1. Basics

1.1. Getting Started

The example demonstrates a reading of the source document from an HTML string and a writing of a conversion result to a temp file.

The conversion relies on default settings of PD4ML: output format is **A4**, **10mm** margins etc.

After the conversion is done, the resulting PDF is open with a default PDF viewer application.

```
PD4ML pd4ml = new PD4ML();

String html = "TEST<pd4ml:page.break><b>Hello, World!</b>";
ByteArrayInputStream bais =
        new ByteArrayInputStream(html.getBytes());

// read and parse HTML
pd4ml.readHTML(bais);

File pdf = File.createTempFile("result", ".pdf");
FileOutputStream fos = new FileOutputStream(pdf);

// render and write the result as PDF
pd4ml.writePDF(fos);

// alternatively or additionally:
// pd4ml.writeRTF(rtfos, false);
// BufferedImage[] images = pd4ml.renderAsImages();

// open the just-generated PDF with a default PDF viewer
Desktop.getDesktop().open(pdf);
```

E001GettingStarted.java

1.2. Set Page Format

Page format and page margins settings are represented with new `com.pd4ml.PageSize` and `com.pd4ml.PageMargins` classes correspondingly.

PageSize class has already predefined constants for commonly used paper formats. A definition of an arbitrary page format (measured in **pt** or **mm**) is also possible, of course.

Both **PageSize** and **PageMargins** settings can be applied to selected range of pages, distinguished by **scope** attribute. Multiple calls of `setPageSize()` or `setPageMargins()` are allowed. By an overlapping/conflict of

the page ranges, a later call wins.

If **scope** attribute is omitted, the setting is applied to all document pages.

```
// define page format for the first page
pd4ml.setPageSize(PageSize.A5, "1");

// define landscape page format for the second and following pages
pd4ml.setPageSize(PageSize.A4.rotate(), "2+");

// reset page margins for the first two pages
pd4ml.setPageMargins(new PageMargins(0, 0, 0, 0), "1-2");

// set page margins for the third and following (if any) pages
pd4ml.setPageMargins(new PageMargins(0, 0, 0, 0), "3+");
```

E002SetPageFormat.java

1.3. Set Page Header And Footer

`setPageHeader()` and `setPageFooter()` methods are used to define page headers/footers using HTML as a layout definition.

Optional **scope** parameter allows to apply particular header or footer to a specified range of pages.

The HTML code may include the placeholders: **$[page]** – to be substituted with current page number; **$[total]** – total number of pages; **$[title]** – document title as defined in **<title>** HTML tag or overridden with `setDocumentTitle()` API call.

```
// define page header for the first page 30px high
pd4ml.setPageHeader("$[title]", 30, "1");

// define page footer for the first page
pd4ml.setPageFooter("Total pages: $[total]", 30, "1");

// define page header for the second page
pd4ml.setPageHeader("<b>$[title]</b> $[page]/$[total]", 30, "2+");

// define page footer for the second page
pd4ml.setPageFooter("<div style='width: 100%; text-align: right'>Page: $[page]
</div>", 30, "2+");
```

E003SetPageHeaderFooter.java

1.4. Set Page Header Footer Inline

Instead of `setPageHeader()` and `setPageFooter()` API calls, you may define header/footer code in your source HTML document inline with proprietary `<pd4ml:page.header>` and `<pd4ml:page.footer>` tags.

Optional **scope** attribute allows to apply particular header or footer to a specified range of pages. `<pd4ml:page.header>` and `<pd4ml:page.footer>` tags may appear in any location of document body (preferable as a direct child of `<body>` element) and impact current and subsequent pages until it is overridden again.

