**Apache Tika**

Tika is a java library which can be used for detecting document types, detecting language of document’s content and extracting content/metadata from various types of file. It uses many existing document parsers libraries such as JDom, Jackson, JSoup, XMLBeans, JAXB, POI, PDF Box, etc. for content/metadata extraction. It utilizes less memory and able to extract content quickly.

It provides APIs to identify content language and detect mime type of file (based on file extension).

It provided generic interface for parsing content and extracting content/metadata by encapsulates all the third party parser libraries within a single parser interface. It is capable of extracting content from various popular files formats such PDF, Word, Excel Sheet, CSV, Text, Images, XML, JSON etc. There is a Tika class which is the simplest and direct way of calling Tika from Java. It follows the facade design pattern. You can find the class in the Tika API at [**org.apache.tika.Tika**](https://tika.apache.org/1.12/api/org/apache/tika/Tika.html)

Search engines, content management systems and machine learning tools uses Tika for content/metadata extraction, document analysis, content analysis, indexing and translation.

**Pre-requisite:**

1. JDK 1.6 or Above
2. Download latest Tika dependencies (1.12 is the latest version currently). See the list of dependencies given below:

|  |
| --- |
| *<dependencies>*  *<dependency>*  *<groupId>org.apache.tika</groupId>*  *<artifactId>tika-core</artifactId>*  *<version>1.12</version>*  *</dependency>*  *<dependency>*  *<groupId>org.apache.tika</groupId>*  *<artifactId>tika-parsers</artifactId>*  *<version>1.12</version>*  *</dependency>*  *<dependency>*  *<groupId>org.apache.tika</groupId>*  *<artifactId>tika-serialization</artifactId>*  *<version>1.12</version>*  *</dependency>*  *<dependency>*  *<groupId>org.apache.tika</groupId>*  *<artifactId>tika-app</artifactId>*  *<version>1.12</version>*  *</dependency>*  *<dependency>*  *<groupId>org.apache.tika</groupId>*  *<artifactId>tika-bundle</artifactId>*  *<version>1.12</version>*  *</dependency>*  *<dependency>*  *<groupId>org.apache.tika</groupId>*  *<artifactId>tika-batch</artifactId>*  *<version>1.12</version>*  *</dependency>*  *<dependency>*  *<groupId>org.apache.tika</groupId>*  *<artifactId>tika-translate</artifactId>*  *<version>1.12</version>*  *</dependency>*  *</dependencies>* |

You can find the TIKA dependencies from below given URL:

<http://mvnrepository.com/search?q=org.apache.tika>

You can also refer to the [GitHub](https://github.com/apache/tika)repository, if you are interested in exploring the source code.

You can refer to documentation here: <http://tika.apache.org/index.html>

Getting started: <http://tika.apache.org/1.12/gettingstarted.html>

Parser Guide: <https://tika.apache.org/1.12/parser_guide.html>

See examples here: <https://tika.apache.org/1.8/examples.html>

Find the JavaDocs here: <https://tika.apache.org/1.12/api/>

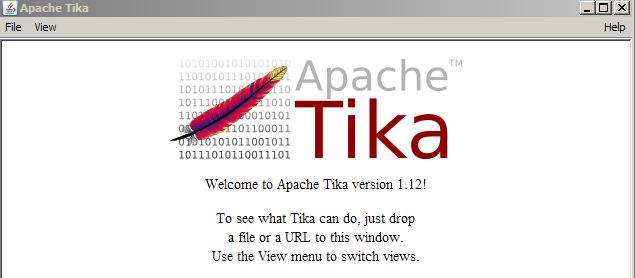
So let’s explore the available APIs and try to do some hands-on. First we will try to use Tika GUI and see the magic of Tika.

Follow the below given steps to use Tika GUI:

1. Open command prompt.
2. Locate the ‘tika-app-1.12.jar’ and copy the full path.
3. Run below command to open Tika GUI.

|  |
| --- |
| C:\Users\Abhinav\.m2\repository\org\apache\tika\tika-app\1.12>java -jar tika-app-1.12.jar -g |

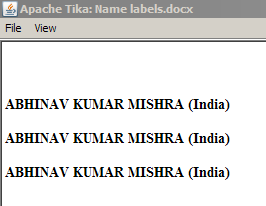
1. It will open below shown GUI.



