

# Tutorial PDFBox v1.0



Ivan Salvadori, ivanls@inf.ufsc.br  
Florianópolis, Santa Catarina  
03/11/2011

“ The Apache PDFBox™ library is an open source Java tool for working with PDF documents. This project allows creation of new PDF documents, manipulation of existing documents and the ability to extract content from documents. Apache PDFBox also includes several command line utilities. Apache PDFBox is published under the [Apache License v2.0](#) ”

Como descrito acima, PDFBox é uma biblioteca que permite manipular arquivos PDF, seja para criação de novos documentos ou mesmo de documentos já existentes, permitindo a extração de seu conteúdo. É uma ferramenta desenvolvido pelo Apache Software Foundation, disponível no endereço <http://pdfbox.apache.org/>.

Essa ferramenta possui as seguintes características:

## Features

- PDF to text extraction
- Merge PDF Documents
- PDF Document Encryption/Decryption
- Lucene Search Engine Integration
- Fill in form data FDF and XFDF
- Create a PDF from a text file
- Create images from PDF pages
- Print a PDF

## Objetivos

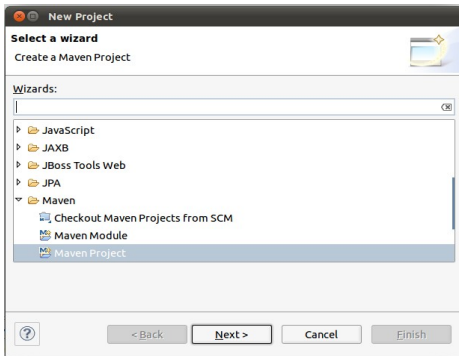
Este tutorial demonstra o uso do PDFBox, implementando exemplos de cada funcionalidade disponível pela biblioteca. Usaremos o Eclipse IDE Indigo, compilando com Java 1.6, juntamente com Apache Maven, porém os exemplos podem ser realizados em qualquer outro ambiente de desenvolvimento, sem comprometer nenhuma funcionalidade.

Com o objetivo secundário, busca apresentar o Apache Maven aos desenvolvedores que estão iniciando com essa ferramenta. Caso você já conheça o maven, pode pular estes detalhes.

# Criando o projeto

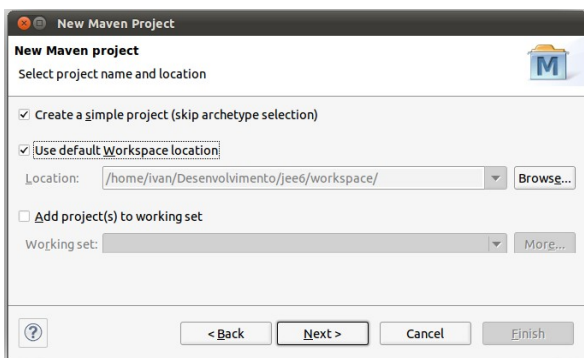
Antes de dar inicio a implementação dos exemplos, vamos configurar o ambiente de desenvolvimento e criar o projeto no Eclipse, que será usado para implementar todas as funcionalidades do PDFBox.

Criar um projeto Maven.

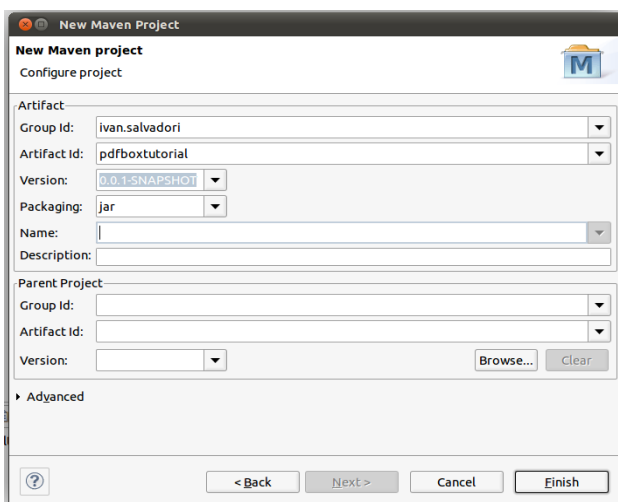


Marcar a opção **Create simple project (skip archetype selection)**.

