APACHE POI PPT - QUICK GUIDE

APACHE POI PPT - OVERVIEW

Many a time, a software application is required to generate reports in Microsoft Office file format. Sometimes, an application is even expected to receive MS- Office files as input data.

Any Java programmer who wants to produce MS Office files as output must use a predefined and read-only API to do so.

What is Apache POI?

Apache POI is a popular API that allows programmers to create, modify, and display MS-Office files using Java programs. It is an open source library developed and distributed by Apache Software Foundation. It contains classes and methods to decode the user input data, or a file into MS Office documents.

Components of Apache POI

Apache POI contains classes and methods to work on all OLE2 Compound documents of MS-Office. The list of components of this API is given below:

- **POIFS** *PoorObfuscationImplementationFileSystem* : This component is the basic factor of all other POI elements. It is used to read different files explicitly.

- **HSSF** *HorribleSpreadSheetFormat* : It is used to read and write .xls format of MS-Excel files.

- **XSSF** *XMLSpreadSheetFormat* : It is used for .xlsx file format of MS-Excel.

- **HPSF** *HorriblePropertySetFormat* : It is used to extract property sets of the MS-Office files.

- **HWPF** *HorribleWordProcessorFormat* : It is used to read and write **.doc** extension files of MS-Word.

- **XWPF** *XMLWordProcessorFormat* : It is used to read and write .docx extension files of MS-Word.

- **HSLF** *HorribleSlideLayoutFormat* : It is used to read, create, and edit PowerPoint presentations.

- **HDGF** *HorribleDiaGramFormat* : It contains classes and methods for MS-Visio binary files.

- **HPBF** *HorriblePuBlisherFormat* : It is used to read and write MS-Publisher files.

This tutorial guides you through the process of working on Microsoft PowerPoint presentation using Java. Therefore the discussion is confined to **XSLF component.**

Note : Older versions of POI support binary file formats such as doc, xls, ppt, etc. Version 3.5 onwards, POI supports OOXML file formats of MS-Office such as docx, xlsx, pptx, etc.

FLAVORS OF JAVA PPT API

This chapter takes you through some of the flavors of Java PowerPoint API and their features. There are many vendors who provide Java PPT related APIs; some of them are considered in this chapter.

Aspose Slides for Java

Aspose slides for Java is a purely licensed Java PPT API, developed and distributed by the vendor **Aspose**. The latest version of this API is 8.1.2, released in July 2014. It is a rich and heavy API *combinationofplainJavaclassesandAWTclasses* for designing the PPT component that can read, write, and manage slides.

The common uses of this API are as follows:

- Build dynamic presentations

- Render and print high-fidelity presentations
- Generate, edit, convert, and print presentations

Apache POI

Apache POI is a 100% open source library provided by Apache Software Foundation. Most of the small and medium scale application developers depend heavily on Apache POI $HSLF + XSLF$. It supports all the basic features of PPT libraries; however, rendering and text extraction are its main features. Given below is the architecture of Apache POI for PPT.
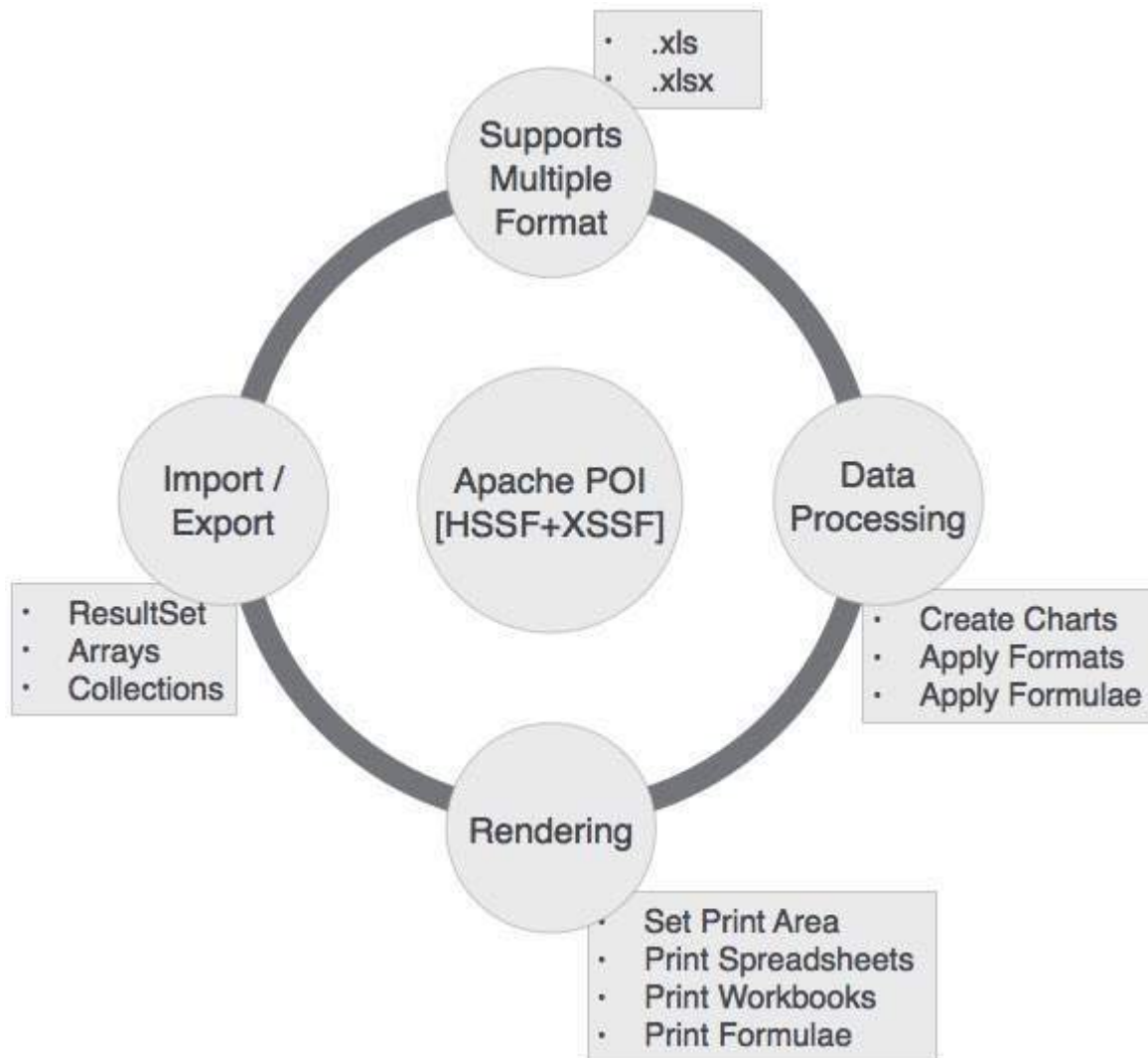
APACHE POI PPT - INSTALLATION

This chapter takes you through the process of setting up Apache POI on Windows and Linux based systems. Apache POI can easily be installed and integrated with your current Java environment, following a few simple steps without any complex setup procedures. User administration is required for installation.

System Requirements

JDK	Java SE 2 JDK 1.5 or above
Memory	1 GB RAM *recommended*
Disk Space	No minimum requirement
Operating System Version	Windows XP or above, Linux

Let us now proceed with the steps to install Apache POI.

Step 1: Verify your Java Installation

First of all, you need to have Java Software Development Kit SDK installed on your system. To verify this, execute any of the following two commands depending on the platform you are working on.

If the Java installation has been done properly, then it will display the current version and specification of your Java installation. A sample output is given in the following table.

Platform	Command	Sample Output
Windows	Open Command Console and type: **\>java –version**	Java version "1.7.0_60" Java TM SE Run Time Environment $build1.7.0_60 - b19$ Java Hotspot TM 64-bit Server VM $build24.60 - b09, mixedmode$
Linux	Open Command terminal and type: **$java –version**	java version "1.7.0_25" Open JDK Runtime Environment $rhel - 2.3.10.4.el6_4 - x86_64$ Open JDK 64-Bit Server VM $build23.7 - b01, mixedmode$

- We assume that the readers of this tutorial have Java SDK version 1.7.0_60 installed on their system.

- In case you do not have Java SDK, download its current version from http://www.oracle.com/technetwork/java/javase/downloads/index.html and install it.

Step 2: Set your Java Environment

Set the environment variable JAVA_HOME to point to the base directory location where Java is installed on your machine. For example,

Platform	Description
Windows	Set JAVA_HOME to C:\ProgramFiles\java\jdk1.7.0_60
Linux	Export JAVA_HOME=/usr/local/java-current

Append the full path of Java compiler location to the System Path.

Platform	Description
Windows	Append the String "C:\Program Files\Java\jdk1.7.0_60\bin" to the end of the system variable PATH.
Linux	Export PATH=$PATH$:JAVA_HOME/bin/

Execute the command **java -version** from the command prompt as explained above.

Step 3: Install Apache POI Library

Download the latest version of Apache POI from http://poi.apache.org/download.html and unzip its contents to a folder from where the required libraries can be linked to your Java program. Let us assume the files are collected in a folder on C drive.

The following images show the directories and the file structures inside the downloaded folder:
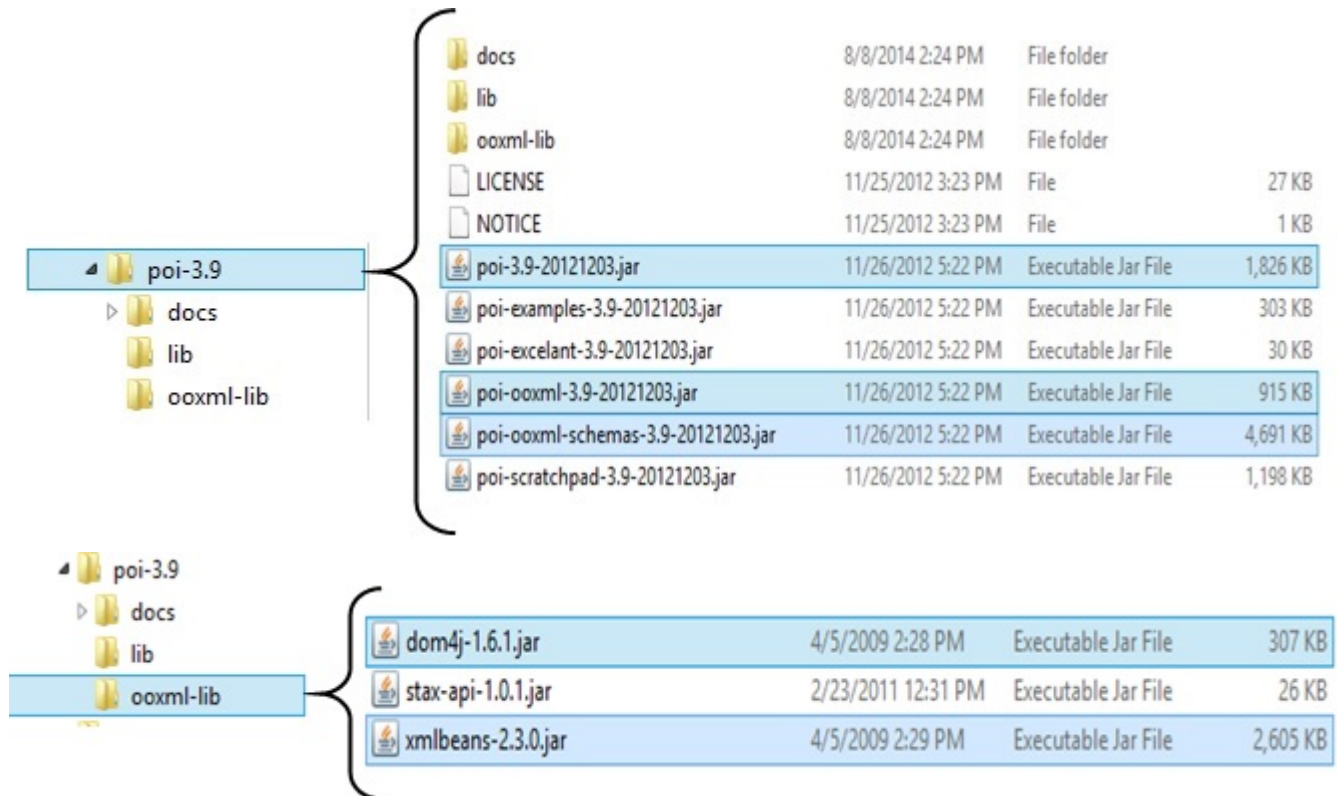
Add the complete path of the five **jars** as highlighted in the above image to the CLASSPATH.