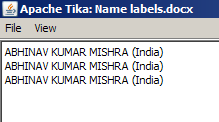
1. Go to ‘File’ menu and select ‘Open’ menu to open any file of your choice. E.g. A word document
2. Once you select the file app will automatically extract the metadata and content from the selected file. You can see all available metadata on selected document. E.g.

|  |
| --- |
| *Application-Name: Microsoft Office Word*  *Application-Version: 14.0000*  *Author: Abhinav*  *Character Count: 3*  *Character-Count-With-Spaces: 3*  *Content-Length: 18700*  *Content-Type: application/vnd.openxmlformats-officedocument.wordprocessingml.document*  *Creation-Date: 2016-02-25T12:49:00Z*  *Last-Author: Abhinav*  *Last-Modified: 2016-02-25T12:56:00Z*  *Last-Save-Date: 2016-02-25T12:56:00Z*  *Line-Count: 1*  *Page-Count: 1*  *Paragraph-Count: 1*  *Revision-Number: 2* |

1. Go to ‘View’ option, you will see options such as “Metadata, Formatted text, Plain text, Main content, Structured text (in xml format) and Recursive JSON”. By default “Metadata” will be selected as view in app.
2. Let’s see how “Formatted text” will look like. I selected “Formatted text” as view; it will display the content as it is in word document. See the screen below:



1. Let’s see how “Plain text” will look like. I selected “Plain text” as view; it will display the content as plain text. See the screen below:



Similarly you can view other options also. This app is helpful in case if you want to test the capabilities of Tika and also want to see how the extracted metadata or content will look like after processing.

**Type Detection in Tika:**

Tika identifies the MimeType of any file. It uses following mechanisms.

1. File Extension: File extension is used to identify the MimeType.
2. Content-type Metadata: If file extension is lost due to some reason, it uses the metadata supplied with the file to detect the MimeType.
3. Magic bytes info: If you observe the raw bytes of a file (open the file in notepad), you will notice some unique character patterns in each type of file. Some files have special byte prefixes called *magic bytes* that are specially made and included in a file for the purpose of identifying the mime type. e.g. you can find CA FE BA BE (hexadecimal format) in a java file and %PDF (ASCII format) in a PDF file. Tika uses this information to identify the media type of a file.
4. Character encodings: It identifies the mime type of plain text files using their character encoding.
5. XML characters (<, />): To identify the xml file type, Tika first parses the xml document and extract the information such as namespace, processing instruction etc.

Let try it with a simple java code. Here we will use the “[**org.apache.tika.Tika**](https://tika.apache.org/1.12/api/org/apache/tika/Tika.html)” class as discussed above.

|  |
| --- |
| **package** com.github.abhinavmishra14.tika;  **import** java.io.File;  **import** java.util.Scanner;  **import** org.apache.tika.Tika;  **public class** FileTypeDetection {  **public static void** main(String[] args) **throws** Exception {  **try** (**final** Scanner scanner = **new** Scanner(System.***in***);) {  System.***out***.println("Please enter a fileName/filePath: ");  **final** String filePath = scanner.nextLine();  **final** File fileObject = **new** File(filePath);  //Instantiate TIKA facade class  **final** Tika tika = **new** Tika();  //Call detect method of the TIKA class  **final** String filetype = tika.detect(fileObject);  System.***out***.println("\nDetected fileType: "+filetype);  }  }  } |

**Output:**

|  |
| --- |
| Please enter a fileName/filePath:  C:\Users\abhinav\Desktop\inside-marklogic-server-r7.pdf  Detected fileType: application/pdf  Please enter a fileName/filePath:  C:\Users\abhinav\Desktop\set-jre-args-share.png  Detected fileType: image/png |

**Content extraction in TIKA:**

Tika uses multiple parsers to extract the content from given file. Tika chooses suitable parser for the given file is decided by [Tika](https://tika.apache.org/1.12/api/org/apache/tika/Tika.html) facade class based on the file type detection. Tika provides multiple overloaded and useful methods to extract the content. The most used method of Tika class is “***parseToString (...)***”. It can extract the content of a file given from file system and it can parse the file from a URL as well. There is one more method ***parse (....)***. It returns the instance of ***java.io.Reader***. It is useful if you want to store the extracted content into file, string, stream etc.

