill() throws Exception{
    long contentsPostion = 96464L; //签名值的起始位置
    String contents = ""; //签名值（16 进制）
    DigitalSignatureByKey digitalSignatureByKey = new DigitalSignatureByKey();
    digitalSignatureByKey.rewriteContents("d:/tmp/2.pdf", contentsPostion, contents);
}
```

4.1.9 一签名多印章

```
public void testDigitalSignatures(){
    KGPdfHummer hummer = null;
    FileInputStream cert = null;
    FileOutputStream    fileOutputStream = null;
    try {
        cert = new FileInputStream("D:/tmp/sign.pfx");
        fileOutputStream = new FileOutputStream("D:/tmp/2.pdf");

        hummer = KGPdfHummer.createSignature("D:/tmp/11.pdf", null,
                true, fileOutputStream, new File("D:/tmp/"),true);
        hummer.setCertificate(cert, "123456", "123456");
        hummer.setPdfSignature(null);
```

```
            PdfSignature4KG pdfSignature4KG = new PdfSignature4KG(
                    "d:/tmp/iSignature.key",1,"123456");
            pdfSignature4KG.setXY(200,200);
            hummer.addExecute(pdfSignature4KG);

            PdfSignature4KG pdfSignature4KG1 = new PdfSignature4KG(
                    "d:/tmp/iSignature.key",0,"123456");
            pdfSignature4KG1.qfz(5, KGQfzModeEnum.ALLPAGE, 200);
            hummer.addExecute(pdfSignature4KG1);

            hummer.doSignature();
        } catch (Exception e) {
            e.printStackTrace();
        }    finally {
            close(cert);
            close(fileOutputStream);
            if(hummer != null) hummer.close();
        }
    }
```

4.2 电子签章实例

4.2.1 添加电子签章

```
    public void testElectronicSeal(){
        KGPdfHummer hummer = null;
        FileOutputStream    fileOutputStream = null;
        try {
            fileOutputStream = new FileOutputStream("D:/tmp/2.pdf");
            hummer = KGPdfHummer. createInstance ("D:/tmp/1.pdf", null,
                    true, fileOutputStream,true);

            PdfElectronicSeal4KG pdfElectronicSeal4KG = new PdfElectronicSeal4KG(
                    "d:/tmp/iSignature.key",1,"123456");
            pdfElectronicSeal4KG.setText("金格科技");
```

```
            hummer.addExecute(pdfElectronicSeal4KG);
            hummer.doExecute();
        } catch (Exception e) {
            e.printStackTrace();
        }   finally {
            close(fileOutputStream);
            if(hummer != null) hummer.close();
        }
    }
```

4.2.2 添加电子签章(公安版)

```
    public void testElectronicSeal(){
        KGPdfHummer hummer = null;
        FileOutputStream    fileOutputStream = null;
        try {
            fileOutputStream = new FileOutputStream("D:/tmp/2.pdf");
            hummer = KGPdfHummer. createInstance ("D:/tmp/1.pdf", null,
                        true, fileOutputStream,true);
            PdfElectronicSeal4GA    pdfElectronicSeal4GA = new
            PdfElectronicSeal4GA("http://192.168.0.90:8080/iSignatureServer/OfficeServer.js
                    p","测试3 9999999999999999993","123456","测试3公章");
            pdfElectronicSeal4GA.setText("金格科技");
            hummer.addExecute(pdfElectronicSeal4GA);
            hummer.doExecute();
        } catch (Exception e) {
            e.printStackTrace();
        }   finally {
            close(fileOutputStream);
            if(hummer != null) hummer.close();
        }
    }
```

4.2.3 添加电子签章*(*国密印章*)*

```
public void testElectronicGM () {
    KGPdfHummer hummer = null;
    FileOutputStream    fileOutputStream = null;
    FileInputStream fileInputStream = null;
    try {
        fileOutputStream = new FileOutputStream("D:/pdf/3.pdf");
        fileInputStream = new FileInputStream("D:/pdf/kinggrid.pdf");
        hummer = KGPdfHummer.createInstance(fileInputStream, null, fileOutputStream, true);
        PdfElectronicSeal4KG pdfElectronicSeal4KG = new PdfElectronicSeal4KG(
    "http://192.168.0.69:8080/iSignatureServer/OfficeServer.jsp",KGServerTypeEnum.AUTO,"12
4455555555GMYZCS","123456","金格科技私章 1X2");
        pdfElectronicSeal4KG.setSealType(KGSealTypeEnum.GM);
        pdfElectronicSeal4KG.setPagen(1);
        pdfElectronicSeal4KG.setXY(30,30);

        DigitalSignatureByServerSM2 digitalSignatureByServerSM2 = new
DigitalSignatureByServerSM2(
    "http://192.168.0.69:8080/iSignatureServer/OfficeServer.jsp","124455555555GMYZCS");
        KGPdfElectronicSig electronicSig =
KGPdfElectronicExecuteFactory.getKGPdfElectronicSigSM2(digitalSignatureByServerSM2);
        pdfElectronicSeal4KG.addExtraExecute(electronicSig);
        hummer.addExecute(pdfElectronicSeal4KG);
        hummer.doExecute();
    } catch (Exception e) {
        e.printStackTrace();
    }  finally {
        close(fileOutputStream);
        if(hummer != null) hummer.close();
    }
}
```

4.2.4 添加电子签章(对接应用系统加盖政务印章)

