PDFBOX

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia https://twitter.com/tutorialspoint

About the Tutorial

Apache PDFBox is an open-source Java library that supports the development and conversion of PDF documents. In this tutorial, we will learn how to use PDFBox to develop Java programs that can create, convert, and manipulate PDF documents.

Audience

This tutorial has been prepared for beginners to make them understand the basics of PDFBox library. This tutorial will help the readers in building applications that involve creation, manipulation and deletion of PDF documents.

Prerequisites

For this tutorial, it is assumed that the readers have a prior knowledge of Java programming language.

Copyright & Disclaimer

Table of Contents

1. PDFBOX – OVERVIEW

The Portable Document Format (PDF) is a file format that helps to present data in a manner that is independent of Application software, hardware, and operating systems.

Each PDF file holds description of a fixed-layout flat document, including the text, fonts, graphics, and other information needed to display it.

There are several libraries available to create and manipulate PDF documents through programs, such as:

- **Adobe PDF Library**: This library provides API in languages such as C++, .NET and Java and using this we can edit, view print and extract text from PDF documents.

- **Formatting Objects Processor**: Open-source print formatter driven by XSL Formatting Objects and an output independent formatter. The primary output target is PDF.

- **iText**: This library provides API in languages such as Java, C#, and other .NET languages and using this library we can create and manipulate PDF, RTF and HTML documents.

- **JasperReports**: This is a Java reporting tool which generates reports in PDF document including Microsoft Excel, RTF, ODT, comma-separated values and XML files.

What is a PDFBox

Apache PDFBox is an open-source Java library that supports the development and conversion of PDF documents. Using this library, you can develop Java programs that create, convert and manipulate PDF documents.

In addition to this, PDFBox also includes a command line utility for performing various operations over PDF using the available Jar file.

Features of PDFBox

Following are the notable features of PDFBox:

- **Extract Text**: Using PDFBox, you can extract Unicode text from PDF files.

- **Split & Merge**: Using PDFBox, you can divide a single PDF file into multiple files, and merge them back as a single file.

- **Fill Forms**: Using PDFBox, you can fill the form data in a document.

- **Preflight**: PDFBox has an optional preflight component; with this, you can verify the PDF files against the PDF/A-1b standard.
- **Print**: Using PDFBox, you can print a PDF file using the standard Java printing API.
- **Save as Image**: Using PDFBox, you can save PDFs as image files, such as PNG or JPEG.
- **Create PDFs**: Using PDFBox, you can create a new PDF file by creating Java programs and, you can also include images and fonts.
- **Signing**: Using PDFBox, you can add digital signatures to the PDF files.

Applications of PDFBox

The following are the applications of PDFBox:

- **Apache Nutch:** Apache Nutch is an open-source web-search software. It builds on Apache Lucene, adding web-specifics, such as a crawler, a link-graph database, parsers for HTML and other document formats, etc.
- **Apache Tika:** Apache Tika is a toolkit for detecting and extracting metadata and structured text content from various documents using existing parser libraries.

Components of PDFBox

The following are the four main components of PDFBox:

- **PDFBox**: This is the main part of the PDFBox. This contains the classes and interfaces related to content extraction and manipulation.
- **FontBox**: This contains the classes and interfaces related to font, and using these classes we can modify the font of the text of the PDF document.
- **XmpBox**: This contains the classes and interfaces that handle XMP metadata.
- **Preflight**: This component is used to verify the PDF files against the PDF/A-1b standard.

2. PDFBOX – ENVIRONMENT

Installing PDFBox

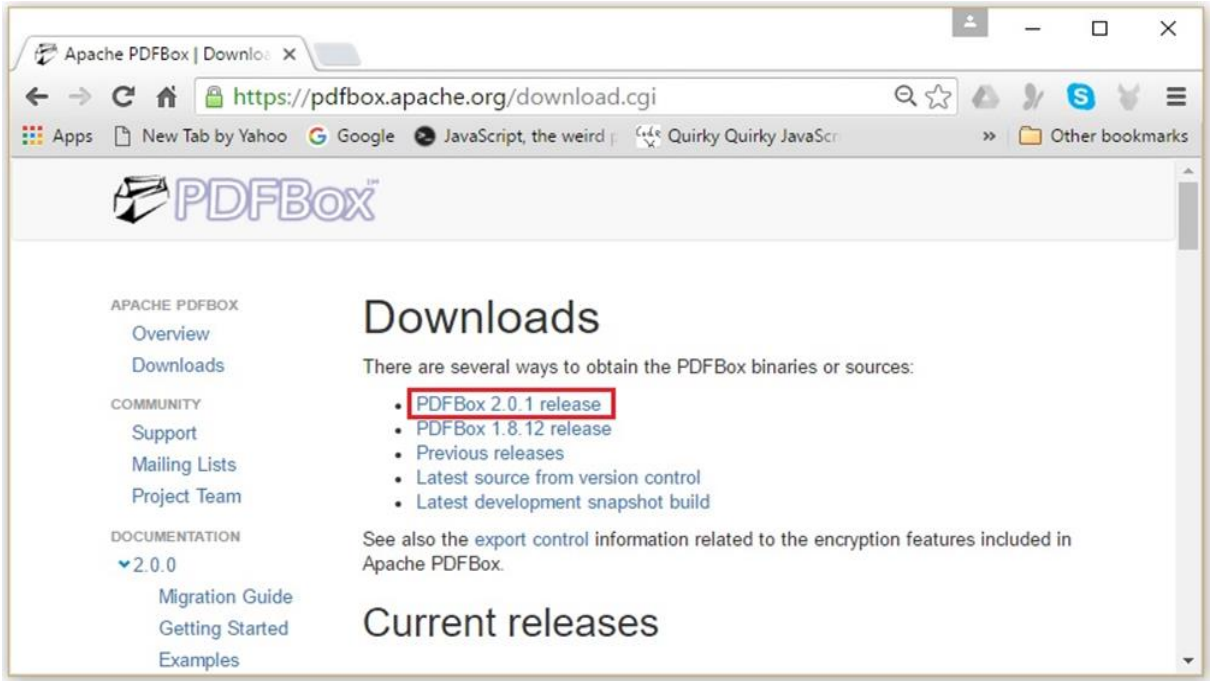
Following are the steps to download Apache PDFBox:

Step 1: Open the homepage of **Apache PDFBox** by clicking on the following link:

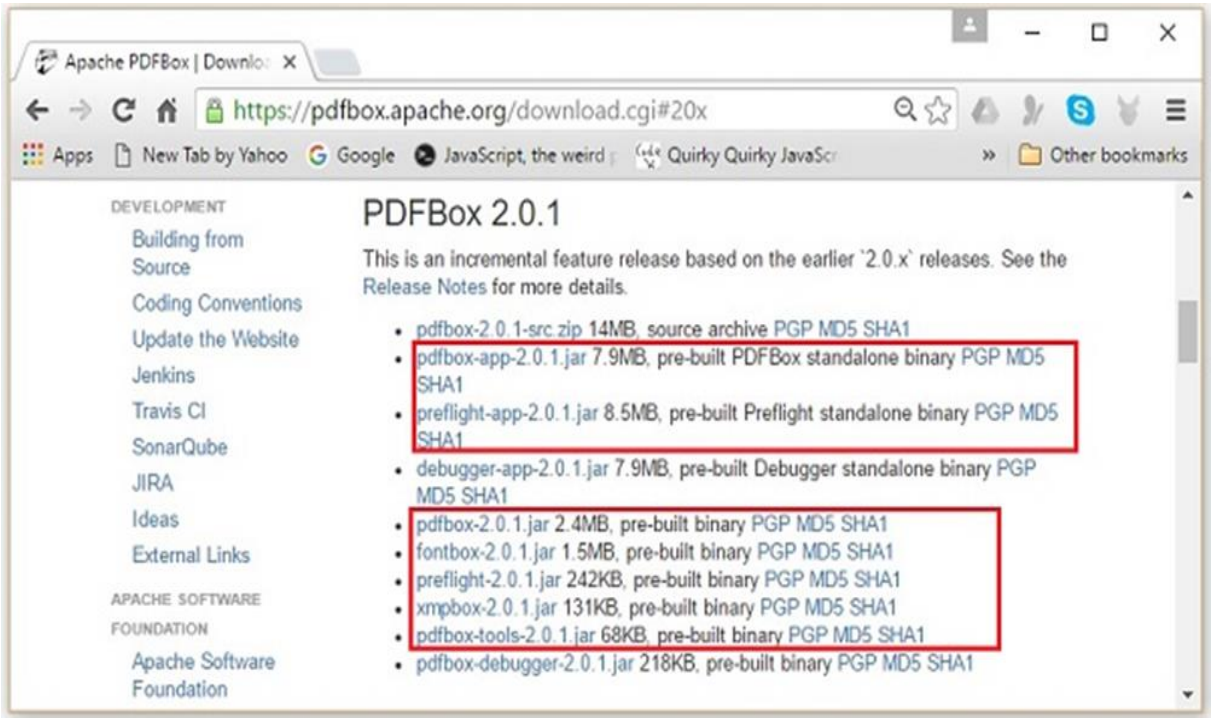
https://pdfbox.apache.org/

Step 2: The above link will direct you to the homepage as shown in the following screenshot:

Step 3: Now, click on the **Downloads** link highlighted in the above screenshot**.** On clicking, you will be directed to the downloads page of PDFBox as shown in the following screenshot.



Step 4: In the Downloads page, you will have links for PDFBox. Click on the respective link for the latest release. For instance, we are opting for **PDFBox 2.0.1** and on clicking this, you will be directed to the required jar files as shown in the following screenshot.



Step 5: Download the jar files pdfbox-2.0.1.jar, fontbox-2.0.1.jar, preflight-2.0.1.jar, xmpbox-2.0.1.jar and, pdfbox-tools-2.0.1.jar.