We will see example of above discussed methods.

See the complete list of useful methods using following link: <https://tika.apache.org/1.12/api/org/apache/tika/Tika.html>

Following steps will be performed by Tika to extract the content:

1. Detect the file type of file using type detection mechanism (as mentioned above).
2. Once type of file is known, choose the suitable parser from its parser library.
3. Parser will parse the content, extract the text, and also throw exceptions for unreadable formats.

Let try it with a simple java code. Here we will use the “[**org.apache.tika.Tika**](https://tika.apache.org/1.12/api/org/apache/tika/Tika.html)” class.

**I have a text file “helloTika.txt” which has following content:**

|  |
| --- |
| *A "Hello, World!" program is a computer program that outputs "Hello, World!" on a display device, often standard output.* |

|  |
| --- |
| **package** com.github.abhinavmishra14.tika;  **import** java.io.File;  **import** java.io.IOException;  **import** java.util.Scanner;  **import** org.apache.tika.Tika;  **import** org.apache.tika.exception.TikaException;  **public** **class** ExtractionTest {  **public** **static** **void** main(String[] args) **throws** IOException, TikaException {  **try** (**final** Scanner scanner = **new** Scanner(System.***in***);) {  System.***out***.println("Please enter a fileName/filePath: ");  **final** String filePath = scanner.nextLine();  **final** File fileObject = **new** File(filePath);  //Instantiate TIKA facade class  **final** Tika tika = **new** Tika();  //Call the parseToString method to get the extracted content as string.  **final** String extractedContent = tika.parseToString(fileObject);  System.***out***.println("\nExtracted content: "+extractedContent);  }  }  } |

**Output:**

|  |
| --- |
| Please enter a fileName/filePath:  C:\Users\abhinav\Desktop\helloTika.txt  Extracted content: A "Hello, World!" program is a computer program that outputs "Hello, World!" on a display device, often standard output. |

**Let’s try the same with as XML file which is accessed via URL:**

<http://www.w3schools.com/xsl/books.xml>

|  |
| --- |
| **package** com.github.abhinavmishra14.tika;  **import** java.io.IOException;  **import** java.net.URL;  **import** java.util.Scanner;  **import** org.apache.tika.Tika;  **import** org.apache.tika.exception.TikaException;  **public** **class** ExtractionTest {  **public** **static** **void** main(String[] args) **throws** IOException, TikaException {  **try** (**final** Scanner scanner = **new** Scanner(System.***in***);) {  System.***out***.println("Please enter a fileURL: ");  **final** String fileURL = scanner.nextLine();  //Create the instance of URL  **final** URL url = **new** URL(fileURL);  //Instantiate TIKA facade class  **final** Tika tika = **new** Tika();  //Call the parseToString method to get the extracted content as string.  **final** String extractedContent = tika.parseToString(url);  System.***out***.println("\nExtracted content: "+extractedContent);  }  }  } |

**Output:**

|  |
| --- |
| Please enter a fileURL:  <http://www.w3schools.com/xsl/books.xml>  Extracted content:    Everyday Italian  Giada De Laurentiis  2005  30.00    Harry Potter  J K. Rowling  2005  29.99  XQuery Kick Start  James McGovern  Per Bothner  Kurt Cagle  James Linn  Vaidyanathan Nagarajan  2003  49.99    Learning XML  Erik T. Ray  2003  39.95 |

**Let’s save the extracted content to a file:**

|  |
| --- |
| **package** com.github.abhinavmishra14.tika;  **import** java.io.File;  **import** java.io.FileWriter;  **import** java.io.IOException;  **import** java.io.Reader;  **import** java.net.URL;  **import** java.util.Scanner;  **import** org.apache.commons.lang.StringUtils;  **import** org.apache.tika.Tika;  **import** org.apache.tika.exception.TikaException;  **public** **class** ExtractAndWriteToAFile {  **public** **static** **void** main(String[] args) **throws** IOException, TikaException {  **try** (**final** Scanner scanner = **new** Scanner(System.***in***);) {  System.***out***.println("Please enter a fileURL: ");  **final** String fileURL = scanner.nextLine();  //Create the instance of URL  **final** URL url = **new** URL(fileURL);  //Instantiate TIKA facade class  **final** Tika tika = **new** Tika();  //Call the parse method to get the extracted content as java.io.Reader object.  **final** Reader reader = tika.parse(url);  //Get the file name from the URL.  //Expected file name will be the name without extension  **final** File fileName = **new** File(  StringUtils.*substringBefore*(StringUtils.substringAfterLast(fileURL, "/"), "."));  **try**(**final** FileWriter fileWriter = **new** FileWriter(fileName);){  System.***out***.println("Writing to file..");  **int** character =-1;  **while** ((character = reader.read()) != -1) {  fileWriter.write(character);  }  System.***out***.println("Writing to file completed!");  }  }  }  } |

