Simple API Cookbook

Version 0.1

Apply to Apache ODF Toolkit 0.5-incubating

1 Text 5
1.1 Text Document 5
1.1.1 Overview 5
1.1.2 Generate TextDocument 5
1.1.3 Get TextDocument 5
1.1.4 Paragraph 6
1.1.5 Section 7
1.1.6 List 8
1.1.7 Header and Footer 11
1.1.8 Text Box 12
1.1.9 Image 12
1.1.10 Span 13
1.1.11 Page Break 14
2 Presentation 14
2.1 Presentation Document 14
2.1.1 Create Presentation Document 14
2.1.2 Get Presentation Document 14
2.1.3 Change Presentation Mode 15
2.2 Slide 15
2.2.1 Add Slide 15
2.2.2 Get Slide 15
2.2.3 Set/Get Slide Name/Index 16
2.2.4 Copy Slide 16
2.2.5 Move/Delete Slide 17
2.2.6 Add Text to Slide 17
2.2.7 Add Image to Slide 18
3 Table 18
3.1 Table 18
3.1.1 Overview 18
3.1.2 Create Table 18
3.1.3 Find Table 19
3.1.4 Delete Table 20
3.1.5 Set Table 20
3.2 Column and Row 20

3.2.1 Get Column/Row 20
3.2.2 Append or Insert Column/Row 21
3.2.3 Remove Columns/Rows 22
3.2.4 Set Column/Row 22
3.3 Cell 22
3.3.1 Get Cell 22
3.3.2 Control Cell Attributes 23
3.3.3 Get&Set Cell Value Type 25
3.3.3.1 Get&Set boolean type Cell 26
3.3.3.2 Get&Set currency type Cell 26
3.3.3.3 Get&Set date type Cell 26
3.3.3.4 Get&Set float type Cell 26
3.3.3.5 Get&Set percentage type Cell 26
3.3.3.6 Get&Set string type Cell 27
3.3.3.7 Deal with the Time Value 27
3.3.3.8 Something about Display Text 27
3.3.4 Set image 27
3.4 Cell Range 28
3.4.1 Get CellRange 28
3.4.2 Merge Text Table 28
3.4.3 Merge Text Column 28
3.4.4 Merge Text Row 29
3.4.5 Merge SpreadSheet 29
4 Chart 29
4.1 Charts 29
4.1.1 Overview 29
4.1.2 Create charts 29
4.1.3 Update charts 31
4.1.4 Get and delete charts 31
5 Style 32
5.1 Style Handling 32
5.1.1 Overview 32
5.1.2 Font handling 32
5.1.3 Advanced font handling 32
5.1.4 Border handling 33
6 Navigation 34

6.1 Manipulate TextSearch 34
6.1.1 TextNavigation 34
6.1.2 TextSelection 34
6.1.2.1 Get Index/Text of TextSelection 34
6.1.2.2 Cut String 35
6.1.2.3 Paste String 35
6.1.2.4 Replace String 36
6.1.2.5 Add Reference to String 36
6.1.2.6 Add Comment 36
6.1.3 FieldSelection 37
6.1.4 TextStyleNavigation 38
7 TextExtractor 38
7.1 TextExtractor 38
7.1.1 Get Text 38
8 Field 39
8.1 Fields 39
8.1.1 Variable Field 39
8.1.2 Condition Field 39
8.1.3 Hidden Field 40
8.1.4 Cross Reference Field 40
8.1.5 Chapter Field 40
8.1.6 Title and Subject Field 40
8.1.7 Author Field 41
8.1.8 Page Number Field 41
8.1.8 Page Number Field 41
8.1.10 Date Field 42
8.1.11 Time Field 42
9 Metadata 42
9.1 Manipulate Metadata 42
9.1.1 Overview 42
9.1.2 Get Meta object 42
9.1.3 Access metadata 43
9.1.4 Access the user defined element 43
9.1.5 Access the document statistics 44

1 Text

1.1 Text Document

1.1.1 Overview

Till now, we support high level APIs to manipulate text document, paragraph, section, list, footer, header, textbox and span.

1.1.2 Generate TextDocument

There are typically four kinds of Text Documents with different modes can be generated: text document,master document, template document and web document.

The following codes generates an empty text document.

```
            TextDocument document1=TextDocument.newTextDocument();
          document1.save(filePath);
```

There are two methods to generate a new document, one with a parameter ofOdfMediaType, and the other one just use the corresponding new function.

The following two lines of codes generate the same kinds of document(Master document).The other three kinds of document are similar.

```
          TextDocument
documentMedia1=TextDocument.newTextDocument(OdfMediaType.TEXT_MASTER);
          TextDocument
documentMedia2=TextDocument.newTextMasterDocument();
```

1.1.3 Get TextDocument

We can get the existing Text Document by using the loadDocument function like follows.The loadDocument function has four kinds of parameters: File, InputStream, OdfPackageDocument andString. Here the one with string parameter is used as an example.

```
          TextDocument
document2=(TextDocument)TextDocument.loadDocument("textdocument.odt");
```

Also we can change the mode of one text document by using the following codes:

```
        document2.changeMode(OdfMediaType.TEXT_WEB);
```

1.1.4 Paragraph

To add a new paragraph to a text document, the following codes can be used,the first one add an empty paragraph to the end of the document, and the second oneadd a new paragraph with the corresponding texts to the document.

```
        document1.addParagraph(null);
        document1.addParagraph("test newParagraph function in
textDocument");
```

The first one will get the second paragraph of this document. The second statement will get the last paragraph with text content.The paragraph without text content will be ignored if the second parameter is true.

The following codes can be used to append text to the end of a paragrah.

```
        para1.appendTextContent("test addText function in
textDocument");
```

The following codes are to get and set the horizontal alignment of a paragraph.

```
        HorizontalAlignmentType align =
para1.getHorizontalAlignment();
        para1.setHorizontalAlignment(HorizontalAlignmentType.CENTER);
```

The following codes are to get and set the font of a paragraph.

```
          Font font = para1.getFont();
          font.setFontStyle(StyleTypeDefinitions.FontStyle.ITALIC);
          para2.setFont(font);
```

The following code is to apply a hyperlink to this paragraph, and append a text with a hyperlink to this paragraph.

```
          para2.applyHyperlink(new
URI("mailto:daisy@odftoolkit.org"));
          para2.appendHyperlink("mail to me",new
URI("mailto:daisy@odftoolkit.org"));
```

The following code is to add a comment at the front of the paragraph.

```
            para2.addComment("This is a comment for para2", "Simple ODF
Commenter");
```

From version 0.6.5, we support heading feature. You can test whether a paragraph is heading,apply a plain text paragraph as heading, and get heading level.

```
            // isHeading() and getHeadingLevel();
            if (para2.isHeading()) {
                  int headingLevel = para2.getHeadingLevel();
                  System.out.println("para2 is a heading, its level is "
+ headingLevel + ".");
            }
            // applyHeading(), default level is 1.
            para1.applyHeading();
            // applyHeading(), heading level is 3.
            para1.applyHeading(true, 3);
```

