iText 中文教程(含代码)

第一部分 iText 的简单应用

（一切实例代码来致于：http://itextdocs.lowagie.com/tutorial/）

第一章 创建一个 Document

利用 iText 五步创建一个 PDF 文件：helloword。

第一步，创建一个 iTextSharp.text.Document 对象的实例：

Document document = new Document();

第二步，为该 Document 创建一个 Writer 实例：

PdfWriter.getInstance(document, new FileStream("Chap0101.pdf", FileMode.Create));

第三步，打开当前 Document

document.Open();

第四步，为当前 Document 添加内容：

document.Add(new Paragraph("Hello World"));

第五步，关闭 Document

document.Close();

完整的代码见示例代码 0101。

在例中，不难看出，制作一个 PDF 文件是非常简单的。

注：如果你将例中“document.Add(new Paragraph("Hello World"));”中的字符串“Hello Word”换成中文，如“这是我的第一个 PDF 文件”，产生的结果一定让你大失所望，因为生成的 PDF 文件中并没有将中文显示出来，不要担心，在第 9 章中 要专门讲解字体问题，中文显示也就迎刃而解了，如果不能正确显示中文，也就没有必要翻译本文了。

```
public class HelloWorld {
    /**
     * Generates a PDF file with the text 'Hello World'
```

```
     *
     * @param args no arguments needed here
     */
    public static void main(String[] args) {
        System.out.println("Hello World");
        // step 1: creation of a document-object
        Document document = new Document();
        try {
            // step 2:
            // we create a writer that listens to the document
            // and directs a PDF-stream to a file
            PdfWriter.getInstance(document,
                    new FileOutputStream("HelloWorld.pdf"));
            // step 3: we open the document
            document.open();
            // step 4: we add a paragraph to the document
            document.add(new Paragraph("Hello World"));
        } catch (DocumentException de) {
            System.err.println(de.getMessage());
        } catch (IOException ioe) {
            System.err.println(ioe.getMessage());
        }
        // step 5: we close the document
        document.close();
    }
}
```

下面对这几步做详细介绍。

第一步 创建一个 Document 实例：

iTextSharp.text.Document-object 共有三个构造函数：

```
public Document();
public Document(Rectangle pageSize);
public Document(Rectangle pageSize,
int marginLeft,
int marginRight,
int marginTop,
int marginBottom);
```

第一个构造函数以 A4 页面作为参数调用第二个构造函数，第二个构造函数以每边 36 磅页边距为参数调用第三个构造函数

- **页面尺寸：**

你可以通过指定的颜色和大小创建你自己的页面，示例代码 0102 创建一个细长的浅黄色背景的页面：

```
Rectangle pageSize = new Rectangle(144, 720);
pageSize.BackgroundColor = new Color(0xFF, 0xFF, 0xDE);
Document document = new Document(pageSize);
```

通常，你不必创建这样的页面，而可以从下面页面尺寸中选择：

A0-A10, LEGAL, LETTER, HALFLETTER, _11x17, LEDGER, NOTE, B0-B5, ARCH_A-ARCH_E, FLSA 和 FLSE

```
/*
 * $Id: DefaultPageSize.java 3373 2008-05-12 16:21:24Z xlv $
 *
 * This code is part of the 'iText Tutorial'.
 * You can find the complete tutorial at the following address:
```

```
 * http://itextdocs.lowagie.com/tutorial/
 *
 * This code is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * itext-questions@lists.sourceforge.net
 */
package com.lowagie.examples.general;
import java.io.FileOutputStream;
import java.io.IOException;
import com.lowagie.text.Document;
import com.lowagie.text.DocumentException;
import com.lowagie.text.PageSize;
import com.lowagie.text.Paragraph;
import com.lowagie.text.pdf.PdfWriter;
/**
 * Demonstrates the use of PageSize.
 * @author blowagie
 */
public class DefaultPageSize {
    /**
     * Creates a PDF document with a certain pagesize
     * @param args no arguments needed here
     */
    public static void main(String[] args) {
```

```
        System.out.println("The default PageSize and some other
standard sizes");
        // step 1: creation of a document-object
        Document document = new Document();
        try {
            // step 2:
            // we create a writer that listens to the document
            // and directs a PDF-stream to a file
            PdfWriter.getInstance(document, new
FileOutputStream("DefaultPageSize.pdf"));
            // step 3: we open the document
            document.open();
            // step 4: we add some paragraphs to the document
            document.add(new Paragraph("The default PageSize is DIN
A4."));
            document.setPageSize(PageSize.A3);
            document.newPage();
            document.add(new Paragraph("This PageSize is DIN A3."));
            document.setPageSize(PageSize.A2);
            document.newPage();
            document.add(new Paragraph("This PageSize is DIN A2."));
            document.setPageSize(PageSize.A1);
            document.newPage();
            document.add(new Paragraph("This PageSize is DIN A1."));
            document.setPageSize(PageSize.A0);
            document.newPage();
            document.add(new Paragraph("This PageSize is DIN A0."));
```

```
            document.setPageSize(PageSize.A5);
            document.newPage();
            document.add(new Paragraph("This PageSize is DIN A5."));
            document.setPageSize(PageSize.A6);
            document.newPage();
            document.add(new Paragraph("This PageSize is DIN A6."));
            document.setPageSize(PageSize.A7);
            document.newPage();
            document.add(new Paragraph("This PageSize is DIN A7."));
            document.setPageSize(PageSize.A8);
            document.newPage();
            document.add(new Paragraph("This PageSize is DIN A8."));
            document.setPageSize(PageSize.LETTER);
            document.newPage();
            document.add(new Paragraph("This PageSize is LETTER."));
            document.add(new Paragraph("A lot of other standard
PageSizes are available."));
        }
        catch(DocumentException de) {
            System.err.println(de.getMessage());
        }
        catch(IOException ioe) {
            System.err.println(ioe.getMessage());
        }
        // step 5: we close the document
        document.close();
    }
```

```
}
```

大多数情况下使用纵向页面，如果希望使用横向页面，你只须使用 rotate() 函数：

```
Document document = new Document(PageSize.A4.rotate());
```

详细代码见示例代码 0103。

```
/*
 * $Id: LandscapePortrait.java 3373 2008-05-12 16:21:24Z xlv $
 *
 * This code is part of the 'iText Tutorial'.
 * You can find the complete tutorial at the following address:
 * http://itextdocs.lowagie.com/tutorial/
 *
 * This code is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * itext-questions@lists.sourceforge.net
 */

package com.lowagie.examples.general;
import java.io.FileOutputStream;
import java.io.IOException;
import com.lowagie.text.Document;
import com.lowagie.text.DocumentException;
import com.lowagie.text.PageSize;
import com.lowagie.text.Paragraph;
```

```
import com.lowagie.text.pdf.PdfWriter;
/**
 * Demonstrates the creating PDF in portrait/landscape.
 * @author blowagie
 */
public class LandscapePortrait {
    /**
     * Creates a PDF document with pages in portrait/landscape.
     * @param args no arguments needed here
     */
    public static void main(String[] args) {

        System.out.println("Documents in Landscape and Portrait
format");
        // step 1: creation of a document-object
        Document document = new Document(PageSize.A4.rotate());
        try {
             // step 2:
            // we create a writer that listens to the document
            // and directs a PDF-stream to a file
            PdfWriter.getInstance(document, new
FileOutputStream("LandscapePortrait.pdf"));
            // step 3: we open the document
            document.open();
            // step 4: we add some content
            document.add(new Paragraph("To create a document in
landscape format, just make the height smaller than the width. For
```

```
instance by rotating the PageSize Rectangle: PageSize.A4.rotate()"));
            document.setPageSize(PageSize.A4);
            document.newPage();
            document.add(new Paragraph("This is portrait again"));
        }
        catch(DocumentException de) {
            System.err.println(de.getMessage());
        }
        catch(IOException ioe) {
            System.err.println(ioe.getMessage());
        }
        // step 5: we close the document
        document.close();
    }
}
```

◆ **页边距：**

当创建一个文件时，你还可以定义上、下、左、右页边距：

```
Document document = new Document(PageSize.A5, 36, 72,
108, 180);
```

在示例代码 0104 中你可以看到该文档有一个 0.5 英寸的左边距和 1 英寸的右边距，上边距为 1.5 英寸，下边距为 2.5 英寸。

说明:

当创建一个矩形或设置边距时，你可能希望知道该用什么度量单位：厘米、英寸或象素，事实上，默认的度量系统以排版单位磅为基础得出其他单位的近似值，如 1 英寸=72 磅，如果你想在 A4 页面的 PDF 中创建一个矩形，你需要计算以下数据:

21 厘米 / 2.54 = 8.2677 英寸

8.2677 英寸* 72 = 595 磅

29.7 厘米 / 2.54 = 11.6929 英寸

11.6929 英寸* 72 = 842 磅

默认边距为 36 磅即半英寸。

如果你修改了页面尺寸，仅仅影响到下一页，如果你修改了页边距，则影响到全部，故慎用。

关于页面的初始值，请参考第三步。

```
/*
 * $Id: CustomPageSize.java 3373 2008-05-12 16:21:24Z xlv $
 *
 * This code is part of the 'iText Tutorial'.
 * You can find the complete tutorial at the following address:
 * http://itextdocs.lowagie.com/tutorial/
 *
 * This code is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * itext-questions@lists.sourceforge.net
 */

package com.lowagie.examples.general;
import java.io.FileOutputStream;
import java.io.IOException;
import com.lowagie.text.Document;
import com.lowagie.text.DocumentException;
```

```
import com.lowagie.text.Paragraph;
import com.lowagie.text.Rectangle;
import com.lowagie.text.pdf.PdfWriter;
/**
 * Demonstrates the use of PageSize.
 * @author blowagie
 */
public class CustomPageSize {
    /**
     * Creates a PDF document with a certain pagesize
     * @param args no arguments needed here
     */
    public static void main(String[] args) {
        System.out.println("Custom PageSize and backgroundcolor");
        // step 1: creation of a document-object
        Rectangle pageSize = new Rectangle(216, 720);
        pageSize.setBackgroundColor(new java.awt.Color(0xFF, 0xFF,
0xDE));
        Document document = new Document(pageSize);
        try {
            // step 2:
            // we create a writer that listens to the document
            // and directs a PDF-stream to a file
            PdfWriter.getInstance(document, new
FileOutputStream("CustomPageSize.pdf"));
            // step 3: we open the document
            document.open();
```

```
            // step 4: we add some paragraphs to the document
            document.add(new Paragraph("The size of this page is
216x720 points."));
            document.add(new Paragraph("216pt / 72 points per inch =
3 inch"));
            document.add(new Paragraph("720pt / 72 points per inch =
10 inch"));
            document.add(new Paragraph("The size of this page is 3x10
inch."));
            document.add(new Paragraph("3 inch x 2.54 = 7.62 cm"));
            document.add(new Paragraph("10 inch x 2.54 = 25.4 cm"));
            document.add(new Paragraph("The size of this page is
7.62x25.4 cm."));
            document.add(new Paragraph("The backgroundcolor of the
Rectangle used for this PageSize, is #FFFFDE."));
            document.add(new Paragraph("That's why the background of
this document is yellowish..."));
        }
        catch(DocumentException de) {
            System.err.println(de.getMessage());
        }
        catch(IOException ioe) {
            System.err.println(ioe.getMessage());
        }
        // step 5: we close the document
        document.close();
    }
```

```
}
```

Margins

In step 4, you will be adding content to the document. If you use high level objects, you needn't worry about margins and page layout. The high level objects take care of that for you. Of course you will have to define the margins first. That is: if you want other margins than the standard 36 points (half an inch).