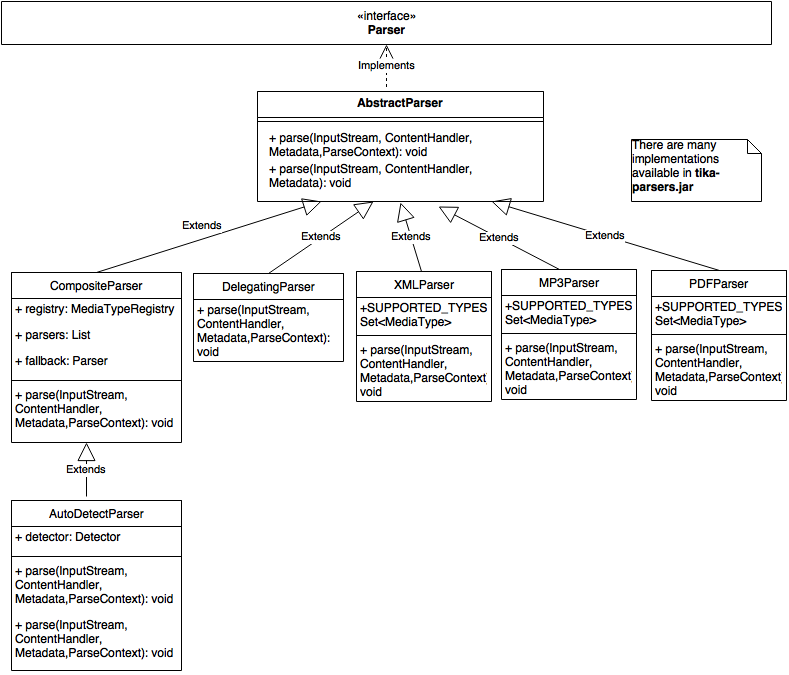
**Output:** (Open the saved file to see the saved content)

|  |
| --- |
| Please enter a fileURL:  http://www.w3schools.com/xsl/books.xml  Writing to file..  Writing to file completed! |

Till now we saw that, we are using the Tika facade class and calling the default implemented methods which are doing the job beautifully. What if you want to have more control over parsing? Hmm

Well we can do that, Tika API provides a [**Parser**](https://tika.apache.org/1.12/api/org/apache/tika/parser/Parser.html) Interface, which you can find under ‘org.apache.tika.parser’ package. This package has an [**AbstractParser**](https://tika.apache.org/1.12/api/org/apache/tika/parser/AbstractParser.html) class which implements the Parser Interface. This package also contains some basic parsers which are created by extending the AbstractParser class. You can also create your own parser implementation by extending AbstractParser or by implementing Parser Interface.

Let’s see how the parsers are arranged using the basic class diagram.



There are multiple parser implementations classes available in Tika, such as XMLParser, MP3Parser, and PDFParser, OfficeParser, OOXMLParser etc.

So, in order to use the parser’s implementations you need to follow below given steps.