1.1.5 Section

From version 0.4, we support high level APIs for section.You can use the following code to get a section by name:

```
            Section aSection =
document1.getSectionByName("ImageSection");
```

or use the following code to get an iterator of sections.All sections in the document would be returned, including sections in footer and header.

```
            Iterator<Section> sections = document1.getSectionIterator();
            while (sections.hasNext()) {
                  Section theSection = sections.next();
            }
```

After you get the object of section,you can use the following codes to set and get the name of the section.

```
            String name = aSection.getName();
            aSection.setName("NewName");
```

You can use the following code to copy and append a section at the end of a document.Copying from a foreign document is supported.You don't need to care whether or not the document is the

owner of the source section.

```
document1.appendSection(aSection, true);
```

If you want to copy and append a section within a same document,you can specify with the second parameter of "copyAppendSection(Section section, boolean isResourceCopied)"whether the linked resources to this section are copied or not.The following code means to copy a section and append it at the end, but the linked resources are shared between the source section and the copied section.

```
document1.appendSection(aSection, false);
```

You can remove this Section from the document content,all the resources that are only linked with this section would be removed too.

```
aSection.remove();
```

1.1.6 List

From version 0.4, we support high level APIs for list.You can use the following code to create a list. The two methods are same.

```
List newList1 = document1.addList();
List newList2 = new List(document1);
```

Actually, you can append list to table cell, list item, presentation slide and notes. They all have the same ability with text document for list.We call them ListContainer, which can append, remove and get the iterator of list.

```
Iterator<List> lists = document1.getListIterator();
List list =null;
while (lists.hasNext()) {
     list = lists.next();
}
```

After you get the object of list,you can use the following methods to set and get the header of the list.

```
String header = list.getHeader();
list.setHeader("NewHeader");
```

You can get all of the existing items in this list. They are returned as java.util.List.

```
            java.util.List<ListItem> items = list.getItems();
```

If you only want to get the item at a specific location, you can use the following code:

```
            int location = 2;
            ListItem item = list.getItem(location);
```

There are several ways to help you append item(s) to the list. Please reference the following code:

```
            // add single item
            ListItem newItem1 = list.addItem(item);
            ListItem newItem2 = list.addItem("itemContent");
            // add clone items
            ListItem[] newItems = new ListItem[]{newItem1, newItem2};
            list.addItems(newItems);
            // add string items
            String[] newItemContents = new String[]{"itemContent1",
"itemContent2"};
            list.addItems(newItemContents);
```

You can specify a location to insert new item(s).

```
            newItem1 = list.addItem(location, item);
            newItem2 = list.addItem(location, "itemContent");
            list.addItems(location, newItems);
            list.addItems(location, newItemContents);
```

If you want to replace a list item with a new one, you can use the following code.

```
            item = list.set(location, newItem1);
            item = list.set(location, "itemContent");
```

You can use the following methods to remove item(s).

```
            list.removeItem(item);
            //item in specific location.
            list.removeItem(location);
            //item in specific collection.
            list.removeItems(items);
```

You can specify with the following methods whether the numbering of the previous list is continued by this list:

```
            // the numbering of the proximate list is continued
            newList1.setContinueNumbering(true);
            // the numbering of the specified list is continued by this
list
            newList2.setContinueList(list);
```

You can use the following method to know whether the list is a number list.

```
            ListType type = list.getType();
```

Now, List API support 3 types of lists, ListType.BULLET, ListType.NUMBER and ListType.IMAGE. The default createdlist is ListType.BULLET, bullet list.How to create a number list or a image list then?ListDecorator has powerful functions that can help you. ListDecorator is an interfacewhich decides how to decorate a list and its list items. We supply 4 implementations of this interface,BulletDecorator, NumberDecorator, ImageDecorator and OutlineDecorator.

```
            // create a number list.
            ListDecorator numberDecorator = new
NumberDecorator(document1);
            List numberList1 = document1.addList(numberDecorator);
            List numberList2 = new List(document1, numberDecorator);
```

You can implement your own ListDecorator as need, or extend the default four implementations.

You can remove a list from the document, the following two methods are same.

```
            list.remove();
            document1.removeList(list);
```

We also provide funtions to manipulate list items.You can use the following methods to get and set the text content of a item.

```
            String itemContent = item.getTextContent();
            item.setTextContent("new item content");
```

You can get the item index and its owner list.

```
            int index = item.getIndex();
            List ownerList = item.getOwnerList();
```

When you want to remove an item, besides List.removeItem(ListItem), you can use the following method directly.

```
            item.remove();
```

With the help of the following code, the start number of list item can be set.

```
            item.setStartNumber(3);
```

Since list item is also a ListContainer, you can add sub list to an item.

```
            item.addList();
```

1.1.7 Header and Footer

From version 0.4.5, we support high level APIs for footer and header.You can use the following code to get the header and footer.

```
            Header docHeader = doc.getHeader();
            Footer footer = doc.getFooter();
```

In order to add contents to header and arrange these contents in a good layout,I suggest you to use table.It's easy to add a none-border table to header.Below codes show how to add a table with 2 columns and 1 row to header, and thenadd some text content to the left cell, and add an image to the right cell.

```
            Table table = docHeader.addTable(1,2);
            table.getCellByPosition(0, 0).setStringValue("header table
cell");
            Cell cell = table.getCellByPosition(1, 0);
            Image image1 = cell.setImage(new URI("file:/c:/image.jpg"));
```

Below codes show how to add a table with 1 column and 1 row to footer, and thenadd some text content to the cell.You can easily set the style of these text content. The codes describe how to putthe text content in the center, set the font of the text content,and set the background of the cell.

```
            table = footer.addTable(1, 1);
            Cell cellByPosition = table.getCellByPosition(0, 0);
            cellByPosition.setStringValue("footer table cell");

cellByPosition.setHorizontalAlignment(HorizontalAlignmentType.CENTER);
            Font myFont = new Font("Arial",
StyleTypeDefinitions.FontStyle.ITALIC, 12, Color.BLUE);
            cellByPosition.setFont(myFont);
            cellByPosition.setCellBackgroundColor(Color.YELLOW);
```

1.1.8 Text Box

From version 0.5, we support high level APIs for text box.Below codes will create a text box and add it to the textdocument with a paragraph as the anchor position.The FrameRectangle specifies an area and the position of this text box.

```
            Paragraph paragraph = doc.getParagraphByIndex(0, false);
            Textbox box = paragraph.addTextbox(new
FrameRectangle(1,1,2,1,SupportedLinearMeasure.IN));
```

Below codes is to set the text content and name of this text box;

```
            box.setTextContent("this is a text box");
            box.setName("MyTextbox");
```