Eclipse Installation

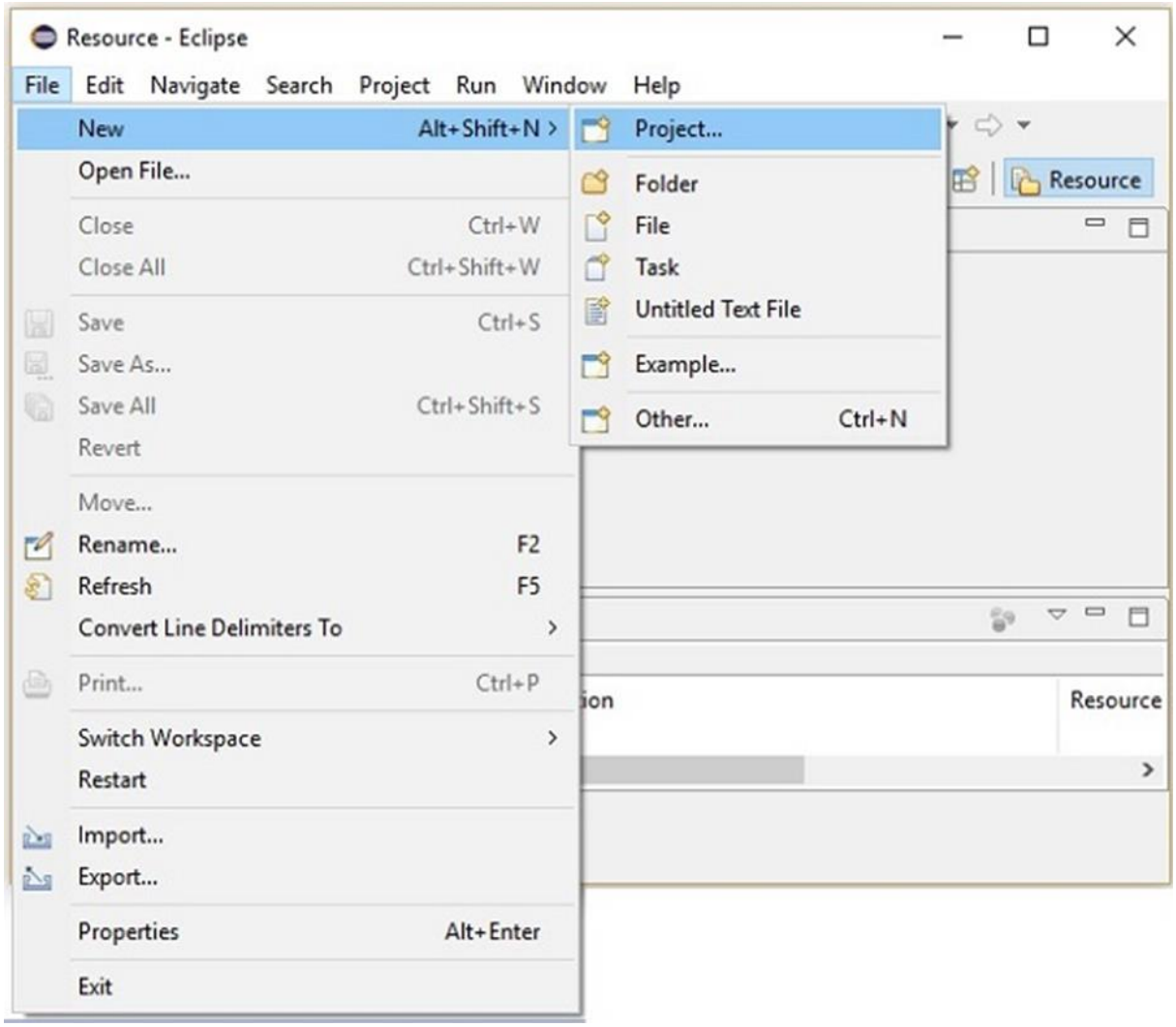
After downloading the required jar files, you have to embed these JAR files to your Eclipse environment. You can do this by setting the Build path to these JAR files and by using **pom.xml.**

Setting Build Path

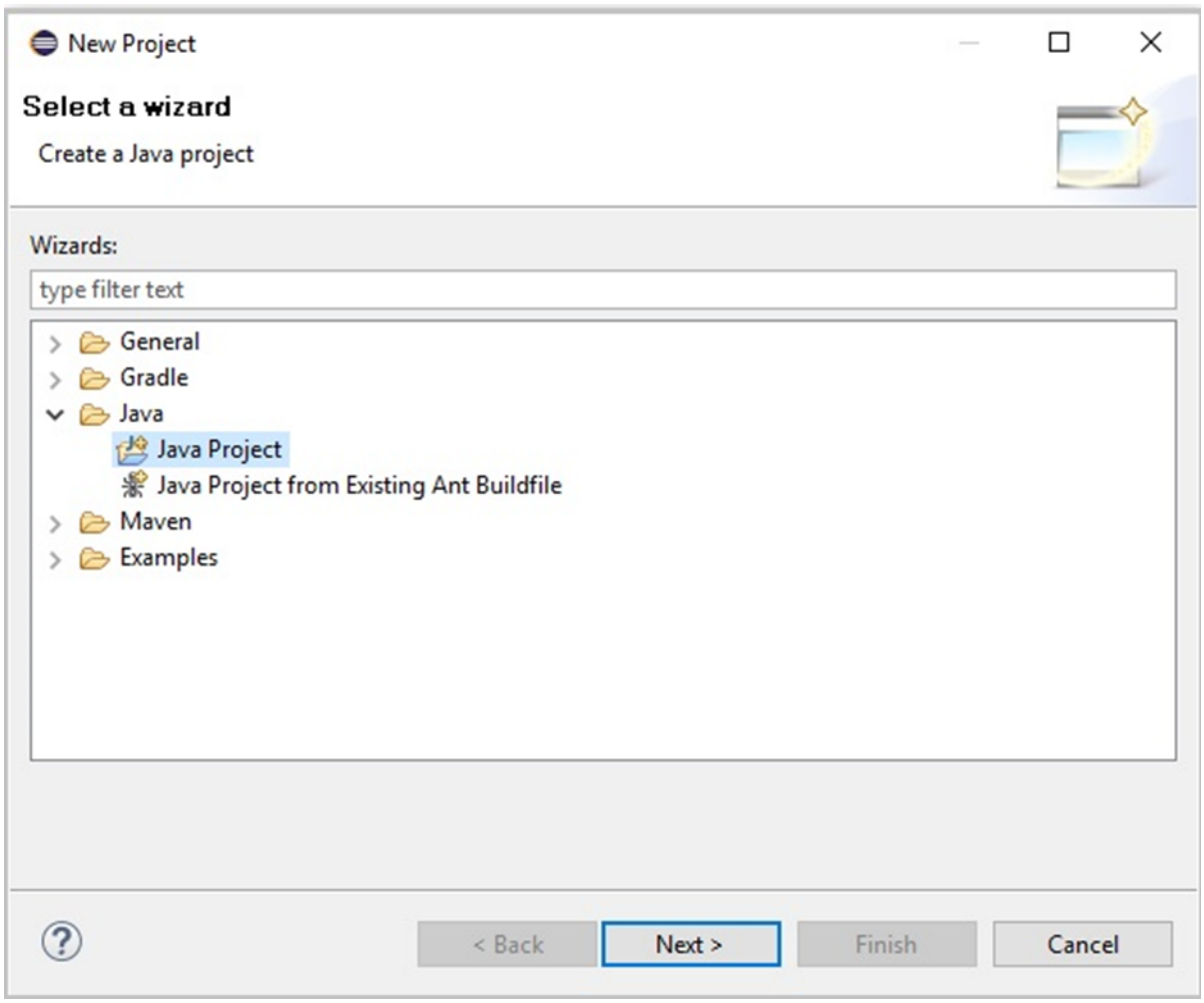
Following are the steps to install PDFBox in Eclipse:

Step 1: Ensure that you have installed Eclipse in your system. If not, download and install Eclipse in your system.

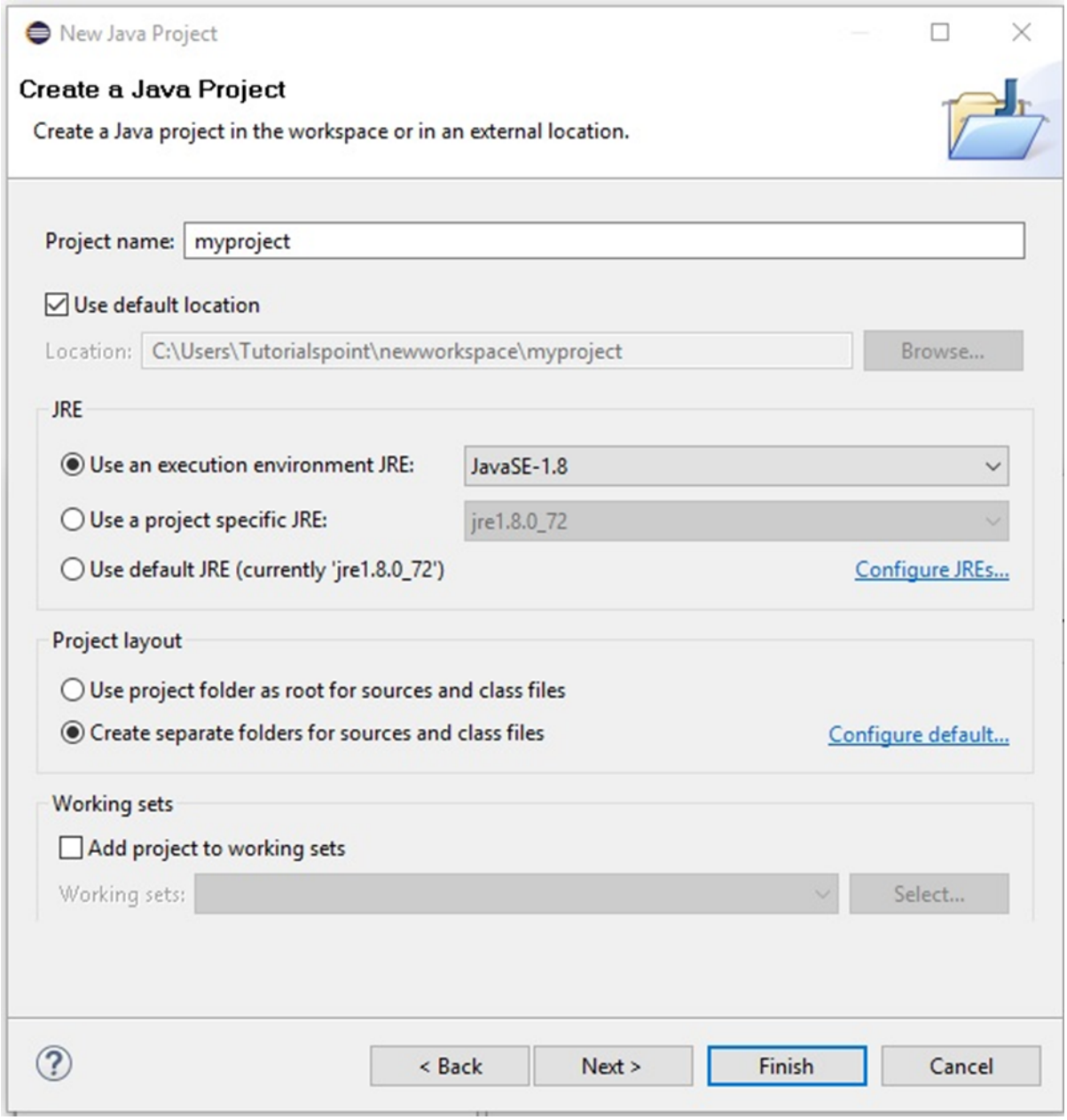
Step 2: Open Eclipse, click on File, New, and Open a new project as shown in the following screenshot.