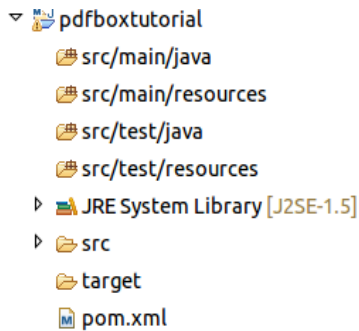
Criaremos um projeto maven simples, sem escolher nenhum modelo de projeto já definido.




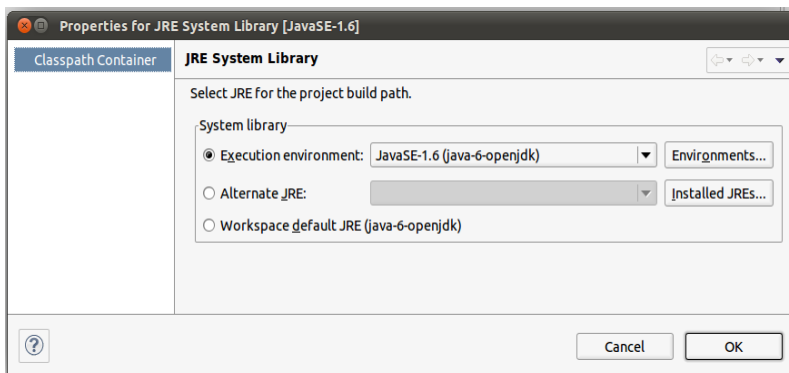
Completar o Group Id e Artifact Id da forma que julgar mais adequado. Group Id representa a entidade que está desenvolvendo o software, Artifact Id é o nome do sistema sendo desenvolvido. Neste tutorial, o software será desenvolvido para ser executado como um programa, executável diretamente pelo usuário, no sistema operacional, mas pode ser desenvolvido como parte integrante de um sistema web ou webservice. Dessa forma usaremos o empacotamento **jar**.



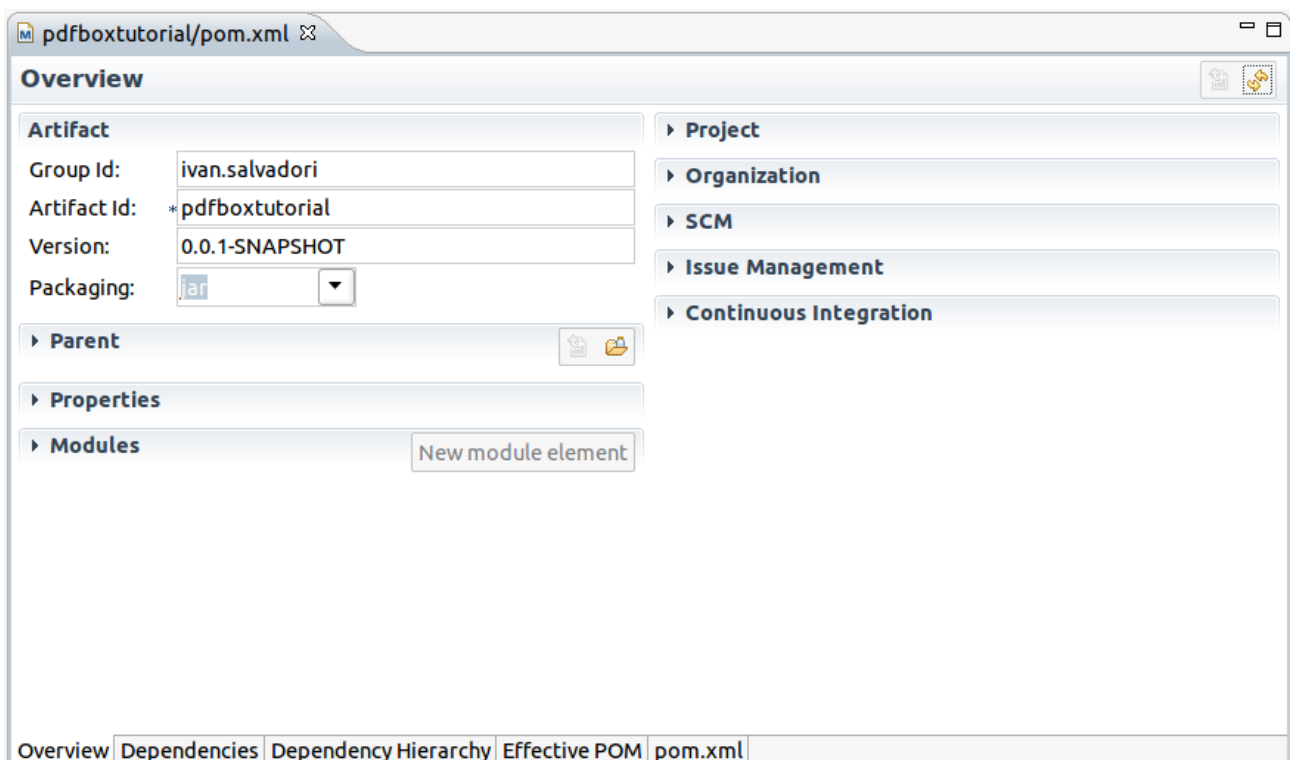
Realizados os passos descrito acima, teremos o projeto criado com a seguinte estrutura.