Below codes is to get a text box by name or get an iterator of text box contained by a paragraph.

```
            Textbox myBox=paragraph.getTextboxByName("MyTextbox");
            Iterator<Textbox> boxIter = paragraph.getTextboxIterator();
```

Below code is to set the background color of a text box.

```
            myBox.setBackgroundColor(Color.BLUE);
```

Below code is to remove a text box;

```
            paragraph.removeTextbox(myBox);
```

1.1.9 Image

From version 0.5.5, we support high level APIs for images.Below codes will create a image and add it to a document with a paragraph as the anchor position.

```
            Paragraph para = doc.getParagraphByIndex(1, false);
            Image image = Image.newImage(para, new
URI("file:/c:/image.jpg"));
```

Below codes can set the properties of image, and even the vertical (or horizontal) alignment.

```
            image.setTitle("Image title");
            image.setDescription("This is a sample image");
            image.setVerticalPosition(FrameVerticalPosition.TOP);
            image.setHorizontalPosition(FrameHorizontalPosition.RIGHT);
```

If you want to add a hyperlink to your image, you can use the following code:

```
            image.setHyperlink(new URI("http://odftoolkit.org"));
```

If you want to handle more style settings of image, you can try FrameStyleHandler.

```
            FrameStyleHandler handler = image.getStyleHandler();
            handler.setAchorType(AnchorType.AS_CHARACTER);
            handler.setHorizontalRelative(HorizontalRelative.PAGE);
            handler.setVerticalRelative(VerticalRelative.PAGE);
```

1.1.10 Span

From version 0.5.5, we support high level APIs for span.You can create a span based on a text selection.With span, you can set a different style to this small unit.

```
            TextNavigation navigation = new TextNavigation("test", doc);
            TextSelection sel = (TextSelection)
navigation.nextSelection();
            Span span = Span.newSpan(sel);
            DefaultStyleHandler styleHandler = span.getStyleHandler();
            Font font1Base = new Font("Arial", FontStyle.ITALIC, 10,
Color.BLACK, TextLinePosition.THROUGH);
            styleHandler.getTextPropertiesForWrite().setFont(font1Base);
```

The following code is to apply a hyperlink to this span.

```
            span.applyHyperlink(new URI("ftp://ftpserver"));
```

1.1.11 Page Break

From version 0.6.5, we support page break feature.You can add page break at the end of text document or after a reference paragraph.It may help you well format your document.

```
            TextDocument newDoc = TextDocument.newTextDocument();
            //add a page break at the end of document.
            newDoc.addPageBreak();
            //add a page break at the end of reference paragraph.
            Paragraph refParagraph = newDoc.addParagraph("before page
break");
            newDoc.addPageBreak(refParagraph);
```

2 Presentation

2.1 Presentation Document

2.1.1 Create Presentation Document

Let's create an empty presentation document first. The following codes generates an empty presentation document with one slide:

```
                PresentationDocument
document=PresentationDocument.newPresentationDocument();
                document.save(filePath);
```

We can also create a presentation template document by using the following codes. The operation of the template document is the same with the normal presentation document.

```
                    PresentationDocument
documentTmp=PresentationDocument.newPresentationTemplateDocument();
```

2.1.2 Get Presentation Document

We can get the existing presentation Document by using the loadDocument function like follows:

Here presentation.odp is the name of the existing presentation document.

Also we can append all the slides of the specified presentation document to the current document by

using the following codes:

Here the slides of the documents "presentation.odp" will be appended to "document".

```
                PresentationDocument presentationmodel;

presentationmodel=(PresentationDocument)PresentationDocument.loadDocume
nt("presentation.odp");
                   document.appendPresentation(presentationmodel);
```

2.1.3 Change Presentation Mode

We can switch the mode presentation documents by using the following codes. Here the first one convert the presentation document to a template, and the second one convert the template to a normal presentation document.

```
                document.changeMode(OdfMediaType.PRESENTATION_TEMPLATE);

                documentTmp.changeMode(OdfMediaType.PRESENTATION);
```

2.2 Slide

2.2.1 Add Slide

If you want to add new slide to the presentation document, you can use the following codes:

Here the SlideLayout is the layout model of the added slide, the first parameter is the index of the added slide and the second parameter is the name of this slide.

```
                PresentationDocument
document=PresentationDocument.newPresentationDocument();

           document.newSlide(1, "new slide", SlideLayout.TITLE_ONLY);
```

2.2.2 Get Slide

You can get the slide through the index in the presentation document, like follows:

```
Slide slide;
slide=document.getSlideByIndex(0);
```

Or you can get the slide through the name of it by using the following code:

```
slide=document.getSlideByName("new slide");
```

To get all the slides in the document,you can do like this:

```
Iterator<Slide> slideList=document.getSlides();
```

Also you can get the number of the slides in the document by the following code:

```
int numSlide=document.getSlideCount();
```

2.2.3 Set/Get Slide Name/Index

If you want to set a new name for a slide, you can use the following two methods:

```
slide.setSlideName("second slide");
document.getSlideByIndex(2).setSlideName("third slide");
```

If you want to know the index and the name of one slide which is being operated, you can use the following codes:

```
int slideIndex=slide.getSlideIndex();
String slideName=slide.getSlideName();
```

2.2.4 Copy Slide

You can copy a slide in the presentation document from one position to another by using the following codes:

Here the first parameter is the source position of the slide need to be copied, the second parameter is the destination position of the slide need to be copied, and the last parameter is the new name of the copied slide.

```
document.copySlide(1, 2, "copied slide");
```

And also you can copy a slide from another document by using the following codes:

Here the first parameter of copyForeignSlide is the new position of the copied slide in the current document, the second parameter is the source document of the copied slide, and the last one is the slide index of the source document that need to be copied.

```
            PresentationDocument documentmodel;

documentmodel=(PresentationDocument)PresentationDocument.loadDocument("
presentation.odp");
            document.copyForeignSlide(1, documentmodel, 2);
```

2.2.5 Move/Delete Slide

To move one slide to another position of this presentation position, you can use the following codes:

Here the first parameter is the current index of the slide that need to be moved, and the second parameter is the index of the destination position before the move action.

```
            document.moveSlide(2, 1);
```

You can delete the slide either by through the index or through the name of the specified slide, like follows:

```
            document.deleteSlideByIndex(1);
            document.deleteSlideByName("third slide");
```

2.2.6 Add Text to Slide

You can set the text content of a slide with text box API since version 0.5.Below codes will get the title text box of a slide, set the text content,and then get the outline text box, set the list content.

```
          Textbox titleBox =
slide.getTextboxByUsage(PresentationClass.TITLE).get(0);
          titleBox.setTextContent("This is the title");
          Textbox outline =
slide.getTextboxByUsage(PresentationClass.OUTLINE).get(0);
          List txtList = outline.addList();
          txtList.addItem("List Item1");
          txtList.addItem("List Item2");
```

To add some text, you can first get the notes of one slide and then add text to this corresponding notes. The following codes shows this process:

```
            Notes note=slide.getNotesPage();
            note.addText("text notes");
```