The HTML code may include the placeholders: **$[page]** – to be substituted with current page number; **$[total]** – total number of pages; **$[title]** – document title as defined in `<title>` HTML tag or overridden with `setDocumentTitle()` API call.

```
1.  <html>
2.  <head>
3.  <title>Header/Footer example</title>
4.  <style>BODY {font-family: Arial}</style>
5.  </head>
6.  <body>
7.
8.  <!-- inline definition of the header/footer for the current and all following
    pages -->
9.  <pd4ml:page.header height=30>$[title]</pd4ml:page.header>
10. <pd4ml:page.footer height=30>Total pages: $[total]</pd4ml:page.footer>
11.
12. First Page
13.
14. <pd4ml:page.break>
15.
16. <!-- here it overrides the header/footer set above starting from the current
    page -->
17. <pd4ml:page.header height=30>
18. <b>$[title]</b> $[page]/$[total]
19. </pd4ml:page.header>
20. <pd4ml:page.footer height=30>
21. <div style='width: 100%; text-align: right'>Page: $[page]</div>
22. </pd4ml:page.footer>
23.
24. Second Page
25. </body>
26. </html>
```

E004SetPageHeaderFooterInline.java

1.5. Set Page Background

`setPageBackground()` API call is intended to define target media background layout. The layout is defined in

HTML: in the simplest case it can be just a scanned form image (i.e. `<img width=100%; height=100% src=form.jpg>`) or it can be more sophisticated HTML/CSS/SVG code.

The HTML code may even include the placeholders: **$[page]** – to be substituted with current page number; **$[total]** – total number of pages; **$[title]** – document title as defined in `<title>` HTML tag or overridden with `setDocumentTitle()` API call.

The background is rendered for all available target media space, ignoring specified margins (if any).

Optional **scope** parameter allows to apply background to a specified range of pages.

```
// define page header for the first page
pd4ml.setPageBackground("<div style='width: 100%; height: 100%; background-color: rgb(228,255,228);'></div>", "1");

// define page footer for the first page
pd4ml.setPageBackground("<div style='width: 100%; height: 100%; background-color: rgb(255,228,228);'></div>", "2+");
```

E005SetPageBackground.java

1.6. Set Page Background Inline

Usage of a proprietary `<pd4ml:page.background>` HTML tag as an alternative to API background definition

```
<html>
<head>
<title>Page background example</title>
<style>BODY {font-family: Arial}</style>
</head>
<body>
<pd4ml:page.background>
<div style='width: 100%; height: 100%; background-color: rgb(228,255,228);'>
</div>
</pd4ml:page.background>
First Page

<pd4ml:page.break>
<!--
// override the previously defined background with a new one starting from the current page
// A similar can be achieved if you place the page background definition to the top of the
// doc and set scope='2+' attribute
//
// Note: the style is applied to <pd4ml:page.background> tag in the case
-->
<pd4ml:page.background style='width: 100%; height: 100%; background-color:
```

```
rgb(255,228,228);'></pd4ml:page.background>
Second Page
</body>
</html>
```

E006SetPageBackgroundInline.java

1.7. Set Page Watermark

PD4ML provides an easy way to utilize native PDF watermarking. PDF watermarks can be configured to only be visible in screen viewers, in printed output or both.

As usually in PD4ML, a watermark layout can be defined using HTML/CSS/SVG code (unfortunately no placeholders like **$[page]** or **$[total]** supported). It is possible to control a watermark position, opacity, angle, scale and a page range to apply.

See `setWatermark()` API call documentation.

```
// define watermark for the first page
pd4ml.setWatermark("<b>WATERMARK</b>",
20, // offset X
0, // offset Y
.3f, // opacity
30, // angle
9, // scale (1 = 100%)
true, // should the watermark be visible in PDF viewers?
true, // should the watermark be printed?
"1"); // page range to apply

// define watermark for the second and following pages
pd4ml.setWatermark("<b style='color: tomato'>WATERMARK</b>", 20, 0, .3f, 30, 9,
true, true, "2+");
```