```
public void testElectronicGB() {
    Security.addProvider(new BouncyCastleProvider());
    KGPdfHummer hummer = null;
    FileOutputStream    fileOutputStream = null;
    FileInputStream fileInputStream = null;
    try {
        fileOutputStream = new FileOutputStream("D:/pdf/3.pdf");
        fileInputStream = new FileInputStream("D:/pdf/kinggrid.pdf");
        hummer = KGPdfHummer.createInstance(fileInputStream, null, fileOutputStream, true);

        String stampSystemUrl = "http://192.168.0.184:8081/iSignatureStamp";
        String appid = "340BE82F1C3949D39796E7A2AA76B4D4";
        String esid = "36000000000155";
        String salt = "2ad57f1d158b4a63ac248c423ef09cd8";

        PdfElectronicSeal4GB pdfElectronicSeal4GB = new
PdfElectronicSeal4GB(stampSystemUrl, salt, appid, esid);
        pdfElectronicSeal4GB.setPagen(1);
        pdfElectronicSeal4GB.setXY(200, 200);

        DigitalSignatureByGB digitalSignatureByGB = new
DigitalSignatureByGB(stampSystemUrl, salt, appid, esid, "1", "title");
        KGPdfElectronicSig electronicSig =
KGPdfElectronicExecuteFactory.getKGPdfElectronicSigGB(digitalSignatureByGB);
        pdfElectronicSeal4GB.addExtraExecute(electronicSig);
        hummer.addExecute(pdfElectronicSeal4GB);
        hummer.doExecute();
    } catch (Exception e) {
        e.printStackTrace();
    }  finally {
        close(fileOutputStream);
        if(hummer != null) hummer.close();
    }
```

```
}
```

4.2.5 电子签章信息

```
public void testElectronicSealDetails(){
    KGPdfHummer hummer = null;
    try {
        hummer = KGPdfHummer.createInstance("D:/tmp/2.pdf", null, true);

        PdfElectronicSealDetails pdfElectronicSealDetails =
                new PdfElectronicSealDetails();
        hummer.addExecute(pdfElectronicSealDetails);
        hummer.doExecute();

        List<Signinfo> signinfos = pdfElectronicSealDetails.getSeals();
        for(Signinfo signinfo : signinfos){
            System.out.println(signinfo.getIndex());
            System.out.println(signinfo.getKeySn());
            System.out.println(signinfo.getUserName());
            System.out.println(signinfo.getCompName());
            System.out.println(signinfo.getSignSn());
            System.out.println(signinfo.getSignName());
            System.out.println(signinfo.getSignTime());
            System.out.println(new String(signinfo.getCert()));
            System.out.println("左下角：" + signinfo.getRect().getLeft() + " " +
                                    signinfo.getRect().getBottom() +

                                "右上角："+ signinfo.getRect().getRight() + " "+
                                    signinfo.getRect().getTop());
            // true 为篡改，false 为未篡改
            System.out.println(signinfo.isTamper());
        }
    } catch (Exception e) {
        e.printStackTrace();
    }  finally {
```

```
            if(hummer != null) hummer.close();
        }
    }
```

4.2.6 电子签章脱保

```
public void testDeviateProtectElectronicSeal() {
    KGPdfHummer hummer = null;
    FileOutputStream    fileOutputStream = null;
    try {
        fileOutputStream = new FileOutputStream("D:/pdf/4.pdf");
        hummer = KGPdfHummer.createInstance("D:/pdf/3.pdf", null,
                    true, fileOutputStream, true);

        DeviateProtectElectronicSeal deleteElectronicSeal = new
DeviateProtectElectronicSeal();
        hummer.addExecute(deleteElectronicSeal);

        hummer.doExecute();
    } catch (Exception e) {
        e.printStackTrace();
    }   finally {
        close(fileOutputStream);
        if(hummer != null) hummer.close();
    }
}
```

4.2.7 删除电子签章

```
    public void testDelElectronicSeal(){
        KGPdfHummer hummer = null;
        FileOutputStream    fileOutputStream = null;
        try {
            fileOutputStream = new FileOutputStream("D:/tmp/3.pdf");
            hummer = KGPdfHummer.createInstance("D:/tmp/2.pdf", null,
```

```
                    true, fileOutputStream, true);

            DeleteElectronicSeal deleteElectronicSeal = new DeleteElectronicSeal();
            hummer.addExecute(deleteElectronicSeal);

            hummer.doExecute();
        } catch (Exception e) {
            e.printStackTrace();
        }   finally {
            close(fileOutputStream);
            if(hummer != null) hummer.close();
        }
    }
```

4.2.8 骑缝章及双侧

```
    public void testElectronicSealQfz(){
        KGPdfHummer hummer = null;
        FileOutputStream    fileOutputStream = null;
        try {
            fileOutputStream = new FileOutputStream("D:/tmp/2.pdf");
            hummer = KGPdfHummer.createInstance("D:/tmp/11.pdf", null,
                    true, fileOutputStream, true);