You can also change the margins while you are adding content. Note that the changes will only be noticed on the NEXT page. If you want the margins mirrored (odd and even pages), you can do this with this method: setMarginMirroring(true).

```
/*
 * $Id: Margins.java 3373 2008-05-12 16:21:24Z xlv $
 *
 * This code is part of the 'iText Tutorial'.
 * You can find the complete tutorial at the following address:
 * http://itextdocs.lowagie.com/tutorial/
 *
 * This code is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * itext-questions@lists.sourceforge.net
 */

package com.lowagie.examples.general;
import java.io.FileOutputStream;
import java.io.IOException;
import com.lowagie.text.Document;
```

```
import com.lowagie.text.DocumentException;
import com.lowagie.text.Element;
import com.lowagie.text.PageSize;
import com.lowagie.text.Paragraph;
import com.lowagie.text.pdf.PdfWriter;
/**
 * Demonstrates the use of margins.
 * @author blowagie
 */
public class Margins {
    /**
     * Creates a PDF document with different pages that have
different margins.
     * @param args no arguments needed here
     */
    public static void main(String[] args) {
        System.out.println("Document margins");
        // step 1: creation of a document-object
        Document document = new Document(PageSize.A5, 36, 72, 108,
180);
        try {
            // step 2:
            // we create a writer that listens to the document
            // and directs a PDF-stream to a file
            PdfWriter.getInstance(document, new
FileOutputStream("Margins.pdf"));
            // step 3: we open the document
```

```
            document.open();
            // step 4:
            document.add(new Paragraph("The left margin of this
document is 36pt (0.5 inch); the right margin 72pt (1 inch); the top
margin 108pt (1.5 inch); the bottom margin 180pt (2.5 inch). "));
            Paragraph paragraph = new Paragraph();
            paragraph.setAlignment(Element.ALIGN_JUSTIFIED);
            document.add(new
Paragraph("**********************************************************
****"));
            for (int i = 0; i < 10; i++) {
                paragraph.add("Hello World, Hello Sun, Hello Moon,
Hello Stars, Hello Sea, Hello Land, Hello People. ");
            }
            document.add(new
Paragraph("**********************************************************
****"));
            document.add(paragraph);
            document.setMargins(180, 108, 72, 36);
            document.add(new
Paragraph("**********************************************************
****"));
            document.add(new Paragraph("Now we change the margins.
You will see the effect on the next page."));
            document.add(new
Paragraph("**********************************************************
****"));
```

```
            document.add(paragraph);
            document.add(new
Paragraph("************************************************************
****"));
            document.setMarginMirroring(true);
            document.add(new Paragraph("Starting on the next page,
the margins will be mirrored."));
            document.add(new
Paragraph("************************************************************
****"));
            document.add(paragraph);
        }
        catch(DocumentException de) {
            System.err.println(de.getMessage());
        }
        catch(IOException ioe) {
            System.err.println(ioe.getMessage());
        }
        // step 5: we close the document
        document.close();
    }
}
```

第二步 创建 Writer 实例

PdfWriter ： 生成 PDF 文档
RtfWriter ： 生成 RTF 文档
HtmlWriter 生成 HTML 文档。

一旦创建了 document，我们可以创建该文档的多个 Writer 的实例，所有这些 Writer 实例均继承自抽象类“iTextSharp.text.DocWriter”。

同时还有另外一种情况，你可以用 iTextSharp.text.pdf.PdfWriter 产生文档 PDF 文件，如果你想创建一个 TeX 文档，你可以使用 iTextSharp.text.TeX.TeXWriter 包。

Writer 类的构造函数是私有的，你只能通过下面的方法创建一个实例：

```
public static xxxWriter getInstance(Document document,
Stream os);(xxx 是 Pdf 或 Xml)
```

你可以通过下面的方法创建一个实例：

```
PdfWriter writer = PdfWriter.getInstance(document, new
FileStream("Chap01xx.pdf"));
```

但是你几乎永远不会用到 Writer 实例（除非你想创建高级 PDF 或者希望用一些非常特殊的函数，如 ViewerPreferences 或 Encryption）。所以通过下面的办法得到实例已经足够了： PdfWriter.getInstance(document, new FileStream("Chap01xx.pdf"));

在第一步中创建一个文档时，第一个参数意义不大，第二个参数可以是任何一种流，到目前为止我们一直使用 System.IO.FileStream 将 Document 写入文件中，示例代码 0105 用到了 System.IO.MemoryStream（这不是一个独立的例子，你必须在 Servlet Engine 中测试这些代码。

当你试图用它创建一个纯文本的时候，你不可能得到你想要的结果：

```
/*
 * $Id: HelloSystemOut.java 3373 2008-05-12 16:21:24Z xlv $
 *
 * This code is part of the 'iText Tutorial'.
```

```
 * You can find the complete tutorial at the following address:
 * http://itextdocs.lowagie.com/tutorial/
 *
 * This code is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * itext-questions@lists.sourceforge.net
 */

package com.lowagie.examples.general;
import java.io.FileOutputStream;
import java.io.IOException;
import com.lowagie.text.Document;
import com.lowagie.text.DocumentException;
import com.lowagie.text.Paragraph;
import com.lowagie.text.pdf.PdfWriter;
/**
 * Generates a simple 'Hello World' PDF file.
 *
 * @author blowagie
 */
public class HelloSystemOut {
    /**
     * Generates a PDF file with the text 'Hello World'
     *
     * @param args no arguments needed here
```

```
     */
    public static void main(String[] args) {
        System.out.println("Hello World");
        // step 1: creation of a document-object
        Document document = new Document();
        try {
            // step 2:
            // we create a writer that listens to the document
            // and directs a PDF-stream to System.out (and a txt file)
            PdfWriter w = PdfWriter.getInstance(document, System.out);
            w.setCloseStream(false); // System.out should not be
closed
            PdfWriter.getInstance(document,
                    new FileOutputStream("HelloWorld.doc"));
            // step 3: we open the document
            document.open();
            // step 4: we add a paragraph to the document
            document.add(new Paragraph("Hello World"));
        } catch (DocumentException de) {
            System.err.println(de.getMessage());
        } catch (IOException ioe) {
            System.err.println(ioe.getMessage());
        }
        // step 5: we close the document
        document.close();
    }
}
```

Remark that you can have different writers listening to the same document at the same time. For instance: if you want to sent a PDF to a browser directly from a servlet, but keep a copy of the PDF in a file on the server, you could invoke getInstance twice with the same document-object, but a different output stream.
At Ghent University, we write sheets with information on Study Programmes for the Course Catalogue in 2 versions: one in HTML for the website and one in PDF for printing. This is done simultaneously using the same source code, even using the same document object. The only difference is that at some point, we pause the PdfWriter and we add a link from the HTML page to the PDF document. This is demonstrated in the next example:

创建 PDF，RTF，HTML：

```
/*
 * $Id: HelloWorldMultiple.java 3373 2008-05-12 16:21:24Z xlv $
 *
 * This code is part of the 'iText Tutorial'.
 * You can find the complete tutorial at the following address:
 * http://itextdocs.lowagie.com/tutorial/
 *
 * This code is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * itext-questions@lists.sourceforge.net
 */
package com.lowagie.examples.general;
import java.io.FileOutputStream;
import java.io.IOException;
```

```
import com.lowagie.text.Anchor;
import com.lowagie.text.Chunk;
import com.lowagie.text.Document;
import com.lowagie.text.DocumentException;
import com.lowagie.text.Paragraph;
import com.lowagie.text.html.HtmlWriter;
import com.lowagie.text.pdf.PdfWriter;
import com.lowagie.text.rtf.RtfWriter2;
/**
 * Generates simple 'Hello World' PDF, RTF and HTML files.
 *
 * @author blowagie
 */

public class HelloWorldMultiple {
    /**
     * Generates simple PDF, RTF and HTML files using only one
Document object.
     *
     * @param args no arguments needed here
     */
    public static void main(String[] args) {
        System.out.println("Hello World in PDF, RTF and HTML");
        // step 1: creation of a document-object
        Document document = new Document();
        try {
            // step 2:
```

```
            // we create 3 different writers that listen to the
document
            PdfWriter pdf = PdfWriter.getInstance(document,
                    new FileOutputStream("HelloWorldPdf.pdf"));
            RtfWriter2 rtf = RtfWriter2.getInstance(document,
                    new FileOutputStream("HelloWorldRtf.rtf"));
            HtmlWriter.getInstance(document,
                    new FileOutputStream("HelloWorldHtml.html"));
            // step 3: we open the document
            document.open();
            // step 4: we add a paragraph to the document
            document.add(new Paragraph("Hello World"));
            // we make references
            Anchor pdfRef = new Anchor("see Hello World in PDF.");
            pdfRef.setReference("./HelloWorldPdf.pdf");
            Anchor rtfRef = new Anchor("see Hello World in RTF.");
            rtfRef.setReference("./HelloWorldRtf.rtf");
            // we add the references, but only to the HTML page:
            pdf.pause();
            rtf.pause();
            document.add(pdfRef);
            document.add(Chunk.NEWLINE);
            document.add(rtfRef);
            pdf.resume();
            rtf.resume();

        } catch (DocumentException de) {
```

```
            System.err.println(de.getMessage());
        } catch (IOException ioe) {
            System.err.println(ioe.getMessage());
        }
        // step 5: we close the document
        document.close();
    }
}
```

第三步 打开 Document

◆ **摘要**

在你写入任何实际数据之前，你可能希望通过以下几种方法写入一些关于本文档的摘要：

```
public boolean addTitle(String title)
public boolean addSubject(String subject)
public boolean addKeywords(String keywords)
public boolean addAuthor(String author)
public boolean addCreator(String creator)
public boolean addProducer()
public boolean addCreationDate()
public boolean addHeader(String name, String content)
```

你可以选择自己的标题、主题、关键字、作者、创建程序，但以下产品信息将始终被添加：iTextSharp（或者 iTextSharp 的引用)和创建时间（实际上这两种方法是自动调用的）。