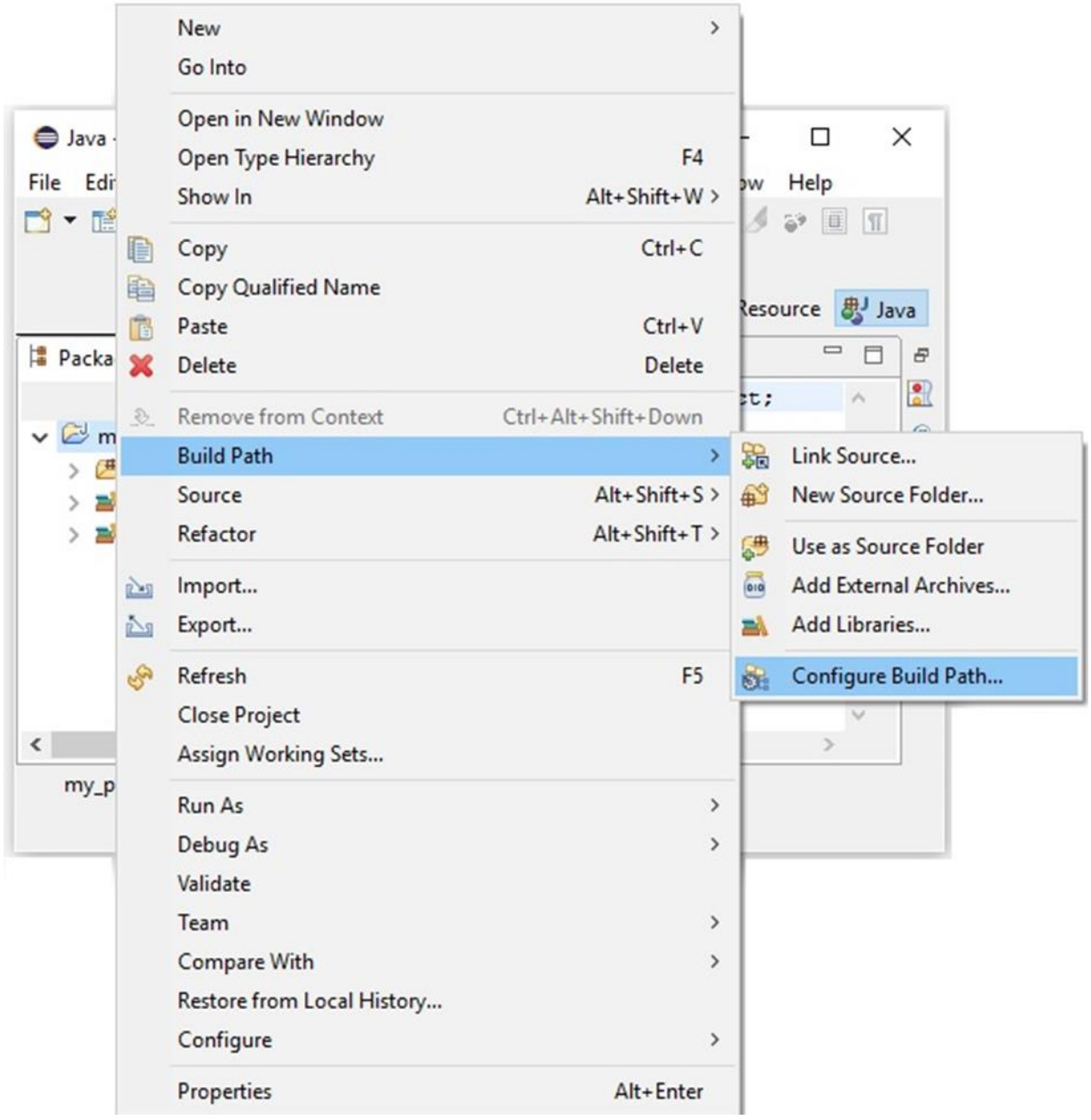
Step 3: On selecting the project, you will get **New Project** wizard. In this wizard, select Java project and proceed by clicking **Next** button as shown in the following screenshot.