            PdfElectronicSeal4KG pdfElectronicSeal4KG = new PdfElectronicSeal4KG(
                    "d:/tmp/iSignature.key",0,"123456");
            pdfSignature4KG.setStartPage(2);
            pdfSignature4KG.setEndPage(5);
            pdfElectronicSeal4KG.qfz(5, KGQfzModeEnum.ALLPAGE, 100);
            // 双侧对开
            //pdfElectronicSeal4KG.qfzBilateralOff(200);

            hummer.addExecute(pdfElectronicSeal4KG);
            hummer.doExecute();
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }   finally {
            close(fileOutputStream);
            if(hummer != null) hummer.close();
        }
    }
```

4.2.9 数字签名

```
public void testElectronicSealSig(){
        KGPdfHummer hummer = null;
        FileOutputStream    fileOutputStream = null;
        try {
            fileOutputStream = new FileOutputStream("D:/tmp/2_d.pdf");
            hummer = KGPdfHummer.createInstance("D:/tmp/11.pdf", null,
                    true, fileOutputStream, true);
            PdfElectronicSeal4KG pdfElectronicSeal4KG = new PdfElectronicSeal4KG(
                    "http://127.0.0.1:8080/iSignatureServer/OfficeServer.jsp",
                    KGServerTypeEnum.AUTO,"001","123456",0);
            pdfElectronicSeal4KG.setXY(200,200);

            // 电子签章数字签名
            KGPdfElectronicSig electronicSig = new KGPdfElectronicSig();
            //electronicSig.setCertMsg(new FileInputStream("d:/tmp/sign.pfx"), "123456");
            electronicSig.setCertMsg(new DigitalSignatureByServer
                    ("http://127.0.0.1:8080/iSignatureServer/OfficeServer.jsp","001"));

            pdfElectronicSeal4KG.addExtraExecute(electronicSig);

            hummer.addExecute(pdfElectronicSeal4KG);
            hummer.doExecute();
        } catch (Exception e) {
            e.printStackTrace();
        }   finally {
            close(fileOutputStream);
            if(hummer != null) hummer.close();
        }
    }
```

4.3 水印实例

4.3.1 添加水印

```
public void testWatermark(){
    KGPdfHummer hummer = null;
    FileOutputStream fileOutputStream = null;
    try {
        fileOutputStream = new FileOutputStream("D:/tmp/2.pdf");
        hummer = KGPdfHummer.createInstance("D:/tmp/1.pdf", null, true,
                fileOutputStream, true);

        PdfWatermark pdfWatermark = new PdfWatermark();
        pdfWatermark.setText("江西金格科技股份有限公司");
        pdfWatermark.setWatermarkName("watermarkName");

        hummer.addExecute(pdfWatermark);
        hummer.doExecute();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        close(fileOutputStream);
        if (hummer != null) hummer.close();
    }
}
```

4.3.2 删除水印

```
public void testDelWatermark(){
    KGPdfHummer hummer = null;
    FileOutputStream fileOutputStream = null;
    try {
        fileOutputStream = new FileOutputStream("D:/tmp/2.pdf");
        hummer = KGPdfHummer.createInstance("D:/tmp/1.pdf", null, true,
                fileOutputStream, false);
```

```
            DeleteWatermark deleteWatermark = new DeleteWatermark();
            deleteWatermark.setWatermarkName("watermarkName");

            hummer.addExecute(deleteWatermark);
            hummer.doExecute();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            close(fileOutputStream);
            if (hummer != null) hummer.close();
        }
    }
```

4.3.3 跨页水印

```
    public void testCrossPageWatermark(){
        KGPdfHummer hummer = null;
        FileOutputStream fileOutputStream = null;
        try {
            fileOutputStream = new FileOutputStream("D:/tmp/2.pdf");
            hummer = KGPdfHummer.createInstance("D:/tmp/1.pdf", null, true,
                    fileOutputStream, true);
            PdfWatermark pdfWatermark = new PdfWatermark();
            pdfWatermark.setText("江西金格科技");
            pdfWatermark.setFontSize(20);
            pdfWatermark.setRotation(65F);
            pdfWatermark.setColor(BaseColor.CYAN);
            pdfWatermark.setPosition(200F, 10F);
            pdfWatermark.setFillOpacity(1F);
            pdfWatermark.setCrossPage(true);
            pdfWatermark.setHideFirstAndLast(true);

            hummer.addExecute(pdfWatermark);
            hummer.doExecute();
        } catch (Exception e) {
```

```
            e.printStackTrace();
        } finally {
            close(fileOutputStream);
            if (hummer != null) hummer.close();
        }
    }
```

4.3.4 密度水印

```
    public void testTextWatermarkByDensity() {
        KGPdfHummer hummer = null;
        FileOutputStream fileOutputStream = null;
        try {
            fileOutputStream = new FileOutputStream("D:/tmp/2.pdf");
            hummer = KGPdfHummer.createInstance("D:/tmp/1.pdf", null, true,
                    fileOutputStream, true);
            PdfWatermark pdfWatermark = new PdfWatermark();
            pdfWatermark.setText("江西金格科技");
            pdfWatermark.setFontSize(20);
            pdfWatermark.setRotation(45F);
            pdfWatermark.setColor(BaseColor.RED);

            pdfWatermark.setDensity(0.2F);

            pdfWatermark.setCrossPage(false);

            hummer.setPagen(0);
            hummer.addExecute(pdfWatermark);
            hummer.doExecute();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            close(fileOutputStream);
            if (hummer != null) hummer.close();
        }
    }
```

4.4 防复印码实例

4.4.1 防复印码