1. Read the file into java.io.InputStream.
2. Create the instance of Parser. You can use any of these individual document parsers. Or you can use either [CompositeParser](https://tika.apache.org/1.12/api/org/apache/tika/parser/CompositeParser.html) or [AutoDetectParser](https://tika.apache.org/1.12/api/org/apache/tika/parser/AutoDetectParser.html) that uses all the parser classes internally and extracts the contents of a document using a suitable parser.
3. Create the instance of content handler; there are many content handlers available in org.apache.tika.sax package. Some of them are: [org.apache.tika.sax.BodyContentHandler](https://tika.apache.org/1.12/api/org/apache/tika/sax/BodyContentHandler.html) **,** [org.apache.tika.sax.LinkContentHandler](https://tika.apache.org/1.12/api/org/apache/tika/sax/LinkContentHandler.html) , [org.apache.tika.sax.PhoneExtractingContentHandler](https://tika.apache.org/1.12/api/org/apache/tika/sax/PhoneExtractingContentHandler.html), [org.apache.tika.sax.TeeContentHandler](https://tika.apache.org/1.12/api/org/apache/tika/sax/TeeContentHandler.html), [org.apache.tika.sax.ElementMappingContentHandler](https://tika.apache.org/1.12/api/org/apache/tika/sax/ElementMappingContentHandler.html), [org.xml.sax.helpers.DefaultHandler](https://docs.oracle.com/javase/6/docs/api/org/xml/sax/helpers/DefaultHandler.html) , Etc. See Javadocs for details on each handler.
4. Create the instance of [org.apache.tika.metadata.Metadata](https://tika.apache.org/1.12/api/org/apache/tika/metadata/Metadata.html)
5. Create the instance of [org.apache.tika.parser.ParseContext](https://tika.apache.org/1.12/api/org/apache/tika/parser/ParseContext.html)
6. Call the parse (…) method of Parser created at step 1 and pass the reference of InputStream, contentHandler, metadata and parseContext.

Let’s try it out.

|  |
| --- |
| **package** com.github.abhinavmishra14.tika;  **import** java.io.File;  **import** java.io.FileInputStream;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** org.apache.tika.exception.TikaException;  **import** org.apache.tika.metadata.Metadata;  **import** org.apache.tika.parser.AutoDetectParser;  **import** org.apache.tika.parser.ParseContext;  **import** org.apache.tika.parser.Parser;  **import** org.apache.tika.sax.BodyContentHandler;  **import** org.xml.sax.ContentHandler;  **import** org.xml.sax.SAXException;  **public** **class** XMLParserTest {  **public** **static** **void** main(String[] args) **throws** IOException,  SAXException, TikaException {  **try** (**final** InputStream inputStream = **new** FileInputStream(  **new** File("C:/Users/abhinav/Desktop/books.xml"));) {  //Create the instance of parser. Here I am using AutoDetectParser.  //You can create the instance of  //XMLParser,MP3Parser,PDFParser,OfficeParser,OOXMLParser based on your need  **final** Parser parser = **new** AutoDetectParser();  **final** ContentHandler contentHandler = **new** BodyContentHandler();  **final** Metadata metadata = **new** Metadata();  **final** ParseContext parseCtx = **new** ParseContext();  parser.parse(inputStream, contentHandler, metadata, parseCtx);  System.***out***.println("Extracted content: "+contentHandler.toString());  }  }  } |

**Output:**

|  |
| --- |
| Extracted content:    Everyday Italian  Giada De Laurentiis  2005  30.00  Harry Potter  J K. Rowling  2005  29.99  XQuery Kick Start  James McGovern  Per Bothner  Kurt Cagle  James Linn  Vaidyanathan Nagarajan  2003  49.99    Learning XML  Erik T. Ray  2003  39.95 |

Let’s try to extract the content from a text file.

|  |
| --- |
| **package** com.github.abhinavmishra14.tika;  **import** java.io.File;  **import** java.io.FileInputStream;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** org.apache.tika.exception.TikaException;  **import** org.apache.tika.metadata.Metadata;  **import** org.apache.tika.parser.AutoDetectParser;  **import** org.apache.tika.parser.ParseContext;  **import** org.apache.tika.parser.Parser;  **import** org.apache.tika.sax.ToTextContentHandler;  **import** org.xml.sax.ContentHandler;  **import** org.xml.sax.SAXException;  **public** **class** TextParserTest {  **public** **static** **void** main(String[] args) **throws** IOException,  SAXException, TikaException {  **try** (**final** InputStream inputStream = **new** FileInputStream(  **new** File("C:/Users/abhinav/Desktop/hello.txt"));) {  // Create the instance of parser.  **final** Parser parser = **new** AutoDetectParser();  **final** ContentHandler contentHandler = **new** ToTextContentHandler();  **final** Metadata metadata = **new** Metadata();  **final** ParseContext parseCtx = **new** ParseContext();  parser.parse(inputStream, contentHandler, metadata, parseCtx);  System.***out***.println("Extracted content: "+contentHandler.toString());  }  }  } |