E007SetPageWatermark.java

1.8. Set Page Watermark Inline

Usage of a proprietary `<pd4ml:watermark>` HTML tag as an alternative to API watermark definition

```
<html>
<head>
<title>Watermarking example</title>
<style>BODY {font-family: Arial}</style>
```

```
</head>
<body>

<pd4ml:watermark style="opacity: 30%; left: 20px; top: 0; scale: 900%; angle: 30deg; media: screen, print;" scope="1">
<b>WATERMARK</b>
</pd4ml:watermark>

<pd4ml:watermark style="opacity: 30%; left: 20px; top: 0; scale: 900%; angle: 30deg; media: screen, print;" scope="2+">
<b style='color: tomato'>WATERMARK</b>
</pd4ml:watermark>

First Page

<pd4ml:page.break>

Second Page

</body>
</html>
```

E008SetPageWatermarkInline.java

1.9. Set Document Password

`setPermissions()` method allows to apply the standard PDF security options: define a document password or restrict particular document actions (like a hi-res print).

See a list of applicable permission flags (`Allow*` and `DefaultPermissions`).

It is possible to define a positive list of permissions:

```
pd4ml.setPermissions(null, Constants.AllowAnnotate |
Constants.AllowDegradedPrint);
```

or to disable only selected ones:

```
pd4ml.setPermissions(null, Constants.DefaultPermissions ^
Constants.AllowModify);
```

```
// protect the document with "test" password. No permission restrictions applied
pd4ml.setPermissions("test", Constants.DefaultPermissions);
```

E009SetDocumentPassword.java

1.10. Inject Html

With PD4ML API it is possible to inject an arbitrary HTML portion either just after opening `<body>` or right before closing `</body>` tag of a source HTML document.

Be careful: with HTML portion it is easy to corrupt the original document layout. As an extreme case, if you inset a beginning of HTML comment with `pd4ml.injectHtml("<--", true);` API call, you obviously get a blank PDF document.

```
1.  // insert some content just after the opening <body> tag:
2.  pd4ml.injectHtml("Some new content to the top of the document", true);
3.
4.  // insert some content before the closing </body> tag:
5.  pd4ml.injectHtml("<p style='color: tomato'>Content to append", false);
```

E010InjectHtml.java

2. HTML

2.1. Add Style Programmatically

`addStyle()` API call applies an extra stylesheet to the source document. It can be specified as a style string or an external resource reference.

Multiple invocations of the method are possible. The method takes effect only if called before `readHTML()` .

```
1.  pd4ml.setHtmlWidth(900); // render HTML in a virtual frame 900px wide
2.  pd4ml.addStyle(
3.  // specify TTF font file for "Consolas" font face (only "plain" style, in the
    case).
4.
5.  // Here we use free FiraMono-Regular instead of the original Consolas.
6.  // Other font faces to be mapped to PDF viewer standard built-in fonts.
7.
8.  // In the resulting PDF you can see '?' symbols instead of some character
    glyphs.
9.  // That means the missing glyphs are not defined by any of the available fonts.
10.
```

```
// As a workaround create a font dir, place a set of fonts there to cover the
// desired language or character range, index fonts and refer to the dir
// with pd4ml.useTTF() API call. Optionally the font dir can be packed to
// a fonts.jar
"@font-face {\n" +
" font-family: \"Consolas\";\n" +
" src: url(\"java:/html/rc/FiraMono-Regular.ttf\") format(\"ttf\"),\n" +
"}\n", false);

// read and parse HTML
pd4ml.readHTML(new URL("html/H001.htm"));
```

H001ConvertHtml.java

2.2. Add TOC

`<pd4ml:toc>` proprietary tag is substituted with a table of contents, auto-generated from `<H1>`-`<H6>` hierarchy.

The generated TOC is an HTML table, whose appearance can be customized using CSS style.