```
public void testAnti_Copy(){
    KGPdfHummer hummer = null;
    FileOutputStream fileOutputStream = null;
    try {
        fileOutputStream = new FileOutputStream("D:/tmp/2.pdf");
        hummer = KGPdfHummer.createInstance("D:/tmp/1.pdf", null, true,
                fileOutputStream, true);
        // 防复印水印
        Image antiCopyImg = hummer.generateAntiCopyImage("金格科技",
                6D, 4D, 48, 8, 2, "黑体");

        PdfWatermark pdfWatermark = new PdfWatermark();
        pdfWatermark.setFillOpacity(1.0F);
        pdfWatermark.setImage(antiCopyImg);
        pdfWatermark.setPosition(200,200);

        hummer.addExecute(pdfWatermark);
        hummer.doExecute();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        close(fileOutputStream);
        if (hummer != null) hummer.close();
    }
}
```

4.4.2 光学防伪

```
public void testOptic_Image (){
    KGPdfHummer hummer = null;
    FileOutputStream fileOutputStream = null;
```

```
        try {
            fileOutputStream = new FileOutputStream("D:/tmp/2.pdf");
            hummer = KGPdfHummer.createInstance("D:/tmp/1.pdf", null, true,
                    fileOutputStream, true);
            // 光学防伪
            Image antiCopyImg = hummer.generateOpticImage("金格科技", 6D, 3D, 3);
            PdfWatermark pdfWatermark = new PdfWatermark();
            pdfWatermark.setFillOpacity(1.0F);
            pdfWatermark.setImage(antiCopyImg);
            pdfWatermark.setPosition(200,200);

            hummer.addExecute(pdfWatermark);
            hummer.doExecute();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            close(fileOutputStream);
            if (hummer != null) hummer.close();
        }
    }
```

4.5 文档加密实例

文档加密，在初始化 *KGPdfHummer* 实例时，不能采用追加模式。

4.5.1 口令加密

```
    public void testEncryptPdf(){
        FileOutputStream    fileOutputStream = null;
        KGPdfHummer hummer = null;
        try{
            fileOutputStream = new FileOutputStream("D:/tmp/2.pdf");
            hummer = KGPdfHummer.createInstance("D:/tmp/金格科技〔2006〕6 号.pdf", null,
                    true, fileOutputStream,false);
            hummer.encryptPdf("123456", "111111", PdfWriter.ALLOW_COPY);
            hummer.doExecute();
```

```
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            close(fileOutputStream);
            if(hummer != null) hummer.close();
        }
    }
```

4.5.2 证书加密

```
    public void testCertEncryptPdf(){
        FileOutputStream    fileOutputStream = null;
        KGPdfHummer hummer = null;
        try{
            fileOutputStream = new FileOutputStream("D:/tmp/2.pdf");
            hummer = KGPdfHummer.createInstance("D:/tmp/金格科技〔2006〕6 号.pdf", null,
                    true, fileOutputStream,false);
            // 公钥证书
            CertificateFactory factory = CertificateFactory.getInstance("X.509");
            X509Certificate x509Certificate = (X509Certificate)factory.generateCertificate(
                    new FileInputStream("d:/tmp/sign.cer"));
            hummer.encryptPdf(x509Certificate, new int[]{PdfWriter.ALLOW_COPY});
            hummer.doExecute();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            close(fileOutputStream);
            if(hummer != null) hummer.close();
        }
    }
```

4.5.3 口令解密

```
    public void testDecryptPdf (){
        FileOutputStream    fileOutputStream = null;
        KGPdfHummer hummer = null;
```

```
        try{
            fileOutputStream = new FileOutputStream("D:/pdf/111.pdf");
            hummer = KGPdfHummer.createInstance("D:/pdf/1.pdf", null,
                    true, null,false);
            hummer.decryptPdf(fileOutputStream);
            hummer.doExecute();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            close(fileOutputStream);
            if(hummer != null) hummer.close();
        }
}
```

4.6 抽取合并实例

4.6.1 抽取文档

```
    public void testExtractPages() {
        OutputStream    os = null;
        KGPdfHummer hummer = null;
        try {
            String fileName = "D:/tmp/1.pdf";
            os = new FileOutputStream("D:/tmp/2.pdf");
            hummer = KGPdfHummer.createInstance(fileName, null, true);
            List<Integer> pagesToKeep = new ArrayList<Integer>();
            pagesToKeep.add(1);
            pagesToKeep.add(2);
            pagesToKeep.add(3);
            hummer.getKGPdfHummerUtils().extractPages(pagesToKeep, os);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            close(os);
```

```
            if(hummer != null) hummer.close();
        }
    }
    public void testExtractPages2() {
        OutputStream    os = null;
        KGPdfHummer hummer = null;
        try {
            String fileName = "D:/tmp/1.pdf";
            os = new FileOutputStream("D:/tmp/2.pdf");
            hummer = KGPdfHummer.createInstance(fileName, null, true);
            hummer.getKGPdfHummerUtils().extractPages("1-5", os);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            close(os);
            if(hummer != null) hummer.close();
        }
    }
```

4.6.2 合并文档

