在本文中，讓我們策劃一個故事並使用 iText PDF 庫將我們的故事轉換為 PDF 格式。我們將編寫一個簡單的程序StoryTime 。首先，我們將聲明兩個String類型的變量。我們將我們的故事存儲在聲明的變量中：

**public** **class** **StoryTime** {

**String** aliceStory = "I am ...";

**String** paulStory = "I am Paul ..";

}

為簡單起見，我們將縮短String值。然後，讓我們聲明一個String類型的變量，它將存儲我們生成的 PDF 的輸出路徑：

**public** **static** **final** **String** OUTPUT\_DIR = "output/alice.pdf";

最後，讓我們創建一個包含程序邏輯的方法。我們將創建一個PdfWriter實例來指定我們的輸出路徑和名稱。

接下來，我們將創建一個PdfDocument實例來處理我們的 PDF 文件。要將我們的String值添加到 PDF 文檔，我們將創建一個新的Document實例：

**public** **void** **createPdf**(String output) **throws** IOException {

**PdfWriter** writer = **new** **PdfWriter**(output);

**PdfDocument** pdf = **new** **PdfDocument**(writer);

**try** (**Document** document = **new** **Document**(pdf, PageSize.A4, false)) {

document.add(**new** **Paragraph**(aliceSpeech)

.setFont(PdfFontFactory.createFont(StandardFonts.TIMES\_ROMAN)));

document.add(**new** **Paragraph**(paulSpeech)

.setFont(PdfFontFactory.createFont(StandardFonts.TIMES\_ROMAN)));

document.close();

}

}

我們的方法將生成一個新的 PDF 文件並將其存儲在OUTPUT\_DIR中。

5.為生成的PDF添加水印

在上一節中，我們使用 iText PDF 庫生成了一個 PDF 文件。**首先生成 PDF 有助於了解頁面大小、旋轉和頁數。這有助於有效地添加水印**。讓我們為我們的簡單程序添加更多邏輯。我們的程序將為生成的 PDF 添加水印。

首先，讓我們創建一個方法來指定水印的屬性。我們將設置水印的Font 、 fontSize和Opacity ：

**public** Paragraph **createWatermarkParagraph**(String watermark) **throws** IOException {

**PdfFont** font = PdfFontFactory.createFont(StandardFonts.HELVETICA);

**Text** text = **new** **Text**(watermark);

text.setFont(font);

text.setFontSize(56);

text.setOpacity(0.5f);

**return** **new** **Paragraph**(text);

}

接下來，讓我們創建一個方法，其中包含向我們的 PDF 文檔添加水印的邏輯。該方法將Document 、 Paragraph和offset作為參數。我們將計算位置和旋轉以放置我們的水印段落：

**public** **void** **addWatermarkToGeneratedPDF**(Document document, **int** pageIndex,

Paragraph paragraph, **float** verticalOffset) {

**PdfPage** pdfPage = document.getPdfDocument().getPage(pageIndex);

**PageSize** pageSize = (PageSize) pdfPage.getPageSizeWithRotation();

**float** x = (pageSize.getLeft() + pageSize.getRight()) / 2;

**float** y = (pageSize.getTop() + pageSize.getBottom()) / 2;

**float** xOffset = 100f / 2;

**float** rotationInRadians = (**float**) (PI / 180 \* 45f);

document.showTextAligned(paragraph, x - xOffset, y + verticalOffset,

pageIndex, CENTER, TOP, rotationInRadians);

}

我們通過調用showTextAligned()方法將水印段落添加到我們的文檔中。接下來，讓我們編寫一個生成新 PDF 並添加水印的方法。我們將調用createWatermarkParagraph()方法和addWatermarkToGeneratedPDF()方法：

**public** **void** **createNewPDF**() **throws** IOException {

**StoryTime** storyTime = **new** **StoryTime**();

**String** waterMark = "CONFIDENTIAL";

**PdfWriter** writer = **new** **PdfWriter**(storyTime.OUTPUT\_FILE);

**PdfDocument** pdf = **new** **PdfDocument**(writer);

**try** (**Document** document = **new** **Document**(pdf)) {

document.add(**new** **Paragraph**(storyTime.alice)

.setFont(PdfFontFactory.createFont(StandardFonts.TIMES\_ROMAN)));

document.add(**new** **Paragraph**(storyTime.paul));

**Paragrapgh** paragraph = storyTime.createWatermarkParagraph(waterMark);

**for** (**int** i = 1; i <= document.getPdfDocument().getNumberOfPages(); i++) {

storyTime.addWatermarkToGeneratedPDF(document, i, paragraph, 0f);