Note que o java 1.5 está selecionado como padrão, vamos alterar para utilizar o java 1.6; Com o direito do mouse, clique em JRE System Library,  JRE System Library [J2SE-1.5] escolha o menu propriedades. Alterar para JavaSE 1.6.



O arquivo pom.xml guarda as configurações do Apache Maven. Clicando duas vezes sobre o pom.xml, será exibido um detalhamento destas configurações.



Clicando na aba **pom.xml** no canto inferior direito, Overview Dependencies Dependency Hierarchy Effective POM pom.xml  
O conteúdo do pom.xml será exibido.

```
pdfboxtutorial/pom.xml
1 <project xmlns="http://maven.apache.org/POM/4.0.0"
2       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3       xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apa
4 <modelVersion>4.0.0</modelVersion>
5 <groupId>ivan.salvadori</groupId>
6 <artifactId>pdfboxtutorial</artifactId>
7 <version>0.0.1-SNAPSHOT</version>
8 </project>
```

Para poder trabalhar com o Apache PDFBox, é necessário fazer o download da biblioteca, baixando o jar da ferramenta. Porém com o Apache Maven isso não é necessário, basta informar que o software que está sendo desenvolvido depende da biblioteca do PDFBox. Isso é feito no pom.xml, adicionando a dependência do PDFBox. O pom.xml ficará da seguinte forma:

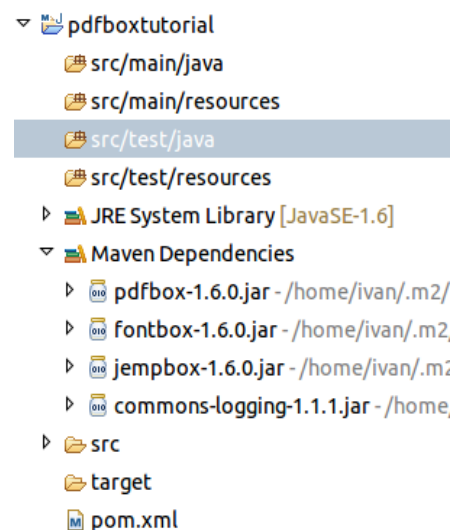
```
*pdfboxtutorial/pom.xml
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3
2       xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apac
3 <modelVersion>4.0.0</modelVersion>
4 <groupId>ivan.salvadori</groupId>
5 <artifactId>pdfboxtutorial</artifactId>
6 <version>0.0.1-SNAPSHOT</version>
7
8 <dependencies>
9
10 <dependency>
11 <groupId>org.apache.pdfbox</groupId>
12 <artifactId>pdfbox</artifactId>
13 <version>1.6.0</version>
14 </dependency>
15
16 </dependencies>
17
18 </project>
19
```

Na tag version, colocamos a ultima versão disponível do PDFBox.

```
<dependency>
  <groupId>org.apache.pdfbox</groupId>
  <artifactId>pdfbox</artifactId>
  <version>1.6.0</version>
</dependency>
```