```
    public void testMergeMultiPdf() {
        OutputStream os = null;
        InputStream is1 = null;
        InputStream is2 = null;
        InputStream is3 = null;
        try {
            os = new FileOutputStream("D:/tmp/2.pdf");
            List<InputStream> srcs = new ArrayList<InputStream>();
            is1 = new FileInputStream("D:/tmp/test1.pdf");
            is2 = new FileInputStream("D:/tmp/test2.pdf");
            is3 = new FileInputStream("D:/tmp/test3.pdf");
            srcs.add(is1);
            srcs.add(is2);
            srcs.add(is3);
            KGPdfHummerUtils.mergeMultiPdf(srcs, os);
```

```
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            close(is1);
            close(is2);
            close(is3);
            close(os);
        }
    }
```

4.7 文本域实例

4.7.1 填充文本域实例

```
    public void testFieldsFillContent(){
        KGPdfHummer hummer = null;
        FileInputStream in = null;
        FileOutputStream    fileOutputStream = null;
        try {
            in = new FileInputStream("d:/tmp/fields.xml");
            byte[] buff = new byte[in.available()];
            in.read(buff);
            String contents = new String(buff,"UTF-8");
            fileOutputStream = new FileOutputStream("D:/tmp/2_f.pdf");
            hummer = KGPdfHummer.createInstance("d:/tmp/ 1.pdf", null, true,
                    fileOutputStream, true);

            PdfFieldsFillText pdfFieldsFillContent = new PdfFieldsFillText(contents);
            hummer.addExecute(pdfFieldsFillContent);
            hummer.doExecute();
        } catch (Exception e) {
            e.printStackTrace();
        }   finally {
            close(fileOutputStream);
            if(hummer != null) hummer.close();
```

```
        }
    }
```

4.7.2 提取域内容实例

```
    public void testFieldsExtractContent(){
        KGPdfHummer hummer = null;
        try {
            hummer = KGPdfHummer.createInstance("d:/tmp/1.pdf", null, true);
            PdfFieldsExtractText pdfFieldsExtractContent = new PdfFieldsExtractText();
            hummer.addExecute(pdfFieldsExtractContent);
            hummer.doExecute();
            System.out.println(pdfFieldsExtractContent.getFieldsContent());
        } catch (Exception e) {
            e.printStackTrace();
        }  finally {
            if(hummer != null) hummer.close();
        }
    }
```

4.8 批注实例

4.8.1 插入批注实例

iWebPDF 或 *iAppPDF* 导出的批注信息。

```
    public void testPostil() throws IOException{
        KGPdfHummer hummer = null;
        FileOutputStream    fileOutputStream = null;
        try {
            fileOutputStream = new FileOutputStream("D:/tmp/2.pdf");
            hummer = KGPdfHummer.createInstance("D:/tmp/1.pdf", null, true,
                    fileOutputStream, true);
            PdfPostil pdfPostil = new PdfPostil();
            List<String> list = new ArrayList<String>();
            //批注信息
            String postils = "";
            list.add(postils);
            pdfPostil.setPostilArray(list);
```

```
            hummer.addExecute(pdfPostil);
            hummer.doExecute();
        } catch (Exception e) {
            e.printStackTrace();
        }   finally {
            fileOutputStream.close();
            if(hummer != null) hummer.close();
        }
    }
```

4.8.2 删除批注实例

```
public void deletePostil() throws IOException{
        KGPdfHummer hummer = null;
        FileOutputStream    fileOutputStream = null;
        try {
            fileOutputStream = new FileOutputStream("D:/tmp/yuan_d.pdf");
            hummer = KGPdfHummer.createInstance("D:/tmp/yuan.pdf", null, true,
fileOutputStream, true);
            DeletePdfPostil deletePdfPostil = new DeletePdfPostil();
            /*deletePdfPostil.setSubtype(PdfName.TEXT);
            deletePdfPostil.setAuthorName("Administrator");*/
            deletePdfPostil.setSubtype(PdfName.LINK);
            hummer.addExecute(deletePdfPostil);
            hummer.doExecute();
        } catch (Exception e) {
            e.printStackTrace();
        }   finally {
            fileOutputStream.close();
            if(hummer != null) hummer.close();
        }

    }
```

5 XML 模板生成 PDF

XML 模板生成 *PDF*【*V4.0.0.240*】，*XML* 模板附件及图片如下：

XML转PDF.zip

说明：参照模板设计 *XML*

代码实例

```
KGPdfExporter pdfExporter = KGPdfExporter.getInstance();
// String content = ""; // 模板内容
//pdfExporter. loadObject (content);
pdfExporter.loadObjectFromFile("resources/template/PdfTemplates.xml");
pdfExporter.export(new FileOutputStream("d:/tmp/templates.pdf"));
```

6 印模生成

6.1.1 生成圆章

Json 格式：

circle（注释版）.json

代码实例：