The example illustrates how to inject a table of contents to the top of a document fro Java API.

```
pd4ml.injectHtml("<pd4ml:toc>", true);

// forces PD4ML to process <pd4ml:toc> tag as it was in the source HTML
// just after opening <body> tag.
```

An attribute **pd4toc="nopagenum"** added to `<H1>`-`<H6>` tags suppresses a page number generation for the marked TOC entries.

H002AddTOC.java

2.3. Page Number Tag

By default `<pd4ml:page.number>` tag is substituted with a current page number. Optional **OF** attribute should refer to an HTML element with matching **ID** attribute value – in the case the tag is substituted with a page number where the referenced element is located.

```
<html>
<body>
Total pages: <pd4ml:page.number><br>
<a href="#continue1"><b>Section 1</b></a> on page <pd4ml:page.number
```

```
of="continue1"><br>
<a href="#continue2"><b>Section 2</b> on page <pd4ml:page.number
of="continue2"></a><br>
<pd4ml:page.break>
<a name="continue1">Section 1</a>
<pd4ml:page.break>
<div id="continue2">Section 2</div>
</body>
</html>
```

2.4. Create Bookmarks

PD4ML supports three methods of bookmarks (aka PDF outlines) generation:

- `<H1>`-`<H6>`
- `<a name="chapter1">Chapter 1</a>`
- `<pd4ml:bookmark>`

The API call illustrates the first method.

```
pd4ml.generateBookmarksFromHeadings(true);
```

See also `generateBookmarksFromAnchors()`

Bookmarks defined with `<pd4ml:bookmark>` are included into bookmarks structure regardless if it is generated with method one or two.

H003CreateBookmarks.java

2.5. Apply Page Breaks

To force a page break you may use either standard CSS method

```
pd4ml.addStyle(
"H3 { page-break-before: always; }\n" +
"H3:first-of-type { page-break-before: auto; }", true);
```

or PD4ML's proprietary `<pd4ml:page.break>` tag.

In PD4ML versions prior to v4 `<pd4ml:page.break>` supports some useful features relevant for PDF output: to rotate page, to change HTML-to-PDF scale factor etc. Also the page break can be conditional. The features are going to be ported to v4 in the forthcoming releases.

H004ApplyPageBreaks.java

2.6. Add Attachment

`<pd4ml:attachment>` tag makes possible to include an arbitrary document or a binary file to resulting PDF as an attachment. The resource to attach is referenced by **SRC** attribute.

`<pd4ml:attachment>` tag can be placed to any reasonable location of a document. In PDF the tag will be substituted with a clickable icon, which opens the attachment with a default viewer application for the attached file type.

There are icon options:

where **area** is a special "invisible" icon, which only turns a neighbor region into a clickable area. The region dimensions are specified with **WIDTH** and **HEIGHT** attributes.

The example shows the way how to add an attachment to the top part of the document with an API call.

```
1.  // with the below code we embed the document source as an attachment to the
    resulting PDF
2.  // The attachment icon will appear on the top (right side) of the document
    layout
3.  pd4ml.injectHtml("<div style=\"text-align: right; width: 100%\">"
4.  + "<pd4ml:attachment style=\"align: right\" type=\"paperclip\"
    src=\"H001.htm\"/>"
5.  + "</div>", true);
```

H005AddAttachment.java

2.7. Footnotes / Endnotes

`<pd4ml:footnote>` tag forces PD4ML to move its nested content to the bottom part of a current page and print a footnote auto-incremented index instead. If footnotes area takes to much space, not fitting footnotes are moved to subsequent page(s).

An appearance of **noref** attribute suppresses the footnote index.

`<pd4ml:footnote.caption>` allows to specify a delimiter between the main document content and footnotes area.