Platform	Description
Windows	Append the following strings to the end of the user variable CLASSPATH: "C:\poi-3.9\poi-3.9-20121203.jar;" "C:\poi-3.9\poi-ooxml-3.9-20121203.jar;" "C:\poi-3.9\poi-ooxml-schemas-3.9-20121203.jar;" "C:\poi-3.9\ooxml-lib\dom4j-1.6.1.jar;" "C:\poi-3.9\ooxml-lib\xmlbeans-2.3.0.jar;.;"
Linux	Export CLASSPATH=$CLASSPATH: /usr/share/poi-3.9/poi-3.9-20121203.tar: /usr/share/poi-3.9/poi-ooxml-schemas-3.9-20121203.tar: /usr/share/poi-3.9/poi-ooxml-3.9-20121203.tar: /usr/share/poi-3.9/ooxml-lib/dom4j-1.6.1.tar: /usr/share/poi-3.9/ooxml-lib/xmlbeans-2.3.0.tar

APACHE POI PPT - CLASSES AND METHODS

In this chapter, we will learn about a few classes and methods under Apache POI API that are crucial to work on PPT files using Java programs.

Presentation

To create and manage a presentation, you have a class called XMLSlideShow in the package *org.apache.poi.xslf.usermodel*. Given below are some important methods and a constructor of this class.

Class : XMLSlideShow

Package : org.apache.poi.xslf.usermodel

S.No	Constructor and Description
1	**XMLSlideShow***java. io. InputStreaminputStream* You can instantiate this class by passing an inputstream class object to it.

S.No	Methods and Description
1	**int addPicture** *byte[]pictureData, intformat* Using this method, you can add a picture to a presentation.
2	**XSLFSlide createSlide** Creates a blank slide in a presentation.
3	**XSLFSlide createSlide***XSLFSlideLayoutlayout* Creates a slide with a given slide layout.
4	**java.util.List <XSLFPictureData> getAllPictures** Returns an array of all the pictures in a presentation.
5	**java.awt.Dimension getPageSize** Using this method, you can get to know the current page size.
6	**XSLFSlideMaster[] getSlideMasters** Returns the array of all the slides in a presentation.
7	

XSLFSlide[] getSlides

Returns all the slides in a presentation.

8

XSLFSlide removeSlide*intindex*

Using this method, you can remove a slide from a presentation.

9

void setPageSize*java. awt. DimensionpgSize*

Using this method, you can reset the page size.

10

void setSlideOrder*XSLFSlideslide, intnewIndex*

Using this method, you can reorder the slides.

Slide

To create and manage a slide in a presentation, the methods of the **XSLFSlide** class are used. Some important methods of this class are mentioned below.

Class : XSLFSlide

Package : org.apache.poi.xslf.usermodel

S.No	Methods and Description
1	**XSLFBackground getBackground** Returns the **XSLFBackground** object which can be used to retrieve details like color and anchor of the background of the slide. You can also draw shapes in the slide using this object.
2	**XSLFSlideLayout getSlideLayout** Provides access to the **XSLFSlideLayout** object of the current slide.
3	**XSLFSlideMaster getSlideMaster** Provides access to the slide master of the current slide.
4	**XSLFTheme getTheme** Returns the **XSLFTheme** object of the current slide.
5	**java.lang.String getTitle** Returns the title of the current slide.
6	

XSLFSlide importContent*XSLFSheetsrc*

Copies the contents of another slide to this slide.

Slide Master

It is the component of the presentation having different slide layouts. The **XSLFSlideMaster** class gives you access to it. Mentioned below are some important methods of this class.

Class : XSLFSlideMaster

Package : org.apache.poi.xslf.usermodel

S.No	Description
1	**XSLFBackground getBackground** Returns the common background of the slide master.
2	**XSLFSlideLayout getLayout***SlideLayouttype* Returns the XSLFSlideLayout object.
3	**XSLFSlideLayout[] getSlideLayouts** Returns all the slide layouts in this slide master.

Slide Layout

The POI library has a class called **XSLFSlideLayout**, using which you can manage the layouts of a slide.

Class : XSLFSlideLayout

Package : org.apache.poi.xslf.usermodel

S.No	Method and Description
1	**void copyLayout***XSLFSlideslide* This method will copy the placeholders from this layout to the given slide.

Text Paragraph

You can write content to the slide using **XSLFTextParagraph** class. Below mentioned are some important methods of this class.

Class : XSLFTextParagraph

Package : org.apache.poi.xslf.usermodel

S.No	Methods and Description
1	**XSLFTextRun addLineBreak** Inserts a line break in a paragraph.
2	**XSLFTextRun addNewTextRun** Adds a new run of text in a paragraph.
3	**void setBulletAutoNumber***ListAutoNumberscheme, intstartAt* Applies automatic numbered bullet points to the paragraph.
4	**void setIndent***doublevalue* Sets the indent to the text in the paragraph.
5	**void setLeftMargin***doublevalue* This method is used to add the left margin of the paragraph.
6	**void setLineSpacing***doublelinespacing* This method is used to set line spacing in the paragraph.
7	**void setTextAlign***TextAlignalign* This method is used to set alignment that is to be set to the paragraph.

Text Run

This is the lowest level of text separation within a text body. You have **XSLFTextRun** class to manage the text run of a paragraph. Below mentioned are some important methods of this class.

Class : XSLFTextParagraph

Package : org.apache.poi.xslf.usermodel

S.No	Methods and Description
1	**XSLFHyperlink createHyperlink** Creates a hyperlink in the presentation.
2	**XSLFHyperlink getHyperlink** This method is used to get the hyperlink.

3

java.lang.String getText

Returns the value of this Text node as a Java string.

4

void setBold*booleanbold*

This method is used to set the text in Bold.

5

void setCharacterSpacing*doublespc*

Sets the spacing between characters within a text run.

6

void setFontColor*java. awt. Colorcolor*

Sets the font color of the text.

7

void setFontSize*doublefontSize*

Sets the font size of the text.

8

void setItalic*booleanitalic*

This method is used to make the paragraph italicized.

9

void setStrikethrough*booleanstrike*

This method is used to format a run of text as strikethrough text.

10

void setSubscript*booleanflag*

This method is used to format the text as subscript.

11

void setSuperscript*booleanflag*

This method is used to format the text in this run as superscript.

12

void setText*java. lang. Stringtext*

This method is used to set the text in a run.

13

void setUnderline*booleanunderline*

This method is used to underline the text in a text run.

Text shape

In PPT, we have shapes that can hold text within them. We can manage these using

XSLFTextShape class. Mentioned below are some important methods of this class.

Class : XSLFTextShape

Package : org.apache.poi.xslf.usermodel

S.No	Methods and Description
1	**void setPlaceholder***Placeholderplaceholder* Using this method, you can choose a place holder.
2	**Placeholder getTextType** Returns the type of the current placeholder.
3	**void clearText** Clears the text area of the current text shape.
4	**XSLFTextParagraph addNewTextParagraph** Adds a new paragraph run to a shape.
5	**void drawContent***java. awt. Graphics2Dgraphics* This method allows you to draw any content on the slide.

HyperLink

The POI library has a class called **XSLFHyperlink** using which you can create a hyperlink in the presentation. Mentioned below are some important methods of this class.

Class : XSLFHyperlink

Package : org.apache.poi.xslf.usermodel

S.No	Methods and Description
1	**java.net.URI getTargetURL** Returns the URL existing in a slide of the presentation.
2	**void setAddress***java. lang. Stringaddress* This method is used to set address to a URL.
3	**void setAddress***XSLFSlideslide* Sets address to the URL present in a slide of the presentation.

APACHE POI PPT - PRESENTATION

Generally, we use MS-PowerPoint to create presentations. Now let us see how to create presentations using Java. After completion of this chapter, you will be able to create new MS-PowerPoint presentations and open existing PPTs with your Java program.

Creating Empty Presentation

To create an empty presentation, you have to instantiate the **XMLSlideShow** class of the *org.poi.xslf.usermodel* package:

```
XMLSlideShow ppt = new XMLSlideShow();
```

Save the changes to a PPT document using the **FileOutputStream** class:

```
File file=new File("C://POIPPT//Examples//example1.pptx");
FileOutputStream out = new FileOutputStream(file);
ppt.write(out);
```

Given below is the complete program to create a blank MS-PowerPoint presentation.

```
import java.io.FileOutputStream;
import java.io.IOException;

import org.apache.poi.xslf.usermodel.XMLSlideShow;
import org.apache.poi.xslf.usermodel.XSLFSlide;

public class CreatePresentation {

   public static void main(String args[]) throws IOException{

      //creating a new empty slide show
      XMLSlideShow ppt = new XMLSlideShow();

      //creating an FileOutputStream object
      File file =new File("example1.pptx");
      FileOutputStream out = new FileOutputStream(file);

      //saving the changes to a file
      ppt.write(out);
      System.out.println("Presentation created successfully");
      out.close()
   }
}
```

Save the above Java code as **CreatePresentation.java**, and then compile and execute it from the command prompt as follows:

```
$javac  CreatePresentation.java
$java CreatePresentation
```

If your system environment is configured with the POI library, it will compile and execute to generate a blank PPT file named **example1.pptx** in your current directory and display the following output on the command prompt:

```
Presentation created successfully
```

The blank PowerPoint document appears as follows:
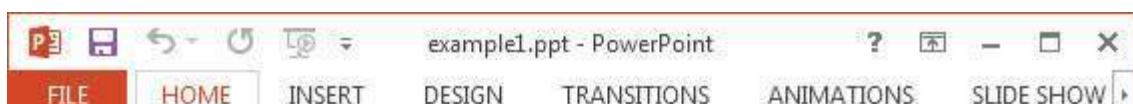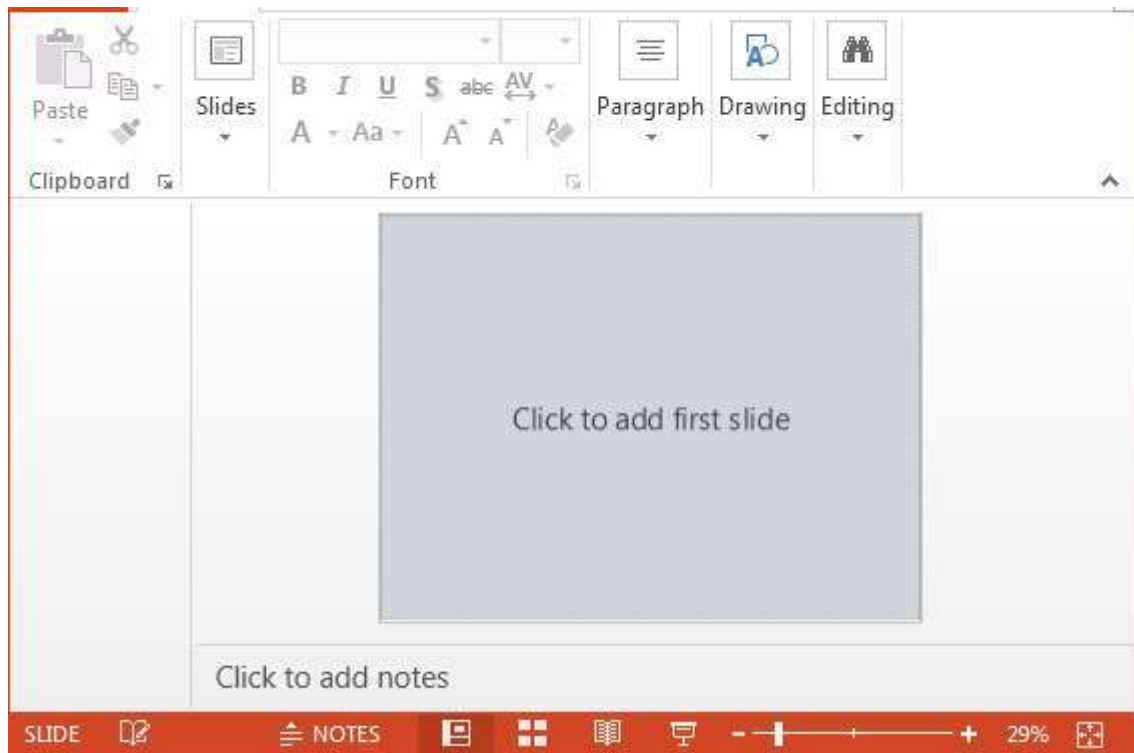
Editing an Existing Presentation

To open an existing presentation, instantiate the **XMLSlideShow** class and pass the **FileInputStream** object of the file to be edited, as an argument to the **XMLSlideShow** constructor.

```
File file=new File("C://POIPPT//Examples//example1.pptx");
FileInputstream inputstream =new FileInputStream(file);
XMLSlideShow ppt = new XMLSlideShow(inputstream);
```

You can add slides to a presentation using the **createSlide** method of the XMLSlideShow class which is in the *org.poi.xslf.usermodel* package.

```
XSLFSlide slide1= ppt.createSlide();
```

Given below is the complete program to open and add slides to an existing PPT:

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

import org.apache.poi.xslf.usermodel.XMLSlideShow;
import org.apache.poi.xslf.usermodel.XSLFSlide;

public class EditPresentation {

   public static void main(String ar[]) throws IOException{

      //opening an existing slide show
      File file = new File("example1.pptx");
      FileInputStream inputstream=new FileInputStream(file);
      XMLSlideShow ppt = new XMLSlideShow(inputstream);