**Output:**

|  |
| --- |
| Extracted content: Hello world is simple test program |

Let’s try to extract the content from a excel sheet:

|  |
| --- |
| **package** com.github.abhinavmishra14.tika;  **import** java.io.File;  **import** java.io.FileInputStream;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** org.apache.tika.exception.TikaException;  **import** org.apache.tika.metadata.Metadata;  **import** org.apache.tika.parser.ParseContext;  **import** org.apache.tika.parser.Parser;  **import** org.apache.tika.parser.microsoft.ooxml.OOXMLParser;  **import** org.apache.tika.sax.BodyContentHandler;  **import** org.xml.sax.ContentHandler;  **import** org.xml.sax.SAXException;  **public** **class** ExcelSheetExtractionTest {  **public** **static** **void** main(String[] args) **throws** IOException,  SAXException, TikaException {  **try** (**final** InputStream inputStream = **new** FileInputStream(  **new** File("C:/Users/abhinav/Desktop/Students.xlsx"));) {  //final Parser parser = new AutoDetectParser();  //Create the instance OOXMLParser. AutoDetectParser will also work.  **final** Parser parser = **new** OOXMLParser();  **final** ContentHandler contentHandler = **new** BodyContentHandler();  **final** Metadata metadata = **new** Metadata();  **final** ParseContext parseCtx = **new** ParseContext();  parser.parse(inputStream, contentHandler, metadata, parseCtx);  System.***out***.println("Extracted content: "+contentHandler.toString());  }  }  } |

**Output:**

|  |
| --- |
| Extracted content: Sheet1  Name Age RollNo Grade  Abhinav 27 101 12  Ashutosh 18 102 10  Abhishek 22 103 8 |

Similarly you can perform content extraction it for PDF, MS Word and PPT etc.

**Metadata extraction in TIKA:**

Metadata is nothing but the additional information supplied with a file. For e.g. in an audio file, the artist, album, title, year, composer etc. are metadata information. Whenever we parse a file using parse(…), we pass reference of an empty metadata object as a parameter. parse(…) method extracts the metadata of the given file (if there are any), and copies them into the metadata object. So, after parsing the file using parse(), we can extract the metadata from that object.

Let’s try with as example:

|  |
| --- |
| **package** com.github.abhinavmishra14.tika;  **import** java.io.File;  **import** java.io.FileInputStream;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** java.util.Arrays;  **import** java.util.List;  **import** org.apache.tika.exception.TikaException;  **import** org.apache.tika.metadata.Metadata;  **import** org.apache.tika.parser.ParseContext;  **import** org.apache.tika.parser.Parser;  **import** org.apache.tika.parser.microsoft.ooxml.OOXMLParser;  **import** org.apache.tika.sax.BodyContentHandler;  **import** org.xml.sax.ContentHandler;  **import** org.xml.sax.SAXException;  **public** **class** ExcelSheetExtractionTest {  **public** **static** **void** main(String[] args) **throws** IOException,  SAXException, TikaException {  **try** (**final** InputStream inputStream = **new** FileInputStream(  **new** File("C:/Users/abhinav/Desktop/Students.xlsx"));) {  //final Parser parser = new AutoDetectParser();  //Create the instance OOXMLParser. AutoDetectParser would also work.  **final** Parser parser = **new** OOXMLParser();  **final** ContentHandler contentHandler = **new** BodyContentHandler();  **final** Metadata metadata = **new** Metadata();  **final** ParseContext parseCtx = **new** ParseContext();  parser.parse(inputStream, contentHandler, metadata, parseCtx);  System.***out***.println("Extracted content: "+contentHandler.toString());  System.***out***.println("------------Extracted metadata-----------”);  //Extract the metadata information from metadata object  **final** List<String> metadataProps = Arrays.*asList*(metadata.names());  **for** (**final** String metadataProp : metadataProps) {  System.***out***.println(metadataProp + ": " + metadata.get(metadataProp));  }  }  }  } |