2.2.7 Add Image to Slide

To add an image to slide, you can use below codes to simply add the image to the last slide of the presentation document.

```
            URI imageuri=new URI("namdaemun.jpg");
            document.newImage(imageuri);
```

Or you can use following code to add an image to a specific position you want

```
            Slide slide1 = document.getSlideByIndex(1);
            Image image = Image.newImage(slide1, new
URI("http://www.xxx.com/a.jpg"));
            FrameRectangle rect = image.getRectangle();
            rect.setX(4);
            rect.setY(5.7);
            image.setRectangle(rect);
```

3 Table

3.1 Table

3.1.1 Overview

This *Table* API supports to manipulate tables in text and spreadsheet documents. It covers the table definition in *ODF Specification 1.2 Committee Draft05*

3.1.2 Create Table

Let's create an empty table first.By default,the code below create a table with 5 columns and 2 rows.

```
            TextDocument document = TextDocument.newTextDocument();
             Table table1 = Table.newTable(document);
             table1.setTableName("table1");
             document.save(filePath);
```

If you want to create table with specified column and row,you can do like this:

```
            int row=4;
             int column=3;
             Table table2=Table.newTable(document, row, column);
             table2.setTableName("table2");
```

If you want to put some numbers into a table while creating it, you can use the constructor Table.newTable(document,rowlabels,columnlabels, data)which you should specify a 2 dimension array as the data and 2 String arrays as table labels,one for row and the other for column.

```
       int rowcount = 10, columncount = 4;
     double[][] data = new double[rowcount][columncount];
     String[] rowlabels = new String[rowcount];
     String[] columnlabels = new String[columncount];
     Table table3=Table.newTable(document,rowlabels,columnlabels,
data);
     table3.setTableName("dataTable");
```

You can also fill table with string values while creating it, to do this you should provide a 2 dimension string array instead of double array.

```
     String[][] stringData = new String[rowcount][columncount];
     Table table4 = Table.newTable(document, rowlabels, columnlabels,
stringData);
     table4.setTableName("stringTable");
```

3.1.3 Find Table

To get all the tables in the document,you can do like this:

```
       List<Table> tableList=document.getTableList();
```

If you want to get a single table,you can use the table name to find it.If it's not found,the method returns null.

```
Table emptyTable=document.getTableByName("table1");
```

3.1.4 Delete Table

```
Table table = document.getTableByName("DeletedTable");
 if (table != null) {
table.remove();
 }
```

3.1.5 Set Table

You can set or update table name,which can be regarded as table identifier in a document.

```
table1.getTableName();
 table1.setTableName("EnglishScore");
```

If you want to change table width,you can do like this:

```
Table tableWidth=document.getTableByName("table1");
 if(tableWidth!=null) {
     long width=500;
     tableWidth.setWidth(width);
     tableWidth.getWidth();
 }
```

Each table in the document has a protect attribute to show whether it is protected or not.

```
boolean isProtected=table1.isProtected();
 table1.setProtected(true);
```

3.2 Column and Row

3.2.1 Get Column/Row

The class Column/Row represents the column/row of table.To get all the columns or rows you can use the getColumnList or getRowList of the table instance.

```
List<Column> columns = table.getColumnList();
List<Row> rows = table.getRowList();
```

You can also get single column/row by specifying the index of the column/row.

The column/row index start from 0.If not found, null will be returned.

```
Column column = table.getColumnByIndex(2);
Row row = table.getRowByIndex(0);
```

If you want to know the count of header column/row in the table,you can do like this:

```
int headerColumnCount = table.getHeaderColumnCount();
int headerRowCount = table.getHeaderRowCount();
```

If you want to know the index of the column/row,you can use the method below:

```
int columnIndex=column.getColumnIndex();
int rowIndex=row.getRowIndex();
```

Can I get the previous or next Column/Row by the current column/row instance?

Yes,you can ask the column/row instance itself,if it doesn't exist,null will be returned.

```
Column previousCol=column.getPreviousColumn();
Column nextCol=column.getNextColumn();
Row previousRow=row.getPreviousRow();
Row nextRow=row.getNextRow();
```

3.2.2 Append or Insert Column/Row

You can add a column to the end or insert many columns before the specified index

The appendColumn/Row method add an empty column/row at the end and return the new appended column/row

```
Column newColumn=table.appendColumn();
Row newRow=table.appendRow();
```

What can I do if I want to insert a column/row into the specified position?

You can use the insertColumn/RowBefore method,whose first parameter is the index of the column/row to be inserted before; The second parameter is the number of columns/rows to be inserted.

```
List<Column> cols = table.insertColumnsBefore(1, 2);
List<Row> newRows = table.insertRowsBefore(0, 2);
```

3.2.3 Remove Columns/Rows

You can delete a number of columns/rows by index

The first parameter is the index of the first column/row to delete; The second parameter is the number of columns/rows to delete.

The code below remove 1 column whose index is 2;remove 2 rows whose index is 1,2.

```
table.removeColumnsByIndex(2, 1);
table.removeRowsByIndex(1, 2);
```

3.2.4 Set Column/Row

If you want to change the width of the column or the height of the row,you can use it like this:

If the second parameter of row's setHeight is true, the row can fit the height to the text, vice versa.

```
column.setWidth(column.getWidth()/2);
row.setHeight(row.getHeight()/2, true);
```

3.3 Cell

3.3.1 Get Cell

If you want to get the specified cell in a table,you can use the getCellByPosition method of Table.

The first parameter is the column index,the second parameter is the row index.

```
        TextDocument document = (TextDocument)
TextDocument.loadDocument(filePath);
      Table table=document.getTableByName("stringTable");
      Cell cell=table.getCellByPosition(1, 1);
```

If you are manipulating a spreadsheet,you can get the cell by its address:

```
      Table sheet1 = document.getTableByName("Sheet1");
      Cell odsCell=sheet1.getCellByPosition("A1");
```

If you want to get a cell from a row,you can specify the index of the cell in the row.

```
      Row row=table.getRowByIndex(1);
      Cell cell2=row.getCellByIndex(1);
      System.out.println(cell2.getStringValue());
```

What can I do if I have a Cell instance and want to know which column and row it belongs to ?

The code below shows how you can do that:

```
      Row row1=cell.getTableRow();
      Column column1=cell.getTableColumn();
```

3.3.2 Control Cell Attributes

Use the getStyleName() method you can get the style name of the cell.

If you want to change the style,it must be set when you set the display text.

```
      String cellSytle=cell.getStyleName();
      cell.setDisplayText("content", cellSytle);
```

What can I do if I want to control the display alignment of the cell?

You can set the horizontal and vertical alignment to do so.