      //adding slides to the slodeshow
      XSLFSlide slide1 = ppt.createSlide();
      XSLFSlide slide2 = ppt.createSlide();

      //saving the changes
      FileOutputStream out = new FileOutputStream(file);
      ppt.write(out);
```

```
        System.out.println("Presentation edited successfully");
        out.close();
    }
}
```
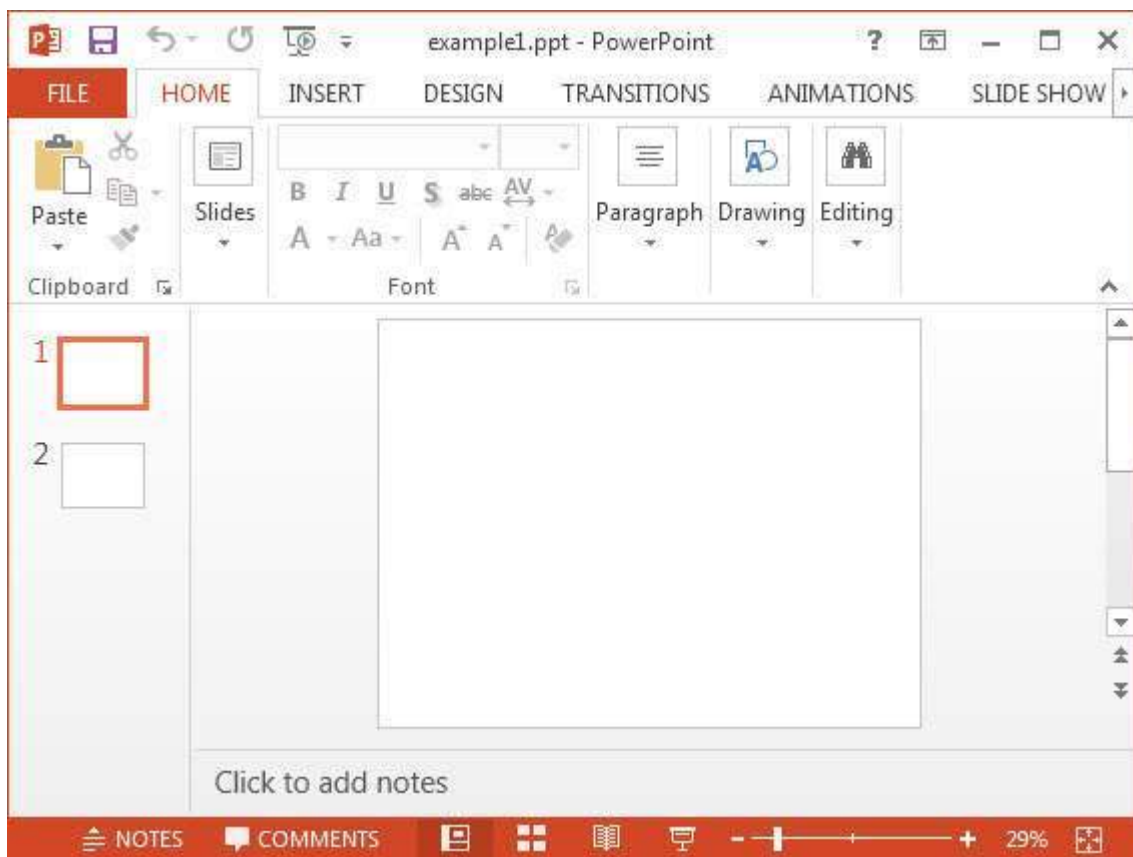
Save the above Java code as **EditPresentation.java**, and then compile and execute it from the command prompt as follows:

```
$javac   EditPresentation.java
$java EditPresentation
```

It will compile and execute to generate the following output:

```
slides successfully added
```

The output PPT document with newly added slides looks as follows:

After adding slides to a PPT, you can add, perform, read, and write operations on the slides.

APACHE POI PPT - SLIDE LAYOUTS

In the previous chapter, you have seen how to create empty slides and how to add slides to it. In this chapter, you will learn how to get the list of available slides, and how to create a slide with different layouts.

Available Slide layouts

PowerPoint presentations have slide layouts, and you can choose a desired layout to edit a slide. First of all, let us find out the list of all the slide layouts available.

- There are different slide masters and in each slide master, there are several slide layouts.

- You can get the list of the slide masters using the **getSlideMasters** method of the **XMLSlideShow** class.

- You can get the list of the slide layouts from each slide master using the **getSlideLayouts**

method of the **XSLFSlideMaster** class.

- You can get the name of the slide layout from the layout object using the **getType** method of the **XSLFSlideLayout** class.

Note : All these classes belongs to *org.poi.xslf.usermodel* package.

Given below is the complete program to get the list of available slide layouts in the PPT:

```java
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;

import org.apache.poi.xslf.usermodel.XMLSlideShow;
import org.apache.poi.xslf.usermodel.XSLFSlideLayout;
import org.apache.poi.xslf.usermodel.XSLFSlideMaster;
public class SlideLayouts {
    public static void main(String args[]) throws IOException{

    //create an empty presentation
       XMLSlideShow ppt = new XMLSlideShow();
       System.out.println("Available slide layouts:")

       //getting the list of all slide masters
       for(XSLFSlideMaster master : ppt.getSlideMasters()){

           //getting the list of the layouts in each slide master
           for(XSLFSlideLayout layout : master.getSlideLayouts()){

               //getting the list of available slides
               System.out.println(layout.getType());
           }
       }
    }
}
```

Save the above Java code as **SlideLayouts.java** , and then compile and execute it from the command prompt as follows:
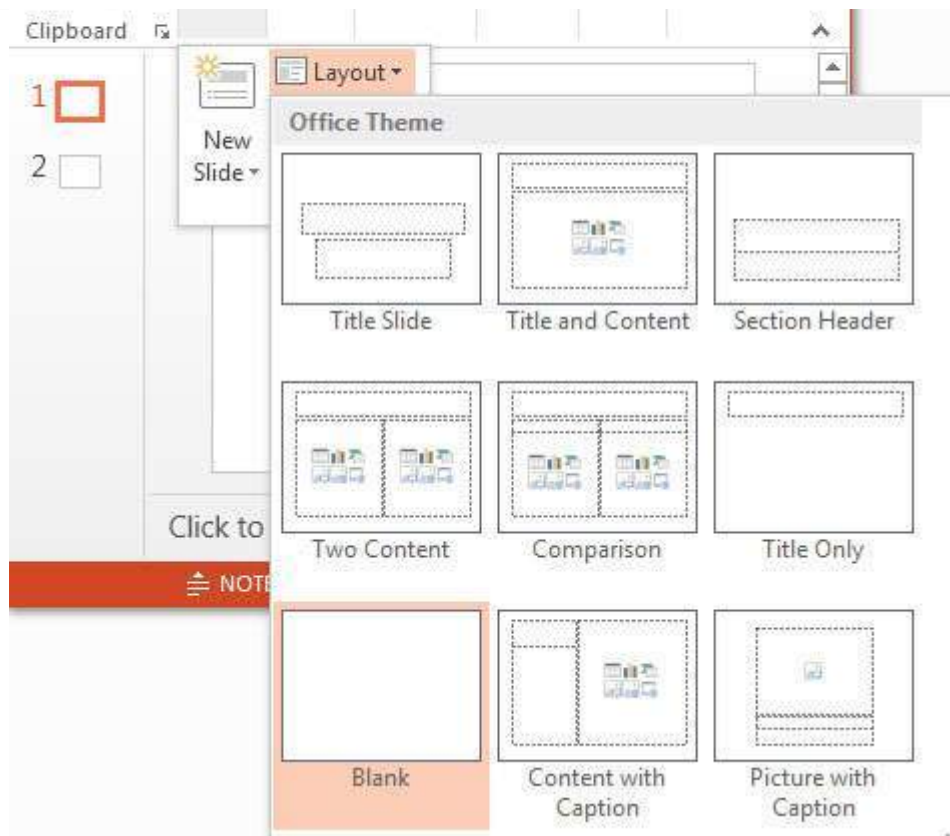
```
$javac   SlideLayouts.java
$java SlideLayouts
```

It will compile and execute to generate the following output:

```
Available slide layouts:
TITLE
PIC_TX
VERT_TX
TWO_TX_TWO_OBJ
BLANK
VERT_TITLE_AND_TX
TITLE_AND_CONTENT
TITLE_ONLY
SECTION_HEADER
TWO_OBJ
OBJ_TX
```

Shown below are some of the sample slide layouts available with MS-Office 360, 2013 edition.
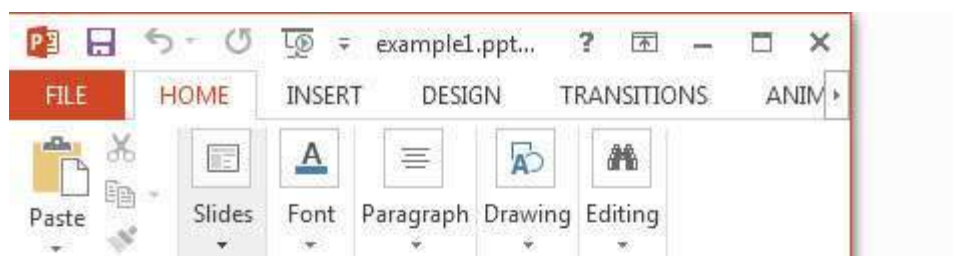
Title Layout

Let us create a slide in a PPT using Title layout. Follow the steps given below:

Step 1 : Create an empty presentation by instantiating the **XMLSlideShow** class as shown below:

```
XMLSlideShow ppt = new XMLSlideShow();
```

Step 2 : Get the list of slide masters using the **getSlideMasters** method. Thereafter, select the desired slide master using the index as shown below:

```
XSLFSlideMaster slideMaster = ppt.getSlideMasters()[0];
```

Here we are getting the default slide master which is in the 0th location of the slide masters array.

Step 3 : Get the desired layout using the **getLayout** method of the **XSLFSlideMaster** class. This method accepts a parameter where you have to pass one of the static variable of the **SlideLayoutclass**, which represents our desired layout. There are several variables in this class where each variable represents a slide layout.

The code snippet given below shows how to create a title layout:

```
XSLFSlideLayout titleLayout = slideMaster.getLayout(SlideLayout.TITLE);
```

Step 4 : Create a new slide by passing a slide layout object as parameter.

```
XSLFSlide slide = ppt.createSlide(titleLayout);
```

Step 5 : Select a placeholder using the **getPlaceholder** method of the **XSLFSlide** class. This method accepts an integer parameter. By passing 0 to it, you will get the **XSLFTextShape** object, using which you can access the title text area of the slide. Set the title using the setText method as shown below:

```
XSLFTextShape title1 = slide.getPlaceholder(0);
//setting the title init
title1.setText("Tutorials point");
```

Given below is the complete program to create a slide with Title layout in a presentation:

```java
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;

import org.apache.poi.xslf.usermodel.SlideLayout;
import org.apache.poi.xslf.usermodel.XMLSlideShow;
import org.apache.poi.xslf.usermodel.XSLFSlide;
import org.apache.poi.xslf.usermodel.XSLFSlideLayout;
import org.apache.poi.xslf.usermodel.XSLFSlideMaster;
import org.apache.poi.xslf.usermodel.XSLFTextShape;

public class TitleLayout {

    public static void main(String args[]) throws IOException{

        //creating presentation
        XMLSlideShow ppt = new XMLSlideShow();

        //getting the slide master object
        XSLFSlideMaster slideMaster = ppt.getSlideMasters()[0];

        //get the desired slide layout
        XSLFSlideLayout titleLayout = slideMaster.getLayout(SlideLayout.TITLE);

        //creating a slide with title layout
        XSLFSlide slide1 = ppt.createSlide(titleLayout);

        //selecting the place holder in it
        XSLFTextShape title1 = slide1.getPlaceholder(0);

        //setting the title init
        title1.setText("Tutorials point");

        //create a file object
        File file=new File("C://POIPPT//Examples//Titlelayout.pptx");
        FileOutputStream out = new FileOutputStream(file);