`<pd4ml:endnote>` and `<pd4ml:endnote.caption>` acts identically, however the endnote content is moved not to the bottom part of a page, but to the end of the document.

```
1.  <pd4ml:footnote.caption>
2.  Footnotes
3.  <hr>
4.  </pd4ml:footnote.caption>
5.
```

```
<pd4ml:footnote noref>This footnote has no reference from the main text</pd4ml:footnote>

A note is a string of text placed at the bottom of a page in a book or document or at the end of a chapter,
volume or the whole text<pd4ml:footnote>In some editions of the Bible, notes are placed in a narrow column
in the middle of each page between two columns of biblical text.
</pd4ml:footnote>.
The note can provide an author's comments on the main text or citations of a
reference work in support of the text, or both.
<p>
Footnotes are notes at the foot of the page while
endnotes<pd4ml:footnote>Unlike footnotes, endnotes have the advantage of not
affecting the layout of the main text, but may cause inconvenience to readers who have to move back
and forth between the main text and the endnotes.</pd4ml:footnote> are collected under a separate heading at
the end of a chapter, volume, or entire work.
<p>
```

H006.htm

3. i18n

3.1. Embedding TTF Fonts

To support non-Latin charsets, all referenced TTF fonts need to be shaped (unused glyphs removed) and embedded to the resulting PDF. To do that PD4ML needs a direct access to TTF font files, as `java.awt.Font` object unfortunately provides no way to read font file bytes.

So in order to work with non-Latin charsets, PD4ML needs to be informed, where it can find font files and which ones can be used.

`useTTF()` API methods let PD4ML know a font directory location or font folder in a resource JAR. Multiple invocations of `useTTF()` are also allowed.

In a font directory PD4ML expects to find **pd4fonts.properties** index file with **font face name -> font file name** mapping information. If the file is not there, it is possible to enable an auto-indexing.

```
public final static String FONTS_DIR = "c:/windows/fonts";

...

PD4ML pd4ml = new PD4ML();
pd4ml.useTTF(FONTS_DIR, true); // The second parameter forces to index fonts in FONTS_DIR.
```

```
// As the indexing of a font directory with a big number of fonts is
time/resource consuming,
// it is a good idea to prepare the font mapping file in advance.
// See the next example how to index.
```

On Windows platform a typical font directory location is **c:/windows/fonts**, but unfortunately it is write-protected and it is not recommended to store **pd4fonts.properties** there.

If you want to use fonts from there, you may rely on auto-index, but limit the scope of indexed fonts with a pattern for a better performance.

N001TtfFonts.java

3.2. Preparing TTF Fonts

A generation of **pd4fonts.properties** from a Java application:

```
// Index available fonts. As the indexing time/resource consuming,
// it is a good idea to prepare the font mapping file in advance.
File index = File.createTempFile("pd4fonts", ".properties");
index.deleteOnExit();
FontCache.generateFontPropertiesFile(FONTS_DIR, index.getAbsolutePath(),
(short)0);

System.out.println("font indexing is done.");
// The same can be done with a command line call:
// java -jar pd4ml.jar -configure.fonts <font.dir> [index.file.location]

...

pd4ml.useTTF(index.getAbsolutePath());
```

A similar can be achieved with a command line call:

```
 java -jar pd4ml.jar -Xmx512m -configure.fonts c:/windows/fonts
d:/write/enabled/dir/pd4fonts.properties
```

After the font dir is indexed and an index file is stored to **d:/write/enabled/dir/**, you may refer the fonts with the API call

```
 pd4ml.useTTF("d:/write/enabled/dir/", false);
```

N002TtfFonts.java

4. Advanced

4.1. Popup Print Dialog

The code below makes PDF print dialog popup as soon as the document is opened in a PDF viewer.

```
pd4ml.addDocumentActionHandler("printdialog", null);
// similarly:
// pd4ml.addDocumentActionHandler("OpenAction", "this.print(true);");
```

A001PopupPrintDialog.java

4.2. Silent Print

The code forces PDF viewer to initiate a printing to default printer as soon as the document is open. Modern PDF viewers normally ask for a confirmation in the case, so a really “silent print” is fortunately not possible.

```
pd4ml.addDocumentActionHandler("silentprint", null);
// similarly:
// pd4ml.addDocumentActionHandler("OpenAction", "this.print({bUI: false,
bSilent: true});");
```