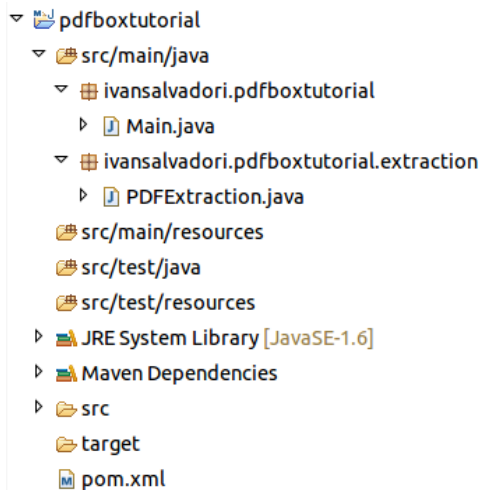
Ao salvar o pom.xml, todos os jar necessários para trabalhar com o PDFBoz serão automaticamente baixados da internet, sem a necessidade de fazer o download manualmente e adicioná-los ao projeto.

Com isso o projeto já está configurado para dar inicio a implementação das funcionalidades do PDFBox.

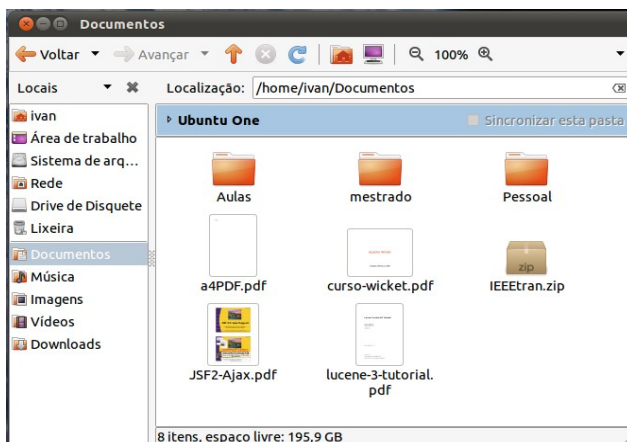


# Extração de textos de documentos PDF

A extração de texto em PDF é uma das principais funcionalidades do PDFBox. Para realizar a extração, vamos criar duas Classes, PDFExtraction, que implementará a funcionalidade de extração e a Classe Main, que executará os métodos implementados.



Para o exemplo, vamos extrair o texto de um arquivo PDF existente no disco.



Vamos implementar um metodo que irá extrair o texto contido em um arquivo PDF, recebendo como parâmetro o nome e o caminho completo do arquivo desejado.

```
1 public String extrairTexto(String caminhoArquivo) {
2
3     String textoExtraido = null;
4
5     File doc = new File(caminhoArquivo);
6
7     PDDocument pdDoc;
8     try {
9         pdDoc = PDDocument.load(doc);
10        PDFTextStripper stripper = new PDFTextStripper();
11        textoExtraido = stripper.getText(pdDoc);
12    } catch (IOException e) {
13        e.printStackTrace();
14    }
15
16    return textoExtraido;
17 }
18 }
```

## Explicando

**Linha 1:** Criado um método que retorna a String do texto extraído de um arquivo PDF. O método recebe o nome do arquivo com o seu caminho completo.

**Linha 3:** Variável que receberá o texto extraído.

**Linha 5:** File para acessar o arquivo na classe.

**Linha 7:** Representação do documento PDF.

**Linha 9:** Carrega as informações do File para o objeto PDDocument.

**Linha 10:** Criação do **PDFTextStripper**, que é o responsável pela extração do texto.

**Linha 11:** Variável recebe o conteúdo do arquivo PDF através do método **getText()**.

## Executando Exemplo

Para executar o método implementado acima, criamos a classe Main com o metodo main.

```
package ivansalvadori.pdfboxtutorial;

import ivansalvadori.pdfboxtutorial.extraction.PDFExtraction;

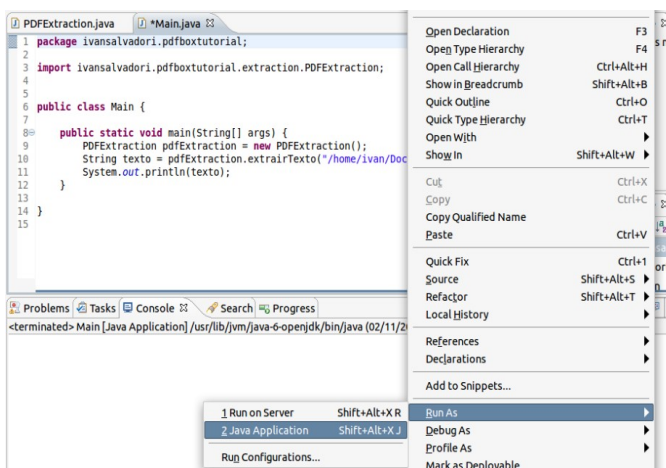
public class Main {

    public static void main(String[] args) {
        PDFExtraction pdfExtraction = new PDFExtraction();
        String texto = pdfExtraction.extrairTexto("/home/ivan/Documentos/lucene-3-tutorial.pdf");
        System.out.println(texto);
    }
}
```



