**HTML language tags and attributes**

<https://www.w3schools.com/html/default.asp>

HTML is the standard markup language for creating Web pages.

* HTML stands for Hyper Text Markup Language
* HTML describes the structure of Web pages using markup
* HTML elements are the building blocks of HTML pages
* HTML elements are represented by tags
* HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
* Browsers do not display the HTML tags, but use them to render the content of the page

<!DOCTYPE html>  
<html>  
<head>  
<title>Page Title</title>  
</head>  
<body>  
  
<h1>My First Heading</h1>  
<p>My first paragraph.</p>  
  
</body>  
</html>

**Example Explained**

* The <!DOCTYPE html> declaration defines this document to be HTML5
* The <html> element is the root element of an HTML page
* The <head> element contains meta information about the document
* The <title> element specifies a title for the document
* The <body> element contains the visible page content
* The <h1> element defines a large heading
* The <p> element defines a paragraph

HTML tags are element names surrounded by angle brackets:

<tagname>content goes here...</tagname>

* HTML tags normally come in pairs like <p> and </p>
* The first tag in a pair is the start tag, the second tag is the end tag
* The end tag is written like the start tag, but with a forward slash inserted before the tag name

Tip: The start tag is also called the opening tag, and the end tag the closing tag.

Below is a visualization of an HTML page structure:

<html>

<head>

<title>Page title</title>

</head>

<body>

<h1>This is a heading</h1>

<p>This is a paragraph.</p>

<p>This is another paragraph.</p>

</body>

</html>

**Note:** Only the content inside the <body> section (the white area above) is displayed in a browser.

## The <!DOCTYPE> Declaration

The <!DOCTYPE> declaration represents the document type, and helps browsers to display web pages correctly.

It must only appear once, at the top of the page (before any HTML tags).

The <!DOCTYPE> declaration is not case sensitive.

The <!DOCTYPE> declaration for HTML5 is:

****

**Apache POI**

The Apache POI Project's mission is to create and maintain Java APIs for manipulating various file formats based upon the Office Open XML standards (OOXML) and Microsoft's OLE 2 Compound Document format (OLE2). **In short, you can read and write MS Excel files using Java.**

**Apache POI terminologies**

Apache POI excel library revolves around following four key interfaces -

**Workbook**: A workbook is the high-level representation of a Spreadsheet.

**Sheet**: A workbook may contain many sheets.

**Row**: As the name suggests, It represents a row in the spreadsheet.

**Cell**: A cell represents a column in the spreadsheet.

HSSF and XSSF implementations -

Apache POI library consists of two different implementations for all the above interfaces.

**HSSF (Horrible SpreadSheet Format)**: HSSF implementations of POI’s high-level interfaces like HSSFWorkbook, HSSFSheet, HSSFRow and HSSFCell are used to work with excel files of the older binary file format - .xls

XSSF (XML SpreadSheet Format): XSSF implementations are used to work with the newer XML based file format - .xlsx.



**import** java.io.FileInputStream;

**import** java.io.FileOutputStream;

**import** org.apache.poi.ss.usermodel.DateUtil;

**import** org.apache.poi.xssf.usermodel.XSSFCell;

**import** org.apache.poi.xssf.usermodel.XSSFRow;

**import** org.apache.poi.xssf.usermodel.XSSFSheet;

**import** org.apache.poi.xssf.usermodel.XSSFWorkbook;

**class** XLS\_POI {

**public** String excel\_path;

**public** FileInputStream fis = **null**;

**public** FileOutputStream fos = **null**;

**public** XSSFWorkbook workbook = **null**;

**public** XSSFSheet sheet = **null**;

**public** XSSFRow row = **null**;

**public** XSSFCell cell = **null**;

// Contrustor to create XLS\_POI object with the Excel file path

// path of excel file is passed as argument

**public** XLS\_POI(String excel\_path) {

**this**.excel\_path = excel\_path;

**try** {

**this**.fis = **new** FileInputStream(excel\_path);

**this**.workbook = **new** XSSFWorkbook(fis);

// this.sheet = workbook.getSheetAt(0);

// this.fis.close();

// this.workbook.close();