```
String circle = new String(FileUtils.file2byte(new File("d:/tmp/
circle.json")),"utf-8");
CircleSignCreator circleSignCreator = CircleSignCreator.getInstance();
circleSignCreator.load(circle);
BufferedImage bufferedImage = circleSignCreator.create();
ImageIO.write(bufferedImage, "PNG", new File("d:/tmp/ circle.png"));
```

6.1.2 生成方章

Json 格式

square（注释版）.json

代码实例：

```
String square = new String(FileUtils.file2byte(new
File("d:/tmp/square.json")),"utf-8");
SquarepictureCreator creator =
    SquarepictureCreator.getInstance();
creator.load(square);
BufferedImage bufferedImage = creator.create();
ImageIO.write(bufferedImage,    "PNG",    new    File("d:/tmp/
square.png"));
```

7 PDF 转图片

把 *PDF* 转为图片（*BMP*、*JPG*、*PNG* 格式），同时支持域签名和电子签章。【*V4.0.0.276*】

代码实例

```
KGPdf2Image pdf2Image = null;
try{
    pdf2Image = new KGPdf2Image("d:/tmp/2.pdf");
    // 按 PDF 高宽等比缩放图片，放大后图片清晰度更高
    pdf2Image.scalePercent(2.0F);

    for(int i=1;i<=pdf2Image.getNumberOfPages();i++){
        FileOutputStream fileOutputStream = new FileOutputStream(
"d:/tmp/" + i + ".png" );
        pdf2Image.save2Image(i, fileOutputStream, "PNG");
        fileOutputStream.close();
    }
} finally {
    if(pdf2Image != null){ pdf2Image.close();}
}
```

8 二维码*/PDF417*

生成二维码*/PDF417* 图片，可以以水印的方式或印章方式插入到 *PDF* 文档。

8.1 二维码实例

```
public void testQR(){
    String content = "江西金格科技股份有限公司 - china think you!";
    try {
        // 返回图片（PNG）字节数组，后面两参数是宽度高度（单位像素）
        byte[] imgb = KGQRCode.qrCode(content, 200, 200);
        FileOutputStream fileOutputStream = new FileOutputStream("d:/qr.png");
        fileOutputStream.write(imgb);
        fileOutputStream.flush();
        fileOutputStream.close();
    } catch (IOException e) {
        e.printStackTrace();
    } catch (WriterException e) {
        e.printStackTrace();
    }
}
```

8.2 PDF417 实例

```
public void testPDF417() throws IOException {
    String content = "江西金格科技股份有限公司 - china think you!";
    FileOutputStream fos = new FileOutputStream(new File("D:/pdf417.png"));
    KGPdfHummerUtils.createPdf417(content, fos, 7 , 2);
}
```

9 其他说明

9.1 印章水印类说明

9.1.1 KGTextInfo 水印类

除了默认无参构造方法，还提供 *2* 种创建对象方式：

public KGTextInfo(String signText)	默认为宋体,黑色,*12* 号字,附加信息写在原印章图片右下角,超出图片宽度自动换行 参数说明: *signText* 附件文字信息
public KGTextInfo(String signText, Font font)	指定字体*(*包括 *fontName:*字体,*fontSize:*字体大小,*fontStyle:*字体样式*)*,附加信息写在原印章图片右下角,超出图片宽度自动换行 参数说明: *signText* 附件文字信息 *font* 字体样式

属性方法说明:

setFontStyle(int fontStyle)	设置 *Font* 的样式常量
public void setFontName(String fontName)	设置字体名称
setFontColor(int fontColor)	设置字体颜色,*RGB* 值
setFontSize(int fontSize)	设置字体大小
setPosType(TextInfoPosType floatType)	设置文字位置 参数说明: *TextInfoPosType* :枚举类型,附加文字定位,包括内部 *9* 个方位,外部 *16* 个方位
setXY(float x, float y)	设置坐标定位的坐标值【*V4.0.0.288*】
setXYPercent(float xPercent, float yPercent)	设置坐标百分比定位的坐标值【*V4.0.0.288*】
setCenter(boolean center)	坐标定位及坐标百分比定位时,文本以坐标值为中心。默认为文本的左下角。【*V4.0.0.288*】

9.1.2 TextInfoPosType 枚举类

XY	坐标定位【*V4.0.0.288*】
XYPERCENT	坐标百分比定位【*V4.0.0.288*】
WEST_MIDDLE	内部-左中位置
EAST_MIDDLE	内部-右中位置

NORTH_LEFT	内部-左上角位置
NORTH_RIGHT	内部-右上角位置
NORTH_CENTER	内部-顶部居中位置
SOUTH_LEFT	内部-左下角位置
SOUTH_RIGHT	内部-右下角位置
SOUTH_CENTER	内部-底部居中位置
CENTER_MIDDLE	内部-垂直居中位置
OUT_SOUTH_EAST	外部-东南位置
OUT_SOUTH_RIGHT	外部-南右位置
OUT_SOUTH_MIDDLE	外部-南中位置
OUT_SOUTH_LEFT	外部-南左位置
OUT_EAST_TOP	外部-东上位置
OUT_EAST_MIDDLE	外部-东中位置
OUT_EAST_BUTTOM	外部-东下位置
OUT_EAST_NORTH	外部-东北位置
OUT_NORTH_LEFT	外部-北左位置
OUT_NORTH_CENTER	外部-北中位置
OUT_NORTH_RIGHT	外部-北右位置
OUT_SOUTH_WEST	外部-西南位置
OUT_WEST_BUTTOM	外部-西下位置
OUT_WEST_MIDDLE	外部-西中位置
OUT_WEST_TOP	外部-西上位置
OUT_NORTH_WEST	外部-西北位置

TextInfoPosType 位置效果图

	北				
西	*OUT_NORTH_WEST*	*OUT_NORTH_LEFT*	*OUT_NORTH_CENTER*	*OUT_NORTH_RIGHT*	*OUT_EAST_NORTH*
	OUT_WEST_TOP	*NORTH_LEFT*	*NORTH_CENTER*	*NORTH_RIGHT*	*OUT_EAST_TOP*

OUT_WEST_MIDDLE	WEST_MIDDLE	CENTER_MIDDLE	EAST_MIDDLE	OUT_EAST_MIDDLE
OUT_WEST_BUTTOM	SOUTH_LEFT	SOUTH_CENTER	SOUTH_RIGHT	OUT_EAST_BUTTOM
OUT_SOUTH_WEST	OUT_SOUTH_LEFT	OUT_SOUTH_MIDDLE	OUT_SOUTH_RIGHT	OUT_SOUTH_EAST
南				

9.2 修改签章服务器用户密码说明