        //save the changes in a PPt document
        ppt.write(out);
        System.out.println("slide cretated successfully");
        out.close();
    }
}
```
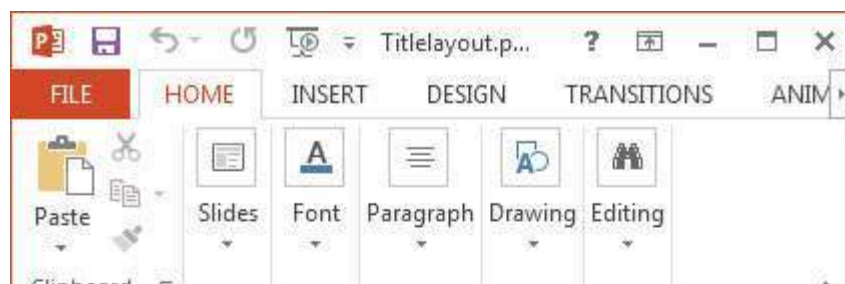
Save the above Java code as TitleLayout.java, and then compile and execute it from the command prompt as follows:

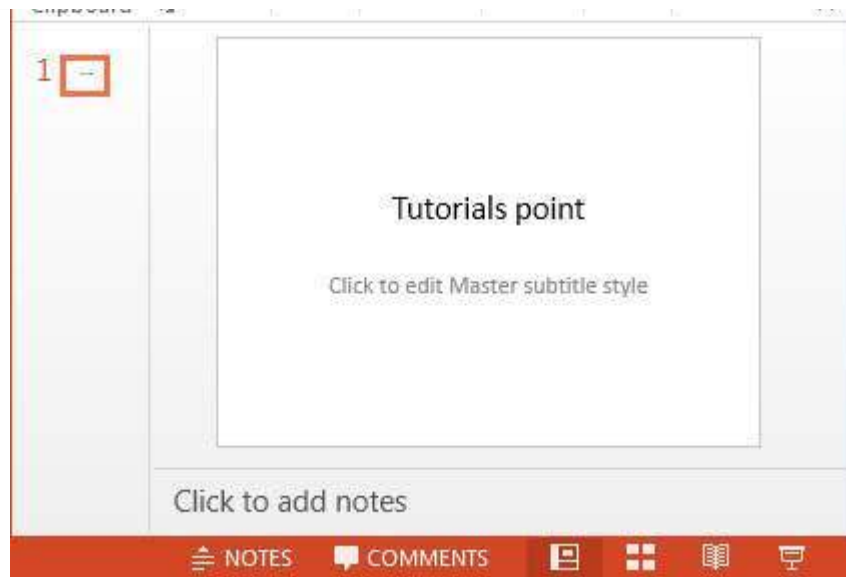```
$javac TitleLayout.java
$java TitleLayout
```

It will compile and execute to generate the following output.

```
slide created successfully
```

The PPT document with newly added Title layout slide appears as follows:

Title and content Layout

Let us create a slide in a PPT using Title and content layout. Follow the steps given below.

Step 1 : Create an empty presentation by instantiating the **XMLSlideShow** class as shown below:

```
XMLSlideShow ppt = new XMLSlideShow();
```

Step 2 : Get the list of slide masters using the **getSlideMasters** method. Select the desired slide master using the index as shown below:

```
XSLFSlideMaster slideMaster = ppt.getSlideMasters()[0];
```

Here we are getting the default slide master which is in the 0th location of the slide masters array.

Step 3 : Get the desired layout using the **getLayout** method of the **XSLFSlideMaster** class. This method accepts a parameter where you have to pass one of the static variable of the **SlideLayout** class which represents our desired layout. There are several variables in this class that represent slide layouts.

The following code snippet shows how to create title and content layout:

```
XSLFSlideLayout contentlayout = slideMaster.getLayout(SlideLayout.TITLE_AND_CONTENT);
```

Step 4 : Create a new slide by passing the slide layout object as parameter.

```
XSLFSlide slide = ppt.createSlide(SlideLayout.TITLE_AND_CONTENT);
```

Step 5 : Select a placeholder using the **getPlaceholder** method of the **XSLFSlide** class. This method accepts an integer parameter. By passing 1 to it, you will get the **XSLFTextShape** object, using which you can access the content area of the slide. Set the title using the setText method as shown below:

```
XSLFTextShape title1 = slide1.getPlaceholder(1);
//setting the title init
title1.setText("Introduction");
```

Step 6 : Clear the existing text in the slide using the **clearText** method of the **XSLFTextShape** class.

```
body.clearText();
```

Step 7 : Add new paragraph using the **addNewTextParagraph** method. Now add a new text run to the paragraph using the **addNewTextRun** method. Now to the text run, add text using the

setText method as shown below:

```
body.addNewTextParagraph().addNewTextRun().setText("this is  my first slide body");
```

Given below is the complete program to create a slide with Title layout in a presentation:

```java
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;

import org.apache.poi.xslf.usermodel.SlideLayout;
import org.apache.poi.xslf.usermodel.XMLSlideShow;
import org.apache.poi.xslf.usermodel.XSLFSlide;
import org.apache.poi.xslf.usermodel.XSLFSlideLayout;
import org.apache.poi.xslf.usermodel.XSLFSlideMaster;
import org.apache.poi.xslf.usermodel.XSLFTextShape;

public class TitleAndBodyLayout {

    public static void main(String args[]) throws IOException{

        //creating presentation
        XMLSlideShow ppt = new XMLSlideShow();

        //getting the slide master object
        XSLFSlideMaster slideMaster = ppt.getSlideMasters()[0];

        //select a layout from specified list
        XSLFSlideLayout slidelayout = slideMaster.getLayout(SlideLayout.TITLE_AND_CONTENT);

        //creating a slide with title and content layout
        XSLFSlide slide = ppt.createSlide(slidelayout);
        //selection of title place holder
        XSLFTextShape title = slide.getPlaceholder(0);

        //setting the title in it
        title.setText("introduction");

        //selection of body placeholder
        XSLFTextShape body = slide.getPlaceholder(1);

        //clear the existing text in the slide
        body.clearText();

        //adding new paragraph
        body.addNewTextParagraph().addNewTextRun().setText("this is  my first slide body");

        //create a file object
        File file=new File("contentlayout.pptx");
        FileOutputStream out = new FileOutputStream(file);

        //save the changes in a file
        ppt.write(out);
        System.out.println("slide cretated successfully");
        out.close();
    }
}
```
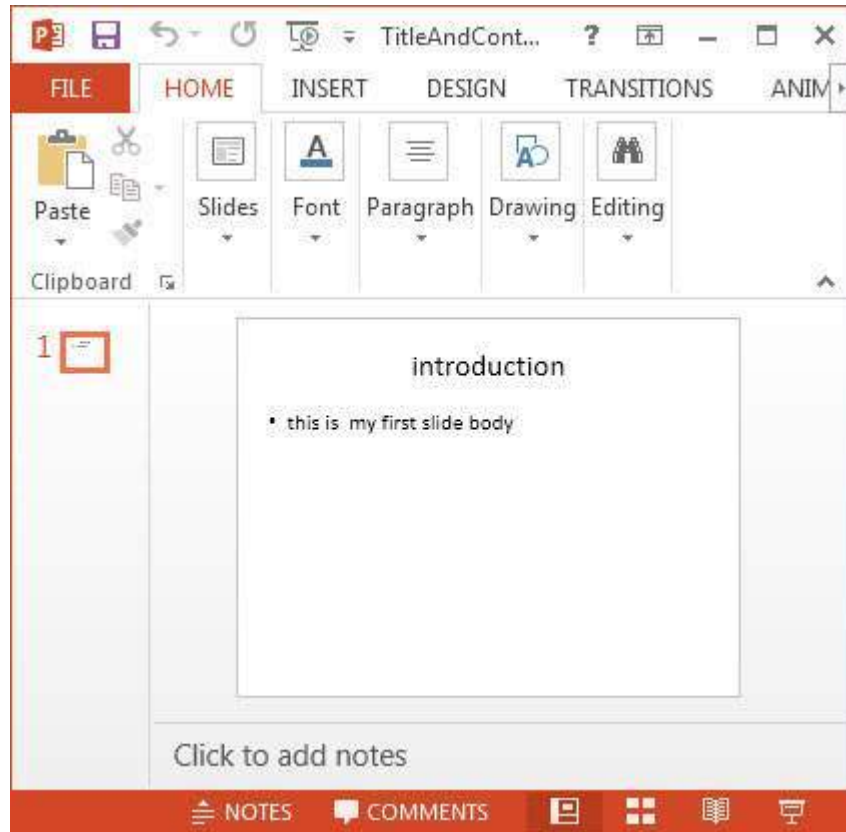
Save the above Java code as **TitleLayout.java**, and then compile and execute it from the command prompt as follows:

```
$javac  TitleLayout.java
$java TitleLayout
```

It will compile and execute to generate the following output:

```
slide created successfully
```

The PPT document with newly added Title layout slide appears as follows:

In the same way, you can create slides with different layouts as well.

APACHE POI PPT - MANAGEMENT OF SLIDES

After completing this chapter, you will be able to delete, reorder, and perform read and write operations on a slide.

Changing a Slide

We can change the page size of a slide using the **setPageSize** method of the **XMLSlideShow** class.

Initially create a presentation as shown below:

```
File file=new File("C://POIPPT//Examples// TitleAndContentLayout.pptx");
//create presentation
XMLSlideShow ppt = new XMLSlideShow(new FileInputStream(file));
```

Get the size of the current slide using the **getPageSize** method of the **XMLSlideShow** class.

```
java.awt.Dimension pgsize = ppt.getPageSize();
```

Set the size of the page using the **setPageSize** method.

```
ppt.setPageSize(new java.awt.Dimension(1024, 768));
```

The complete program for changing the size of a slide is given below:

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import org.apache.poi.xslf.usermodel.XMLSlideShow;
```

```java
public class ChangingSlide {

    public static void main(String args[]) throws IOException{

        //create file object
        File file = new File("TitleAndContentLayout.pptx");

        //create presentation
        XMLSlideShow ppt = new XMLSlideShow();

        //getting the current page size
        java.awt.Dimension pgsize = ppt.getPageSize();
        int pgw = pgsize.width; //slide width in points
        int pgh = pgsize.height; //slide height in points
        System.out.println("current page size of the PPT is:");
        System.out.println("width :" + pgw);
        System.out.println("height :" + pgh);

        //set new page size
        ppt.setPageSize(new java.awt.Dimension(2048,1536));

        //creating file object
        FileOutputStream out = new FileOutputStream(file);

        //saving the changes to a file
        ppt.write(out);
        System.out.println("slide size changed to given dimentions ");
        out.close();
    }
}
```

Save the above Java code as **ChangingSlide.java**, and then compile and execute it from the command prompt as follows:
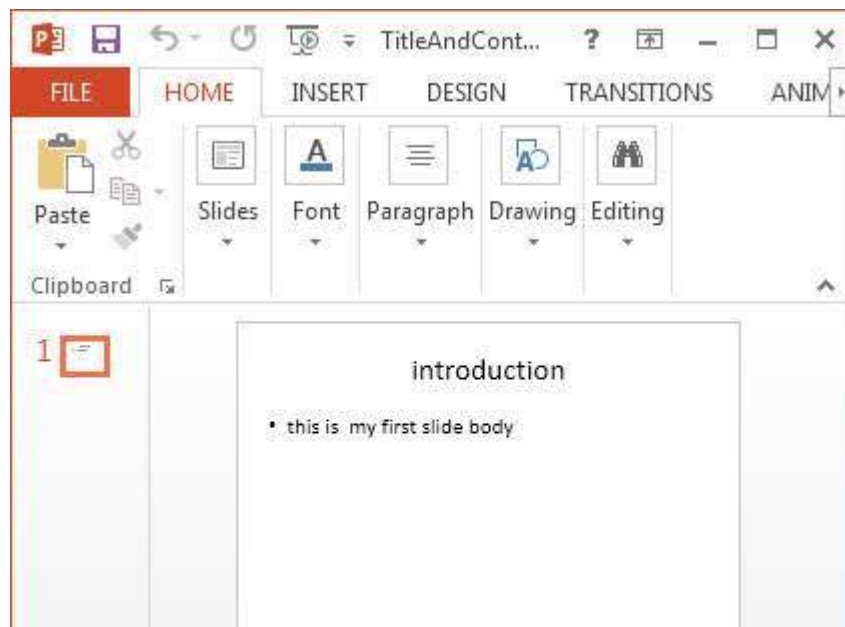
```
$javac ChangingSlide.java
$java ChangingSlide
```

It will compile and execute to generate the following output.
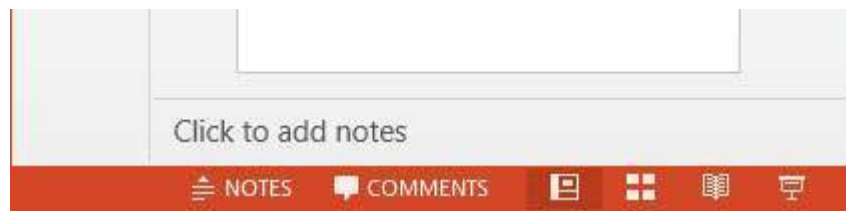
```
current page size of the presentation is :
width :720
height :540
slide size changed to given dimensions
```
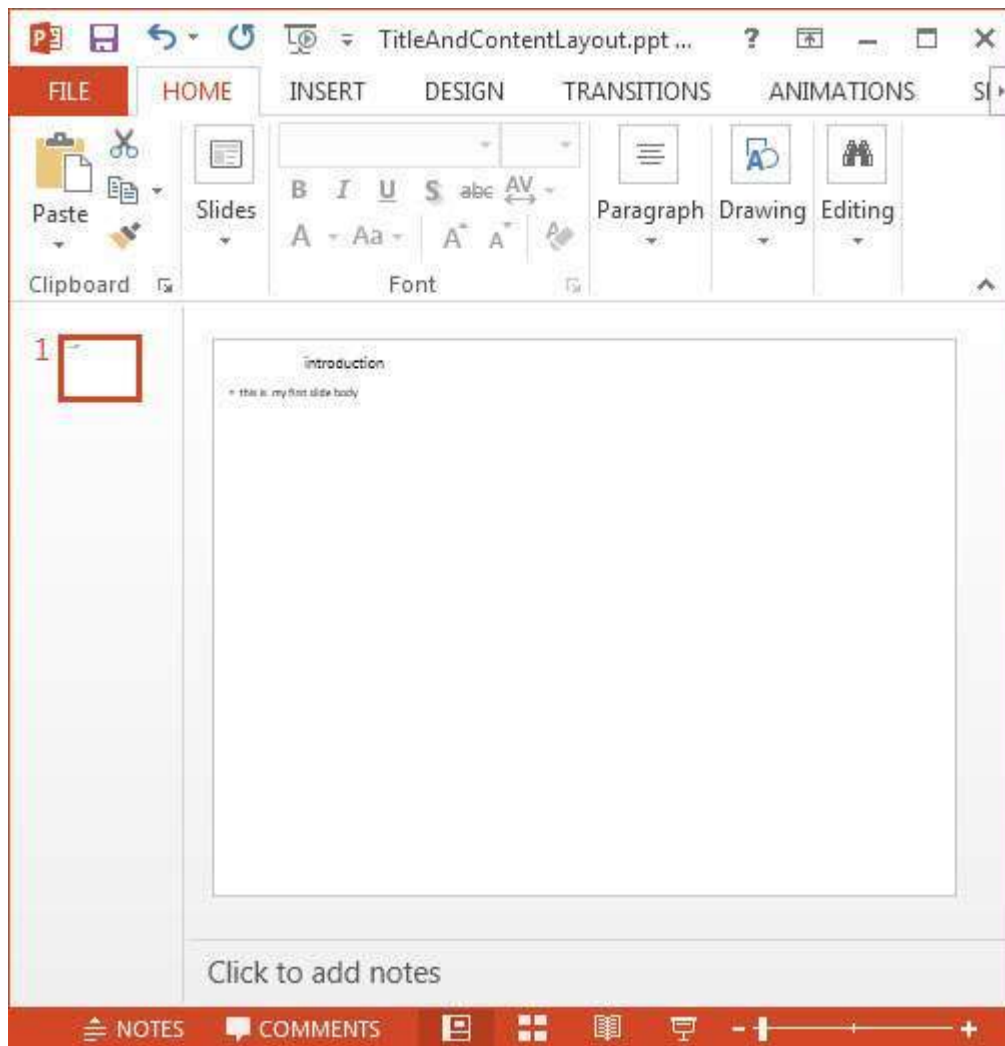
Given below is the snapshot of the presentation before changing the slide size:

The slide appears as follows after changing its size:

Reordering Slides

You can set the slide order using the **setSlideOrder** method. Given below is the procedure to set the order of the slides.

Open an existing PPT document as shown below:

```
File file=new File("C://POIPPT//Examples//example1.pptx");
XMLSlideShow ppt = new XMLSlideShow(new FileInputStream(file));
```

Get the slides using the **getSlides** method as shown below:

```
XSLFSlide[] slides =ppt.getSlides();
```

Select a slide from the array of the slides, and change the order using the **setSlideOrder** method as shown below:

```
//selecting the fourth slide
XSLFSlide selectesdslide= slides[4];
//bringing it to the top
ppt.setSlideOrder(selectesdslide, 1);
```

Given below is the complete program to reorder the slides in a presentation:

```java
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import org.apache.poi.xslf.usermodel.XMLSlideShow;
import org.apache.poi.xslf.usermodel.XSLFSlide;

public class ReorderSlide {

    public static void main(String args[]) throws IOException{

        //opening an existing presentation
        File file=new File("example1.pptx");
        XMLSlideShow ppt = new XMLSlideShow(new FileInputStream(file));

        //get the slides
        XSLFSlide[] slides = ppt.getSlides();

        //selecting the fourth slide
        XSLFSlide selectesdslide = slides[13];

        //bringing it to the top
        ppt.setSlideOrder(selectesdslide, 0);

        //creating an file object
        FileOutputStream out = new FileOutputStream(file);

        //saving the changes to a file
        ppt.write(out);
        out.close();
    }
}
```
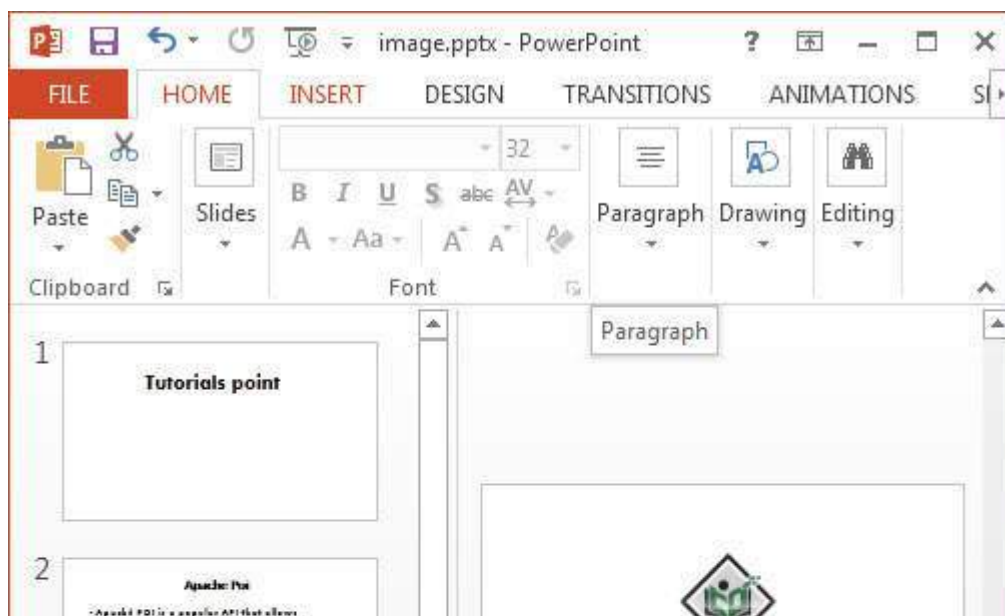
Save the above Java code as **ReorderSlide.java**, and then compile and execute it from the command prompt as follows:

```
$javac   ReorderSlide.java
$java ReorderSlide
```

It will compile and execute to generate the following output.
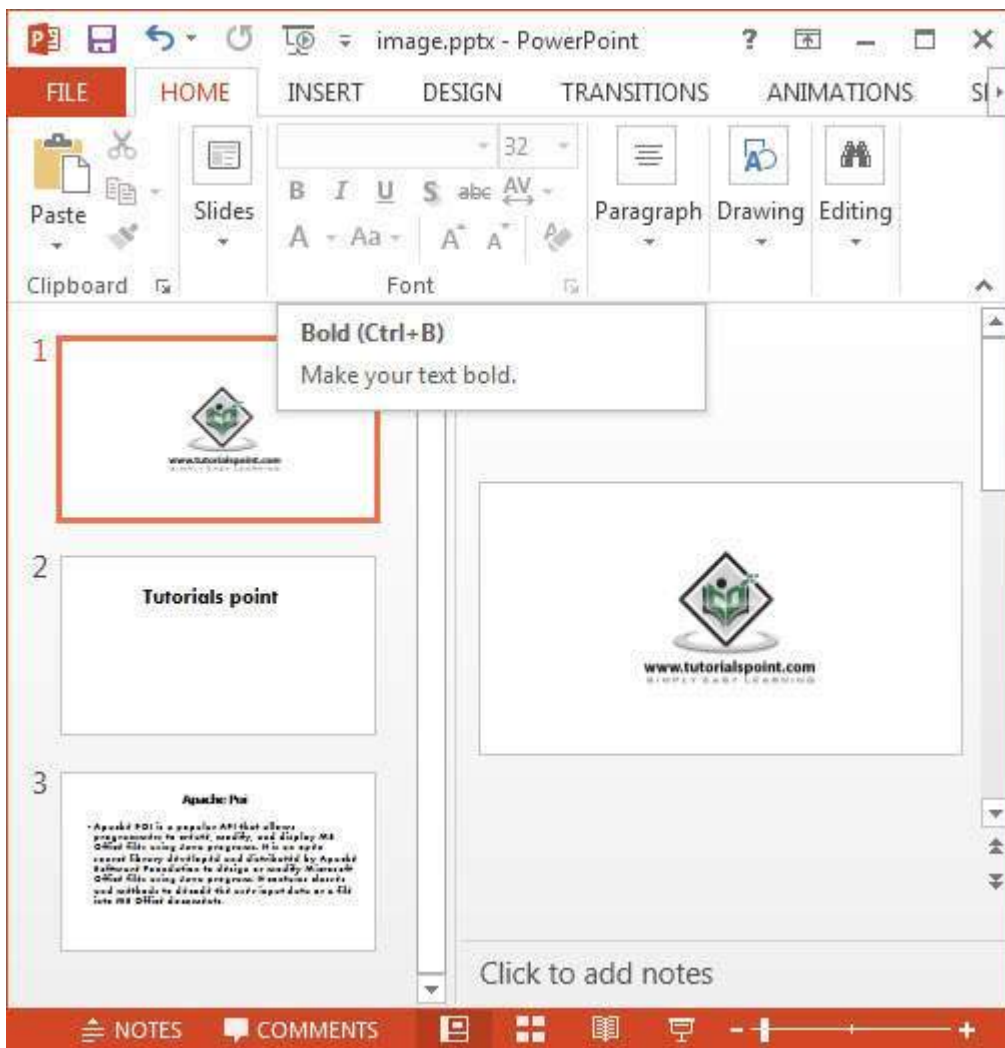
```
Reordering of the slides is done
```

Given below is the snapshot of the presentation before reordering the slides:

After reordering the slides, the presentation appears as follows. Here we have selected the slide with image and moved it to the top.

Deleting Slides

You can delete the slides using the **removeSlide** method. Follow the steps given below to delete slides.

Open an existing presentation using the **XMLSlideShow** class as shown below:

```
File file=new File("C://POIPPT//Examples//image.pptx");
XMLSlideShow ppt = new XMLSlideShow(new FileInputStream(file));
```

Delete the required slide using the **removeSlide** method. This method accepts an integer parameter. Pass the index of the slide that is to be deleted to this method.

```
ppt.removeSlide(1);
```

Given below is the program to delete slides from a presentation:

```java
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

import org.apache.poi.xslf.usermodel.XMLSlideShow;

public class Deleteslide {

    public static void main(String args[]) throws IOException{

        //Opening an existing slide
        File file=new File("image.pptx");
        XMLSlideShow ppt = new XMLSlideShow(new FileInputStream(file));

        //deleting a slide
        ppt.removeSlide(1);

        //creating a file object
        FileOutputStream out = new FileOutputStream(file);

        //Saving the changes to the presentation
        ppt.write(out);
        out.close();
    }
}
```
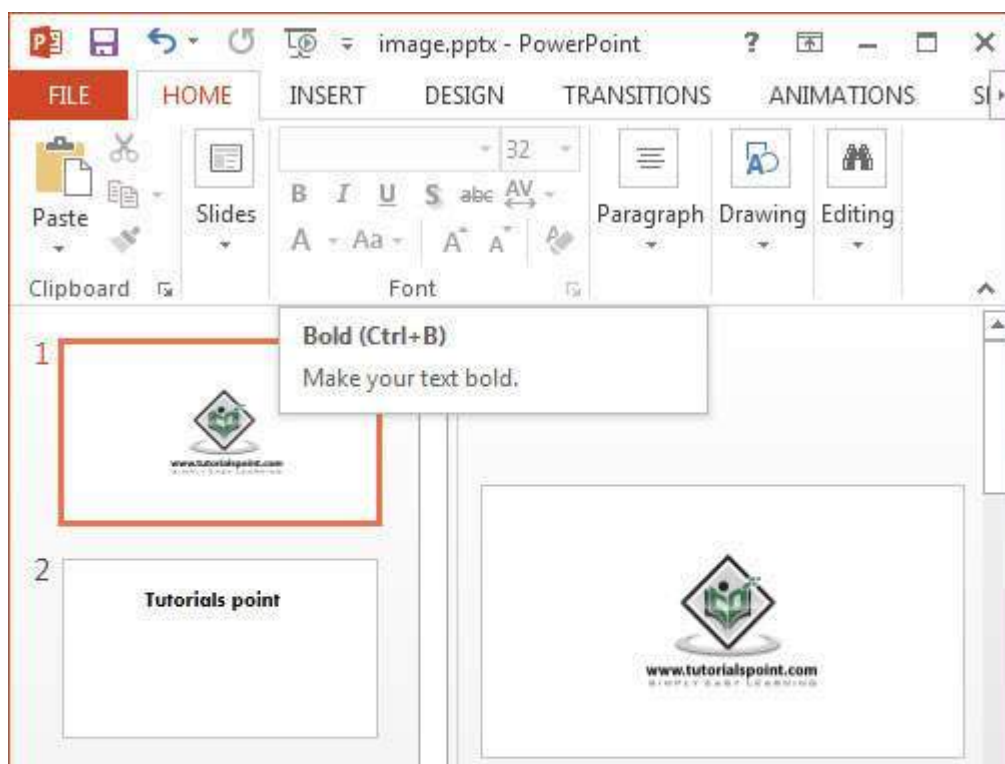
Save the above Java code as **Deleteslide.java**, and then compile and execute it from the command prompt as follows:

```
$javac   Deleteslide.java
$java Deleteslide
```
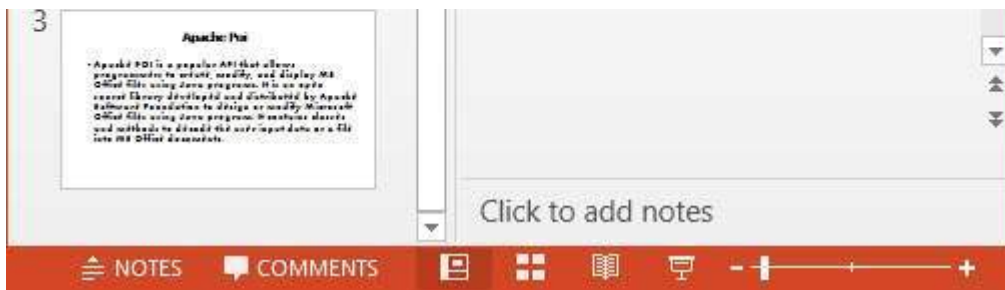
It will compile and execute to generate the following output:

```
reordering of the slides is done
```
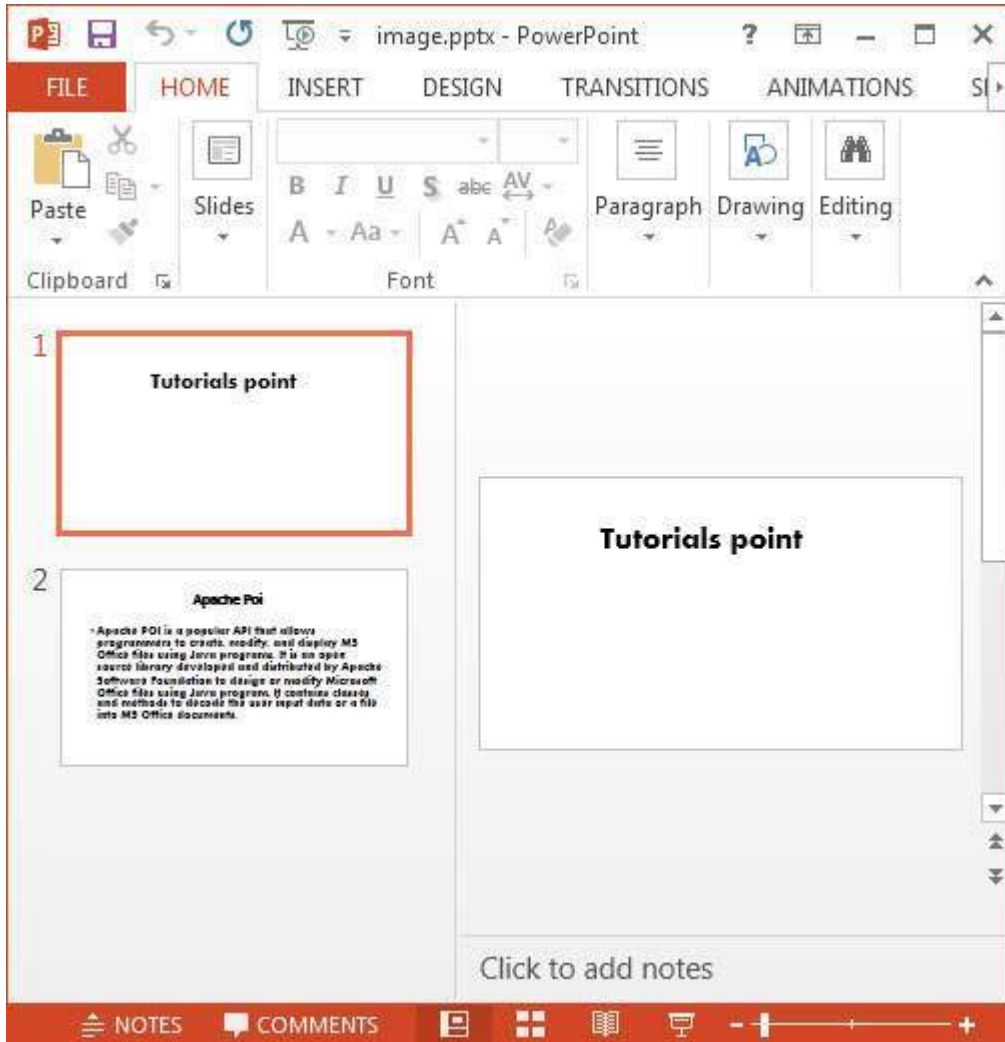
The snapshot below is of the presentation before deleting the slide:

After deleting the slide, the presentation appears as follows:

APACHE POI PPT - IMAGES

In this chapter, you will learn how to add an image to a PPT and how to read an image from it.

Adding Image

You can add images to a presentation using the **createPicture** method of **XSLFSlide**. This method accepts image in the form of byte array format. Therefore, you have to create a byte array of the image that is to be added to the presentation.

Follow the given procedure to add an image to a presentation. Create an empty slideshow using **XMLSlideShow** as shown below:

```
XMLSlideShow ppt = new XMLSlideShow();
```

Create an empty presentation in it using **createSlide**.

```
XSLFSlide slide = ppt.createSlide();
```

Read the image file that is to be added and convert it into byte array using **IOUtils.toByteArray** of the **IOUtils** class as shown below:

```
//reading an image
File image=new File("C://POIPPT//boy.jpg");
//converting it into a byte array
byte[] picture = IOUtils.toByteArray(new FileInputStream(image));
```

Add the image to the presentation using **addPicture**. This method accepts two variables: byte array format of the image that is to be added and the static variable representing the file format of the image. The usage of the **addPicture** method is shown below:

```
int idx = ppt.addPicture(picture, XSLFPictureData.PICTURE_TYPE_PNG);
```

Embed the image to the slide using **createPicture** as shown below:

```
XSLFPictureShape pic = slide.createPicture(idx);
```

Given below is the complete program to add an image to the slide in a presentation:

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

import org.apache.poi.util.IOUtils;
import org.apache.poi.xslf.usermodel.XMLSlideShow;
import org.apache.poi.xslf.usermodel.XSLFPictureData;
import org.apache.poi.xslf.usermodel.XSLFPictureShape;
import org.apache.poi.xslf.usermodel.XSLFSlide;

public class AddingImage {

    public static void main(String args[]) throws IOException{

        //creating a presentation
        XMLSlideShow ppt = new XMLSlideShow();

        //creating a slide in it
        XSLFSlide slide = ppt.createSlide();

        //reading an image
        File image=new File("C://POIPPT//boy.jpg");

        //converting it into a byte array
        byte[] picture = IOUtils.toByteArray(new FileInputStream(image));

        //adding the image to the presentation
        int idx = ppt.addPicture(picture, XSLFPictureData.PICTURE_TYPE_PNG);

        //creating a slide with given picture on it
        XSLFPictureShape pic = slide.createPicture(idx);

        //creating a file object
        File file=new File("addingimage.pptx");
        FileOutputStream out = new FileOutputStream(file);

        //saving the changes to a file
        ppt.write(out)
        System.out.println("image added successfully");
        out.close();
    }
}
```
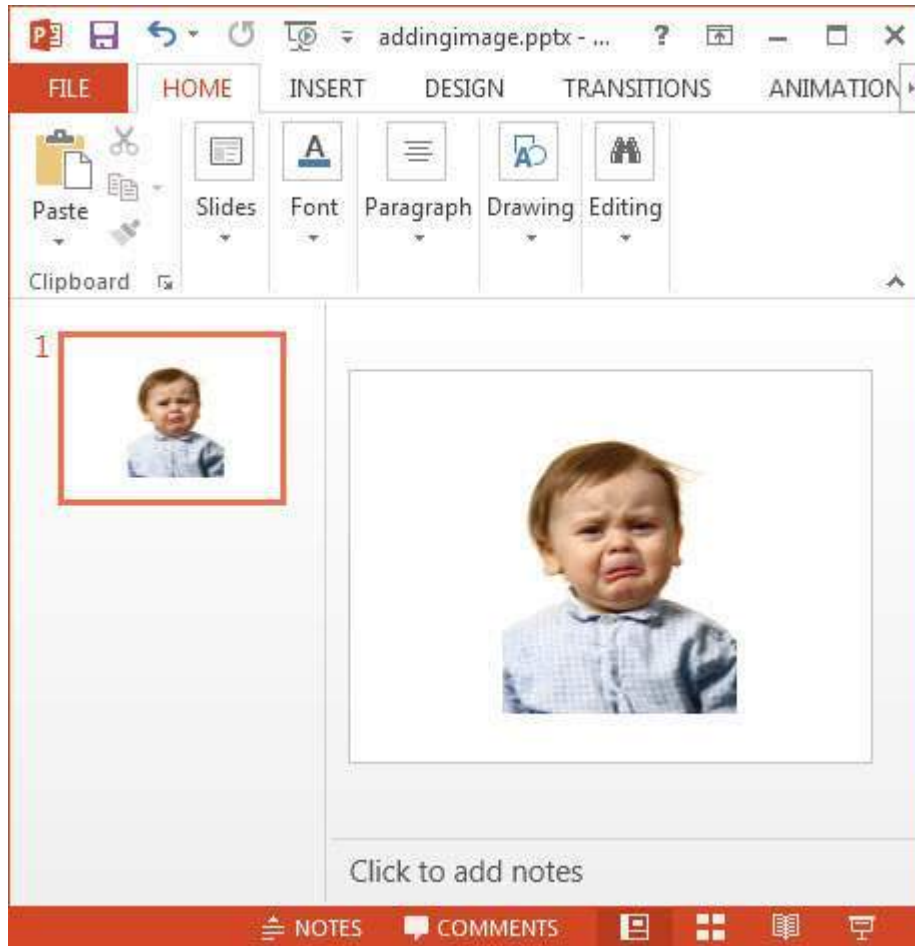
Save the above Java code as **AddingImage.java**, and then compile and execute it from the command prompt as follows:

```
$javac  AddingImage.java
$java AddingImage
```

It will compile and execute to generate the following output:

```
reordering of the slides is done
```

The presentation with the newly added slide with image appears as follows:

Reading Image

You can get the data of all the pictures using the **getAllPictures** method of the **XMLSlideShow** class. The following program reads the images from a presentation:

```java
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

import org.apache.poi.xslf.usermodel.XMLSlideShow;
import org.apache.poi.xslf.usermodel.XSLFPictureData;

public class Readingimage {

   public static void main(String args[]) throws IOException{

       //open an existing presentation
       File file = new File("addingimage.pptx");
       XMLSlideShow ppt = new XMLSlideShow(new FileInputStream(file));

       //reading all the pictures in the presentation
       for(XSLFPictureData data : ppt.getAllPictures()){
```

```
            byte[] bytes = data.getData();
            String fileName = data.getFileName();
            int pictureFormat = data.getPictureType();
            System.out.println("picture name: " + fileName);
            System.out.println("picture format: " + pictureFormat);
        }

        //saving the changes to a file
        FileOutputStream out = new FileOutputStream(file);
        ppt.write(out);
        out.close();
    }
}
```

Save the above Java code as **Readingimage.java**, and then compile and execute it from the command prompt as follows:

```
$javac  Readingimage.java
$java Readingimage
```

It will compile and execute to generate the following output:

```
picture name: image1.png
picture format: 6
```

APACHE POI PPT - CREATING HYPERLINKS

In this chapter you will learn how to create hyperlinks in a presentation.

Creating Hyperlinks

You can read the hyperlinks in a presentation using the **createHyperlink** method of the **XSLFTextRun** class. Follow the procedure given below to create a hyperlink in a presentation.