The code below shows how to get and set the alignment.You can refer to the javadoc about the alignment type.

```
        StyleTypeDefinitions.HorizontalAlignmentType
horizontalAlign=cell.getHorizontalAlignmentType();
        StyleTypeDefinitions.VerticalAlignmentType
verticalAlign=cell.getVerticalAlignmentType();

cell.setHorizontalAlignment(StyleTypeDefinitions.HorizontalAlignmentTyp
e.CENTER);

cell.setVerticalAlignment(StyleTypeDefinitions.VerticalAlignmentType.BO
TTOM);
```

If the content of the cell is too long,you can set the wrap option of the cell.

```
        cell.setTextWrapped(true);
```

If don't know the cell is wrapped or not,you can use the method:

```
        boolean isTextWrapped=cell.isTextWrapped();
```

If you want to set the background color of the cell,be care that the color type is org.odftoolkit.odfdom.type.Color.

```
         Color cellBackgroundColor=cell.getCellBackgroundColor();
         cell.setCellBackgroundColor(Color.valueOf("#000000"));
```

How can I control the spanned number of the column/row:

```
         int spannedNum=cell.getcgetColumnSpannedNumber();
         cell.setColumnSpannedNumber(spannedNum);
         int rowSpannedNum=cell.getRowSpannedNumber();
         cell.setRowSpannedNumber(rowSpannedNum);
```

For column,maybe you want to know the column repeated number:

```
         int repeatedNum=cell.getColumnsRepeatedNumber();
         cell.setColumnsRepeatedNumber(repeatedNum);
```

How about formatting a cell's content? You can set the format string of the cell to do so.

For example you want to format the date to yyyy-MM-dd ,you can:

```
            String cellFormatStr=cell.getFormatString();
            cell.setDateValue(new GregorianCalendar(2010,5,1));
            cell.setFormatString("yyyy-MM-dd");
```

Be care that the setFormatString only works for float, date and percentage.

You may be confused by the difference between getFormatString and getFormula,the difference is that:

For the setFormula method,it just sets as a formula attribute,the cell value will not be calculated.

```
            String formula=cell.getFormula();
            cell.setFormula(formula);
```

How can I clear the content of the cell?

RemoveContent remove all of the cell while the removeTextContent only remove the text content of the cell.

```
        cell.removeContent();
        cell.removeTextContent();
```

3.3.3 Get&Set Cell Value Type

The cell value can have different types,for the setValueType method: the parameter can be
"boolean"
"currency"
"date"
"float"
"percentage"
"string"
"time"
"void"

If the parameter type is not a valid cell type, an IllegalArgumentException will be thrown.

```
        String valueType=cell.getValueType();
        cell.setValueType(valueType);
```

For the following getXXXValue() method:it gets the cell value as xxx type.An IllegalArgumentException will be thrown if the cell type is not xxx.

3.3.3.1 Get&Set boolean type Cell

For setBooleanValue method:it sets the cell value as a boolean and sets the value type to be boolean.

```
boolean booleanValue=cell.getBooleanValue();
cell.setBooleanValue(booleanValue);
```

3.3.3.2 Get&Set currency type Cell

For the following getting methods,if the value type is not "currency", an IllegalArgumentException will be thrown.

The currency code of the cell is like "USD", "EUR", "CNY", and the currency symbol is like "$"

```
String currencyCode=cell.getCurrencyCode();
cell.setCurrencyCode("USD");
```

You can also set currency value and currency format.Please note the overall format includes the symbol character, for example: $#,##0.00.

```
cell.setCurrencyValue(100.00, "USD");
cell.setCurrencyFormat("$", "$#,##0.00");
```

3.3.3.3 Get&Set date type Cell

```
Calendar dateValue=cell.getDateValue();
cell.setDateValue(new GregorianCalendar(2010,5,1));
```

3.3.3.4 Get&Set float type Cell

```
double floatValue=cell.getDoubleValue();
cell.setDoubleValue(new Double(22.99f));
```

3.3.3.5 Get&Set percentage type Cell

```
double percentageValue=cell.getPercentageValue();
cell.setPercentageValue(0.89);
```

3.3.3.6 Get&Set string type Cell

If the cell type is not string, the display text will be returned.

```
        String stringValue=cell.getStringValue();
        cell.setStringValue("simple");
```

3.3.3.7 Deal with the Time Value

If you want to get the string type of time value,you can format it:

```
        cell.setTimeValue(Calendar.getInstance());
        SimpleDateFormat simpleFormat = new
SimpleDateFormat("'PT'HH'H'mm'M'ss'S'");
          String  timeString=
simpleFormat.format(cell.getTimeValue().getTime());
```

3.3.3.8 Something about Display Text

Please note the display text in ODF viewer might be different from the value set by this method,because the displayed text in ODF viewer is calculated and set by editor.

```
        String displayText=cell.getDisplayText();
        cell.setDisplayText(displayText);
```

3.3.4 Set image

From version 0.5.5, we support high level APIs for images.You can use following codes to set an image to a cell.

```
          Image myImage = cell.setImage(new
URI("http://www.xxx.com/a.jpg"));
```

You can use following codes to access an image in a cell.

```
        Image image = cell.getImage();
        String imagename = image.getName();
        FrameRectangle rect = image.getRectangle();
        rect.setX(1);
        rect.setY(1);
        image.setRectangle(rect);
```

3.4 Cell Range

3.4.1 Get CellRange

You can get cell range by providing start and end index of the column and row,or just provide start and end address of the cell(if you are using the spreadsheet.)

```
        CellRange cellRange = table.getCellRangeByPosition(1, 0, 2,
0);
        CellRange cellRangeAdd =
table.getCellRangeByPosition("$E1","$E6");
```

3.4.2 Merge Text Table

The code below merges all of the selected cells into one:

```
        Table table1 = document.getTableByName("Table1");
        CellRange cellrange = table1.getCellRangeByPosition(0, 0,
table1.getColumnCount()-1, table1.getRowCount()-1);
        cellRange.merge();
```

3.4.3 Merge Text Column

The code below shows how to merge the cells of the first column into one ：

```
         table1 = document.getTableByName("Table1");
         CellRange firstColumn = table1.getCellRangeByPosition(0, 0,
0, table1.getRowCount()-1);
         firstColumn.merge();
```

3.4.4 Merge Text Row

The code below shows how to merge the cells of the first 2 rows into one ：

```
            table1 = document.getTableByName("Table1");
            int rowCount = table1.getRowCount();
               CellRange firstTwoRow = table1.getCellRangeByPosition(0,
0, table1.getColumnCount()-1, 1);
               firstTwoRow.merge();
```

3.4.5 Merge SpreadSheet

Merge a spreadsheet's cell is the same as text document.Especially,when getting the cell range of spreadsheet,you can use special address instead of index.

```
            Table sheet1 = document.getTableByName("Sheet1");
            CellRange cellRange2 =
sheet1.getCellRangeByPosition("$E1","$E6");
            cellRange2.setCellRangeName("TimeCellRange");
            cellRange2.merge();
```

4 Chart

4.1 Charts

4.1.1 Overview

Since 0.6, Simple ODF provides methods to manipulate charts in text document, spreadsheet document and presentation document.You can create, update and delete charts with these methods.