}

}

}

最後，讓我們編寫一個單元測試來驗證水印的存在：

**@Test**

**public** **void** **givenNewTexts\_whenGeneratingNewPDFWithIText**() **throws** IOException {

**StoryTime** storyTime = **new** **StoryTime**();

**String** waterMark = "CONFIDENTIAL";

**LocationTextExtractionStrategy** extStrategy = **new** **LocationTextExtractionStrategy**();

**try** (**PdfDocument** pdfDocument = **new** **PdfDocument**(**new** **PdfReader**(storyTime.OUTPUT\_FILE))) {

**for** (**int** i = 1; i <= pdfDocument.getNumberOfPages(); i++) {

**String** textFromPage = getTextFromPage(pdfDocument.getPage(i), extStrategy);

assertThat(textFromPage).contains(waterMark);

}

}

}

我們的測試驗證了我們生成的 PDF 中是否存在水印。

6. 為現有 PDF 添加水印

**iText PDF 庫使向現有 PDF 添加水印變得容易**。我們將首先將我們的 PDF 文檔加載到我們的程序中。並使用 iText 庫來操作我們現有的 PDF。

首先，我們需要創建一個方法來添加水印段落。由於我們在上一節中創建了一個，因此我們也可以在此處使用它。

接下來，我們將創建一個方法，其中包含幫助我們向現有 PDF 添加水印的邏輯。該方法將接受Document 、 Paragraph 、 PdfExtGState, pageIndex, and offSet作為參數。在該方法中，我們將創建一個新的PdfCanvas實例以將數據寫入我們的 PDF 內容流。

然後，我們將計算水印在 PDF 上的位置和旋轉。我們將刷新文檔並釋放狀態以提高性能：

**public** **void** **addWatermarkToExistingPDF**(Document document, **int** pageIndex,

Paragraph paragraph, PdfExtGState graphicState, **float** verticalOffset) {

**PdfDocument** pdfDocument = document.getPdfDocument();

**PdfPage** pdfPage = pdfDocument.getPage(pageIndex);

**PageSize** pageSize = (PageSize) pdfPage.getPageSizeWithRotation();

**float** x = (pageSize.getLeft() + pageSize.getRight()) / 2;

**float** y = (pageSize.getTop() + pageSize.getBottom()) / 2;

**PdfCanvas** over = **new** **PdfCanvas**(pdfDocument.getPage(pageIndex));

over.saveState();

over.setExtGState(graphicState);

**float** xOffset = 14 / 2;

**float** rotationInRadians = (**float**) (PI / 180 \* 45f);

document.showTextAligned(paragraph, x - xOffset, y + verticalOffset,

pageIndex, CENTER, TOP, rotationInRadians);

document.flush();

over.restoreState();

over.release();

}

最後，讓我們編寫一個方法來為現有的 PDF 添加水印。我們將調用createWatermarkParagraph()來添加水印段落。此外，我們將調用addWatermarkToExistingPDF()來處理向頁面添加水印的任務：

**public** **void** **addWatermarkToExistingPdf**() **throws** IOException {

**StoryTime** storyTime = **new** **StoryTime**();

**String** outputPdf = "output/aliceNew.pdf";

**String** watermark = "CONFIDENTIAL";

**try** (**PdfDocument** pdfDocument = **new** **PdfDocument**(**new** **PdfReader**("output/alice.pdf"),

**new** **PdfWriter**(outputPdf))) {

**Document** document = **new** **Document**(pdfDocument);

**Paragraph** paragraph = storyTime.createWatermarkParagraph(watermark);

**PdfExtGState** transparentGraphicState = **new** **PdfExtGState**().setFillOpacity(0.5f);

**for** (**int** i = 1; i <= document.getPdfDocument().getNumberOfPages(); i++) {

storyTime.addWatermarkToExistingPage(document, i, paragraph,

transparentGraphicState, 0f);

}

}

}

讓我們編寫一個單元測試來驗證水印的存在：

**@Test**

**public** **void** **givenAnExistingPDF\_whenManipulatedPDFWithITextmark**() **throws** IOException {

**StoryTime** storyTime = **new** **StoryTime**();

**String** outputPdf = "output/aliceupdated.pdf";

**String** watermark = "CONFIDENTIAL";

**LocationTextExtractionStrategy** extStrategy

= **new** **LocationTextExtractionStrategy**();

**try** (**PdfDocument** pdfDocument = **new** **PdfDocument**(**new** **PdfReader**(outputPdf))) {

**for** (**int** i = 1; i <= pdfDocument.getNumberOfPages(); i++) {

**String** textFromPage = getTextFromPage(pdfDocument.getPage(i), extStrategy);

assertThat(textFromPage).contains(watermark);

}

}

}

我們的測試驗證了現有 PDF 中是否存在水印。

七、結論

在本教程中，我們通過生成新的 PDF 探索了 iText PDF 庫。我們在生成的 PDF 中添加了水印，隨後又在現有 PDF 中添加了水印。 iText 庫在處理 PDF 方面看起來很強大。 [GitHub 上](https://github.com/eugenp/tutorials/tree/master/libraries-files)提供了完整的代碼。