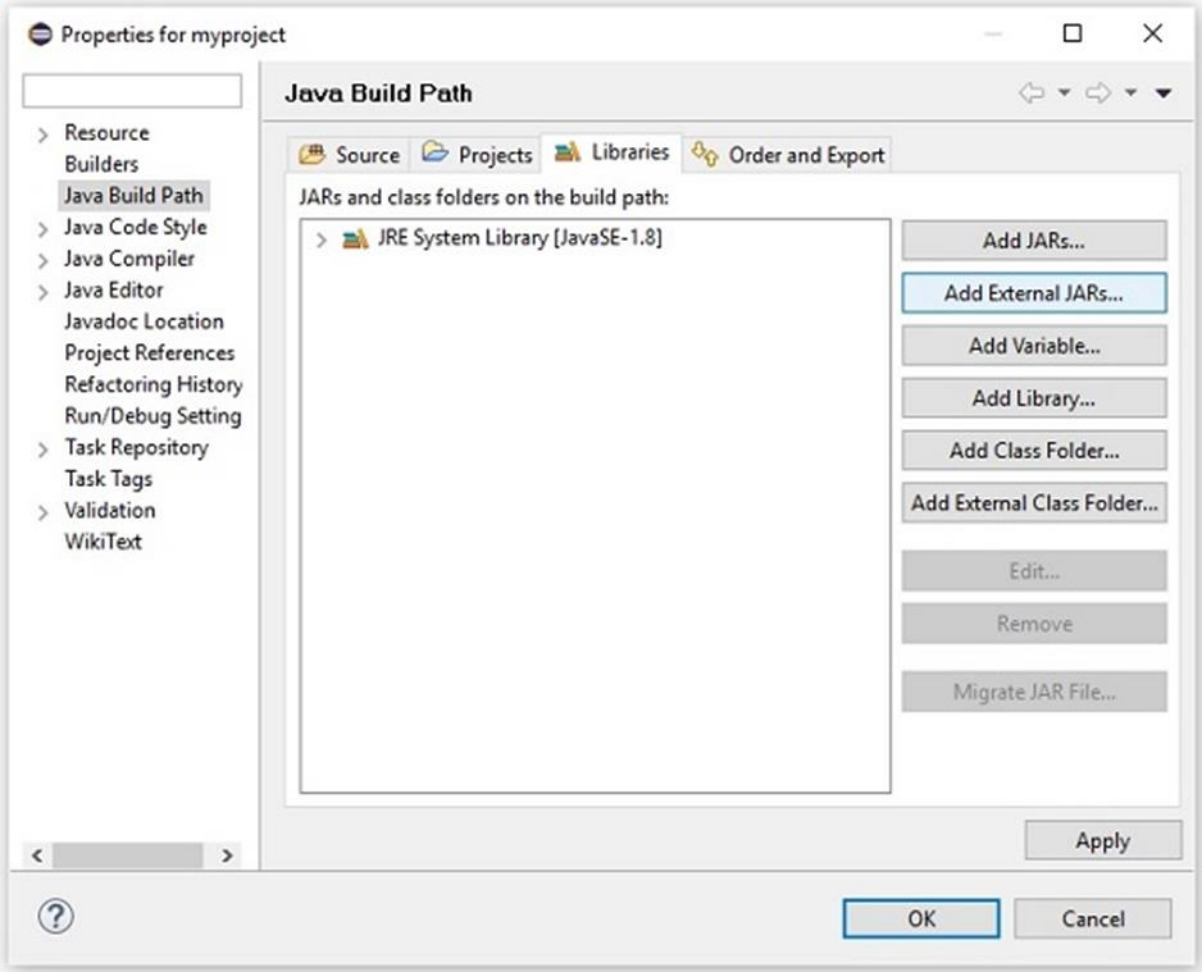
Step 4: On proceeding forward, you will be directed to the **New Java Project wizard**. Create a new project and click on **Next** as shown in the following screenshot.



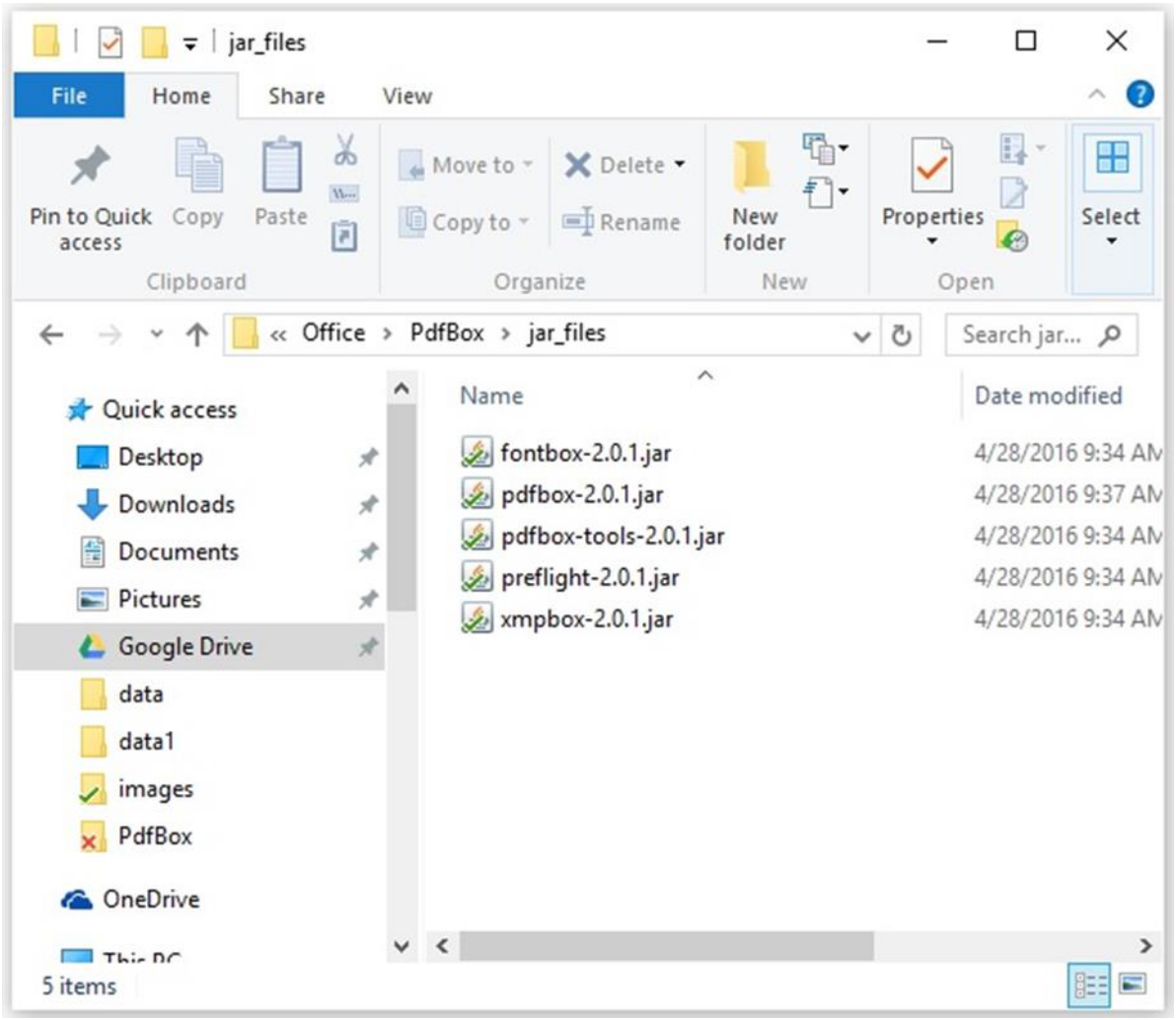
Step 5: After creating a new project, right click on it; select **Build Path** and click on **Configure Build Path...** as shown in the following screenshot.