Create an empty presentation using the **XMLSlideShow** class as shown below:

```
XMLSlideShow ppt = new XMLSlideShow();
```

Create an empty slide and create a textbox and body of the slide using body and content layout.

```
//create an empty presentation
XSLFSlideMaster slideMaster = ppt.getSlideMasters()[0];
//creating a slide with title and content layout
XSLFSlideLayout slidelayout = slideMaster.getLayout(SlideLayout.TITLE_AND_CONTENT);
XSLFSlide slide = ppt.createSlide(slidelayout);
//selection of body place holder
XSLFTextShape body = slide.getPlaceholder(1);
//clear the existing text in the slide
body.clearText();
```

Create a text run object and set text to it as shown below:

```
XSLFTextRun textRun=body.addNewTextParagraph().addNewTextRun();
textRun.setText("Tutorials point");
```

Create a hyperlink using the **createHyperlink** method of the **XSLFTextRun** class as shown below:

```
XSLFHyperlink link = textRun.createHyperlink();
```

Set the link address to the hyperlink using the **setAddress** method of **XSLFHyperlink** class as shown below:

```
link.setAddress("http://www.tutorialspoint.com/");
```

Given below is the complete program to create hyperlink in a presentation:

```java
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;

import org.apache.poi.xslf.usermodel.SlideLayout;
import org.apache.poi.xslf.usermodel.XMLSlideShow;
import org.apache.poi.xslf.usermodel.XSLFHyperlink;
import org.apache.poi.xslf.usermodel.XSLFSlide;
import org.apache.poi.xslf.usermodel.XSLFSlideLayout;
import org.apache.poi.xslf.usermodel.XSLFSlideMaster;
import org.apache.poi.xslf.usermodel.XSLFTextRun;
import org.apache.poi.xslf.usermodel.XSLFTextShape;

public class CreatingHyperlinks {

    public static void main(String args[]) throws IOException{

        //create an empty presentation
        XMLSlideShow ppt = new XMLSlideShow();

        //getting the slide master object
        XSLFSlideMaster slideMaster = ppt.getSlideMasters()[0];

        //select a layout from specified list
        XSLFSlideLayout slidelayout = slideMaster.getLayout(SlideLayout.TITLE_AND_CONTENT);

        //creating a slide with title and content layout
        XSLFSlide slide = ppt.createSlide(slidelayout);

        //selection of title place holder
        XSLFTextShape body = slide.getPlaceholder(1);

        //clear the existing text in the slid
        body.clearText();

        //adding new paragraph
        XSLFTextRun textRun = body.addNewTextParagraph().addNewTextRun();

        //setting the text
        textRun.setText("Tutorials point");
        //creating the hyperlink
        XSLFHyperlink link = textRun.createHyperlink();

        //setting the link address
        link.setAddress("http://www.tutorialspoint.com/");

        //create the file object
        File file=new File("hyperlink.pptx");
        FileOutputStream out = new FileOutputStream(file);

        //save the changes in a file
        ppt.write(out);
        System.out.println("slide cretated successfully");
        out.close();
    }
}
```
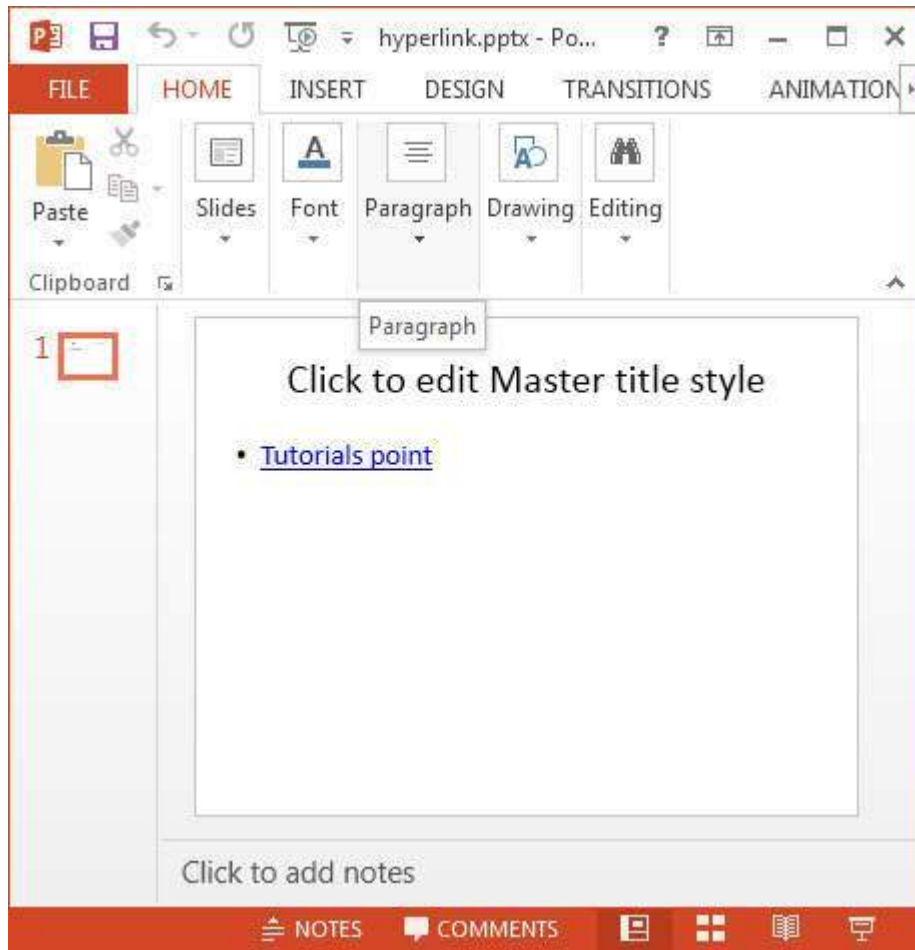
Save the above Java code as **CreatingHyperlinks.java**, and then compile and execute it from the command prompt as follows:

```
$javac  CreatingHyperlinks.java
$java CreatingHyperlinks
```

It will compile and execute to generate the following output:

```
slide cretated successfully
```

The newly added slide with the hyperlink in its body looks as follows:

APACHE POI PPT - READING SHAPES

Reading Shapes from a Presentation

You can get a count of the number of shapes used in a presentation using the method **getShapeName** of the **XSLFShape** class. Given below is the program to read the shapes from a presentation:

```java
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

import org.apache.poi.xslf.usermodel.XMLSlideShow;
import org.apache.poi.xslf.usermodel.XSLFShape;
import org.apache.poi.xslf.usermodel.XSLFSlide;

public class ReadingShapes {

   public static void main(String args[]) throws IOException{

      //creating a slideshow
      File file = new File("shapes.pptx");
      XMLSlideShow ppt = new XMLSlideShow(new FileInputStream(file));

      //get slides
      XSLFSlide[] slide = ppt.getSlides();

      //getting the shapes in the presentation
      System.out.println("Shapes in the presentation:");
      for (int i = 0; i < slide.length; i++){
```

```
            XSLFShape[] sh = slide[i].getShapes();
            for (int j = 0; j < sh.length; j++){

                //name of the shape
                System.out.println(sh[j].getShapeName());
            }
        }
        FileOutputStream out = new FileOutputStream(file);
        ppt.write(out);
        out.close();
    }
}
```

Save the above Java code as **ReadingShapes.java**, and then compile and execute it from the command prompt as follows:

```
$javac   ReadingShapes.java
$java ReadingShapes
```
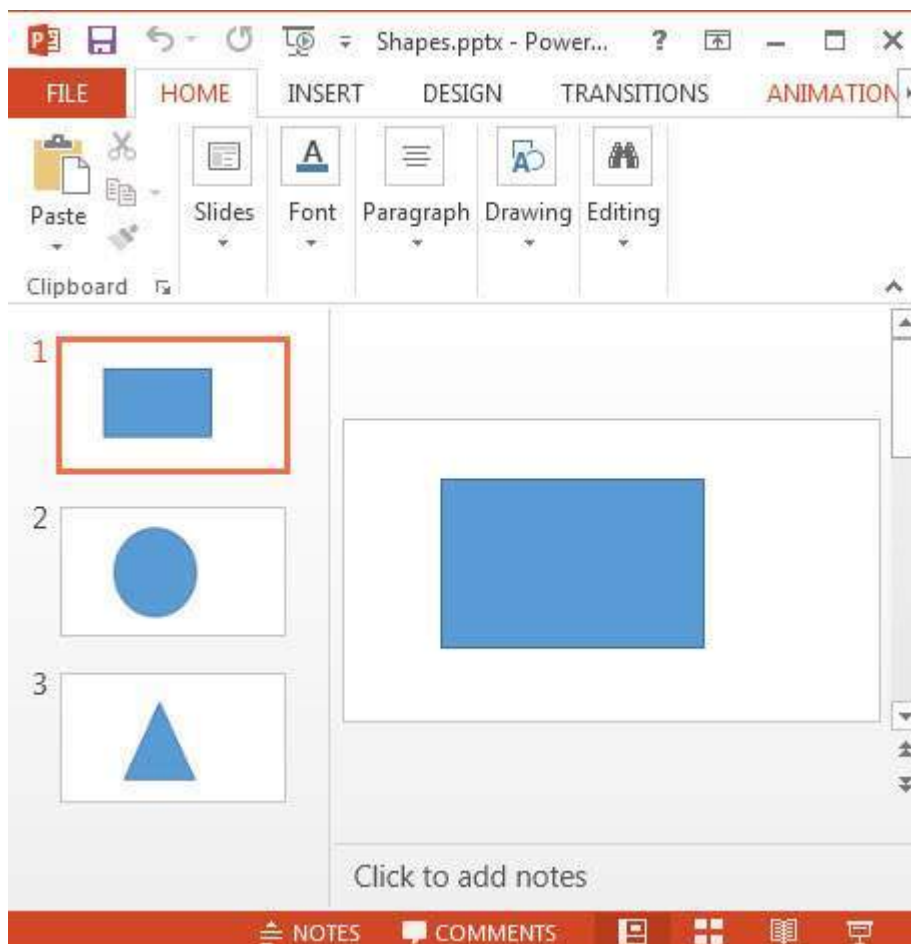
It will compile and execute to generate the following output.

```
Shapes in the presentation:
Rectangle 1
Oval 1
Isosceles Triangle 1
```

The newly added slide with the various shapes appears as follows:

APACHE POI PPT - FORMATTING TEXT

Formatting Text in a Presentation

The text in a presentation can be formatted using the methods of the **XSLFTextRun** class. For that, you have to create an **XSLFTextRun** class object by selecting one of the slide layouts as

shown below:

```
//create the empty presentation
XMLSlideShow ppt = new XMLSlideShow();
//getting the slide master object
XSLFSlideMaster slideMaster = ppt.getSlideMasters()[0];
//select a layout from specified list
XSLFSlideLayout slidelayout = slideMaster.getLayout(SlideLayout.TITLE_AND_CONTENT);
//creating a slide with title and content layout
XSLFSlide slide = ppt.createSlide(slidelayout);
//selection of title place holder
XSLFTextShape body = slide.getPlaceholder(1);
//clear the existing text in the slide
body.clearText();
//adding new paragraph
XSLFTextParagraph paragraph=body.addNewTextParagraph();
//creating text run object
XSLFTextRun run = paragraph.addNewTextRun();
```

You can set the font size of the text in the presentation using **setFontSize**.

```
run.setFontColor(java.awt.Color.red);
run.setFontSize(24);
```

The following code snippet shows how to apply different formatting styles *bold, italic, underline, strikeout* to the text in a presentation.

```
//change the text into bold format
run.setBold(true);
//change the text it to italic format
run.setItalic(true)
// strike through the text
run.setStrikethrough(true);
//underline the text
run.setUnderline(true);
```

To have line breaks between paragraphs, use **addLineBreak** of the **XSLFTextParagraph** class as shown below:

```
paragraph.addLineBreak();
```

Given below is the complete program to format the text using all the above methods:

```
import java.io.FileOutputStream;
import java.io.IOException;

import org.apache.poi.xslf.usermodel.SlideLayout;
import org.apache.poi.xslf.usermodel.XMLSlideShow;
import org.apache.poi.xslf.usermodel.XSLFSlide;
import org.apache.poi.xslf.usermodel.XSLFSlideLayout;
import org.apache.poi.xslf.usermodel.XSLFSlideMaster;
import org.apache.poi.xslf.usermodel.XSLFTextParagraph;
import org.apache.poi.xslf.usermodel.XSLFTextRun;
import org.apache.poi.xslf.usermodel.XSLFTextShape;

public class TextFormating {

    public static void main(String args[]) throws IOException{

        //creating an empty presentation
        XMLSlideShow ppt = new XMLSlideShow();

        //getting the slide master object
        XSLFSlideMaster slideMaster = ppt.getSlideMasters()[0];

        //select a layout from specified list
```

```java
        XSLFSlideLayout slidelayout = slideMaster.getLayout(SlideLayout.TITLE_AND_CONTENT);

        //creating a slide with title and content layout
        XSLFSlide slide = ppt.createSlide(slidelayout);

        //selection of title place holder
        XSLFTextShape body = slide.getPlaceholder(1);

        //clear the existing text in the slide
        body.clearText();

        //adding new paragraph
        XSLFTextParagraph paragraph=body.addNewTextParagraph();

        //formatting line1
        XSLFTextRun run1 = paragraph.addNewTextRun();
        run1.setText("This is a colored line");
        //setting color to the text
        run1.setFontColor(java.awt.Color.red);
        //setting font size to the text
        run1.setFontSize(24);
        //moving to the next line
        paragraph.addLineBreak();

        //formatting line2
        XSLFTextRun run2 = paragraph.addNewTextRun();
        run2.setText("This is a bold line");
        run2.setFontColor(java.awt.Color.CYAN);
        //making the text bold
        run2.setBold(true);
        paragraph.addLineBreak();

        //formatting line3
        XSLFTextRun run3 = paragraph.addNewTextRun();
        run3.setText(" This is a striked line");
        run3.setFontSize(12);
        //making the text italic
        run3.setItalic(true);
        //strike through the text
        run3.setStrikethrough(true);
        paragraph.addLineBreak();

        //formatting line4
        XSLFTextRun run4 = paragraph.addNewTextRun();
        run4.setText(" This an underlined line");
        run4.setUnderline(true);
        //underlining the text
        paragraph.addLineBreak();
        //creating a file object
        File file=new File("TextFormat.pptx");
        FileOutputStream out = new FileOutputStream(file);
        saving the changes to a file
        ppt.write(out);
        out.close();
    }
}
```