A002SilentPrint.java

4.3. Read Resources From Classpath

To address resources via Java Classloader, PD4ML provides support for a non-standard **“java:”** protocol.

```
// read and parse HTML
pd4ml.readHTML(new URL("java:/advanced/A003.htm"));

// If you need to handle "java:" URLs in your application, run once the
following code
// e.g. in "static { }" section

URL.setURLStreamHandlerFactory(new URLStreamHandlerFactory() {
public URLStreamHandler createURLStreamHandler(String protocol) {
return "java".equals(protocol) ? new URLStreamHandler() {
protected URLConnection openConnection(URL url) throws IOException {
return new URLConnection(url) {
public void connect() throws IOException {
}
```

```
14.   };
15.   }
16.   } : null;
17.   }
18.   });
```

The `URL.setURLStreamHandlerFactory()` call is implicitly done by `PD4ML()` instantiation to suppress `java.net.MalformedURLException: Unknown protocol` . Do the same in your application if you need to deal with **"java:"** URLs.

A003ReadHtmlFromClasspath.java

4.4. Add Progress Listener

HTML conversion of big documents may take a while. If you use PD4ML in a GUI application, probably you would like to show a progress bar which informs the user about the conversion state instead of just showing the empty page.

PD4ML provides a callback API for that.

In the example all progress events are just dumped to STDOUT. It is up to you how to use the progress data in your application for a better user experience.

```
1.  public static class ProgressMeter implements ProgressListener {
2.
3.  private long startTime = -1;
4.
5.  /**
6.  * callback method triggered by progress event. The implementation      dumps
    the events to STDOUT.
7.  * Alternatively it could control GUI progress bar etc.
8.  */
9.  public void progressUpdate(int messageID, int progress, String note, long msec)
    {
10.
11.  if ( startTime < 0 ) {
12.  startTime = msec;
13.  }
14.
15.  String tick = String.format( "%7d", msec - startTime );
16.  String progressString = String.format( "%3d", progress );
17.
18.  String step = "";
19.  switch ( messageID ) {
20.  case CONVERSION_BEGIN:
21.  step = "conversion begin";
22.  break;
23.  case MAIN_DOC_READ:
24.  step = "doc read";
25.  break;
```

```
26.   case HTML_PARSED:
27.   step = "html parsed";
28.   break;
29.   case RENDERER_TREE_BUILT:
30.   step = "document tree structure built";
31.   break;
32.   case HTML_LAYOUT_IN_PROGRESS:
33.   step = "layouting...";
34.   break;
35.   case HTML_LAYOUT_DONE:
36.   step = "layout done";
37.   break;
38.   case PAGEBREAKS_ALIGNED:
39.   step = "pagebreaks aligned";
40.   break;
41.   case TOC_GENERATED:
42.   step = "TOC generated";
43.   break;
44.   case DOC_RENDER_IN_PROGRESS:
45.   step = "generating doc page";
46.   break;
47.   case RTF_PRE_RENDER_DONE:
48.   step = "RTF pre-render done";
49.   break;
50.   case DOC_WRITE_BEGIN:
51.   step = "writing doc...";
52.   break;
53.   case CONVERSION_END:
54.   step = "done.";
55.   break;
56.   }
57.   System.out.println( tick + " " + progressString + " " + step + " " + note );
58.   }
59.   }
60.
61.   ...
62.
63.   pd4ml.monitorProgressWith(new ProgressMeter());
```

A004AddProgressListener.java

4.5. Add Custom Resource Loader

If some HTML resources like images or stylesheets are not accessible with the standard methods (file read, HTTP(S), etc), you may define your own resource reading “driver”.

First, define a resource addressing syntax, that matches your needs. For example `<a src="database:table=pictures;id=4711">`

Second, implement a resource loader, which knows what to do with **“database:table=pictures;id=4711”** URL.

