Introducing pdfHTML

Technical presentation

Introduction

Introduction

iText 7

- Released in May 2016
- Reimplementation for speed, elegance & scalability
- Modular & extensible
- Introducting iText as a platform

pdfHTML

- Successor to iText 5's XMLWorker
- Part of iText 7 add-on suite
  - [core + add-ons graphic]

Use case

Convert HTML content to PDF

Why not generate from a browser?

- Re-use the structural information from HTML so that you can easily create PDF/A, PDF/UA, or Tagged PDF
- Headless process means no visual rendering overhead which keeps performance high
  - Visual rendering can cause performance challenges that pdfHTML avoids

Changes from iText 5's XMLWorker

- Supports more HTML tags & CSS features such as:
  - <abbr>
  - floating & fixed positioning
  - @media
- More easily extensible for custom tags
- Integrates seamlessly with other iText functionality such as:
  - Barcodes
  - PDF/A
  - pdfCalligraph
- Does not require XHTML to function
  - XHTML is a very strict subset of HTML

Architecture

Steps

Parse HTML file

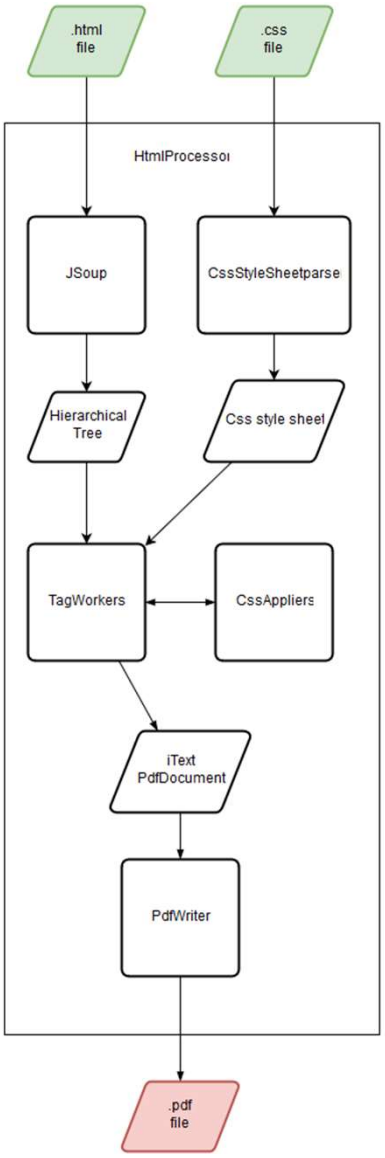
Resolve resource locations (images, CSS, ...)	Parse DOM •Correct DOM errors •Construct DOM hierarchy	Parse CSS

Translate HTML DOM to iText layout objects

Create iText object from HTML object + contents •LIFO (stack)	Resolve CSS to DOM •Apply CSS	Set iText properties from CSS

Feed iText layout objects to PdfDocument

Write PDF file



Basic usage

- A few lines of pdfHTML code:
  - Specify input
  - Specify output
  - Specify config (optional)

```
ConverterProperties props = new ConverterProperties();
HtmlConverter.convertToPdf(new File("in.html"), new File("out.pdf"), props);
```

- Diverse options:
  - Input:
    - String
    - InputStream
    - File
  - Return value:
    - Document object
    - List of iText elements
    - void, but write to output
  - Output
    - OutputStream
    - File
    - PdfDocument

Configuration

Resources

BaseURI

- Root location for resources
  - Images
  - CSS & JavaScript
- Default is either:
  - Location of input HTML file
  - Location where code is executed

FontProvider

- Locations of all fonts
- Default
  - Standard 14 PDF fonts
  - Free fonts included in pdfHTML
- Default can easily be adapted
  - Alternative constructor for specific font types
    - All fonts in default font folder can be added
  - Add or remove a specific font

Resources

```
ConverterProperties props = new ConverterProperties();

// baseUri
props.setBaseUri("./static/");

// fonts
FontProvider dfp = new DefaultFontProvider();
dfp.addFont("/path/to/MyFont.ttf");
props.setFontProvider(dfp);

// media device
props.setMediaDeviceDescription(new MediaDeviceDescription(MediaType.PRINT));

// execute logic
File input = new File("/path/to/input.html");
File output = new File("/path/to/output.pdf");
HtmlConverter.convertToPdf(input, output, props);
```

Customizations

HTML

- Custom HTML tags
  - Input: <qrcode content="http://itextpdf.com" />
  - Output: (image)
- Custom use of standard HTML tags
  - E.g. If text contains "total price", put in **bold**
- Technique: Custom implementation of ITagWorker interface

CSS

- Custom CSS attributes or values
  - e.g. use specific colors in company logo
- Custom use of standard CSS attributes
  - Color-blindness:
    - Simulate what color-blind person sees
    - Maximize color contrasts for color-blind people
- Technique: Custom implementation of ICSSApplier interface