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

// Checks whether the given sheet exist in the xlxs file

// Name of the sheet is passed as argument

// returns 1 if "sheetName" is present

// returns 0 if "sheetName" is not present

**public** **int** isSheetPresent(String sheetName) {

**int** index = workbook.getSheetIndex(sheetName);

**if** (index == -1) {

**return** 0;

} **else** {

**return** 1;

}

}

// returns number of columns in sheet

// Name of the sheet is passed as argument

// returns -1 if "sheetName" is not present

// return -2 if "sheetName" is present however there are no rows

**public** **int** getColumnCount(String sheetName) {

// if sheet does not exist then return -1

**if** (isSheetPresent(sheetName) == 0)

**return** -1;

sheet = workbook.getSheet(sheetName);

row = sheet.getRow(0);

**if** (row == **null**) {

**return** -2;

} **else** {

**return** row.getLastCellNum();

}

}

// returns number of rows present in the sheet "sheetName"

// returns -1 if "sheetName" not present

**public** **int** getRowCount(String sheetName) {

// if sheet does not exist then return -1

**if** (isSheetPresent(sheetName) == 0)

**return** -1;

sheet = workbook.getSheet(sheetName);

**return** (sheet.getLastRowNum() + 1);

}

// returns the celldata in a sheet and at particular row and column name

// Name of the sheet, row number and column name is passed as arguments

// returns -1 if sheet not present

// returns -2 if the row number does not exist

// returns -3 if column number does not exist

// returns -4 if cell value is blank

// returns -5 if cell is not boolean, String, Numeric, formula or blank

**public** String getCellData(String sheetName, **int** rowNum, String colName) {

**int** colNum = -1;

**if** (isSheetPresent(sheetName) == 0) {

**return** "-1";

}

XSSFSheet worksheet = workbook.getSheet(sheetName);

row = worksheet.getRow(0);

**if** (row == **null**) {

**return** "-2";

}

**for** (**int** i = 0; i < row.getLastCellNum(); i++) {

**if** (row.getCell(i).getStringCellValue().trim().equalsIgnoreCase(colName)) {

colNum = i;

**break**;

}

}

**if** (colNum == -1) {

System.***out***.println(colName + " - column not present in the sheet");

**return** "-3";

}

**try** {

sheet = workbook.getSheet(sheetName);

row = sheet.getRow(rowNum - 1);

**if** (row == **null**) {

System.***out***.println("Rownumber " + rowNum + " not present");

**return** "-2";

}

cell = row.getCell(colNum);

**if** (cell == **null**) {

**return** "-3";

} **else** {

**switch** (cell.getCellTypeEnum()) {

**case** ***BOOLEAN***:

**return** String.*valueOf*(cell.getBooleanCellValue());

**case** ***STRING***:

**return** cell.getStringCellValue();

**case** ***NUMERIC***:

**if** (DateUtil.*isCellDateFormatted*(cell)) {

**return** String.*valueOf*(cell.getDateCellValue());

} **else** {

**return** String.*valueOf*(cell.getNumericCellValue());

}

**case** ***FORMULA***:

**return** String.*valueOf*(cell.getCellFormula());

**case** ***BLANK***:

System.***out***.println("Cell value blank");

**return** ("-4");

**default**:

System.***out***.println("None of the above");

**return** ("-5");

}

}

} **catch** (Exception e) {

e.printStackTrace();

}

**return** "-5";

}

// returns the celldata in a sheet and at particular row and column number

// Name of the sheet, row number and column number is passed as arguments

// returns -1 if sheet not present

// returns -2 if the row number does not exist

// returns -3 if column number does not exist

// returns -4 if cell value is blank

// returns -5 if cell is not boolean, String, Numeric, formula or blank

**public** String getCellData(String sheetName, **int** rowNum, **int** colNum) {

**try** {

**if** (isSheetPresent(sheetName) == 0) {

**return** "-1";

}

sheet = workbook.getSheet(sheetName);

row = sheet.getRow(rowNum - 1);

**if** (row == **null**) {

**return** "-2";

}

cell = row.getCell(colNum - 1);