The loader has to be derived from **com.pd4ml.ResourceProvider** class and to implement two methods:
`boolean canLoad(String resource, FileCache cache)` to test if it can read the URL;
`BufferedInputStream getResourceAsStream(String resource, FileCache cache)` to actually read the resource bytes.

```
1.  public class DummyProvider extends ResourceProvider {
2.
3.  public final static String PROTOCOL = "dummy";
4.
5.  @Override
6.  public BufferedInputStream getResourceAsStream(String resource, FileCache
    cache) throws IOException {
7.  if (!resource.toLowerCase().startsWith(PROTOCOL)) {
8.  return null;
9.  }
10.
11. // interpret the "resource" parameter according to your protocol (e.g. as a key
    to a database record etc)
12.
13. // in the example we simply dump the resource parameter string
14. String buf = "[" + resource.substring(PROTOCOL.length()+1) + "]";
15. ByteArrayInputStream baos = new ByteArrayInputStream(buf.getBytes());
16. return new BufferedInputStream(baos);
17. }
18.
19. @Override
20. public boolean canLoad(String resource, FileCache cache) {
21. if (resource.toLowerCase().startsWith(PROTOCOL)) {
22. return true;
23. }
24. return false;
25. }
26. }
27.
28. pd4ml.addCustomResourceProvider("advanced.DummyProvider");
```

A005AddCustomResourceLoader.java

4.6. Substitute Placeholders

A simple way to add dynamic content to your static HTML templates.

Add **$[var1]**, **$[my.variable]** etc placeholders to your HTML.

During conversion specify dynamic content for the placeholders this way:

```
1.  HashMap<String, String> map = new HashMap<>();
2.  map.put("var1", "value 1");
```

```
map.put("var2", "[value 2]");
map.put("var3", "* value 3 *");
map.put("my.variable", "Dynamically inserted text");
pd4ml.setDynamicData(map);
```

$[page], **$[total]** and **$[title]** placeholders are reserved.

A006SubstitutePlaceholders.java

4.7. Rendering Status Info

Receiving some conversion statistics and diagnostics data:

```
// render and write the result as PDF/A
pd4ml.writePDF(fos, Constants.PDFA);

System.out.println("pages: " +
(Long)pd4ml.getLastRenderInfo(Constants.PD4ML_TOTAL_PAGES));

// reports actual HTML document layout height in pixels
// (as a rule the value depends on htmlWidth conversion parameter)
System.out.println("height: " +
(Long)pd4ml.getLastRenderInfo(Constants.PD4ML_DOCUMENT_HEIGHT_PX));

// reports default width of the HTML document layout in pixels.
// If the document has root-level elements with width="100%",
// the returned value is almost always going to be equal htmlWidth parameter.
// If the returned value is smaller htmlWidth, probably it is optimal htmlWidth
for the given document.
System.out.println("right edge: " +
(Long)pd4ml.getLastRenderInfo(Constants.PD4ML_RIGHT_EDGE_PX));

StatusMessage[] msgs =
(StatusMessage[])pd4ml.getLastRenderInfo(Constants.PD4ML_PDFA_STATUS);

for ( int i = 0; i < msgs.length; i++ ) {
System.out.println( (msgs[i].isError() ? "ERROR: " : "WARNING: ") +
msgs[i].getMessage());
}
```

A007RenderingStatusInfo.java

4.8. Adding Custom Tag Renderer

PD4ML provides a way to introduce your own HTML tags. The example illustrates a way, how to define `<star>` tag, which renders (surprise!) a star. See StarTag class implementation

```
String html = "TEST STAR [<star height=20 width=20 style='border: 1 solid blue'>]";
pd4ml.addCustomTagHandler("star", new StarTag());

ByteArrayInputStream bais = new ByteArrayInputStream(html.getBytes());
pd4ml.readHTML(bais);
```

FYI: Using this API PD4ML plugs external MathML and SVG renderers in.