```
KGHttpUtils httpUtils = new KGHttpUtils();
try {
    // 通过密钥盘序列号修改密码
    httpUtils. modifyUserPwdByKeySN
                ("http://127.0.0.1:8080/iSignatureServer/OfficeServer.jsp",
              "keySN","oldPwd" ,newPwd");
    // 通过用户编码修改密码
    httpUtils. modifyUserPwdByUserCode
                ("http://127.0.0.1:8080/iSignatureServer/OfficeServer.jsp",
                 "userCode","oldPwd" ,newPwd");
    System.out.println("用户密码修改成功！");
} catch (Exception e) {
    System.out.println("用户密码修改失败：" + e.toString());
}
```

9.3 KGPdfHummerUtils 辅助类说明

类名：*com.kinggrid.pdf. KGPdfHummerUtils*

方法：

1. ***public List<Rectangle> getPositionByText(int pageNumber,String text,boolean once)***

说明： 查找某页文本所在的位置，返回矩形框。【*BDE*】

参数：

pageNumber：页数。

text：文本

once：是否找到对应文本第一个位置就返回。

返回：

文本矩形区域列表

2. ***public void addFieldWithoutSignature(String fieldName,Rectangle rectangle,int page)***

说明： 添加未签名的域【*BDE*】

参数：

filename ：域名称

rectangle ：域所在的矩形框

page ：页码

3. ***public void sealVector(String sealInfo)*** 【*BDE*】

说明：添加矢量图

参数：

sealInfo ：*iwebpdf* 生成的矢量图数据信息

4. ***public boolean isExistFont(String baseFont)***

说明：判断 *PDF* 文件中是否支持字体

参数：

baseFont ：字体名称，如：宋体

返回：

Boolean

5. ***public TextField createTextField(String fieldName, Rectangle rectangle,String defaultText)***

说明：创建文本域对象，完成后调用 *addTextField* 方法添加。【*4.0.0.236*】

参数：

fieldname ：域名称

rectangle ：域的矩形框

defaultText ：默认值

返回：

TextField 文本域对象

6. ***public void addTextField(TextField text, int page, boolean multiline, boolean password, int maxLen)***

说明：添加文本域。【*4.0.0.236*】

参数：

text ： 文本域

page ： 页码

multiline ： 是否多行

password ： 是否密码文本域

maxLen ： 文字最多输入的个数

返回：

void

7. ***public void addTextField(TextField text, int page, boolean multiline, boolean readonly, boolean password, int maxLen)***

说明：添加文本域。**【*6.0.0.554*】**

参数：

text ： 文本域

page ： 页码

multiline ： 是否多行

readonly ： 是否只读

password ： 是否密码文本域

maxLen ： 文字最多输入的个数

返回：

void

8. ***public void addBookMarks(InputStream in)***

说明：根据规定格式的 *XML* 文件增加书签。**【*4.0.0.240*】**

参数：

in ： *xml* 文件流，格式如下