**if** (cell == **null**) {

**return** "-3";

} **else** {

**switch** (cell.getCellTypeEnum()) {

**case** ***BOOLEAN***:

**return** String.*valueOf*(cell.getBooleanCellValue());

**case** ***STRING***:

**return** cell.getStringCellValue();

**case** ***NUMERIC***:

**if** (DateUtil.*isCellDateFormatted*(cell)) {

**return** String.*valueOf*(cell.getDateCellValue());

} **else** {

**return** String.*valueOf*(cell.getNumericCellValue());

}

**case** ***FORMULA***:

**return** String.*valueOf*(cell.getCellFormula());

**case** ***BLANK***:

System.***out***.println("Cell value blank");

**return** ("-4");

**default**:

System.***out***.println("None of the above");

**return** "-5";

}

}

} **catch** (Exception e) {

e.printStackTrace();

}

**return** "-5";

}

// set the celldata in a sheet and at particular row and column name

// Name of the sheet, row number and column name and cellvalue are passed as arguments

// returns -1 if sheet not present

// returns -2 if the row number does not exist

// returns true if celldata is updated

**public** String setCellData(String sheetName, **int** rowNum, String colName, String CellValue) {

**try** {

**int** colNum = -1;

**if** (isSheetPresent(sheetName) == 0) {

System.***out***.println(sheetName + " - not present in the workbook");

**return** "-1";

}

XSSFSheet worksheet = workbook.getSheet(sheetName);

row = worksheet.getRow(0);

**if** (row == **null**) {

**return** "-2";

}

**for** (**int** i = 0; i < row.getLastCellNum(); i++) {

**if** (row.getCell(i).getStringCellValue().trim().equalsIgnoreCase(colName)) {

colNum = i;

**break**;

}

}

row = worksheet.getRow(rowNum - 1);

**if** (row == **null**) {

row = worksheet.createRow(rowNum - 1);

}

cell = row.getCell(colNum);

**if** (cell == **null**) {

cell = row.createCell(colNum);

}

cell.setCellValue(CellValue);

// fis.close();

fos = **new** FileOutputStream(excel\_path);

workbook.write(fos);

fos.flush();

fos.close();

} **catch** (Exception e) {

// **TODO**: handle exception

}

**return** String.*valueOf*(**true**);

}

// set the celldata in a sheet and at particular row and column number

// Name of the sheet, row number and column name and cellvalue are passed as arguments

// returns -1 if sheet not present

// returns -2 if the row number does not exist

// returns true if celldata is updated

**public** String setCellData(String sheetName, **int** rowNum, **int** colNum, String CellValue) {

**try** {

**if** (isSheetPresent(sheetName) == 0) {

System.***out***.println(sheetName + " - not present in the workbook");

**return** "-1";

}

XSSFSheet worksheet = workbook.getSheet(sheetName);

row = worksheet.getRow(rowNum - 1);

**if** (row == **null**) {

row = worksheet.createRow(rowNum - 1);

}

cell = row.getCell(colNum - 1);

**if** (cell == **null**) {

cell = row.createCell(colNum - 1);

}

cell.setCellValue(CellValue);

// fis.close();

fos = **new** FileOutputStream(excel\_path);

workbook.write(fos);

fos.flush();

fos.close();

} **catch** (Exception e) {

e.printStackTrace();

}

**return** String.*valueOf*(**true**);

}

//adds a sheet having name in variable sheetName

// returns -1 if sheet already present

// returns true if sheet is added successfully

**public** String addSheet(String sheetName) {

**if** (isSheetPresent(sheetName) == 1) {

**return** "-1";

}

**try** {

fos = **new** FileOutputStream(excel\_path);

sheet = workbook.createSheet(sheetName);

workbook.write(fos);

fos.flush();

fos.close();

} **catch** (Exception e) {

e.printStackTrace();

}

**return** String.*valueOf*(**true**);

}

//removes sheet from workbook

// returns -1 if sheet not present

// returns true if sheet is removed successfully

**public** String removeSheet(String sheetName) {

**if** (isSheetPresent(sheetName) == 0) {

**return** "-1";

}

**try** {

fos = **new** FileOutputStream(excel\_path);

workbook.removeSheetAt(workbook.getSheetIndex(sheetName));

workbook.write(fos);

fos.flush();

fos.close();

} **catch** (Exception e) {

e.printStackTrace();