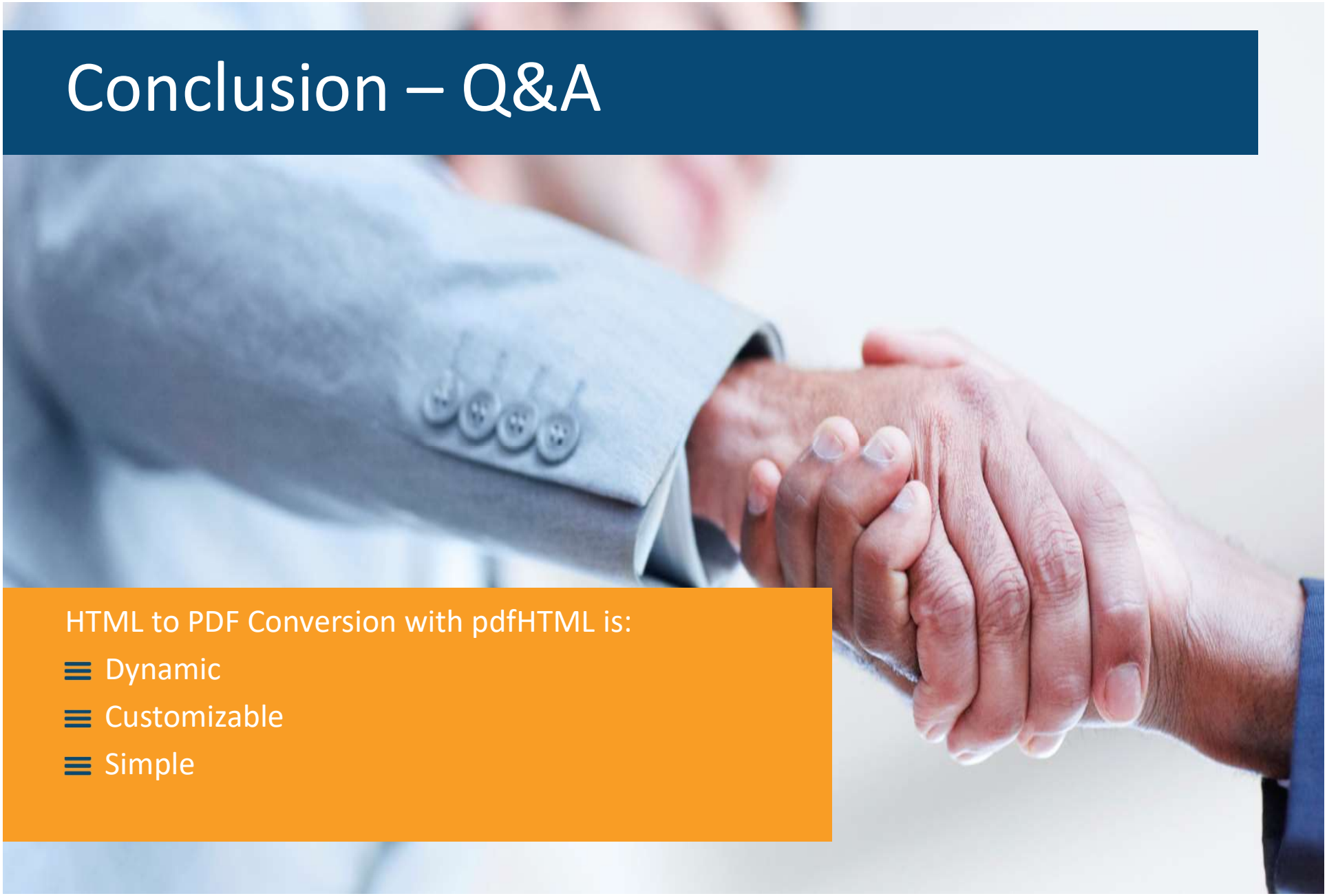
Customizations

```
class MyTagWorkerFactory extends DefaultTagWorkerFactory {
    @Override
    public ITagWorker getCustomTagWorker(IElementNode tag, ProcessorContext context) {
        if (tag.name().equals("qrcode")) {
            return new CustomTagWorker(); // implements ITagWorker
        }
        if (tag.name().equals ("p")) {
            return new CustomParagraphTagWorker(); // extends ParagraphTagWorker
        }
        return null;
    }
}

ConverterProperties props = new ConverterProperties();

// Tag worker registration
ITagWorkerFactory fact = new MyTagWorkerFactory();
props.setTagWorkerFactory(fact);
```

Conclusion – Q&A



HTML to PDF Conversion with pdfHTML is:

- Dynamic
- Customizable
- Simple