```
<?xml version="1.0" encoding="UTF-8"?>
<Bookmark>
    <Title Page="1 Fit" Action="GoTo">一级书签 1
        <Title Page="8 Fit" Action="GoTo">二级书签 1</Title>
        <Title Page="9 Fit" Action="GoTo">二级书签 2
            <Title Page="10 Fit" Action="GoTo">三级书签</Title>
        </Title>
    </Title>
    <Title Page="14 Fit" Action="GoTo">一级书签 2
        <Title Page="15 Fit" Action="GoTo">二级书签 3</Title>
    </Title>
```

<Title Page="16 Fit" Action="GoTo">一级书签 3</Title>

</Bookmark>

返回：

void

9. ***public void addFileAttachment(InputStream attachment, String attachmentFileName, String description)***

说明：添加附件【*4.0.0.240*】

参数：

attachment ：附件

attachmentFileName ：附件名称

description ：附件描述

返回：

void

10. ***public void extractPages(List<Integer> pagesToKeep, OutputStream os)***

说明：抽取 *PDF* 文档【*4.0.0.240*】

参数：

pagesToKeep：指定的要抽取的页码

os：由抽取页面组成的 *PDF*

返回：

void

11. ***public void extractPages(String ranges,OutputStream os)***

说明：抽取 PDF 文档【*4.0.0.240*】

参数：

ranges：指定的要抽取页码的范围，表达式：*[1-5]*

os：由抽取页面组成的 *PDF*

返回：

void

12. ***public static void mergeMultiPdf(List<InputStream> srcs, OutputStream os)***

说明：合并 *PDF* 文档【*4.0.0.240*】

参数：

srcs：多份 *PDF* 文档流

os：合并后的 *PDF*

返回：

void

13. ***public void addBookMarks(List<String> bookmarkList)***

说明：根据规定格式的 *List* 增加书签【*4.0.0.246*】

参数：

bookmarkList：*List<String>*形式的参数。

元素格式为*"n.n.n 多级书签"*，可看作是两部分：

第一部分*"n.n.n"*表示书签的层级，

第二部分*"多级书签"*表示书签具体的值，二者之间用一个空格分隔。

对于*"n.n.n"*，*"n"*的个数表示几级书签，*n* 的值表示为该级的第几个书签。

例如：*"1 第一章"*表示文档的第一个一级书签；

*"1.1 第一章第一节"*表示一个二级书签，是第一个一级书签下的第一个二级书签；

*"1.2.3 第一章第二节第三段"*表示一个三级书签，是第一个一级书签下的第二个二级书签下的第三个三级书签。

注意：

*1.*书签层级和书签值之间必须有至少一个空格分隔，否则抛异常；

*2.*书签值的前后可有多个空格，但生成的书签中这些空格均会被忽略；

*3.*此方法根据 *List* 中元素顺序从前到后依次增加书签，调用者必须确保 *List* 中书签的顺序与期望的顺序一致；

*4.*如果书签文本在文档中不存在，则报错，整个书签添加失败

返回：

void

示例：

```
List<String> bookmarkList = Arrays.asList("1 第一章", "1.1 第一节", "1.1.1 第一段", "1.2 第二节", "1.2.1 第一段", "2 第二章", "2.1 第一节", "3 第三段");
```

14. public List<Rectangle> getSigFieldsRectOfPageNum(int pageNum)

说明：获取域数字签名的坐标【*4.0.0.300*】

参数：

pageNum：页码

返回：

当前页域数字签名坐标列表

实例：

```
public void getSigFieldsRectOfPageNum(){
    FileOutputStream    fileOutputStream = null;
    String fileName = "D:/tmp/2.pdf";
    KGPdfHummer hummer = null;
    try{
        hummer = KGPdfHummer.createInstance(fileName, null, true);
```

```
            KGPdfHummerUtils hummerUtils = hummer.getKGPdfHummerUtils();
            int numberOfPages = hummer.getNumberOfPages();
            for(int i=1;i<=numberOfPages;i++){
                System.out.println("页码 : " + i);
                List<Rectangle> rects = hummerUtils.getSigFieldsRectOfPageNum(i);
                for(Rectangle rect : rects){
                    System.out.println(" x=" + (rect.getLeft() + rect.getRight())/2F
                        + "   y=" + (rect.getBottom()+rect.getTop())/2F);
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            close(fileOutputStream);
            if(hummer != null) hummer.close();
        }
    }
```

15. ***public static void CreatePDF4ImageList(List<Image> images, OutputStream os)***

说明：新建 *PDF* 文档并把图片居中插入到文档中，图片高宽大于 *A4* 纸时，等比缩放 【*4.0.0.286*】

参数：

images：图片列表

os：*PDF* 输出流

返回：

Void

16. ***public static void tiff2Pdf(String fileName,OutputStream os)***

说明：*tiff* 图片转 *pdf*

参数：

fileName：文件名称

os：*PDF* 输出流

返回：

Void

实例：

```
public    void testTiff2Pdf() throws IOException, Exception {
```

KGPdfHummerUtils.tiff2Pdf("d:/tmp/11.tif", new FileOutputStream("d:/tmp/tiff.pdf"));

}

17. ***public static void image2Pdf (String fileName,OutputStream os)***

说明：单个图片转成 *PDF*

参数：

fileName：文件名称

os：*PDF* 输出流

返回：

Void

实例：

public void testImage2Pdf() throws IOException, Exception {

KGPdfHummerUtils.image2Pdf("d:/tmp/d.bmp", new FileOutputStream("d:/tmp/Img.pdf"));

}

10 文档声明

本文档内容改动及版本更新将不再另行通知。本文档的范例中使用的人名、公司名和数据如果没有特别指明，均属虚构。对于本文档、及本文档涉及的技术和产品，江西金格科技股份有限公司拥有其专利、商标、著作权或其它知识产权，除非得到江西金格科技股份有限公司的书面许可，本文档不授予这些专利、商标、著作权或其它知识产权的许可。

- *Kinggrid*、*iWebOffice*、*iSignature*和*DBPacket*是江西金格科技股份有限公司的商标。
- 其它标牌和产品名称是其各自公司的商标或注册商标。
- 本文档最后更新时间：*2017.09.26*。

（完）