**Output**:

|  |
| --- |
| Extracted content: Sheet1  Name Age RollNo Grade  Abhinav 27 101 12  Ashutosh 18 102 10  Abhishek 22 103 8    ---------Extracted metadata---------------  meta:last-author: abhinav  meta:creation-date: 2016-04-08T11:08:29Z  dcterms:modified: 2016-04-08T11:10:12Z  meta:save-date: 2016-04-08T11:10:12Z  Last-Author: abhinav  Application-Name: Microsoft Excel  dc:creator: abhinav  dcterms:created: 2016-04-08T11:08:29Z  Author: abhinav  Last-Modified: 2016-04-08T11:10:12Z  Application-Version: 12.0000  date: 2016-04-08T11:10:12Z  modified: 2016-04-08T11:10:12Z  creator: abhinav  extended-properties:AppVersion: 12.0000  Creation-Date: 2016-04-08T11:08:29Z  protected: false  meta:author: abhinav  extended-properties:Application: Microsoft Excel  Content-Type: application/vnd.openxmlformats-officedocument.spreadsheetml.sheet  Last-Save-Date: 2016-04-08T11:10:12Z |

You can also **add/update** metadata to the documents. [Metadata](https://tika.apache.org/1.12/api/org/apache/tika/metadata/Metadata.html) class provides methods to **add/update** metadata. Let’s try it with an example:

|  |
| --- |
| **package** com.github.abhinavmishra14.tika;  **import** java.io.File;  **import** java.io.FileInputStream;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** java.util.Arrays;  **import** java.util.List;  **import** org.apache.tika.exception.TikaException;  **import** org.apache.tika.metadata.Metadata;  **import** org.apache.tika.parser.ParseContext;  **import** org.apache.tika.parser.Parser;  **import** org.apache.tika.parser.microsoft.ooxml.OOXMLParser;  **import** org.apache.tika.sax.BodyContentHandler;  **import** org.xml.sax.ContentHandler;  **import** org.xml.sax.SAXException;  **public** **class** ExcelSheetExtractionTest {  **public** **static** **void** main(String[] args) **throws** IOException,  SAXException, TikaException {  **try** (**final** InputStream inputStream = **new** FileInputStream(  **new** File("C:/Users/abhinav/Desktop/Students.xlsx"));) {  //final Parser parser = new AutoDetectParser();  //Create the instance OOXMLParser. AutoDetectParser would also work.  **final** Parser parser = **new** OOXMLParser();  **final** ContentHandler contentHandler = **new** BodyContentHandler();  **final** Metadata metadata = **new** Metadata();  **final** ParseContext parseCtx = **new** ParseContext();  parser.parse(inputStream, contentHandler, metadata, parseCtx);  System.***out***.println("Extracted content: "+contentHandler.toString());  System.***out***.println("---------Extracted metadata---------------");  //Extract the metadata information from metadata object  **final** List<String> metadataProps = Arrays.*asList*(metadata.names());  **for** (**final** String metadataProp : metadataProps) {  System.***out***.println(metadataProp + ": " + metadata.get(metadataProp));  }  System.***out***.println("------------------------------------------");  //Add metadata information  metadata.add("Usage", "Used for example");  //Update metadata information  metadata.set("Author", "Abhinav Mishra");  System.***out***.println("\n---------Updated metadata---------------");  **final** List<String> updatedMetadataProps = Arrays.*asList*(metadata.names());  **for** (**final** String metadataProp : updatedMetadataProps) {  System.***out***.println(metadataProp + ": " + metadata.get(metadataProp));  }  }  }  } |