}

**return** String.*valueOf*(**true**);

}

// adds a column at the end of the table

// sheetName and Column Name are passed as arguments

// returns -1 if the sheet is not present in the workbook

// returns true if column is added successfully

**public** String addColumn(String sheetName, String colName) {

**if** (isSheetPresent(sheetName) == 0) {

**return** "-1";

}

**try** {

fos = **new** FileOutputStream(excel\_path);

XSSFSheet worksheet = workbook.getSheet(sheetName);

row = worksheet.getRow(0);

**if** (row == **null**) {

row = worksheet.createRow(0);

}

**if** (row.getLastCellNum() == -1) {

cell = row.createCell(0);

} **else** {

cell = row.createCell(row.getLastCellNum());

}

cell.setCellValue(colName);

workbook.write(fos);

fos.flush();

fos.close();

} **catch** (Exception e) {

e.printStackTrace();

}

**return** String.*valueOf*(**true**);

}

// do not use this function now as when we try to remove two columns (one after the another) the xlsx file gets corrupted

**public** String removeColumn(String sheetName, String colName) {

**if** (isSheetPresent(sheetName) == 0) {

**return** "-1";

}

**try** {

fos = **new** FileOutputStream(excel\_path);

**int** colNum = -1;

XSSFSheet worksheet = workbook.getSheet(sheetName);

row = worksheet.getRow(0);

**for** (**int** i = 0; i < row.getLastCellNum(); i++) {

**if** (row.getCell(i).getStringCellValue().trim().equalsIgnoreCase(colName)) {

colNum = i;

**break**;

}

}

**for** (**int** i = 0; i < getRowCount(sheetName); i++) {

row = worksheet.getRow(i);

**if** (row != **null**) {

cell = row.getCell(colNum);

**if** (cell != **null**) {

row.removeCell(cell);

}

}

}

workbook.write(fos);

fos.flush();

fos.close();

**return** colName;

} **catch** (Exception e) {

e.printStackTrace();

}

**return** colName;

}

// not implemented yet – HomeWork

**public** String addRow(String sheetName, **int** rowNum, String rowValue[]) {

**return** String.*valueOf*(**false**);

}

// not implemented yet – HomeWork

**public** String removeRow(String sheetName, **int** rowNum) {

**return** String.*valueOf*(**false**);

}

}

----------------------------------------

**import** org.apache.poi.ss.usermodel.Cell;

**import** org.apache.poi.ss.usermodel.CellStyle;

**import** org.apache.poi.ss.usermodel.DateUtil;

**import** org.apache.poi.xssf.usermodel.\*;

**import** java.io.\*;

**public** **class** XLS\_Operation {

**public** **static** **void** main(String arg[]) {

XLS\_POI datatable = **new** XLS\_POI("D:\\Test\_POI.xlsx");

// System.out.println("Number of Columns = " +

// datatable.getColumnCount("Suite1"));

/\*

\* for (int row = 0; row < datatable.getRowCount("Suite1"); row++) { for (int

\* col = 0; col < datatable.getColumnCount("Suite1"); col++) {

\* System.out.print(datatable.getCellData("Suite1", row, col) + " "); }

\* System.out.println(); }

\*/

// datatable.set\_cell\_value("Suite1", 1, 1, "Hello how are you");

System.***out***.println(datatable.addSheet("Jigar"));

// datatable.setCellData("Sheet1", 2,"DOB","hello");

// System.out.println(datatable.getCellData("Sheet1", 5, "name"));

// System.out.println(datatable.getCellData("Sheet1", 2, 3));

// System.out.println(datatable.getRowCount("Sheet12"));

// System.out.println(datatable.getCellData("TestCases1", 2, "Description"));

// System.out.println(datatable.addSheet("TestCases1"));

// System.out.println(datatable.removeSheet("TestCases1"));

// System.out.println(datatable.addColumn("Sheet1", "TC"));

//System.out.println(datatable.removeColumn("Sheet1", "TC3"));

// System.out.println(datatable.getCellData("TestCases", 2, "Description"));

// System.out.println(datatable.setCellData("TestCases1", 5, "TCID", "hello"));

System.***out***.println("done");

// datatable.setCellData("Suite1", 8, 6, "ffffffffffffffffffffff");

}

}