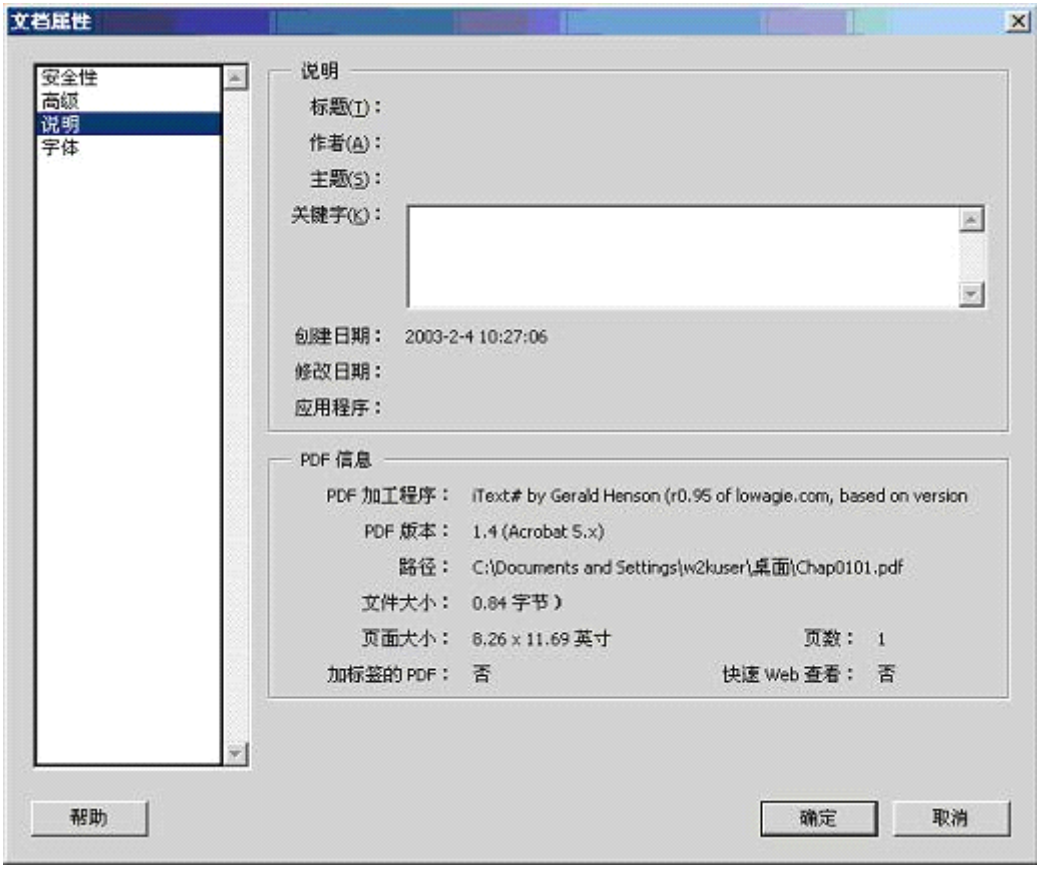
你还可以将自定义的名称添加为“报头信息”，但是这对于 PdfWriter 没有任何作用，如果看看实例代码 0101 产生的 pdf 文件的“文档属性”，我们可以

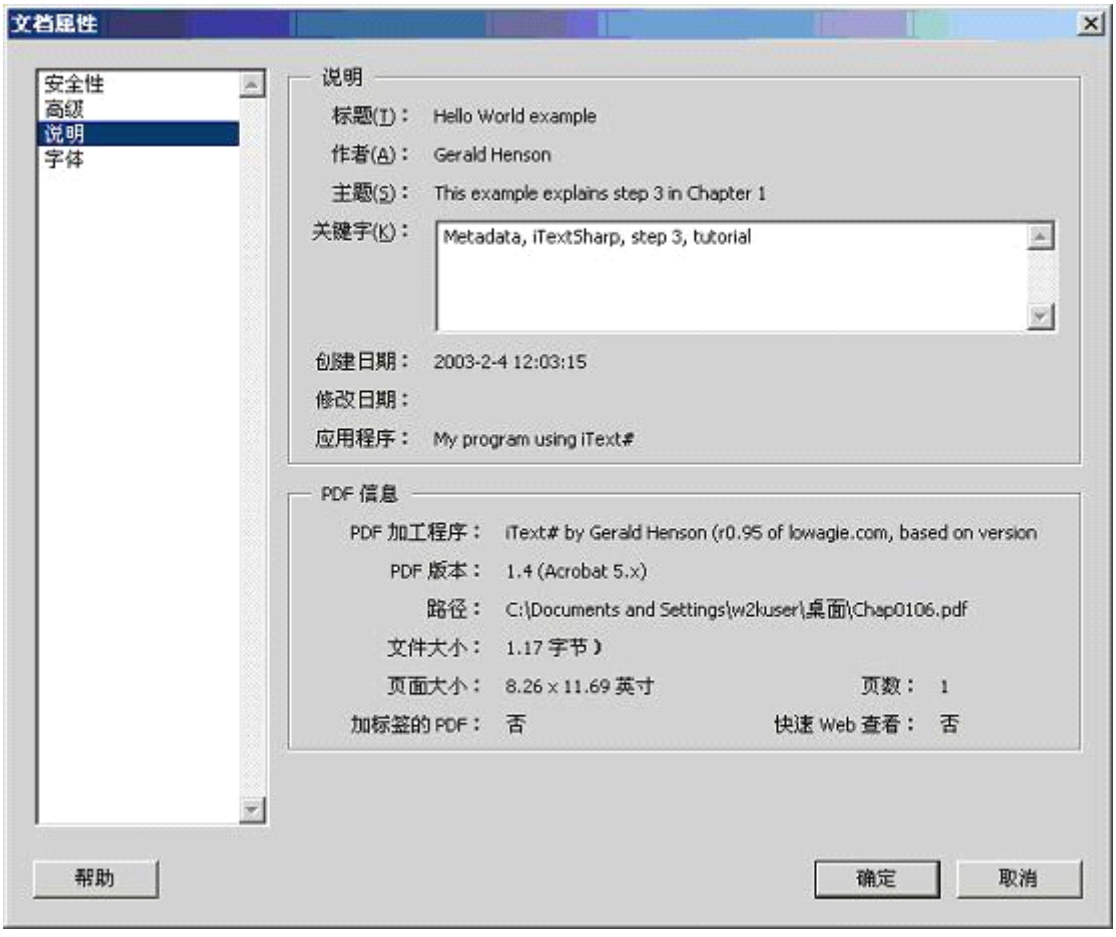
看到仅仅有 PDF 创建程序和产品日期，而示例代码 0106 的“文档属性”框中有更多的信息。

```
/*
 * $Id: HelloWorldMeta.java 3373 2008-05-12 16:21:24Z xlv $
 *
 * This code is part of the 'iText Tutorial'.
 * You can find the complete tutorial at the following address:
 * http://itextdocs.lowagie.com/tutorial/
 *
 * This code is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *
 * itext-questions@lists.sourceforge.net
 */
package com.lowagie.examples.general;
import java.io.FileOutputStream;
import java.io.IOException;
import com.lowagie.text.Document;
import com.lowagie.text.DocumentException;
import com.lowagie.text.Paragraph;
import com.lowagie.text.pdf.PdfWriter;
/**
 * Generates a simple PDF file with metadata.
 *
 * @author blowagie
 */
```

```
public class HelloWorldMeta {
    /**
     * Generates a PDF file with metadata
     *
     * @param args no arguments needed here
     */
    public static void main(String[] args) {
        System.out.println("Metadata");
        // step 1: creation of a document-object
        Document document = new Document();
        try {
            // step 2:
            // we create a writer that listens to the document
            // and directs a PDF-stream to a file
            PdfWriter.getInstance(document,
                    new FileOutputStream("HelloWorldMeta.pdf"));
            // step 3: we add some metadata open the document
            document.addTitle("Hello World example");
              document.addSubject("This example explains how to add
metadata.");
              document.addKeywords("iText, Hello World, step 3,
metadata");
              document.addCreator("My program using iText");
              document.addAuthor("Bruno Lowagie");
            document.open();
            // step 4: we add a paragraph to the document
            document.add(new Paragraph("Hello World"));
```

```
        } catch (DocumentException de) {
            System.err.println(de.getMessage());
        } catch (IOException ioe) {
            System.err.println(ioe.getMessage());
        }
        // step 5: we close the document
        document.close();
    }
}
```


文档属性
安全性
高级
说明
字体
说明
标题(T)：
作者(A)：
主题(S)：
关键字(K)：
创建日期： 2003-2-4 10:27:06
修改日期：
应用程序：
PDF 信息
PDF 加工程序： iText# by Gerald Henson (r0.95 of lowagie.com, based on version
PDF 版本： 1.4 (Acrobat 5.x)
路径： C:\Documents and Settings\w2kuser\桌面\Chap0101.pdf
文件大小： 0.84 字节)
页面大小： 8.26 x 11.69 英寸
页数： 1
加标签的 PDF： 否
快速 Web 查看： 否
帮助
确定
取消


文档属性
安全性
高级
说明
字体
说明
标题(T)： Hello World example
作者(A)： Gerald Henson
主题(S)： This example explains step 3 in Chapter 1
关键字(K)： Metadata, iText5harp, step 3, tutorial
创建日期： 2003-2-4 12:03:15
修改日期：
应用程序： My program using iText#
PDF 信息
PDF 加工程序： iText# by Gerald Henson (r0.95 of lowagie.com, based on version
PDF 版本： 1.4 (Acrobat 5.x)
路径： C:\Documents and Settings\w2kuser\桌面\Chap0106.pdf
文件大小： 1.17 字节)
页面大小： 8.26 x 11.69 英寸
页数： 1
加标签的 PDF： 否
快速 Web 查看： 否
帮助
确定
取消

打开 document 前要做的事：

你只能在Open方法调用之前添加摘要,这是iText开发工具提供的一个选择。

在 HTML 中，报头信息被放在文档前面报头标识中间，调用 Open 方法将导致报头信息写入流，因而在 Document 被打开后无法更改这些数据。

PDF 报头信息不包括摘要，看起来有类似于：

%PDF-1.2

该行显示生成的文档是一个版本为 1.2 的 PDF 格式的文件，在 PDF 中，摘要保存在 PdfInfo 对象中，当文档关闭时已经写入 PdfWriter 中了，因此，没有关于为什么不能修改库来满足任何时候添加或更改摘要的技术原因

- **页面初始化**

Open 方法在不同的 Witer 中同时会产生初始化事件，举例来说，如果你需要一个水印或者页眉页角对象出现在文档第一页的开始处,你需要在打开文档前添加这些，同样的用于设置该文档其他页水印、页眉、页角、页数和尺寸。

当调用下列方法：

```
public bool setPageSize(Rectangle pageSize)
public bool Add(Watermark watermark)
public void removeWatermark()
setting Header property
public void resetHeader()
setting Footer property
public void resetFooter()
public void resetPageCount()
setting PageCount property
```

产生的结果只能在下一个新页中看到（当在本页调用初始化方法时），代码见示例代码 0107，你必须要准备一张名为 watermark.jpg 的图片，如下图：

◆ **阅读器参数：**

你可以通过下面的办法为 PDF 文件指定一些阅读器（如 Adobe Reader）参数：

```
public void setViewerPreferences(int preferences)
```

在示例代码 0108 中，指定了下面一些参数：

```
writerA.setViewerPreferences(PdfWriter.PageLayoutTwoColumn
Left);
writerB.setViewerPreferences(PdfWriter.HideMenubar |
PdfWriter.HideToolbar);
writerC.setViewerPreferences(PdfWriter.PageLayoutTwoColumn
Left | PdfWriter.PageModeFullScreen |
PdfWriter.NonFullScreenPageModeUseThumbs);
```

正如你所看到的，参数可以使用以下一些常量：

- 文件被打开时，页面布局用到下面的其中一个 ：
  - PdfWriter.PageLayoutSinglePage – 同时只显示一个页面
  - PdfWriter.**PageLayoutOneColumn** –单列显示
  - PdfWriter.**PageLayoutTwoColumnLeft** –双列显示,奇数页在左
  - PdfWriter.**PageLayoutTwoColumnRight** -双列显示,奇数页在右
- 文件打开时，页面模式用到下面其中之一：
  - PdfWriter.**PageModeUseNone** – 既不显示大钢也不显示缩略图

- PdfWriter.**PageModeUseOutlines** – 显示大纲
- PdfWriter.**PageModeUseThumbs** – 显示缩略图
- PdfWriter.**PageModeFullScreen** – 全屏模式，没有菜单、windows 控件或者其他任何 windows 可见控件
  - PdfWriter.**HideToolbar** – 当文档激活时，是否隐藏阅读程序（如 Adobe Reader）的工具条
  - PdfWriter.**HideMenubar** -当文档激活时，是否隐藏阅读程序的菜单.
  - PdfWriter.**HideWindowUI** -当文档激活时，是否隐藏阅读程序的界面元素，如滚动条、导航条等，而仅仅保留文档显示
  - PdfWriter.**FitWindow** – 是否调整文档窗口尺寸以适合显示第一页。
  - PdfWriter.CenterWindow – 是否将文档窗口放到屏幕中央
  - 在全屏模式下，指定如何显示界面元素（选择一个）
- PdfWriter.**NonFullScreenPageModeUseNone** -既不显示大钢也不显示缩略图
- PdfWriter.**NonFullScreenPageModeUseOutlines** – 显示大钢
- PdfWriter.**NonFullScreenPageModeUseThumbs** – 显示缩略图