A008AddingCustomTagRenderer.java

5. PDF Tools

5.1. Convert And Merge With PDF

With `merge()` API call you may specify a PDF document to merge HTML conversion result with. It can be entire static PDF document or only selected pages of the document.

```
URL pdfUrl = new URL("java:/pdftools/PDFOpenParameters.pdf");
PdfDocument pdf = new PdfDocument(pdfUrl, null);

File f = File.createTempFile("result", ".pdf");

pd4ml.setPageHeader("HEADER $[page] of $[total]", 40, "1+");

// merge only with pages from 2 to 4. The pages will be appended to the converted PDF
pd4ml.merge(pdf, 2, 4, true);

pd4ml.readHTML(new ByteArrayInputStream(html.getBytes()));
pd4ml.writePDF(new FileOutputStream(f));
```

P001ConvertAndMergeWithPDF.java

5.2. Merge Two PDFs

PD4ML also provides a set of useful tools to deal with PDF.

The example illustrates how to merge two static PDFs to a single doc. It is straightforward.

```
1.  URL pdfUrl1 = new URL("java:/pdftools/doc1.pdf");
2.  URL pdfUrl2 = new URL("java:/pdftools/doc2.pdf");
3.  PdfDocument pdf1 = new PdfDocument(pdfUrl1, null);
4.  PdfDocument pdf2 = new PdfDocument(pdfUrl2, null);
5.
6.  File f = File.createTempFile("pdf", ".pdf");
7.
8.  pdf1.append(pdf2);
9.  pdf1.write(new FileOutputStream(f));
```

P002MergeTwoPDFs.java

5.3. Merge Two PDFs And Protect With Password

As an extension of the previous example, the resulting document is also protected with a password and reduced permissions.

```
1.  URL pdfUrl1 = new URL("java:/pdftools/doc1.pdf");
2.  URL pdfUrl2 = new URL("java:/pdftools/doc2.pdf");
3.  PdfDocument pdf1 = new PdfDocument(pdfUrl1, null);
4.  PdfDocument pdf2 = new PdfDocument(pdfUrl2, null);
5.
6.  File f = File.createTempFile("pdf", ".pdf");
7.
8.  pdf1.append(pdf2);
9.  pdf1.write(new FileOutputStream(f), "test", // Protect the resulting PDF with
    password "test"
10. Constants.AllowDegradedPrint | Constants.AllowAnnotate);
```

P003MergeTwoPDFsAndProtectWithPassword.java

5.4. Update Pdf Meta Info

PD4ML's PDF tools make possible to update PDF document meta info.

```
1.  PdfDocument doc = new PdfDocument(pdfUrl, null);
2.
```

```
System.out.println("document author: " + doc.getAuthor());

doc.setTitle("Document Modification Test");
doc.setSubject("PdfDocument API test");
doc.setKeywords("key1, key2");
doc.setModDate(); // set modification date to NOW

doc.write(new FileOutputStream(f), null, -1); // no password, default
permissions
```

P004UpdatePdfMetaInfo.java

5.5. Underlay/Overlay

A very special way of PDF document merging: overlay and underlay.

```
PdfDocument doc1 = new PdfDocument(pdfUrl, null);
PdfDocument doc2 = new PdfDocument(pdfUrl, null);

// overlay request to place doc2 content over doc1
// "1" limits to use only the first page of doc2 as an overlay content
// "2+" specifies to apply the overlay to the second and all subsequent pages
// "128" is opacity of overlay (doc2) content, which corresponds ~50%
doc1.overlay(doc2, "1", "2+", 128);
// doc1.underlay(doc2, "1", "2+", 128);

File f = File.createTempFile("pdf", ".pdf");

// writing the overlay result as a new PDF document
FileOutputStream fos = new FileOutputStream(f);
doc1.write(fos);
```

P005UnderlayOverlay.java