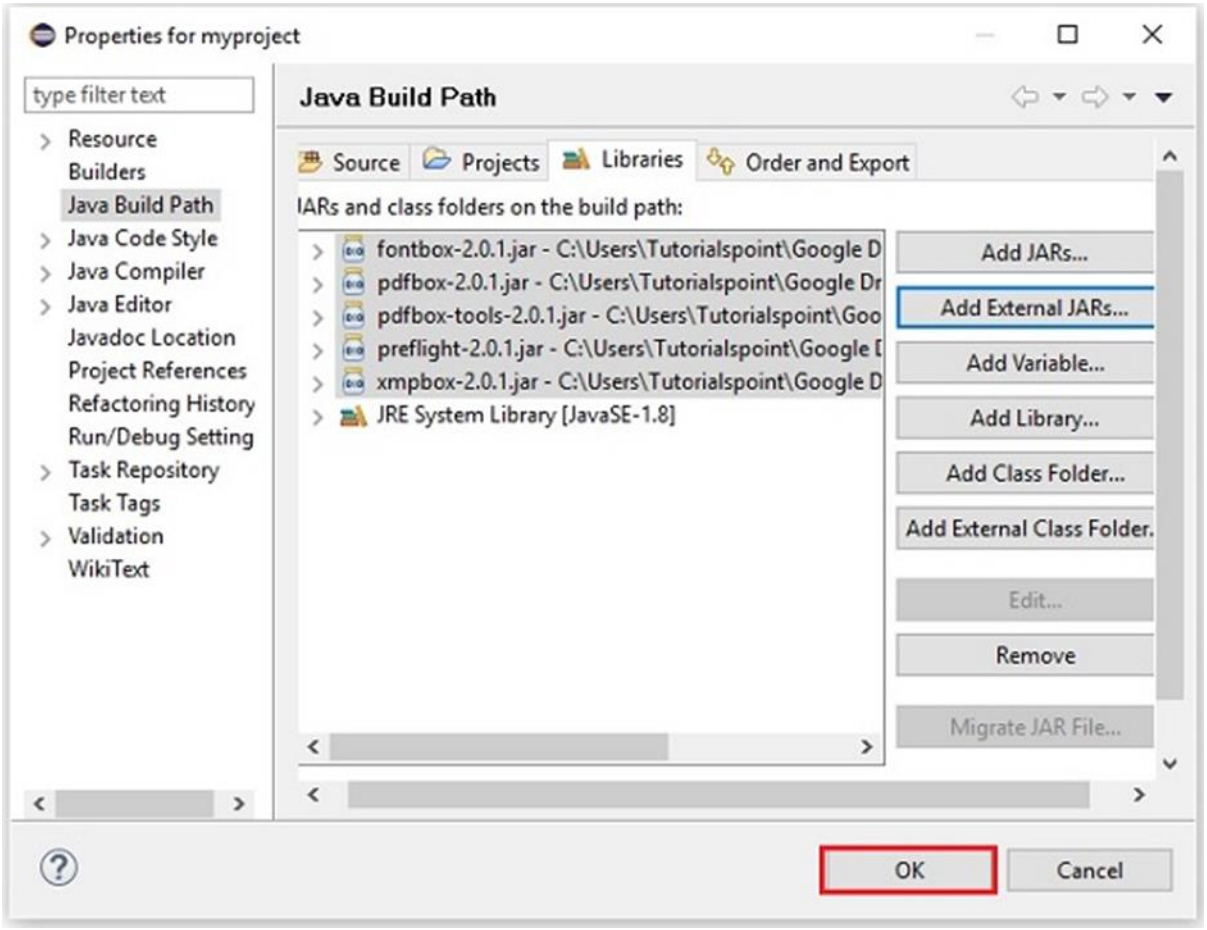
Step 6: On clicking on the **Build Path** option you will be directed to the **Java Build Path wizard**. Select the **Add External JARs** as shown in the following screenshot.