**Output:**

|  |
| --- |
| Extracted content: Sheet1  Name Age RollNo Grade  Abhinav 27 101 12  Ashutosh 18 102 10  Abhishek 22 103 8    ---------Extracted metadata---------------  meta:last-author: abhinav  meta:creation-date: 2016-04-08T11:08:29Z  dcterms:modified: 2016-04-08T11:10:12Z  meta:save-date: 2016-04-08T11:10:12Z  Last-Author: abhinav  Application-Name: Microsoft Excel  dc:creator: abhinav  dcterms:created: 2016-04-08T11:08:29Z  Author: abhinav  Last-Modified: 2016-04-08T11:10:12Z  Application-Version: 12.0000  date: 2016-04-08T11:10:12Z  modified: 2016-04-08T11:10:12Z  creator: abhinav  extended-properties:AppVersion: 12.0000  Creation-Date: 2016-04-08T11:08:29Z  protected: false  meta:author: abhinav  extended-properties:Application: Microsoft Excel  Content-Type: application/vnd.openxmlformats-officedocument.spreadsheetml.sheet  Last-Save-Date: 2016-04-08T11:10:12Z  ------------------------------------------  ---------Updated metadata---------------  meta:last-author: abhinav  meta:creation-date: 2016-04-08T11:08:29Z  dcterms:modified: 2016-04-08T11:10:12Z  meta:save-date: 2016-04-08T11:10:12Z  Last-Author: abhinav  Application-Name: Microsoft Excel  dc:creator: abhinav  dcterms:created: 2016-04-08T11:08:29Z  Author: Abhinav Mishra  Last-Modified: 2016-04-08T11:10:12Z  Application-Version: 12.0000  date: 2016-04-08T11:10:12Z  Usage: Used for example  modified: 2016-04-08T11:10:12Z  creator: abhinav  extended-properties:AppVersion: 12.0000  Creation-Date: 2016-04-08T11:08:29Z  protected: false  meta:author: abhinav  extended-properties:Application: Microsoft Excel  Content-Type: application/vnd.openxmlformats-officedocument.spreadsheetml.sheet  Last-Save-Date: 2016-04-08T11:10:12Z |

**Language Detection using TIKA:**

Tika provide language detection tool as well, it is very useful if you want to differentiate the documents based on language. Tika adds the language information into metadata while parsing the document.

Tika can detect 18 languages from 184 standard languages standardized by ISO 639-1. Language detection in Tika is done using the getLanguage (…) method of the [LanguageIdentifier](https://tika.apache.org/1.5/api/org/apache/tika/language/LanguageIdentifier.html)class. This method returns the code name of the language in String format.  To get the language of the content you have to pass the content to the Constructor of [LanguageIdentifier](https://tika.apache.org/1.5/api/org/apache/tika/language/LanguageIdentifier.html) class.

e.g. [*LanguageIdentifier*](https://tika.apache.org/1.5/api/org/apache/tika/language/LanguageIdentifier.html) *langId = new* [*LanguageIdentifier*](https://tika.apache.org/1.5/api/org/apache/tika/language/LanguageIdentifier.html)*(“Hello Tika”);*

Let’s see the working of language detection tool using an example:

|  |
| --- |
| **package** com.github.abhinavmishra14.tika;  **import** java.io.File;  **import** java.io.FileInputStream;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** org.apache.tika.exception.TikaException;  **import** org.apache.tika.language.LanguageIdentifier;  **import** org.apache.tika.metadata.Metadata;  **import** org.apache.tika.parser.AutoDetectParser;  **import** org.apache.tika.parser.ParseContext;  **import** org.apache.tika.parser.Parser;  **import** org.apache.tika.sax.BodyContentHandler;  **import** org.xml.sax.ContentHandler;  **import** org.xml.sax.SAXException;  **public** **class** LanguageDetectionTest {  **public** **static** **void** main(String[] args) **throws** IOException,  SAXException, TikaException {  **try** (**final** InputStream inputStream = **new** FileInputStream(  **new** File("C:/Users/abhinav/Desktop/hello.txt"));) {  // You can create the instance of  // XMLParser,MP3Parser,PDFParser,OfficeParser based on your need  **final** Parser parser = **new** AutoDetectParser();  **final** ContentHandler contentHandler = **new** BodyContentHandler();  **final** Metadata metadata = **new** Metadata();  **final** ParseContext parseCtx = **new** ParseContext();  parser.parse(inputStream, contentHandler, metadata, parseCtx);  **final** String extractedContent = contentHandler.toString();  System.***out***.println("Extracted content: "+extractedContent);  System.***out***.println("Detecting the content language..");  **final** LanguageIdentifier langIdentifier = **new** LanguageIdentifier(extractedContent);  System.***out***.println("Language of the content is: "+langIdentifier.getLanguage());    }  }  } |

**Output:**

|  |
| --- |
| Extracted content: Hello world is simple test program  Detecting the content language..  Language of the content is: en |