说明:你只能在类 PdfWriter 中调用这些方法。

◆ 加密

打开文档之前还要做的一件事情就是加密（如果你希望该文档加密），要达到这个目的，你可以使用下面的方法：

```
public void setEncryption(boolean strength, String userPassword,
String ownerPassword, int permissions);
```

- strength 是下面两个常量之一：
  - PdfWriter.**STRENGTH40BITS**: 40 位
  - PdfWriter.**STRENGTH128BITS**: 128 位 (Acrobat Reader 5.0 及以上版本支持)
- UserPassword 和 ownerPassword 可以为空或零长度， 这种情况下， ownerPassword 将被随机的字符串代替
- Permissions 为下列常量之一：
  - PdfWriter.**AllowPrinting**

- PdfWriter.**AllowModifyContents**
- PdfWriter.**AllowCopy**
- PdfWriter.**AllowModifyAnnotations**
- PdfWriter.**AllowFillIn**
- PdfWriter.**AllowScreenReaders**
- PdfWriter.**AllowAssembly**
- PdfWriter.**AllowDegradedPrinting**

该功能参见示例代码 0109 和示例代码 0110。

writer.setEncryption(PdfWriter.STRENGTH40BITS, null, null, PdfWriter.AllowCopy);

示例代码 0109 产生的文件能够被打开而无须密码，但用户不能打印、修改本文档。

writer.setEncryption(PdfWriter.STRENGTH128BITS, "userpass", "ownerpass", PdfWriter.AllowCopy | PdfWriter.AllowPrinting);

打你试图打开示例代码 0110 产生的文件时，将要求输入密码（'userpass'），因为添加了 AllowPrinting 参数,你可以打印该文档而不会发生任何问题。

第四步 添加内容

在解释第一步到第三步的不同示例中,你可能已经遇到了一些对象如Phrase, Paragraph 等 在接下来的几章中，所有这些问题都将得到详细解释。

有时你可能想一个 writer 故意忽略 document 产生的行为,如示例代码 0111:

当我们创建了两个 writer： writerA 和 writerB：

PdfWriter writerA = PdfWriter.getInstance(document, new FileStream("Chap0111a.pdf", FileMode.Create));

PdfWriter writerB = PdfWriter.getInstance(document, new FileStream("Chap0111b.pdf", FileMode.Create));

我们可以创建两个有细微差别的文档：

writerA.Pause();

document.add(new Paragraph("This paragraph will only be added to Chap0111b.pdf, not to Chap0111a.pdf"));

writerA.resume();

你可以比较文件：Chap0111a.pdf 和 Chap0111b.pdf 的区别

第五步，关闭 document

关闭 document 非常重要，因为它将关闭正在运行的 Writer 并将内容写入文件，该方法在最后被调用，你应该总是要关闭文档。

高级话题：阅读 PDF 文件

该部分内容介绍了 iText 只能产生 PDF 格式的文件而不能解析 PDF 格式文件，不再翻译。

第二章 块、短句和段落

块

块(Chunk)是能被添加到文档的文本的最小单位，块可以用于构建其他基础元素如短句、段落、锚点等，块是一个有确定字体的字符串，要添加块到文档中时，其他所有布局变量均要被定义。下面一行中，我们创建了一个内容为“hello World”、红色、斜体、COURIER 字体、尺寸 20 的一个块：

Chunk chunk = new Chunk("Hello world", FontFactory.getFont(FontFactory.COURIER, 20, Font.ITALIC, new Color(255, 0, 0)));

- 典型字体 1：

在本指南中，除了第九章外（你可以在这里学会使用其他字体），我们将始终使用典型字体 1，这些是不同的典型字体 1：

- Courier (该字体定宽)
- Helvetica
- Times Roman
- Symbol
- ZapfDingbats

◆ 下划线/删除线

如果你希望一些块有下划线或删除线，你可以通过改变字体风格简单做到：

```
Chunk chunk1 = new Chunk("This text is underlined",
FontFactory.getFont(FontFactory.HELVETICA, 12, Font.UNDERLINE));
Chunk chunk2 = new Chunk("This font is of type ITALIC | STRIKETHRU", FontFactory.getFont(FontFactory.HELVETICA, 12, Font.ITALIC | Font.STRIKETHRU));
```

◆ 上标/下标

在块中有几个方法可以调用，其中大部分将在接下来的章节中介绍，本章中只介绍一个方法 setTextRise(float f). 你可以使用该方法在上标或下标中写块。

◆ 块的背景

如果你想改变块的背景，你可以使用方法 setBackground(Color color). 这将在块文本的下面添加一个彩色矩形：

```
ck.setBackground(new Color(0xFF, 0xFF, 0x00));
```

在示例代码 0101 中，你可以概览典型字体 1 和一个使用 setTextRise, setBackground 等方法的的例子。

短句

短句（Phrases）是一系列以特定间距（两行之间的距离）作为参数的块，一个短句有一个主字体，但短句中的一些块具有不同于主字体的字体，你有更多的选择去创建短句，一些具体使用参见代码 0202。

- **古希腊语**

因为古希腊语经常使用，在类 Phrase 的构造函数中有一个特征：将一个字符串作为参数（如果你想避免这种情况，你只能使用块工作而不能使用字符串），正如你在示例代码 0203 中看到的，这个特征自动地将 913 至 937（除 903）和 945 至 969（古希腊的 ASCII 值）范围内的所有字体改为希腊符号。

- 非主要性

与其说这是一个特征，不如说是一个缺陷，但无论如何，这使创建一个非主要性的短句或段落成为可能，这将产生一个由下向上书写的临时作用（参见示例代码 0204）。如果你想在一页中将一些位置移动到上面时可能有用。

说明，当你穿越上边届时无法检查，也没有办法让你回到前一页。

段落

段落是一系列块和（或）短句。同短句一样，段落有确定的间距。用户还可以指定缩排；在边和（或）右边保留一定空白，段落可以左对齐、右对齐和居中对齐。添加到文档中的每一个段落将自动另起一行。有几种办法建立一个段落，如：

```
Paragraph p1 = new Paragraph(new Chunk("This is my first paragraph.", FontFactory.getFont(FontFactory.HELVETICA, 12)));
Paragraph p2 = new Paragraph(new Phrase("This is my second paragraph.", FontFactory.getFont(FontFactory.HELVETICA, 12)));
```

```
Paragraph p3 = new Paragraph("This is my third
paragraph.", FontFactory.getFont(FontFactory.HELVETICA,
12));
```

所有有些对象将被添加到段落中：

```
p1.add("you can add strings, "); p1.add(new Chunk("you
can add chunks ")); p1.add(new Phrase("or you can add
phrases."));
```

说明：一个段落有一个且仅有一个间距，如果你添加了一个不同字体的短句或块，原来的间距仍然有效，你可以通过 SetLeading 来改变间距，但是段落中所有内容将使用新的中的间距。见示例代码 0205。

◆ 保持段落的整体性

在示例代码 0206 中，我们使用了 setKeepTogether(true)方法来试图将一个段落放在同一页中，该方法并不是始终有效，举个例子，第一段不能刚好在一页中，于是被分成了两部分。第二段被放置在第二页，但第三段顺沿到了第三页上。

字体的延续

你应该掌握字体延续的一些规则，这些规则的应用见示例代码 0207，当我们将一些内容用指定的字体（非默认字体）创建一个短句或者段落后再添加更多内容时，初始对象的字体风格将被延续，请看“Hello 1!”和“Hello 2”：

```
Phrase myPhrase = new Phrase("Hello 2! ", new
Font(Font.TIMES_NEW_ROMAN, 8, Font.BOLD));
myPhrase.Add(new Phrase("some other font ", new
Font(Font.HELVETICA, 8, Font.ITALIC)));
myPhrase.Add(new Phrase("This is the end of the sentence.\n", new
Font(Font.TIMES_NEW_ROMAN, 8, Font.ITALIC)));
document.Add(myPhrase);
```

我们由 Times New Roman 粗体字开始，添加一些文本使用 Helvetica 字体而不指定风格，我们发现文本被改变成了粗体，当我们再加一些文本使用 Times New Roman 字体和斜体风格，结果变成了粗斜体。

如果我们使用 FontFactory 来创建字体，字体风格不会被延续，因为 FontFactory 使用了另外的技术构建一个字体：

```
myPhrase = new Phrase("Hello 1bis! ",
FontFactory.getFont(FontFactory.TIMES_NEW_ROMAN, 8, Font.BOLD));
myPhrase.Add(new Phrase("some other font ",
FontFactory.getFont(FontFactory.HELVETICA, 8, Font.ITALIC)));
myPhrase.Add(new Phrase("This is the end of the sentence.\n",
FontFactory.getFont(FontFactory.TIMES_NEW_ROMAN, 8, Font.ITALIC)));
document.Add(myPhrase);
```

在上面的代码中，使用 Helvetica 字体的文本风字体没有指定（既不是粗体也不是斜体）。采用 Times New Roman 的额外文本仅仅显示为斜体。

你也看到我们添加了一个段落，添加该段落就如同一个短句。

```
Paragraph myParagraph = new Paragraph("Hello 1! ", new
Font(Font.TIMES_NEW_ROMAN, 8, Font.BOLD));
myParagraph.Add(new Paragraph("This is the end of the
sentence.",FontFactory.getFont(new Font.TIMES_NEW_ROMAN, 8)));
document.Add(myParagraph);
```

你可以不这样做，但将失去字体风格的延续，首先不用任何字体创建段落（例中我们仅仅给字体出间距为 1.5 倍），然后添加内容的不同部分。

```
myParagraph = new Paragraph(12);
myParagraph.Add(new Paragraph("Hello 3! ", new
Font(Font.TIMES_NEW_ROMAN, 8, Font.BOLD)));
myParagraph.Add(new Paragraph("This is the end of the sentence.",
new Font(Font.TIMES_NEW_ROMAN, 8, Font.ITALIC)));
```

document.Add(myParagraph);

如果你使用了 Phrase 对象，你同样会失去字体风格的延续：

myPhrase = new Phrase(12);

myPhrase.Add(new Phrase("Hello 4! ", new Font(Font.TIMES_NEW_ROMAN, 8, Font.BOLD)));

myPhrase.Add(new Phrase("This is the end of the sentence.", newFont(Font.TIMES_NEW_ROMAN, 8, Font.ITALIC)));

document.Add(myPhrase);

◆ 更改分割符

通常，当文本不能放在一行时，文本将被分割成不同的部分，iText 首先会查找分割符，如果没有找到，文本将在行尾被截断。有一些预定的分割符如“ ”空格和“-”连字符，但是你可以使用 setSplitCharacter 方法来覆盖这些默认值。在示例代码 0208 中，你可以看到当到达行尾时一个块是如何被分割的。然后分隔符被改成点“.”，该行在该字符处被分割。

第三章 锚点、列表和注释

锚点

我们都知道 HTML 中的超文本链接，当我们点击某些语句，你能够跳转到网上的其他页。在 PDF 中也可以实现这种功能。事实上，在第十一章整个章节中有关于PDF链接的介绍,但这是iText的更高级的应用,本章中我们处理简单的iText。