4.1.2 Create charts

We all know, a chart is associated with a table. In order to create a chart, you must determine the data set of this chart.The data set can be a cell range of a table, for example:

```
            CellRangeAddressList cellRange =
CellRangeAddressList.valueOf("A.A1:A.B3");
            DataSet dataSet = new DataSet(cellRange, spreadsheetDoc,
true, true, false);
```

Or a two dimensional array, for example:

```
int row = 2, column = 3;
double[][] data = new double[column][row];
String[] labels = new String[row];
String[] legends = new String[column];
DataSet dataset = new DataSet(labels, legends, data);
```

You should also use rectangle to define the position and the size of this chart. For example:

```
Rectangle rect = new Rectangle();
rect.x = 2000;
rect.y = 2700;
rect.width = 15000;
rect.height = 8000;
rect.y = 110000;
```

Then you can create a chart:

```
spreadsheetDoc.createChart("Page Visit", dataSet,rect);
```

There are some shortcut methods to create charts, for example, below codes show how to create a chart in a text document:

```
Chart chart = textDoc.createChart(
            "Page Visit", spreadsheetDoc,
            cellRange, true, true, false, rect);
```

If you want to create a chart in a spreadsheet document, you need to specify a cell to be the anchor of this chart, for example:

```
spreadsheetDoc.createChart("Page Visit", spreadsheetDoc,
cellRange,
                    true, true, false, rect,
spreadsheetDoc.getTableByName("C")
                                .getCellByPosition("D10"));
```

If you want to create a chart in a presentation document, you can use the existing layout of a slide, which means, you don't need to specifya rectangle. The layouts that could contain a chart include: TITLE_PLUS_CHART, TITLE_PLUS_2_CHART, TITLE_LEFT_CHART_RIGHT_OUTLINE,TITLE_PLUS_3_OBJECT, and TITLE_PLUS_4_OBJECT. For example:

```
            Slide slide = presentationDoc.newSlide(2, "Slide3",
                        SlideLayout.TITLE_PLUS_2_CHART);
            chart = slide.createChart("Count of Visits", spreadsheetDoc,
                        cellRange, true, true, false, null);
```

4.1.3 Update charts

You can update charts properties, for example, the title, axis title, chart type, whether to apply 3D effect, whether to use legend with API.For example:

```
            chart.setChartTitle("New title");
            chart.setAxisTitle("Hour", "Number");
            chart.setChartType(ChartType.AREA);
            chart.setApply3DEffect(true);
            chart.setUseLegend(true);
```

You can update the data set too. For example:

```
            chart.setChartData(new
DataSet(CellRangeAddressList.valueOf("A.A1:A.C4"), spreadsheetDoc,
true, true, true));
```

4.1.4 Get and delete charts

You can get charts by title e.g.

```
            chart = textDoc.getChartByTitle("New title").get(0);
```

You can also get a chart by its unique ID. The unique ID of a chart in Simple ODF API is the path of the chart document (relative to the ODF document package).The unique ID can be gotten with method:

```
            String chartid = chart.getChartID();
            chart = textDoc.getChartById(chartid);
```

You can also get the count of charts in this document.

```
            int count = textDoc.getChartCount();
```

You can delete a chart by ID or by title, e.g.

```
            textDoc.deleteChartById(chartid);
            textDoc.deleteChartByTitle("New title");
```

5 Style

5.1 Style Handling

5.1.1 Overview

Style handling methods provide convenient methods to set font and borders.

5.1.2 Font handling

The most simple method to define font settings is to create a fontobject, and set it to a cell object.The below code snippet defines a font object to describe "Arial"italic font with size "12pt" and black color,and then set it to a cell. The font will work for western charactersby default.

```
            SpreadsheetDocument document =
SpreadsheetDocument.newSpreadsheetDocument();
            Table table = document.getTableByName("Sheet1");
            Font font = new Font("Arial",
StyleTypeDefinitions.FontStyle.ITALIC, 12, Color.BLACK);
            Cell cell = table.getCellByPosition("A1");
            cell.setFont(font);
```

The most simple method to get font settings of western characters is:

```
            Font theFont = cell.getFont();
            double size = theFont.getSize();
            String fontName = theFont.getFamilyName();
            StyleTypeDefinitions.FontStyle fontStyle =
theFont.getFontStyle();
            Color fontColor = theFont.getColor();
```

5.1.3 Advanced font handling

CellStyleHandler can help you to achieve advanced functions.In Open Document Format, there can be different font settings fordifferent script types.For example, a font setting for English characters and another fontsetting for Chinese characters.If you want to define the font setting for other script types, youcan reference to below codes.The below code snippet defines a font for

Chinese characters.

```
            cell.getStyleHandler().setFont(font, new
Locale(Locale.CHINESE.getLanguage(), Locale.CHINA.getCountry()));
```

The below code snippet shows how to get the font setting for other kinds of scripts.

```
            CellStyleHandler styleHandler = cell.getStyleHandler();
            Font westernFont =
styleHandler.getFont(Document.ScriptType.WESTERN);
            Font chineseFont =
styleHandler.getFont(Document.ScriptType.CJK);
            Font complexFont =
styleHandler.getFont(Document.ScriptType.CTL);
```

5.1.4 Border handling

The most simple way to set border is to create a border object andthen set it to a cell object.Below code snippet illustrates how to set a cell object with fourborders.

```
            cell = table.getCellByPosition("A1");
            cell.setStringValue("four border");
            Border border = new Border(Color.RED, 1,
StyleTypeDefinitions.SupportedLinearMeasure.PT);
            cell.setBorders(CellBordersType.ALL_FOUR, border);
```

Below code snippet illustrates how to set a cell object with left andright borders, top and bottom borders and diagonal lines.

```
            cell.setBorders(CellBordersType.LEFT_RIGHT, border);
            cell.setBorders(CellBordersType.TOP_BOTTOM, border);
            cell.setBorders(CellBordersType.DIAGONAL_LINES, border);
```

Below code snippet illustrates how to set a cell object with leftborder, top border and diagonal from bottom left to top right.

```
            cell.setBorders(CellBordersType.LEFT, border);
            cell.setBorders(CellBordersType.TOP, border);
            cell.setBorders(CellBordersType.DIAGONALBLTR, border);
```

Below code snippet illustrates how to get a border definition.

```
            Border thisBorder = cell.getBorder(CellBordersType.LEFT);
            thisBorder = cell.getBorder(CellBordersType.TOP);
            thisBorder = cell.getBorder(CellBordersType.DIAGONALBLTR);
```

6 Navigation

6.1 Manipulate TextSearch

6.1.1 TextNavigation

First an ODF text document is needed to test the navigation operation. The following codes shows two main functions of TextNavigation:hasNext() and getCurrentItem(). The first parameter of the TextNavigation constructor is the matched pattern String, and the second isthe navigation scope.