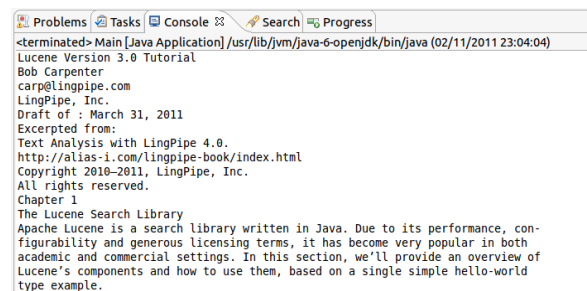
Vamos realizar a extração do arquivo lucene-3-tutorial.pdf informando seu caminho completo.

\* Em breve escreverei artigos sobre Lucene.



Para executar,  
direito do mouse → RunAs → Java Application

## Resultado:



É possível obter mais informações sobre o documento PDF lido do arquivo, obter o número de páginas por exemplo.

```
public int getNumeroPaginas(String caminhoArquivo) {
    int numPaginas = 0;
    File doc = new File(caminhoArquivo);
    PDDocument pdDoc;
    try {
        pdDoc = PDDocument.load(doc);
        numPaginas = pdDoc.getNumberOfPages();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return numPaginas;
}
```

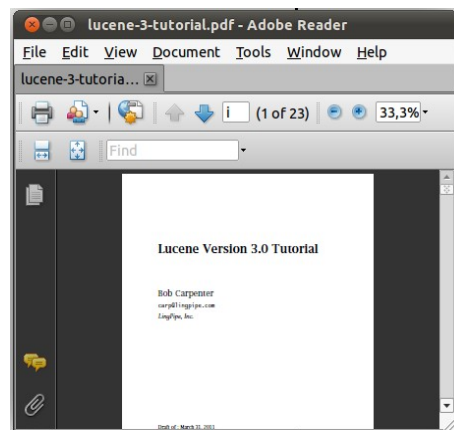
## Execução e Resultado

```
PDFExtraction.java Main.java
1 package ivansalvadori.pdfboxtutorial;
2
3 import ivansalvadori.pdfboxtutorial.extraction.PDFExtraction;
4
5 public class Main {
6
7     public static void main(String[] args) {
8         PDFExtraction pdfExtraction = new PDFExtraction();
9         int paginas = pdfExtraction.getNumeroPaginas("/home/ivan/Documentos/lucene-3-tutorial.pdf");
10        System.out.println(paginas);
11    }
12 }
13
14
15
```

Problems Tasks Console Search Progress

<terminated> Main [Java Application] /usr/lib/jvm/java-6-openjdk/bin/java (02/11/2011 23:26:54)

23



Note que o documento PDF contém exatamente 23 páginas.

Outra opção é especificar o intervalo de páginas para extração do texto.

```
public String extrairTexto(String caminhoArquivo, int paginaInicio, int paginaFinal) {
    String textoExtraido = null;
    File doc = new File(caminhoArquivo);
    PDDocument pdDoc;
    try {
        pdDoc = PDDocument.load(doc);
        PDFTextStripper stripper = new PDFTextStripper();
        stripper.setStartPage(paginaInicio);
        stripper.setEndPage(paginaFinal);
        textoExtraido = stripper.getText(pdDoc);
    } catch (IOException e) {
        e.printStackTrace();
    }
    return textoExtraido;
}
```