如果你想在文档中添加一个外部链接（例如使用 URL 链接到 WEB 上的其他文档），你可以简单地使用 Anchor 对象，它派生于 Phrase 对象，使用方法相同。只有两种额外方法定义两种额外变量：setName 和 setReference。

外部链接示例：

Anchor anchor = new Anchor("website", FontFactory.getFont(FontFactory.HELVETICA, 12, Font.UNDERLINE, new Color(0, 0, 255)));

```
anchor.Reference = "http://itextsharp.sourceforge.net";
anchor.Name = "website";
```

如果你想添加内部链接，你需要选择该链接不同的名称，就象你相位在 HTML 中利用名称作为锚点一样。为达到该目的，你需要添加一个“#”。

内部链接示例：

```
Anchor anchor1 = new Anchor("This is an internal link");
anchor1.Name = "link1";
Anchor anchor2 = new Anchor("Click here to jump to the internal link");
anchor.Reference = "#link1";
```

这两个链接的例子请见示例代码 0301。

列表

通过类 List 和 ListItem，你可以添加列表到 PDF 文件中，对于列表你还可以选择是否排序。

排序列表示例：

```
List list = new List(true, 20);
list.Add(new ListItem("First line"));
list.Add(new ListItem("The second line is longer to see what happens once the end of the line is reached. Will it start on a new line?"));
list.Add(new ListItem("Third line"));
```

结果如下：

1. First line
2. The second line is longer to see what happens once the end of the line is reached. Will it start on a new line?
3. Third line

不排序示例如下：

```
List overview = new List(false, 10);
overview.Add(new ListItem("This is an item"));
overview.Add("This is another item");
```

结果如下：

- This is an item
- This is another item

你可以通过 setListSymbol 方法更改列表符号：

```
// 用字符串作为列表符号
list1.ListSymbol = "*";
// 用 Chunk 作为列表符号（包含“•”字符）
list2.ListSymbol = new Chunk("\u2022",
FontFactory.getFont(FontFactory.HELVETICA, 20));
//用图片作为列表符号
list3.ListSymbol = new Chunk(Image.getInstance("myBullet.gif"), 0,
0);
```

还可以使用 setIndentationLeft 和 setIndentationRight 方法设置缩排，列表符号的缩排在构造函数中设置。更多的例子请参见示例代码 0302。

注释

iText 支持不同风格的注释。

◆ 文本注释：

你可以添加一小段文本到你的文档中，但它并非文档内容的一部分，注释有标题和内容：

```
Annotation a = new Annotation(
"authors",
```

"Maybe it's because I wanted to be an author myself that I wrote iText.");

◆ 外部链接注释：

你需要指定一个可点击的矩形和一个字符串（URL 描述）或 URL 对象：

Annotation annot = new Annotation(100f, 700f, 200f, 800f, new URL("http://www.lowagie.com"));

Annotation annot = new Annotation(100f, 700f, 200f, 800f, "http://www.lowagie.com");

◆ 外部 PDF 文件链接注释：

你需要指定一个可点击的矩形和一个字符串（文件名称）和目的文件或页码。

Annotation annot = new Annotation(100f, 700f, 200f, 800f, "other.pdf", "mark");

Annotation annot = new Annotation(100f, 700f, 200f, 800f, "other.pdf", 2);

◆ 指定行为链接注释

你需要指定一个可点击的矩形和一个指定的行为：

Annotation annot = new Annotation(100f, 700f, 200f, 800f, PdfAction.FIRSTPAGE);

◆ 应用程序链接注释：

你需要指定一个可点击的矩形和一个应用程序：

Annotation annot = new Annotation(300f, 700f, 400f, 800f, "C://winnt/notepad.exe", null, null, null);

我们无须在页面上指定一个位置，iText 会内部处理。你能够看到 iText 添加文本注释在页面上当前位置下面，第一个在段后第一行下面，第二个在短句结束处的下面。

所有其他注释需要指定想匹配的矩形区域，在示例代码 0304 中，我们画了一些正方形（使用的函数将在第十章中介绍），为每个正方形添加了一些链接注释 。

第四章 页眉页脚、章节、区域和绘图对象

使用在第三至第五章中描述的大量简单 iText 对象可以避免更多的高级话题（第九至十二章），紧记这些简单对象限制的功能，大量复杂的功能在第三部分。

页眉页脚

HeaderFooter 对象可以于为文档每页添加页眉和页脚。这样一个页眉或页脚包含一个标准的短句（如果需要）和当前页码，如果你需要更多复杂的页眉和页脚（使用表格或者第几页共几页），请阅读第十二章。

在示例代码 0401 中，你可以看到我们首先添加了一个包含页码没有任何边框的页脚。

```
HeaderFooter footer = new HeaderFooter(new Phrase("This is page: "), true);
footer.Border = Rectangle.NO_BORDER;
document.Footer = footer
```

我们还可以使用下面的构造函数：

```
HeaderFooter footer = new HeaderFooter(new Phrase("This is page "), new Phrase("."));
```

构造函数知道你希望添加一个页码和将其放置在两个短句间，如果你只是设置一个 HeaderFooter 而不改变边框，页眉或页脚的文本上下各有一条直线。

```
HeaderFooter header = new HeaderFooter(new Phrase("This is a header without a page number"), false);
document.Header = header;
```

章节和区域

在第十一章中将描述如何构建一个树的外观，如果你只需要一个简单的章节和（子）区域，你可以用 Chapter 对象和 Section 对象自动构建一个树：

```
Paragraph cTitle = new Paragraph("This is chapter 1", chapterFont);
Chapter chapter = new Chapter(cTitle, 1);
Paragraph sTitle = new Paragraph("This is section 1 in chapter 1",
sectionFont);
Section section = chapter.addSection(sTitle, 1);
```

在示例代码 0402 中，我们添加了一系列的章节和子区域，你可以看到完整的树形,树形结构默认打开,如果你希望部分节点关闭,你必须使用用BookmarkOpen属性其值为 false，详见示例代码 0403。

图形

如果你想添加图形，如直线、圆、几何窗体，你应该阅读读十章，但如果你只需要一些有限的功能，你可以使用 Graphic 对象

```
Graphic grx = new Graphic();
//添加一个矩形
grx.rectangle(100, 700, 100, 100);
// 添加一条斜线
grx.moveTo(100, 700);
grx.lineTo(200, 800);
// 将图形显示出来
grx.stroke();
document.Add(grx);
```

完整的代码请见示例代码 0404，如果想看到全部的方法，请参见 PdfContentByte 对象 API。

当你想给页面加一个边框或者在文本当前位置画一条水平线时，图形对象非常有用。下面的方法用指定的宽度、间距（如果需要）和颜色画一个边框。

```
public void setBorder(float linewidth, float
extraSpace);
public void setBorder(float linewidth, float extraSpace,
 Color color);
```

下面的方法用指定的宽度（如果需要）和颜色画一条水平线，线的长度是指定两边缘间可用面积的的百分比。

```
public void setHorizontalLine(float linewidth, float
percentage)
public void setHorizontalLine(float linewidth, float
percentage, Color color)
```

示例代码 5 中，有一个离边界 5 磅，线宽 3 磅的边框，还有两条水平线，一条为黑色，5 磅宽，可用空间的 100%，另外一条为红色，线宽 3 磅，可用空间的 80%。

第五章 表格

重点：如果你仅仅生成 PDF 文件(没有 XML、HTML、RTF……)，使用类 pdfPTable 代替类 Table 更好。

一些简单的表格

一个表格是包含单元格排列成矩阵的矩形区域。表格的距阵并不要求是 m×n 的，它可以有空洞或者单元格比单个的要大。

创建一个表格最通用的办法是预先知道有几行几列：

```
public Table(int columns, int rows);
```

在示例代码 0501 中，我们构建了一个简单的表：

```
Table aTable = new Table(2,2);
aTable.addCell("0.0");
```

```
aTable.addCell("0.1");
aTable.addCell("1.0");
aTable.addCell("1.1");
```

该表格有两行两列，单元格被自动添加，从第一行第一列开始，然后是第二列，当一行满后，下一单元格自动添加到下一行的第一列中。

也可以将单元格添加到表中指定的位置，如示例代码 0502，别了要添加 System.Drawing.dll 引用，以获得 Point 对象，我们创建了一个 4 行 4 列的表格然后添加一些单元格到随机的位置上：

```
Table aTable = new Table(4,4);
aTable.AutoFillEmptyCells = true;
aTable.addCell("2.2", new Point(2,2));
aTable.addCell("3.3", new Point(3,3));
aTable.addCell("2.1", new Point(2,1));
aTable.addCell("1.3", new Point(1,3));
```

你可以看到我们将 AutoFillEmptyCells 属性设置为 true，这将自动、默认的单元格布局填充空的单元格，如果我们忘记了这样做（就象本例中第二个表格），将没有额外的单元格添加，不包含任何单格的行也将被忽略，在本例中，第一行将不显示，因为该行是空行。

经常用数据库查询结果来填充表格，大多数情况下，你预先并不知道到底需要多少行，这就是为什么还有第二个构造函数的原因：

```
public Table(int columns);
```

iText 根据需要自动添加行，在示例代码 0503 中，初始化了 4 行 4 列，当我们添加第 6 行和第 7 行的单元格时，iText 自动增加行数到 7。

增加列数也是可能的，但是有点麻烦，它不能自动生成，你必须使用 addColumns 方法并设置列宽，详见示例代码 0504。

一些表格参数

前面例子中的表格并不美观，我们可以设置大量的参数来改变表格外观。类 Table 和类 Cell 派生于类 Rectangle，我们可以用大量典型的 Rectangle 方法，让我们来看看示例代码 0505。

```
Table table = new Table(3);
table.BorderWidth = 1;
table.BorderColor = new Color(0, 0, 255);
table.Cellpadding = 5;
table.Cellspacing = 5;
Cell cell = new Cell("header");
cell.Header = true;
cell.Colspan = 3;
table.addCell(cell);
cell = new Cell("example cell with colspan 1 and rowspan 2");
cell.Rowspan = 2;
cell.BorderColor = new Color(255, 0, 0);
table.addCell(cell);
table.addCell("1.1");
table.addCell("2.1");
table.addCell("1.2");
table.addCell("2.2");
table.addCell("cell test1");
cell = new Cell("big cell");
cell.Rowspan = 2;
cell.Colspan = 2;
cell.BackgroundColor = new Color(0xC0, 0xC0, 0xC0);
table.addCell(cell);
table.addCell("cell test2");
document.Add(table);
```

◆ 单元格间距和填距

在第 4 行中，我们设置了表格的填距，就是单元格边界和内容间一定数量的空间，在前面的示例中，我们看到文本紧贴边界，通过使用用特定的填距，就可以避免。

在第 5 行中，我们设置了表格的间距，就是单元格和表格边界间的一定数量的空间，不同的单元格间使用了半数空间，具体代码见示例代码 0506。

◆ 对齐方式

在示例代码 0506 中，我们也改变了单元格“big cell”的对齐方式：

```
cell.HorizontalAlignment = Element.ALIGN_CENTER;
cell.VerticalAlignment = Element.ALIGN_MIDDLE;
```

注：不能总是相信垂直对齐方式。

◆ 边框

如果我们象在第 14 行中那样添加了一个单元格，将使用默认的单元格布局（默认的布局可以 SetDefalut 方法改变），如果我们使用了 Cell 对象，我们可以控制每一个单元格的布局。