The result of function getCurrentItem is a Selection object, so a TextSelection object is used here to check outthe result.Finally the informations of all the String "What" in the text document will be printed out.

```
            TextDocument
textdoc=(TextDocument)TextDocument.loadDocument("textsearch.odt");
            TextNavigation search1;
            search1=new TextNavigation("What",textdoc);
            while (search1.hasNext()) {
                  TextSelection item1 = (TextSelection)
search1.nextSelection();
                  System.out.println(item1);
            }
```

6.1.2 TextSelection

6.1.2.1 Get Index/Text of TextSelection

Run the following codes will get the text content of the searched String "good" and the corresponding index inthe text document.

```
            TextNavigation search2=new TextNavigation("good",textdoc);
            while(search2.hasNext()){
                  TextSelection item2=(TextSelection)
search2.nextSelection();
                  String searchedText=item2.getText();
                  int searchedIndex=item2.getIndex();
                  System.out.println(searchedText);
                  System.out.println(searchedIndex);
            }
```

6.1.2.2 Cut String

To cut some specified string in a text document, you can do like the following codes whichcut off all the String "day" in the document.

```
               search2=new TextNavigation("day",textdoc);
            while(search2.hasNext()){
                  TextSelection item=(TextSelection)
search2.nextSelection();
                  item.cut();
            }
```

6.1.2.3 Paste String

The following codes paste the string "change" both at the front and at the end of the string "good",by using the function pasteAtFrontOf() and pasteAtEndOf().

```
            search2 = null;
            search2 = new TextNavigation("good", textdoc);
            TextSelection pastesource = null;
            TextNavigation search3 = new TextNavigation("change",
textdoc);
            if (search3.hasNext()) {
                  pastesource = (TextSelection) search3.nextSelection();
            }
            while (search2.hasNext()) {
                  TextSelection item = (TextSelection)
search2.nextSelection();
                  //paste "change" at the front of "good"
                  pastesource.pasteAtFrontOf(item);
                  //paste "change" at the end of "good"
                  pastesource.pasteAtEndOf(item);
            }
```

6.1.2.4 Replace String

The following codes replace all the string "replacesource" with the string "replacedest" in the text document.

```
            search2 = null;
            search2 = new TextNavigation("replacesource", textdoc);
            if (search3.hasNext()) {
                  TextSelection item= (TextSelection)
search3.nextSelection();
                  item.replaceWith("replacedest");
            }
```

6.1.2.5 Add Reference to String

To add reference for a string, you can do like the following codes. Herefunction addHref is used, the parameter of it is an URL object. The codes add network address"http://www.ibm.com" to the string "network".

```
            search2 = null;
            search2 = new TextNavigation("network", textdoc);
            while (search2.hasNext()) {
                  TextSelection item = (TextSelection)
search2.nextSelection();
                  item.addHref(new URL("http://www.ibm.com"));
            }
```

6.1.2.6 Add Comment

Adding comment is a useful function when review document, such as spell check and security check.You can do it like the following codes. Here, function addComment is used, the first parameter isthe comment content, the second parameter is the comment author. The codes add a spell suggestionbefore the string "natwork".

```
            TextNavigation search4 = new TextNavigation("natwork",
textdoc);
            while (search4.hasNext()) {
                  TextSelection selection = (TextSelection)
search4.nextSelection();
                  selection.addComment("Please change 'natwork' with
'network'.",     "SpellChecker");
            }
```

6.1.3 FieldSelection

Field Selection is a decorator class of TextSelection, which help user replace a text content with field.Following code can be used to search the document content, and replace with a simple field.

```
            TextDocument doc =
TextDocument.loadDocument("fieldSample.odt");
            TextNavigation search = new
TextNavigation("ReplaceDateTarget", doc);
            while (search.hasNext()) {
                  TextSelection item = (TextSelection)
search.nextSelection();
                  FieldSelection fieldSelection = new
FieldSelection(item);

fieldSelection.replaceWithSimpleField(Field.FieldType.FIXED_DATE_FIELD)
;
            }
```

Following code can be used to search the document content, and replace with a condition field.

```
            TextSelection item = (TextSelection) search.nextSelection();
            FieldSelection fieldSelection = new FieldSelection(item);
            fieldSelection.replaceWithConditionField("test_con_variable
== \"true\"", "trueText", "falseText");
```

Following code can be used to replace with a hidden field.

```
            fieldSelection.replaceWithHiddenTextField("test_con_variable
== \"true\"", "hiddenText");
```

Following code can be used to replace with a reference field.

```
            ReferenceField referenceField =
Fields.createReferenceField(doc.addParagraph("span").getOdfElement(),
"selection-test-ref");
            fieldSelection.replaceWithReferenceField(referenceField,
ReferenceField.DisplayType.TEXT);
```

Following code can be used to replace with a variable field.

```
            VariableField userVariableField =
Fields.createUserVariableField(doc, "selection_user_variable", "test");
            fieldSelection.replaceWithVariableField(userVariableField);
```

6.1.4 TextStyleNavigation

Similar with TextNavigation, TextStyleNavigation has two main functions: getCurrentItem() andhasNext() which is shown in the following codes. The input parameter of TextStyleNavigation constructoris a map of OdfStyleProperty, so here a TreeMap "searchProps" which contains the Style properties is usedto construct the TextStyleNavigation object.

```
            TextStyleNavigation searchStyle1;
            TreeMap<OdfStyleProperty, String> searchProps = new
TreeMap<OdfStyleProperty, String>();
            searchProps.put(StyleTextPropertiesElement.FontName, "Times
New Roman1");
            searchProps.put(StyleTextPropertiesElement.FontSize,
"16pt");
            searchStyle1 = new TextStyleNavigation(searchProps,
textdoc);
            if (searchStyle1.hasNext()) {
                  TextSelection itemstyle = (TextSelection)
searchStyle1.nextSelection();
                  System.out.print((itemstyle.toString()));
            }
```

7 TextExtractor

7.1 TextExtractor

7.1.1 Get Text

TextExtractor provides a method to get the display text of a single element. EditableTextExtractor is a sub class of TextExtractor.It provides a method to return all the text that the user can typically edit in a document, including text in cotent.xml,header and footer in styles.xml, meta data in meta.xml.