## Execução e Resultado

Extrair texto da primeira página do documento

```
PDFExtraction.java Main.java
1 package ivansalvadori.pdfboxtutorial;
2
3 import ivansalvadori.pdfboxtutorial.extraction.PDFExtraction;
4
5 public class Main {
6
7     public static void main(String[] args) {
8         PDFExtraction pdfExtraction = new PDFExtraction();
9         String texto = pdfExtraction.extrairTexto("/home/ivan/Documentos/lucene-3-tutorial.pdf", 1, 1);
10        System.out.println(texto);
11    }
12 }
13
14
15
```

Problems Tasks Console Search Progress

<terminated> Main [Java Application] /usr/lib/jvm/java-6-openjdk/bin/java (02/11/2011 23:35:20)

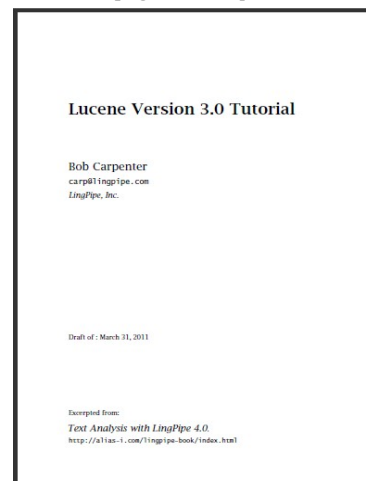
Lucene Version 3.0 Tutorial

Bob Carpenter  
carp@lingpipe.com  
LingPipe, Inc.

Draft of : March 31, 2011

Excerpted from:  
Text Analysis with LingPipe 4.0.  
<http://alias-i.com/lingpipe-book/index.html>

Primeira página do arquivo



É importante apresentar a fonte do documento utilizado no exemplo de extração de texto.

**Lucene Version 3.0 Tutorial** - Bob Carpenter - [carp@lingpipe.com](mailto:carp@lingpipe.com) - LingPipe, Inc.

Respeitando seus direitos autorais e recomendando sua leitura.

## Anexo I

### PDFExtraction.java

```
package ivansalvadori.pdfboxtutorial.extraction;

import java.io.File;
import java.io.IOException;

import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.util.PDFTextStripper;

public class PDFExtraction {

    public String extrairTexto(String caminhoArquivo) {

        String textoExtraido = null;

        File doc = new File(caminhoArquivo);

        PDDocument pdDoc;
        try {
            pdDoc = PDDocument.load(doc);
            PDFTextStripper stripper = new PDFTextStripper();
            textoExtraido = stripper.getText(pdDoc);

        } catch (IOException e) {
            e.printStackTrace();
        }

        return textoExtraido;

    }

    public int getNumeroPaginas(String caminhoArquivo) {

        int numPaginas = 0;

        File doc = new File(caminhoArquivo);

        PDDocument pdDoc;
        try {
            pdDoc = PDDocument.load(doc);
            numPaginas = pdDoc.getNumberOfPages();

        } catch (IOException e) {
            e.printStackTrace();
        }

        return numPaginas;

    }

    public String extrairTexto(String caminhoArquivo, int paginaInicio,
                               int paginaFinal) {

        String textoExtraido = null;

        File doc = new File(caminhoArquivo);

        PDDocument pdDoc;
        try {
            pdDoc = PDDocument.load(doc);
            PDFTextStripper stripper = new PDFTextStripper();
            stripper.setStartPage(paginaInicio);
            stripper.setEndPage(paginaFinal);
            textoExtraido = stripper.getText(pdDoc);

        } catch (IOException e) {
            e.printStackTrace();
        }

        return textoExtraido;

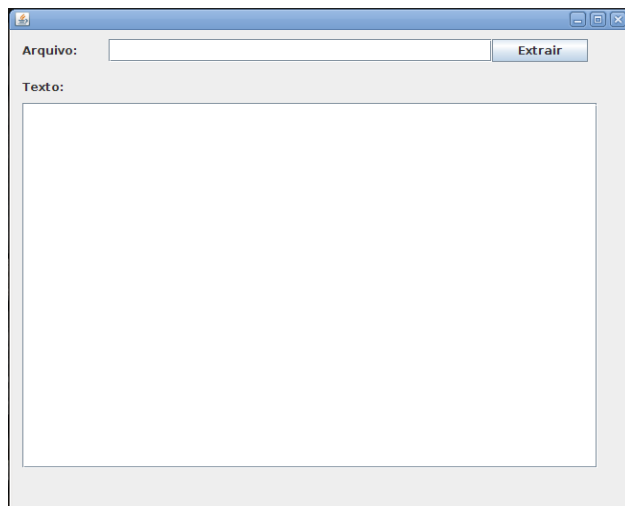
    }

}
```