在第 2 和第三中，我们设置整个表格的边框宽度和边框颜色，我们在单元格上可以使用的方法，在 12 行中，每个单元格用“box”作为边界绘制（就象在 HTML 中），但是示例代码 0507 显示，我们在 PDF 中有大量更多可能。

◆ 颜色

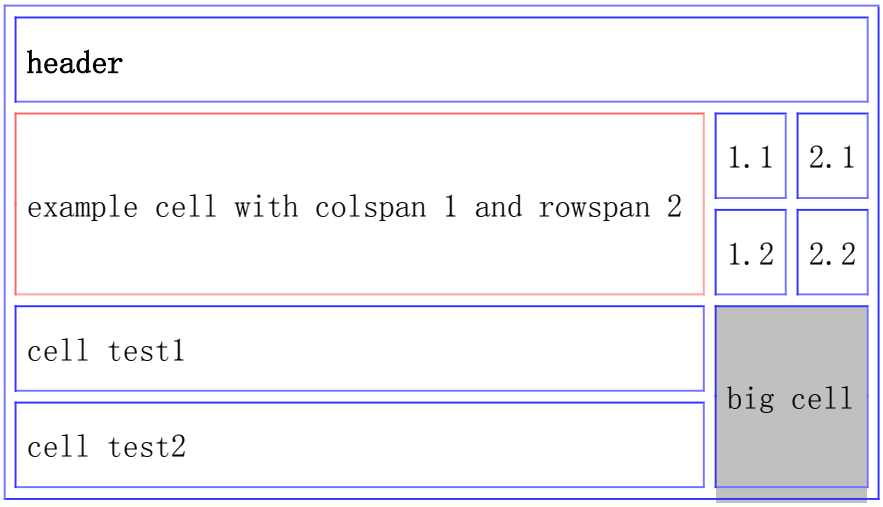
在第 22 行中，你也能定义单元格的背景色，在示例代码 0507 中，我们不使用颜色只是用一定灰度填充。

◆ 行跨和列跨

最后，你也能设置单元格的行跨（11/20 行）和列跨（8/21 列）。通过这种方法可以将几个单元格合并成一个大的单元格。

◆ 备注

第 7 行在 PDF 中没有意义，用于生成 HTML，在 HTML 中并不是总能产生同样的布局，PDF 表格有点象：



◆ 表格分割

如果一个表格不能放在一页中，将自动被分割，示例代码 0508 显示了当一个表格到达页边时发生的情况，这将在下一节中解释。

大表格

跨越几页的表格将自动被分割成不同的部分。示例代码 0509 显示了一个跨越多页的报表。该报表有一个表头，如果你希望这个表头在每页都出现，你可以用 endHeaders()方法标记表头区域的结束点，见示例代码 0510。

为做这样的报表，建议设置单元格间距为 0 和仅使用指定的填距。

你可能已经注意到了，当一个表格被分割时，一些边界好象丢失了。这是因为单元格在前一页被完整地绘制了而不会传递给下一页。

◆ 强行将一个表格或单元格布置到一页上

有有些情况下，你可能希望避免单元格或者整个表被拆分成两个部分，示例代码 0511 差不多和示例代码 0508 完全一样，但我们设置了参数 TableHasToFit 为 true，看看示例代码 0508 和示例代码 0511 结果区别。在示例代码 0512 中我

们修改了示例代码 0510 的 CellsHaveToFit 属性为 true，比较两个示例产生结果的区别。

内存管理

当我们添加一个对象到文档时，该对象一有可能就写入了输出流，但当创建一个表格时，该 Table 对象一直保存着，对于真正的大表格，这将成为一个问题 。

同样，当你正写一个 HttpServletResponse 对象到输出流时，浏览器也可能超时。这就是为什么你自己用 fitsPage()方法控制表分割是有用的，示例代码 0513 告诉你如何做。

嵌套表格

有两种方法嵌套表格，第一种是利用 insertTable 方法明确地将一个表格插入到另外一个表格，示例代码 0514 显示了通过插入到其他表格的办法创建的 5 个 表格。正如你看到的在前面两个表中，所有空的单元格自动得到分割，因为改变了原来的表格。如果一个单元格不空，列跨度和（或）行跨度将自动调整到新的位 置，页面上第三个表格显示所有原表中列的相关宽度都得到了保护，第四个表格显示我们可以在插入了表格后添加其他单元格：该单元格自由地添加到下一个单元格 中。最后是一个深度嵌套的表格。

当你使用 insertTable 方法时，插入表的宽度百分比不会被考虑，如果你希望插入表仅占单元格的 80%（这是默认的宽度百分比），你不得不在单元格中绕排，见示例代码 0515，这也是让一个表结合其他数据存放在同一个单元格中的唯一办法，见示例代码 0516。

备注：你只能将一个表格插入到列跨度和行跨度均为 1 的单元格中。

表格偏移

当一个表格被添加到文档之前，以当前间距为准的新行将被添加（如前一个插入对象的间距）。有时因为前一个插入对象和当前表格间的间距过大或过小你

并不希望这样做，如果你想改变这个空间，你不得不设置表格偏移，如示例代码0517。

表格的绝对位置

iTextSharp.text.Table 是一个通过标准方法创建表格的相当简单的类，但有时你希望表格有一些特殊的行为，这种情况下你将使用更复杂的类com.lowagie.text.pdf.PdfPTable，示例代码 0518 是一个非常简单的例子，在第十章和十二章中将有一些更复杂的例子。

第六章 图片

Image 对象

如果你学习过 API，你可能已经注意到可以通过几种构造函数来创建图片，本手册中，我们将仅仅告诉你最简单的解决方案，如访问通过文件名或 URL 确定的图片生成的 Image 对象。

```
public static Image getInstance(Uri url)

public static Image getInstance(string filename)
```

Image 是一个抽象类，故得到实例的方法将判断给出的图片的类别（GIF、Jpeg、PNG……）并返回对象的类别 Gif、Jpeg、Png……，一些图片会被忽略，如果你想知道哪些图片会被忽略，请查阅FAQ（http://www.lowagie.com/iText/faq.html#images）。

◆ 通过 URL 得到图片实例

这是添加一个图片最简单的办法，见示例代码 0601，我们添加了一个 WMF、一个 Gif、一个 Jpeg 和一个 PNG 图片到文档中，使用 4 个 URL 得到：

```
Image wmf = Image.getInstance(new
URL("../examples/harbour.wmf"));
```

```
Image gif = Image.getInstance(new
URL("../examples/vonnegut.gif"));
Image jpeg = Image.getInstance(new
URL("../examples/myKids.jpg"));
Image png = Image.getInstance(new
URL("../examples/hitchcock.png"));
```

备注：许多 PDF 库在插入一个图片前都将其解压缩并转换成位图格式，下面是几个我为什么不这样做的原因：

- 这将导致 PDF 文件增大，这样产生的 PDF 文件尺寸是不同图片文件尺寸总和的数十倍。
- 面临一个法律问题：LZW 算法受专利保护，所以不允许使用这种算法来解压缩 GIF 等文件。

◆ 通过文件名得到图片实例

通过简单地改变图片引用路径将示例代码 0601 改成示例代码 0602：

```
Image gif = Image.getInstance("vonnegut.gif");
Image jpeg = Image.getInstance("myKids.jpg");
Image png = Image.getInstance("hitchcock.png");
```

同示例代码 0601 的区别只是该图象从本地获取而已，另外一个例子见示例代码 0603。

图片的位置

◆ 对齐方式

通过下面方法设置图片的对齐方式：

```
Alignment = Image.RIGHT
Alignment = Image.MIDDLE
Alignment = Image.LEFT
```

参见示例代码 0604。

我们将 Vonnegut 的图片放在右边，小孩的图片放在中间，hitchcock 的图片放在左边。

◆ 图片和文本

另外，你还可以指定文本相对图片的环绕方式：

```
Alignment = Image.RIGHT | Image.TEXTWRAP
Alignment = Image.MIDDLE
Alignment = Image.LEFT | Image.UNDERLYING
```

见示例代码 0506，文字在 Vonnegut 图片的左侧，并不在我小孩的图处环绕，且排在 Hitchcock 图片的上面。

说明：该功能尚有一些 BUG。

◆ 绝对位置

当制作 PDF 文件时，你可能用到该方法：

```
public void setAbsolutePosition(int absoluteX, int absoluteY)
```

将一个图片放要页面上一个绝对位置的代码见示例代码 0606，我们在不同的坐标处添加了两幅图片，这里使用给定的坐标将图片放在了左下角，通过将图片的宽度和高度作为 X 和 Y 坐标将设置第一个图片，坐标的 2 倍设置第二个图片。

缩放和旋转图片

◆ **缩放**

有几种办法可以缩放图片：

```
public void scaleAbsolute(int newWidth, int newHeight)
public void scalePercent(int percent)
public void scalePercent(int percentX, int percentY)
public void scaleToFit(int fitWidth, int fitHeight)
```

小孩的图片大小为 194×202 象素，如果你想让图片小一些，你可以通过 scaleAbsolute(97, 101)进行缩放，使用 scalePercent(50)也能到达同样的效果。

还可以通过 scaleAbsolute(194, 101)来减小，所以这些例子都放在了示例代码 0607 中。

◆ **对分辨率的影响**

如果一个图片不经任何缩放，其分辨率（resolution）为 72，如果该图片缩放比例为 50%，则分辨率为 144，如果有更低的缩放比，则分辨率将更大，因为象素相同但尺寸变得更小了。使用 72/300=24%的比例放置一个 300dpi 的图片，例：你用 300dpi 扫描了一个 5×5 英寸的图片，图片结果为 1500×1500 象素（5×300），当你用 24%（72/300=0.24）的比例将该图片放置到 PDF 文件中时，PDF 中的图片将为 5×5 英寸 1500X1500 象素 300dpi，该图片将始终为 1500X1500 象素而不管尺寸如何。

◆ **旋转**

可以通过下面的方法旋转图片

```
public void setRotation(double r)
```

详见示例代码 0608。

原始图片数据

到目前为止，所有例子中使用的图片均来自地本地磁盘或者某个网站，但也可能使用包含图片信息的数组来得到图片的实例：

```
public static Image getInstance(byte[] img)
```

该方法同前面方法的效果相同，返回一个新的 Gif，Jpeg 或者 Png 类别的 Image 对象。

在示例代码 0609 中，我们添加一个从一个 Jpeg 文件中读入到字节数组中的图片，很明显，使用其他 getInstance 方法得到实例更优越，但这仅仅是一个例子，该 getInstance 方法在动态创建那些根本不存在的图片时非常有用。

该例子也演示了如何创建和使用一个原始图片。

```
public static Image getInstance(int width, int height, int components, int bpc, byte data[])
```