The following codes use EditableTextExtractor as an example, the text of the document "textExtractor.odt" is extracted for user.For TextExtractor, it can't extract the text from a TextDocument.

```
            TextDocument
textdoc=(TextDocument)TextDocument.loadDocument("textExtractor.odt");
            EditableTextExtractor extractorD =
EditableTextExtractor.newOdfEditableTextExtractor(textdoc);
            String output = extractorD.getText();
            System.out.println(output);
```

In the following codes, the whole document content will be returned.This operation is the same in

TextExtractor.

```
            OdfElement elem=textdoc.getContentRoot();
            EditableTextExtractor extractorE =
EditableTextExtractor.newOdfEditableTextExtractor(elem);
            System.out.println(extractorE.getText());
```

8 Field

8.1 Fields

8.1.1 Variable Field

You can use the following code to create a variable field, and set the value.

```
                TextDocument doc = TextDocument.newTextDocument();
                Paragraph paragraph =
doc.addParagraph("test_con_variable:");
                VariableField simpleVariableField =
Fields.createSimpleVariableField(doc, "test_con_variable");
                simpleVariableField.updateField("true",
paragraph.getOdfElement());
```

Following code can be used to set value to variable field, and append it to an ODF element.

```
                simpleVariableField.updateField("user variable
content", null);

simpleVariableField.displayField(paragraph.getOdfElement());
```

8.1.2 Condition Field

Following code can be used to create a condition field.

```
                Paragraph newParagraph = doc.addParagraph("Condition
Field Test:");
                ConditionField conditionField =
Fields.createConditionField(newParagraph.getOdfElement(),
"test_con_variable == \"true\"",
                            "trueText", "falseText");
```

8.1.3 Hidden Field

Following code can be used to create a hidden field.

```
                newParagraph = doc.addParagraph("Hide Text Field
Test:");
                conditionField =
Fields.createHiddenTextField(newParagraph.getOdfElement(),
"test_con_variable == \"true\"", "hiddenText");
```

8.1.4 Cross Reference Field

Following code can be used to create a reference field.

```
                OdfElement newTextSpanElement =
((TextPElement)doc.addParagraph("Reference
Content:").getOdfElement()).newTextSpanElement();
                newTextSpanElement.setTextContent("This is a test
reference content.");
                ReferenceField referenceField =
Fields.createReferenceField(newTextSpanElement, "test-ref");
```

Following code can be used to append a reference field.

```
                referenceField.appendReferenceTo(doc.addParagraph("User
Reference Field:").getOdfElement(), ReferenceField.DisplayType.TEXT);
```

8.1.5 Chapter Field

Following code can be used to create a chapter field.

```
                ChapterField chapterField =
Fields.createChapterField(doc.addParagraph("Chapter:").getOdfElement())
;
```

8.1.6 Title and Subject Field

Following code can be used to create a title field.

```
                TitleField titleField =
Fields.createTitleField(doc.addParagraph("The
Title:").getOdfElement());
```

Following code can be used to create a subject field.

```
                SubjectField subjectField =
Fields.createSubjectField(doc.addParagraph("The
Subject:").getOdfElement());
```

8.1.7 Author Field

Following code can be used to create a author initial field and a author name field.

```
                AuthorField authorField =
Fields.createAuthorInitialsField(doc.addParagraph("The initials of the
author :").getOdfElement());
                authorField =
Fields.createAuthorNameField(doc.addParagraph("Author:").getOdfElement(
));
```

8.1.8 Page Number Field

Following code can be used to create a current page number field.

```
                PageNumberField numberField =
Fields.createCurrentPageNumberField(doc.addParagraph("Current Page
Number:").getOdfElement());

numberField.setNumberFormat(NumberFormat.UPPERCASE_LATIN_ALPHABET);
                numberField.setDisplayPage(DisplayType.NEXT_PAGE);
```

Following code can be used to create a previous page number and a next page number field.

```
                numberField =
Fields.createPreviousPageNumberField(doc.addParagraph("Previous Page
Number:").getOdfElement());
                numberField =
Fields.createNextPageNumberField(doc.addParagraph("Next Page
Number:").getOdfElement());
```

8.1.9 Page Number Field

Following code can be used to create a page count field, and set the number format.

```
                PageCountField countField =
Fields.createPageCountField(doc.addParagraph("Page
Count:").getOdfElement());

countField.setNumberFormat(NumberFormat.UPPERCASE_LATIN_ALPHABET);
```

8.1.10 Date Field

Following code can be used to create a date field, and set the format.

```
                DateField dateField =
Fields.createDateField(doc.addParagraph("Date:").getOdfElement());
                dateField.formatDate("yy-MM-dd");
```

8.1.11 Time Field

Following code can be used to create a time field, and set the format.

```
                TimeField timeField =
Fields.createTimeField(doc.addParagraph("Time:").getOdfElement());
                timeField.formatTime("HH:mm:ss a");
```

9 Metadata

9.1 Manipulate Metadata

9.1.1 Overview

This *Meta* API supports to access and set document metadata. It covers the metadata definitions in *ODF Specification 1.2 Committee Draft05*

9.1.2 Get Meta object

First you load an ODF text document(for example),then get the meta file DOM and then use the meta DOM to create an instance of the Meta. Use Meta you can access all the supported elements.

```
      TextDocument doc = (TextDocument)
TextDocument.loadDocument("testtable.odt");
      OdfFileDom metadom = doc.getMetaDom();
      Meta metadata = new Meta(metadom);
```

9.1.3 Access metadata

After creating the Meta instance,you can use it to manipulate the metadata: for example, you can set a value for the <meta:generator> like this:

```
      metadata.setGenerator("OpenOffice.org/3.0$Win32
OpenOffice.org_project/300m15$Build-9379");
```

The <meta:keyword> may contain many keywords, you can set the whole list of keywords and add one keyword as you want. the api currently do not provide the direct method for deleting one keyword ,you can get the keyword list first,and then delete the keyword,finally set the list to the element.

```
      metadata.addKeyword("java");
      List<String> keywords=metadata.getKeywords();
      keywords.remove("java");
      metadata.setKeywords(keywords);
```

9.1.4 Access the user defined element

To manipulate the user defined data,you should get the list of their names,and then use the names to update the data or its datatype or delete the whole user defined data. you can use the setUserDefinedData(String name, String type, String value) method to update data,if the name not exists in the document,the method will add the new user defined data.

```
     List<String> names=metadata.getUserDefinedDataNames();
      for (String name : names) {
       metadata.removeUserDefinedDataByName(name);
      }
      String key="newId";

//org.odftoolkit.odfdom.dom.attribute.meta.MetaValueTypeAttribute.Value

      metadata.setUserDefinedData(key, Value.STRING.toString(),
"new001");
      //update the  datatype
      metadata.setUserDefinedDataType(key, Value.BOOLEAN.toString());
      //update the data value
      metadata.setUserDefinedDataValue(key, "false");

      //get the datatype
      String dataType=metadata.getUserDefinedDataType(key);
      //get the data value
      String dataValue=metadata.getUserDefinedDataValue(key);
```

9.1.5 Access the document statistics

if you want to access the document statistics,you should get a DocumentStatistic instance, if the return is null,it means that this ODF document doesn't have any document statistic information,you should create a document statistics object.

```
      DocumentStatistic stat = metadata.getDocumentStatistic();
      if(stat==null) {
          stat=new
DocumentStatistic(metadata.getOfficeMetaElement().newMetaDocumentStatis
ticElement());
      }

      stat.setCellCount(3);
      Integer cellCount=stat.getCellCount();
```