Save the above code as **TextFormating.java**, and then compile and execute it from the command prompt as follows:

```
$javac  TextFormating.java
$java TextFormating
```
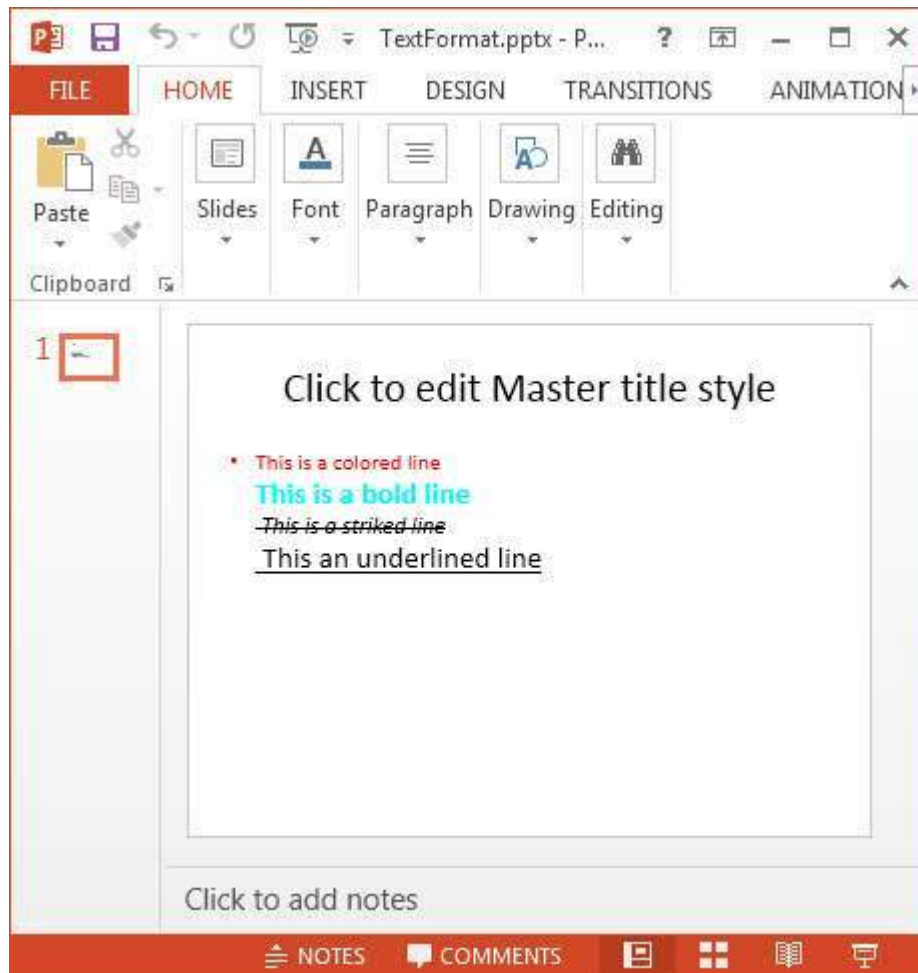
It will compile and execute to generate the following output:

```
Formatting completed successfully
```

The slide with formatted text appears as follows:

APACHE POI PPT - MERGING

Merging Multiple Presentations

You can merge multiple presentations using the **importContent** method of the **XMLSlideShow** class. Given below is the complete program to merge two presentations:

```java
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

import org.apache.poi.xslf.usermodel.XMLSlideShow;
import org.apache.poi.xslf.usermodel.XSLFSlide;

public class MergingMultiplePresentations {

    public static void main(String args[]) throws IOException{

    //creating empty presentation
    XMLSlideShow ppt = new XMLSlideShow();

    //taking the two presentations that are to be merged
    String file1 = "presentation1.pptx";
    String file2 = "presentation2.pptx";
    String[] inputs = {file1, file2};

    for(String arg : inputs){

        FileInputStream inputstream = new FileInputStream(arg);
        XMLSlideShow src = new XMLSlideShow(inputstream);
        for(XSLFSlide srcSlide : src.getSlides()){

            //merging the contents
```

```
            ppt.createSlide().importContent(srcSlide);
        }
    }

    String file3 = "combinedpresentation.pptx";

    //creating the file object
    FileOutputStream out = new FileOutputStream(file3);

    // saving the changes to a file
    ppt.write(out);
    System.out.println("Merging done successfully");
    out.close();
    }
}
```

Save the above code as **MergingMultiplePresentations.java**, and then compile and execute it from the command prompt as follows:
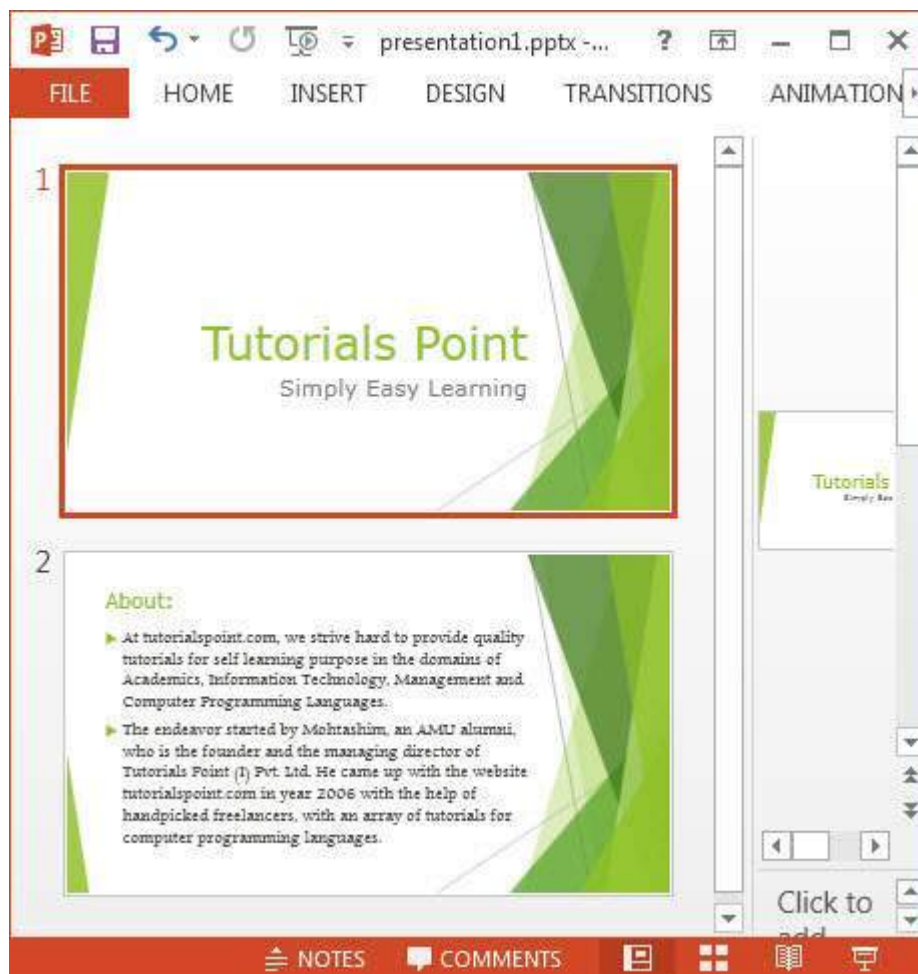
```
$javac  MergingMultiplePresentations.java
$java MergingMultiplePresentations
```

It will compile and execute to generate the following output:

```
Merging done successfully
```

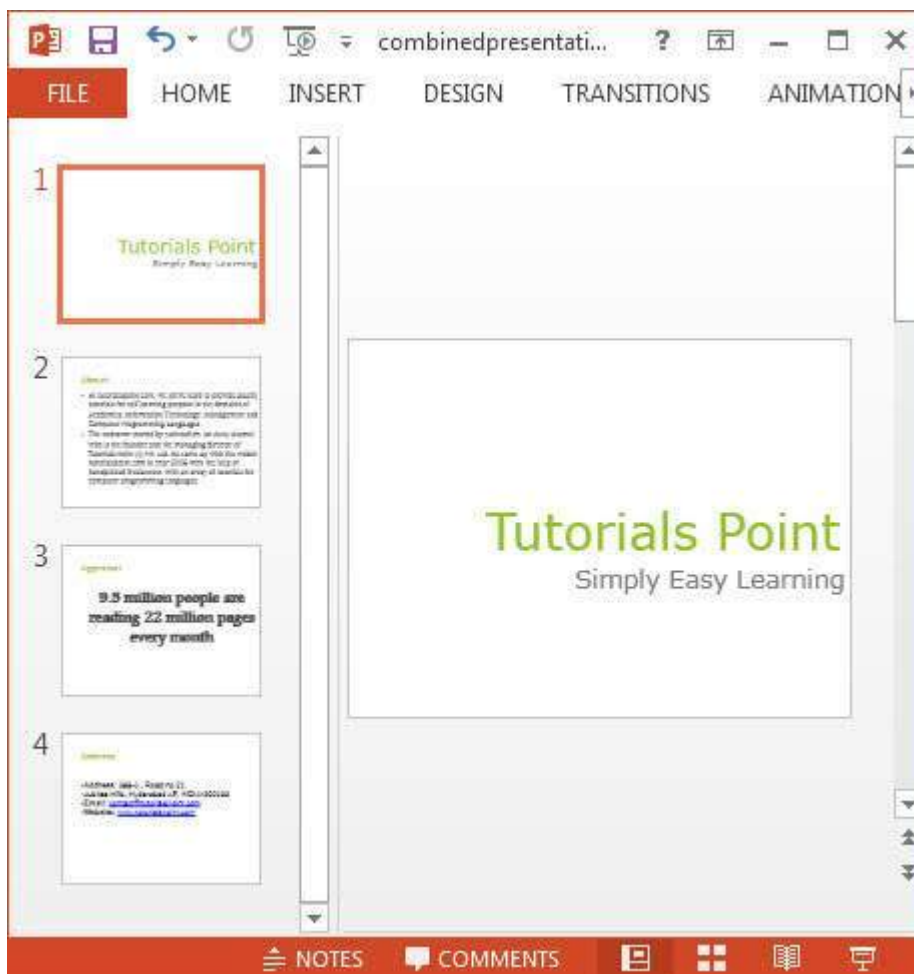The following snapshot shows the first presentation:

The following snapshot shows the second presentation:

Given below is the output of the program after merging the two slides. Here you can see the content of the earlier slides merged together.

APACHE POI PPT - PPT TO IMAGE

Converting Presentation to Image

You can convert a presentation to an image file. The following program shows how to go about it.

```java
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

import org.apache.poi.xslf.usermodel.XMLSlideShow;
import org.apache.poi.xslf.usermodel.XSLFSlide;

public class PpttoPNG {

   public static void main(String args[]) throws IOException{

      //creating an empty presentation
      File file=new File("addingimage.pptx");
      XMLSlideShow ppt = new XMLSlideShow(new FileInputStream(file));

      //getting the dimensions and size of the slide
      Dimension pgsize = ppt.getPageSize();
      XSLFSlide[] slide = ppt.getSlides();
      for (int i = 0; i < slide.length; i++) {
      BufferedImage img = new BufferedImage(pgsize.width,
pgsize.height,BufferedImage.TYPE_INT_RGB);
      Graphics2D graphics = img.createGraphics();

      //clear the drawing area
      graphics.setPaint(Color.white);
      graphics.fill(new Rectangle2D.Float(0, 0, pgsize.width, pgsize.height));

      //render
      slide[i].draw(graphics);

      //creating an image file as output
      FileOutputStream out = new FileOutputStream("ppt_image.png");
      javax.imageio.ImageIO.write(img, "png", out);
      ppt.write(out);
      System.out.println("Image successfully created");
      out.close();
   }
}
```

Save the above Java code as **PpttoPNG.java**, and then compile and execute it from the command prompt as follows:

```
$javac  PpttoPNG.java
$java PpttoPNG
```

It will compile and execute to generate the following output:

```
Image created successfully
```

The following snapshot shows the presentation that is given as input:

Given below is the snapshot of the image created at the specified location.