本例中创建了一个 100×100 象素的图片，因为每个象素用 RGB 描述，所以图片大小为 100×100×3 字节。

System.Drawing.Bitmap

示例代码 0610 是一个比较高级的话题，理由如下：

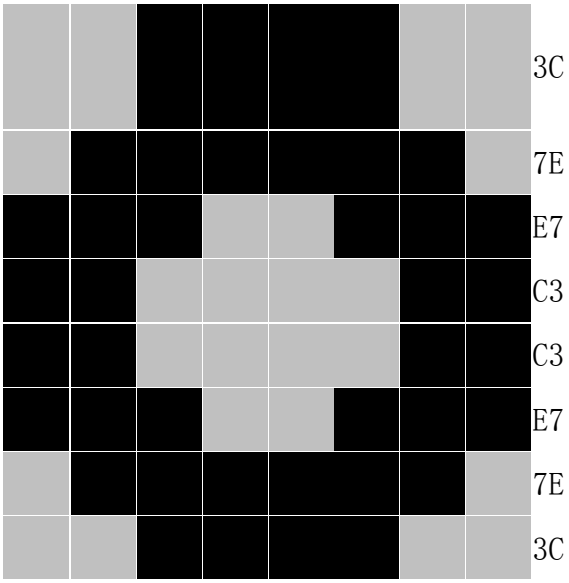
1. 首次使用到了 System.Drawing.Bitmap 类。该类在.net 框架中，可以使用更多类型的图片，如 TIFF、GIF，而这些图片在 iText 中均不支持，你可以检查一下.net 框架看看你需要的图片格式是否得到支持。
2. 前面的例子中，还有一些事情要注意：当添加一个图片时不会出现什么问题，文字始终浮于图片上面，本例中，我们希望图片浮在文字上面。这也是为什么我们将使用 iTextSharp.text.pdf.PdfContentByte 类的原因（见第十章）。
3. 你将发现使用的图片为透明的 gif 格式，你可以到 http://itextsharp.sourceforge.net/examples/h.gif 下载得到。

TIFF 和 CCITT

示例代码 0611 也是一个比较高级的话题，例中转换一个 TIFF 文件到 PDF 文件。

◆ 图片遮罩

示例代码 0613 在，我们创建了一个用作遮罩的图片



该图片尺寸为 8×8 象素，每组一个字节，使用 makeMask()方法可以转换成遮罩。

```
byte maskr[] = {(byte)0x3c, (byte)0x7e, (byte)0xe7, (byte)0xc3,
(byte)0xc3, (byte)0xe7, (byte)0x7e, (byte)0x3c};
Image mask = Image.getInstance(8, 8, 1, 1, maskr);
mask.makeMask();
mask.setInvertMask(true);
```

我们可以用该遮罩直接遮住其他图片的一部分。

```
PdfContentByte cb = writer.DirectContent;
Image image = Image.getInstance("vonnegut.gif");
image.ImageMask = mask;
```

或者我们将该遮罩用于模板遮罪中。

```
PdfContentByte cb = writer.DirectContent;
cb.setRGBColorFill(255, 0, 0);
cb.addImage(mask, mask.scaledWidth() * 8, 0, 0, mask.scaledHeight()
* 8, 100, 400);
```

关于 ContentByte 对象更多信息请参见第十章。

图片和其他对象

◆ 图片在块中

有时，可以方便地将图片置于块中，通过一定偏移将一个图片置于块中：

```
Chunk ck = new Chunk(img, 0, -5);
```

具体代码见示例代码 0614，我们可以添加该特殊图片块到短句、表格等，本例中的图片请到

http://itextsharp.sourceforge.net/examples/pngnow.png 下载。

◆ 图片在表格中

你可以将图片添加到单元格中，但有两个副作用：

- 表格的宽度是确定，当图片超出单元格的宽度时，将自动缩小。
- 你不能进行文字绕排和为图片添加下划线。

参见示例代码 0615。

◆ 图片链接注释

如果你希望得到一个可点击的图片，或者想添加链接注释到图片上，你需要创建一个 Annotation 对象，并添加到图片上，你不需要指定位置（你可以使用 0,0,0,0），该位置会内部更新以适合该图片。

```
gif.Annotation = new Annotation(0, 0, 0, 0, "Chap1102b.pdf", 3);
jpeg.Annotation = new Annotation("picture", "These are my children", 0, 0, 0, 0);
```

参加示例代码 0616。

第二部分 其他文档格式

第七章 XML 和 (X)HTML

本章主要介绍了如何利用 iText 控件生成 XLM 文档和（X）HTML 文档，但我们对这些并不感兴趣，故只介绍本章中提到的将 XML 转为 PDF。

在第一章中，我们通过 5 步生产一个 PDF 文件，为了将一个 XML 文件转换为PDF 文件，只需重写第 3 和第 4 步，第 5 步由解析器自动处理。

```
//第 3 步：创建一个解析器并设置文档句柄：
iTextHandler h = new iTextHandler(document);
//第 4 步，转换该文档：
h.Parse("Chap0701.xml");
```

示例代码见示例代码 0702

第八章 RTF 文件

RTF 包

RTF 包是基于 iText 包扩展出来的，允许 iText 除生成 PDF 文件外还可以输出 RTF 文件，除了一些在 RTF 包中不支持的特性外，大多数 PDF 文件特性都可以使用。

创建一个 RTF 文档

创建一个 RTF 文档和创建一个 PDF 文档方法是一样的，都是这基本的 5 步，唯一的区别是第 2 步中用 RtfWriter 代替了 PdfWriter，见示例代码 0801。

第 1 步 创建一个 the iTextSharp.text.Document 对象的实例：

```
Document document = new Document();
```

第 2 步 创建一个 document 的 RtfWriter 将 document 写入你选择的输出流：

```
RtfWriter.getInstance(document, new
FileStream("Chap0801.rtf"), FileMode.Create);
```

第 3 步 打开 document:

```
document.Open();
```

第 4 步 添加内容到 document

```
document.Add(new Paragraph("Hello World"));
```

第 5 步 关闭 document

```
document.Close();
```

关于如何创建其他对象并添加到 document 中，请参见其他章节的内容。

不支持的特性

- 水印
- 阅读器参数
- 加密

- 内嵌字体
- 块间距
- 段落右缩排
- 列表右缩排
- 无圆点符号列表
- 嵌套表格
- 除 JPEG 和 PNG 的其他图片

RTF 中扩展的页眉和页脚

写入 RTF 时无法在开始新页前通过 setHeader 方法改变文档的页眉或页脚，这里有两个办法来解决这个问题。

- 利用“Chapters”，添加一个新“chapter”到文档前，使用 setHeader 或 setFooter，你可以在不同的“Chapters”中使用不同的页眉或页脚，见示例代码 0802。
- 使用 RtfHeaderFooters 类。该类允许你设置 4 个页眉或页脚，并指定在哪页出现。你当然可以结合 Chapter 创建 4 个不同的页眉或页脚，见示例代码 0803。

◆ 使用 RtfHeaderFooters 类

第 1 步创建一个 RtfHeaderFooters 类：

```
RtfHeaderFooters headers = new RtfHeaderFooters();
```

第 2 步添加 HeaderFooter 对象

```
headers.Add(RtfHeaderFooters.LEFT_PAGES, new HeaderFooter(new Phrase("This header is only on left hand pages")));
headers.Add(RtfHeaderFooters.RIGHT_PAGES, new HeaderFooter(new Phrase("This header is only on right hand pages")));
```

第 3 步如同使用页眉页脚一样使用 RtfHeaderFooters

```
document.Header = headers;
```

使用 RtfHeaderFooters.add(...)的常量：

- **FIRST_PAGE:** 在你文档的第一页使用该页眉或页脚。你将使用 `rtfWriter.HasTitlePage = true` 来完成
- **LEFT_PAGES:** 所有左边页均使用该页眉或页脚
- **RIGHT_PAGES:** 所有右边页均使用该页眉或页脚
- **ALL_PAGES:** 所有页均使用该页眉或页脚，只有和 **FIRST_PAGE** 结合使用才有意义。

有一件事非常重要：如果你使用 LEFT_PAGES 或者 RIGHT_PAGES 来设置页眉或页脚，再使用 ALL_PAGES，页眉和页脚均不会起作用。

表格效果见示例代码 0804。

第三部分 iText 的高级应用

第九章 字体

本章原文讲了许多字体的使用技巧，但就是没有讲如何使用中文，因此，意义不大，再说，如果不支持中文，前面的也就白翻译了，因此，根据原文讲到的一些知识，我摸索出汉字的使用方法，自己写了本章内容，应该算是“原创”了吧^_^(哎呀！谁拿鸡蛋扔我……)。

Windows 中一般都是使用 TrueType 字体，每个中文版 Windows 操作系统均默认安装了宋体、仿宋、黑体和楷体四种字体，你还可以安装其他第三方字体，如安装了 Office 2000 后，会自动安装华文行楷等字体，比较奇怪的是，在 PDF 文件中插入了一种本计算机才有的字体，在打开 PDF 文件的计算机上虽然没有该字体，但仍然能正常显示！这有别于 Word 文件，Word 文件将当前计算机中没有的字体一律用宋体代替，这大概是意外收获吧。

字体文件一般保存在 windir\Fonts 目录中，扩展名为 TTF，还有扩展名为 TTC 的字体文件，也是 TrueType 字体，不过是一个集合，也就是里面有多种字体。下面列出 windows2000 简体中文版四种标准字体的文件名称：

SIMSUN.TTC：宋体和新宋体

SIMKAI.TTF：楷体

SIMHEI.TTF：黑体

SIMFANG.TTF：仿宋体

TrueType 字体应用

按下面的方法写入黑体字文字，大小为 32 磅：

```
BaseFont bfHei = BaseFont.createFont(@"c:\winnt\fonts\SIMHEI.TTF", BaseFont.IDENTITY_H, BaseFont.NOT_EMBEDDED);
Font font = new Font(bfHei, 32);
String text = "这是黑体字测试！";
document.Add(new Paragraph(text, font));
```

不要管 BaseFont.createFont 方法第二、三个参数的意思，依葫芦画瓢就行了，第一个参数显示就是字体文件存放的位置。

后面的代码都非常好理解，不再赘述。

TruType 字体集合的应用

字体集合的使用同上面差不多，只是在在 createFont 方中要指定使用哪种字体。如：

```
BaseFont bfSun=BaseFont.createFont(@"c:\winnt\fonts\SIMSUN.TTC,1", BaseFont.IDENTITY_H, BaseFont.NOT_EMBEDDED);
font = new Font(bfSun, 16);
text = "这是字体集合中的新宋体测试！";
document.Add(new Paragraph(text, font));
```

不难看出，在使用 BaseFont.createFont 方法时，第一个参数 @"c:\winnt\fonts\SIMSUN.TTC,1"中多了一个“,1”，表示使用序号为 1 字体，序号为 0 的字体为宋体。

毕竟我们不是做排版软件，有了上面的办法就基本上够用了，真正很复杂的 PDF 文件制作，不妨做成 XML 文件（最简单的办法就是用 Word 排版，然后另存为 web 页了），然后按第七章的办法转换。

代码见示例代码 0901。

第十章 图象和文本的绝对位置

pdfContentByte

到目前为止，我们已经使用了简单的 iText，我们已经添加了文本、图片、段落、章节、列表、表格等，没有涉及到布局问题。Itext 分割文本到每页中，并将每个单词、句子、段落布置到页面上，但有时我们并不需要这种自动格式，有时我们希望将一些图象或者文本放置在某页的指定位置，为实现该功能，我们将使用 PdfContentByte 类。

为代替第一章，仅用 PdfWriter 类的 getInstance 方法是不够的，你必须真实地拥有一个 PdfWriter 对象，你可以通过在使用 Writer 对象中使用 getDirectContent()方来得到该对象。例：

```
PdfWriter writer = PdfWriter.getInstance(document, new FileOutputStream("test.pdf"));
PdfContentByte cb = writer.DirectContent;
```

说明：当你添加高级对象（如表格）时，两个 PdfContentByte 对象将被内部使用：一个用于文本，一个用用于图象（如边界或单元格背景）。文本绘制浮于图象的上面。

当你通过 getDirectContent()方法直接使用 PdfContentByte 对象时，你所添加的所有对象都将浮于文本和图象。如果你想避免这种情况和希望添加内容在图象或文本的背后，你需要使用用 getDirectContentUnder()方。