## Interface Gráfica

Como o objetivo é desenvolver um software desktop, totalmente executável, vamos então implementar uma interface gráfica para melhorar a experiência do usuário. A Interface consiste em uma janela apenas, possui um campo para digitar o arquivo com caminho completo, um botão para realizar a extração do texto e uma área de texto para exibir o conteúdo extraído. Não vamos discutir os detalhes da implementação da janela gráfica, pois esse não é objetivo do tutorial, mas o seu código é apresentado no Anexo II.

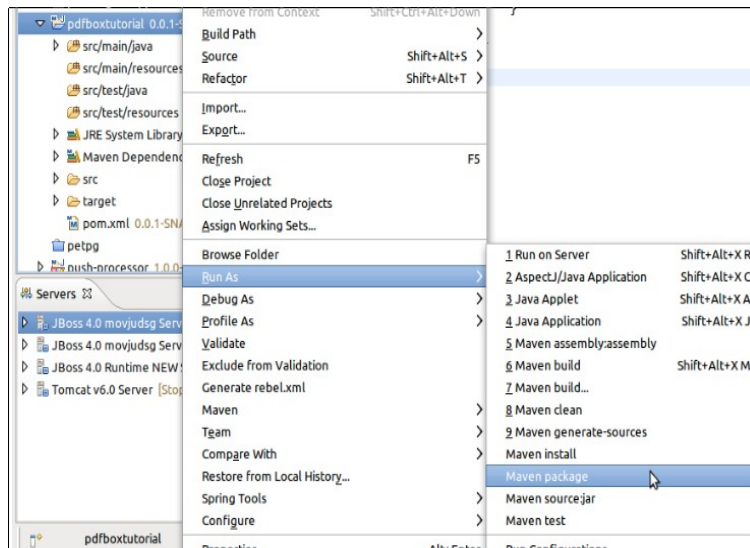


## Gerando o Executável

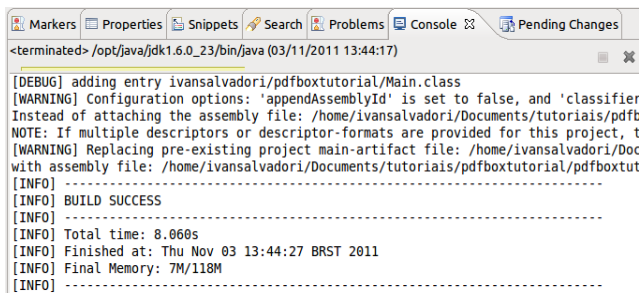
Para gerar o executável através do maven, temos que realizar algumas pequenas configurações, umas delas é especificar qual será a classe que terá seu método main executado, no nosso caso temos apenas uma classe com método main, mesmo assim devemos especificar. Deve-se também orientar o maven a empacotar as dependências utilizadas juntamente com o jar final. Essas configurações são feitas no pom.xml. Foi configurado também a diretiva que determina o java 1.6 como o padrão a ser utilizado. Acrescente as informações abaixo no pom.xml

```
<build>
  <finalName>pdfBoxTutorial</finalName>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-assembly-plugin</artifactId>
      <executions>
        <execution>
          <id>package-jar-with-dependencies</id>
          <phase>package</phase>
          <goals>
            <goal>single</goal>
          </goals>
          <configuration>
            <appendAssemblyId>false</appendAssemblyId>
            <descriptorRefs>
              <descriptorRef>jar-with-dependencies</descriptorRef>
            </descriptorRefs>
            <archive>
              <manifest>
                <mainClass>ivansalvadori.pdfboxtutorial.Main</mainClass>
              </manifest>
            </archive>
          </configuration>
        </execution>
      </executions>
    </plugin>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <configuration>
        <source>1.6</source>
        <target>1.6</target>
      </configuration>
    </plugin>
  </plugins>
</build>
```