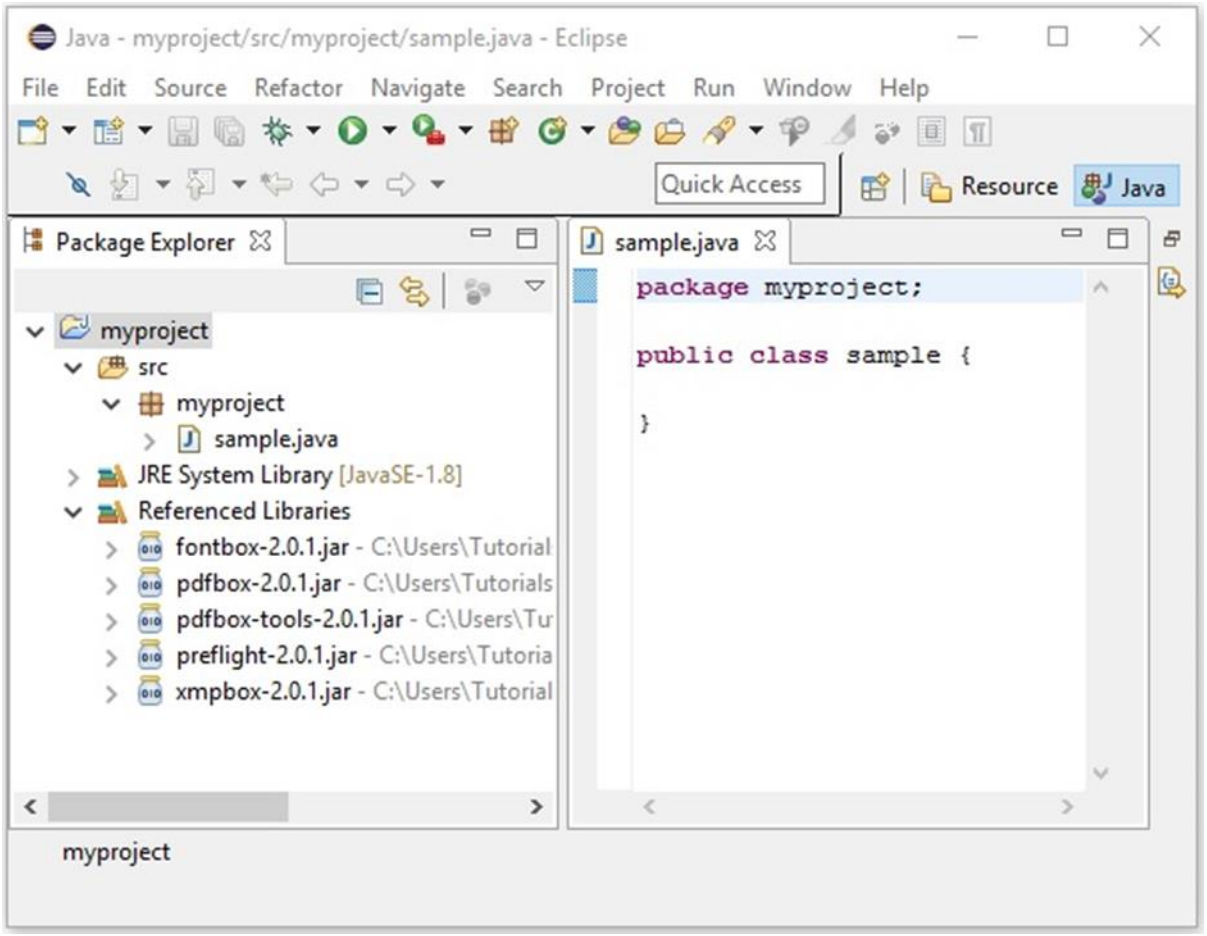
Step 7: Select the jar files **fontbox-2.0.1.jar, pdfbox-2.0.1.jar, pdfbox-tools-2.0.1.jar, preflight-2.0.1.jar, xmpbox-2.0.1.jar** as shown in the following screenshot.



Step 8: On clicking the **Open** button in the above screenshot, those files will be added to your library as shown in the following screenshot.



Step 9: On clicking **OK**, you will successfully add the required JAR files to the current project and you can verify these added libraries by expanding the Referenced Libraries as shown in the following screenshot.



Using pom.xml

Convert the project into maven project and add the following contents to its **pom.xml.**

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>my_project</groupId>
  <artifactId>my_project</artifactId>
  <version>0.0.1-SNAPSHOT</version>



   <build>
    <sourceDirectory>src</sourceDirectory>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.3</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
  <dependencies>
    <dependency>
      <groupId>org.apache.pdfbox</groupId>
      <artifactId>pdfbox</artifactId>
      <version>2.0.1</version>
    </dependency>

    <dependency>
      <groupId>org.apache.pdfbox</groupId>
      <artifactId>fontbox</artifactId>
      <version>2.0.0</version>
    </dependency>
```


```
    <dependency>
      <groupId>org.apache.pdfbox</groupId>
      <artifactId>jempbox</artifactId>
      <version>1.8.11</version>
    </dependency>

    <dependency>
      <groupId>org.apache.pdfbox</groupId>
      <artifactId>xmpbox</artifactId>
      <version>2.0.0</version>
    </dependency>

    <dependency>
      <groupId>org.apache.pdfbox</groupId>
      <artifactId>preflight</artifactId>
      <version>2.0.0</version>
    </dependency>

    <dependency>
      <groupId>org.apache.pdfbox</groupId>
      <artifactId>pdfbox-tools</artifactId>
      <version>2.0.0</version>
    </dependency>

 </dependencies>

</project>
```

3. PDFBOX — CREATING A PDF DOCUMENT

Let us now understand how to create a PDF document using the PDFBox library.

Creating an Empty PDF Document

You can create an empty PDF Document by instantiating the **PDDocument** class. You can save the document in your desired location using the **Save()** method.

Following are the steps to create an empty PDF document.

Step 1: Creating an Empty Document

The **PDDocument** class that belongs to the package ***org.apache.pdfbox.pdmodel***, is an In-memory representation of the PDFDocument. Therefore, by instantiating this class, you can create an empty PDFDocument as shown in the following code block.

```
PDDocument document = new PDDocument();
```

Step 2: Saving the Document

After creating the document, you need to save this document in the desired path, you can do so using the **Save()** method of the **PDDocument** class. This method accepts a string value, representing the path where you want to store the document, as a parameter. Following is the prototype of the save() method of the **PDDocument** class.

```
document.save("Path");
```

Step 3: Closing the Document

When your task is completed, at the end, you need to close the **PDDocument** object using the **close ()** method. Following is the prototype of the close() method of **PDDocument** class.

```
document.close();
```

Example

This example demonstrates the creation of a PDF Document. Here, we will create a Java program to generate a PDF document named **my_doc.pdf** and save it in the path **C:/PdfBox_Examples/**