一句话，当一页完成时，4 层的重叠遵照如下顺序：

1、 通过 getDirectContentUnder()得到的 PdfContentByte

2、 包含图象或高级对象的内部 PdfContentByte

3、 病文本或高级对象的内部 PdfContentByte

4、 通过 getDirectContent()得到的 PdfContentByte

简单图形

在示例代码 1001 中，绘制了一些简单图形，我们使用了诸如 moveTo 和 lineTo 方法来在移动到页面上当前位置然后画一条直线到其他位置。我们使用了诸如 setLineWidth 和 setLineDash 方法来改变直线的外观，如：

```
cb.LineWidth = 10f;
cb.moveTo(100, 700);
cb.lineTo(200, 800);
cb.stroke();
```

说明：当你改变诸如颜色、线宽等属性时，只有你在调用 stroke 方法时才起作用。在例中绘制三角形时，我们设置颜色为绿色，在使用 stroke 方法前我们改变颜色为红色，则绘制三角形的结果为为红色而不是绿色，该例中还有矩形、圆等使用方法。

文本

当你想将文本写入 ContentByte 中时，你必须使用方法 beginText()和 endText，你也必须设置字体和尺寸。就象图形示例中一样，还有许多方法用于写入和放置文本，但你最需要的是方法 showTextAligned 和方法 showText 配合 setTextMatrix。

例 1：

```
BaseFont bf = BaseFont.createFont(BaseFont.HELVETICA, BaseFont.CP1252, BaseFont.NOT_EMBEDDED);  cb.beginText();
cb.setFontAndSize(bf, 12);
cb.showTextAligned(PdfContentByte.ALIGN_CENTER, text + "This text is centered", 250, 700, 0);
cb.endText();
```

例 2：

```
BaseFont bf = BaseFont.createFont(BaseFont.HELVETICA,
BaseFont.CP1252, BaseFont.NOT_EMBEDDED);
cb.beginText();
cb.setFontAndSize(bf, 12);
cb.setTextMatrix(100, 400);
cb.showText("Text at position 100,400.");
cb.endText();
```

请参见示例代码 1002。

模板（Form xObjects）

当我们在第四章讨论页眉和页脚时，我们定义了一小块添加到每一页的信息，实际上，该小块信息写到了文件的每一个新页上。这并不是最经济的解决方案，更好的办法是将该信息作为一个 Form Xobject 仅在文档中添加一次，在其可见位置重复出现。我达到该目的，我们将使用模板。

- 创建一个 **PdfTemplate**
- 创建 PdfTemplate 的最好方法是调用 PdfContentByte 对象中的 createTemplate 方法：

```
PdfContentByte-object:
PdfTemplate template = cb.createTemplate(500, 200);
```

这样，该模板的宽度为 500，高度为 200。

通过该模板我们可以做象 PdfContentByte 同样的事情

```
template.moveTo(0, 200);
template.lineTo(500, 0);
template.stroke();
template.beginText();
```

```
    BaseFont bf = BaseFont.createFont(BaseFont.HELVETICA,
BaseFont.CP1252, BaseFont.NOT_EMBEDDED);
    template.setFontAndSize(bf, 12);
    template.setTextMatrix(100, 100);
    template.showText("Text at the position 100,100
(relative to the template!)");
    template.endText();
```

◆ 添加一个模板到文档

通过象下面一样在绝对位置添加一个模板：

```
cb.addTemplate(template, 0, 400);
```

你还可以做一些有趣的事情，如缩放或旋转他们：

```
//将模板旋转 90 度
cb.addTemplate(template, 0, 1, -1, 0, 500, 200);
// 缩放模板为 50%
cb.addTemplate(template, .5f, 0, 0, .5f, 100, 400);
//缩放模板为 200%
cb.addTemplate(template, 2, 0, 0, 2, -200, 400);
```

具体演示见示例代码 1003。

◆ 第几页共几页

在一些情况下，你希望插入一些你在写本页时外壳无法知道的信息到文本中去，如：在一篇文档的第一页，你并不知道该文档共有几页。只能在完成了整个文档时才知道总的页数。当使用模板时，该问题就不存在了。在示例代码 0103 中，我们在添加模板到 ContentByte 前添加了一些信息到模板中，这是没有必要的。我们可以在任何时候添加信息到模板，因为 iText 添加 Form Xobject 是在PDF 结束的地方（当通过 close 方法关闭该文档时调用）。示例代码 1004 显示了首先创建 4 页然后添加总到页数，该例非常简单和有用。

分栏

在本章以前，你已经掌握了如何将文本放在一个绝对位置，这种情况下，我们要确定文本的开始坐标。如果我们想知道文本的结束位置，我们得做一些计算工作。

现在我们要加一些文本到一个矩形框的内部，希望文本到达右边界时自动换行。超出矩形部分将不显示，可以通过 ColumnText 类实现。

举个例子：

为显示一个指定的短句在坐标(100, 300)和(200, 500)间的矩形内居中，我们使用下面的代码：

```
PdfContentByte cb = writer.DirectContent;
ColumnText ct = new ColumnText(cb);
ct.setSimpleColumn(phrase, 60, 300, 100, 500, 15,
Element.ALIGN_CENTER);
ct.go();
```

通过查看示例代码 1005，你会立即发现通过该方法可以可以画一些复杂的表格而无须 Table 对象。

另一个例子：

没有必要一次性将文本全部添加进去，你可以先定义一个矩形，然后添加一些文本，最后用 go 方法显示分栏。

```
PdfContentByte cb = writer.DirectContent;
ColumnText ct = new ColumnText(cb);
ct.setSim7pleColumn(60, 300, 100, 500, 15,
Element.ALIGN_CENTER);
ct.addText(phrase1);
ct.addText(phrase2);
ct.addText(phrase3);
```

```
ct.go();
```

详见示例代码 1006。

多栏

当然，如果文本超出了矩形范围，我们并不想丢失这些多出的文本，或许我们想将这些文本显示到其他栏中。这就是为什么我们要查看 go 方法返回值的原因。如果返回标识为“NO_MORE_COLUMN”，表示该栏中没有足够的空间存放该文本，如果所有的文本均显示出来，标识将为“NO_MORE_TEXT”。

请参见示例代码 1007。

不规则栏

定义一个非矩形的区域来显示栏也是可能的,通过使用 setColumns 方法,我们为文本定义了一个左右边界。

```
float[] left = {70,790, 70,60};
float[] right = {300,790, 300,700, 240,700, 240,590, 300,590, 300,106, 270,60};
ct.setColumns(left, right);
```

左边界是一条直线，而右边界是不规则的。该函数的结果可以导致一些非常有意思的布局，见示例代码 1008，本例中你将用到一个名为 caesar_coin.jpg 的图片：

PdfTable

在第 5 章中，我们简要地讲述了 PdfPTable 对象，现在我们将讨论该对象更多的的特性。

你可以用 3 种不同的方法创建 PdfTable：

```
PdfPTable(float[] relativeWidths);
PdfPTable(int numColumns);
```

```
PdfPTable(PdfPTable table);
```

你可以给该表设置更多的参数，如表宽度、列宽度、水平对齐方式等，你可以通过下面的办法添加单元格：

```
public void addCell(PdfPCell cell);
public void addCell(PdfPTable table);
public void addCell(Phrase phrase);
public void addCell(String text);
```

除了单元格填距和和间距，这些方法同 Table 对象非常类似。这些参数对每个单元格个体进行了设置，当然，你可以设置单元格的默认值，为改变单元格的默认值，使用 getDefaultCell()和调用一个或更多的类 PdfPCell 的方法（你可以设置对齐方式、间距、边框、颜色甚至最低高度）。

注：通过 PdfPTable，你能改变一个单元格的列跨度，但不能改变行跨度！在 PdfPTable 内部是一些独立的行，要让它支持行跨度更改需要对 PdfPTable 对象进行很大的调整，不要期望在近期内实现，你可以用嵌套表来解决这些问题。

你可以象第 5 章一样将一个 PdfPTable 添加到当前文档中，但你也可以添加一个表在当前页中的绝对位置：

```
public float writeSelectedRows(int rowStart, int rowEnd,
 float xPos, float yPos, PdfContentByte canvas);
```

参数 rowStart 是你想开始的行的数目，参数 rowEnd 是你想显示的最后的行（如果你想显示所有的行，用-1），xPos 和 yPos 是表格的坐标，canvas 是一个 PdfContentByte 对象。在示例代码 1009 中，我们添加了一个表在(100,600)处：

```
table.writeSelectedRows(0, -1, 100, 600,
writer.DirectContent);
```

使用 PdfPTable，你不能设置行跨度和（或）来跨度（怎么和上面的有点矛盾？）你可以使用嵌套表来解决，见示例代码 1010。

最后，示例代码 1011 和示例代码 1012 展示了 PdfTable 可以和 templates 和 columns 一起使用，在示例代码 1012 中将用到 cover.png 图片如下：

颜色（SpotColors）和图案(Patterns)

颜色（spotcolors）的使用见示例代码 1013，示例代码 1014 和示例代码 1015 演示了图案(patterns)的使用方法。

第十一章 本地和异地转向、目标和概要

本地转向

有时你需要一个允许读者从文档的一个地方跳转到另外一个地方的链接，你可以通过类 Chunk 的 setLocalGoto 和 setLocalDestination 两个方法实现，例：

```
Chunk localgoto = new Chunk("this word",
FontFactory.getFont(FontFactory.HELVETICA, 12, Font.NORMAL, new
Color(0, 0, 255))).setLocalGoto("test");
Chunk destination = new Chunk("local destination",
FontFactory.getFont(FontFactory.HELVETICA, 12, Font.NORMAL, new
Color(0, 255, 0))).setLocalDestination("test");
```

见示例代码 1101。

异地转向

在第 3 章中，我们演示了一个锚点如何转向到其他 URL，一个锚点通过不同的字体、风格和颜色，可以包含不同的 Chunks，在 iText 的高级应用中，下面定义链接到 URL 的其他方法：

```
Chunk chunk = new Chunk("anchor",
FontFactory.getFont(FontFactory.HELVETICA, 12)).setAnchor(new
URL("http://www.lowagie.com/iText/"));
```

◆ 转到 PDF 文档中的指定位置

如果你在文档中指定了一个目的地，你可以从另外一个文档跳转到这里，为实现该功能，你可以使用方法：

setRemoteGoto: Chunk chunk = new Chunk("jump", FontFactory.getFont(FontFactory.HELVETICA, 12, Font.ITALIC)).setRemoteGoto("test.pdf", "test"));

test.pdf 是另外一个 pdf 文件，”test”是该文件的一个目的地。

跳转到另一个 PDF 文件指定页

使用方法 setRemoteGoto，用页码参数代替名称参数，可以非常容易地跳转定另外一个文档的指定页：

chunk = new Chunk("jump", FontFactory.getFont(FontFactory.HELVETICA, 12, Font.ITALIC)).setRemoteGoto("test.pdf", 3));

见示例代码 1102

◆ 启动一个应用程序

可以使用下面的方法启动一个应用程序：

public PdfAction(String application, String parameters, String operation, String defaultDir)

如果 application 为“c:/winnt/notepad.exe”（其余参数可以为 null），你可以通过 PDF 文件中的链接来启动记事本程序。

◆ 文件和 URL

如果你想跳转到其他文档或 URL，你需要通过下面的构造函数之一创建一个：

PdfAction(String filename, String name);

PdfAction(String filename, int page);

PdfAction(URL url);

PdfAction(String url);

前面两个构造函数允许你跳转到文件的指定位置或页码，后两个构造函数允许你跳转到其他 URL 上。

其余部分略。