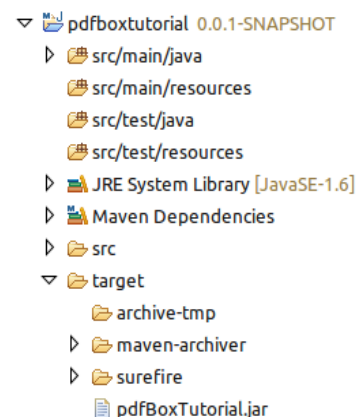
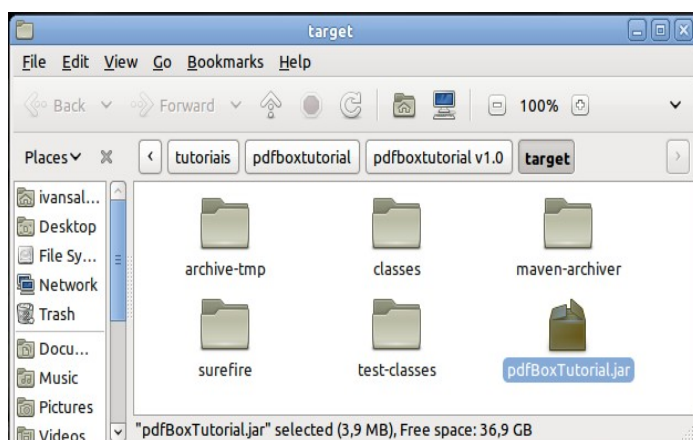
Com o pom.xml configurado corretamente, vamos gerar o JAR executável.  
Direito do mouse sobre o projeto, Run As → Maven package.



Se tudo ocorreu bem, a seguinte mensagem estará no console:



O Jar executável fica disponível na pasta target.



## Anexo II

### Janela.java

```
package ivansalvadori.pdfboxtutorial.view;
import ivansalvadori.pdfboxtutorial.extraction.PDFExtraction;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;

public class Janela extends JFrame {

    private static final long serialVersionUID = -845967882639773609L;

    private JLabel labelCampoArquivo;
    private JTextField campoArquivo;
    private JButton botaoExtrair;
    private JLabel labelCampoTextoExtraido;
    private JTextArea campoTextoExtraido;
    private JScrollPane scrolCampoTextoExtraido;

    public Janela() {
        super.setSize(650, 550);
        super.setLayout(null);

        this.labelCampoArquivo = new JLabel("Arquivo: ");
        this.labelCampoArquivo.setBounds(10, 10, 100, 25);
        add(this.labelCampoArquivo);

        this.campoArquivo = new JTextField();
        this.campoArquivo.setBounds(100, 10, 400, 25);
        add(this.campoArquivo);

        this.botaoExtrair = new JButton("Extrair");
        this.botaoExtrair.setBounds(500, 10, 100, 25);
        add(this.botaoExtrair);

        this.labelCampoTextoExtraido = new JLabel("Texto:");
        this.labelCampoTextoExtraido.setBounds(10, 50, 150, 25);
        add(this.labelCampoTextoExtraido);

        this.campoTextoExtraido = new JTextArea();
        this.scrolCampoTextoExtraido = new JScrollPane(this.campoTextoExtraido);
        add(this.scrolCampoTextoExtraido);
        this.scrolCampoTextoExtraido.setBounds(10, 80, 600, 400);

        this.botaoExtrair.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                botaoExtrairClick(evt);
            }
        });

        private void botaoExtrairClick(java.awt.event.ActionEvent evt) {
            String caminhoArquivo = this.campoArquivo.getText();

            PDFExtraction pdfExtraction = new PDFExtraction();
            this.campoTextoExtraido.setText(pdfExtraction.extrairTexto(caminhoArquivo));

            this.labelCampoTextoExtraido.setText("Texto: " + pdfExtraction.getNumeroPaginas(caminhoArquivo) + "
páginas");
        }
    }
}
```

Configurando a classe Main para exibir a janela quando iniciado o sistema.

### Main.java

```
package ivansalvadori.pdfboxtutorial;
import ivansalvadori.pdfboxtutorial.view.Janela;

public class Main {
    public static void main(String[] args) {
        new Janela().setVisible(true);
    }
}
```

Com isso temos a primeira funcionalidade do PDFBox implementada!  
Outras funcionalidades serão demonstradas posteriormente em nova versão deste tutorial.