PDFUnit

Automated PDF Tests

Purchase orders, invoices and other PDF documents are the output of individual programs.

They might have errors. They can be tested.

more ...

Content

- General requirements for PDF testing tools
- Features of PDFUnit-Java
- PDFUnit-Monitor
- Utilities
- Code Examples, PDFUnit-Java
- Code Examples, PDFUnit-NET
- Code Examples, PDFUnit-Perl
- Code Examples, PDFUnit-XML
- History of PDFUnit
- Inside PDFUnit-Java

General Requirements for every Testing Tool

- A testing tool has to be easy to use
- A testing tool has to cover the functionality requested by the market
- A testing tool has to integrate with common languages
- A testing tool has to be tested itself

PDFUnit fulfills these requirements

Java - available
.NET - available
Perl - available
XML - available

Features of PDFUnit (I)

- Checking for text:

  - on specified pages,
  - in spedified regions
  - configurable whitespace handling
  - rotated text
  - Unicode proved (french, swedisch, greek, russian, ...)
  - right-to-left text (arabic, hebrew, ...)

- Comparing two PDF documents:

  - on specified pages
  - in specified regions
  - text
  - layout (as rendered images)

Features of PDFUnit (II)

- Validating:
  - actions, acrofields, bookmarks,
    custom properties, destinations, document infos,
    embedded files, fast web view, fonts, font types,
    JavaScript, layers, page format
    signature data, text overflow in acrofields

- Tests with images:
  - on specified pages
  - in page regions
  - comparing images of PDF against image files
  - counting visible images
  - size

Features of PDFUnit (III)

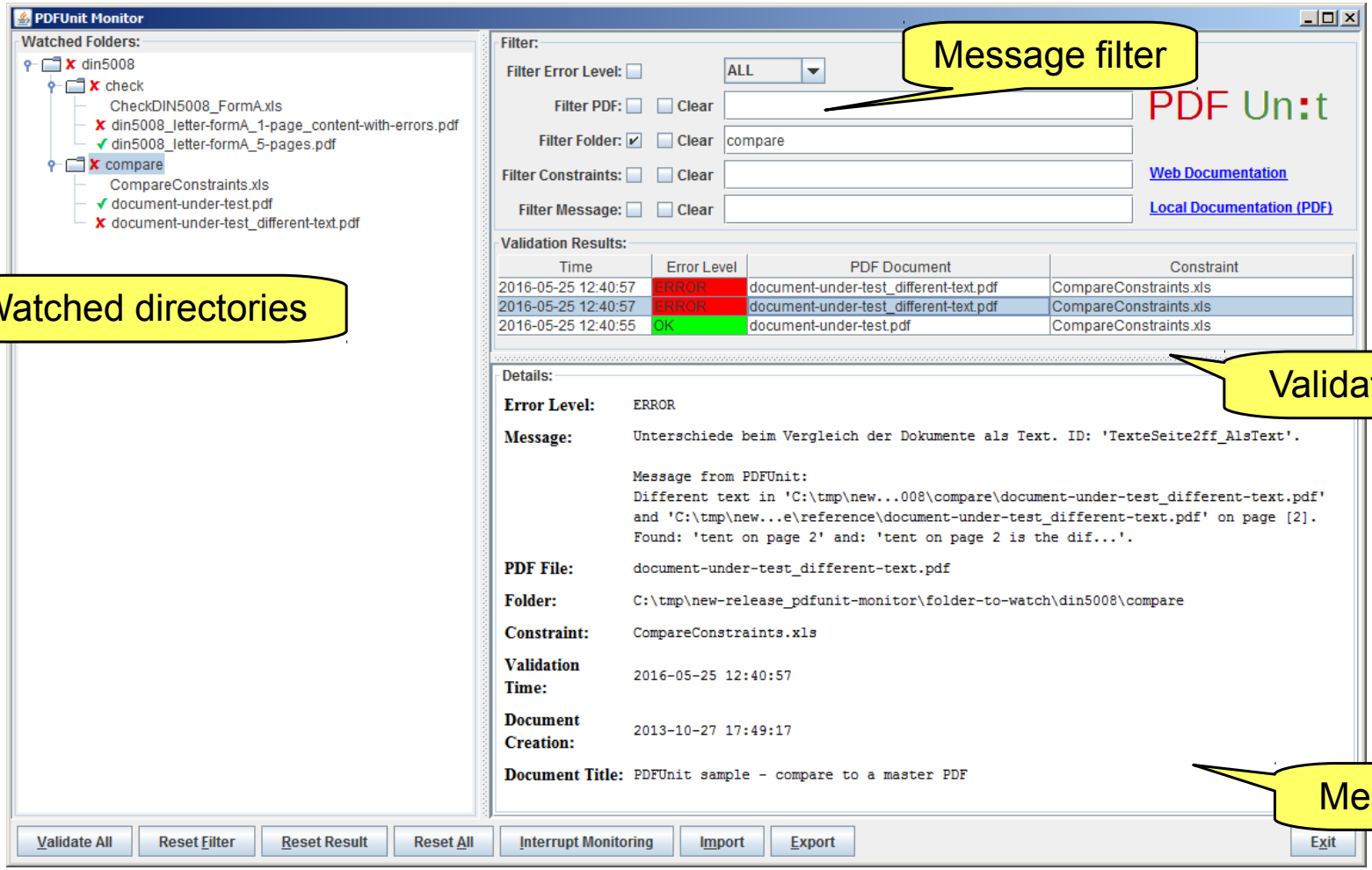
- Special test features:

  - Bar code and
  - QR code inside PDF
  - ZUGFeRD data
  - XFA data
  - XMP data
  - OCR processing of images

- PDF validation against rules written in Excel files

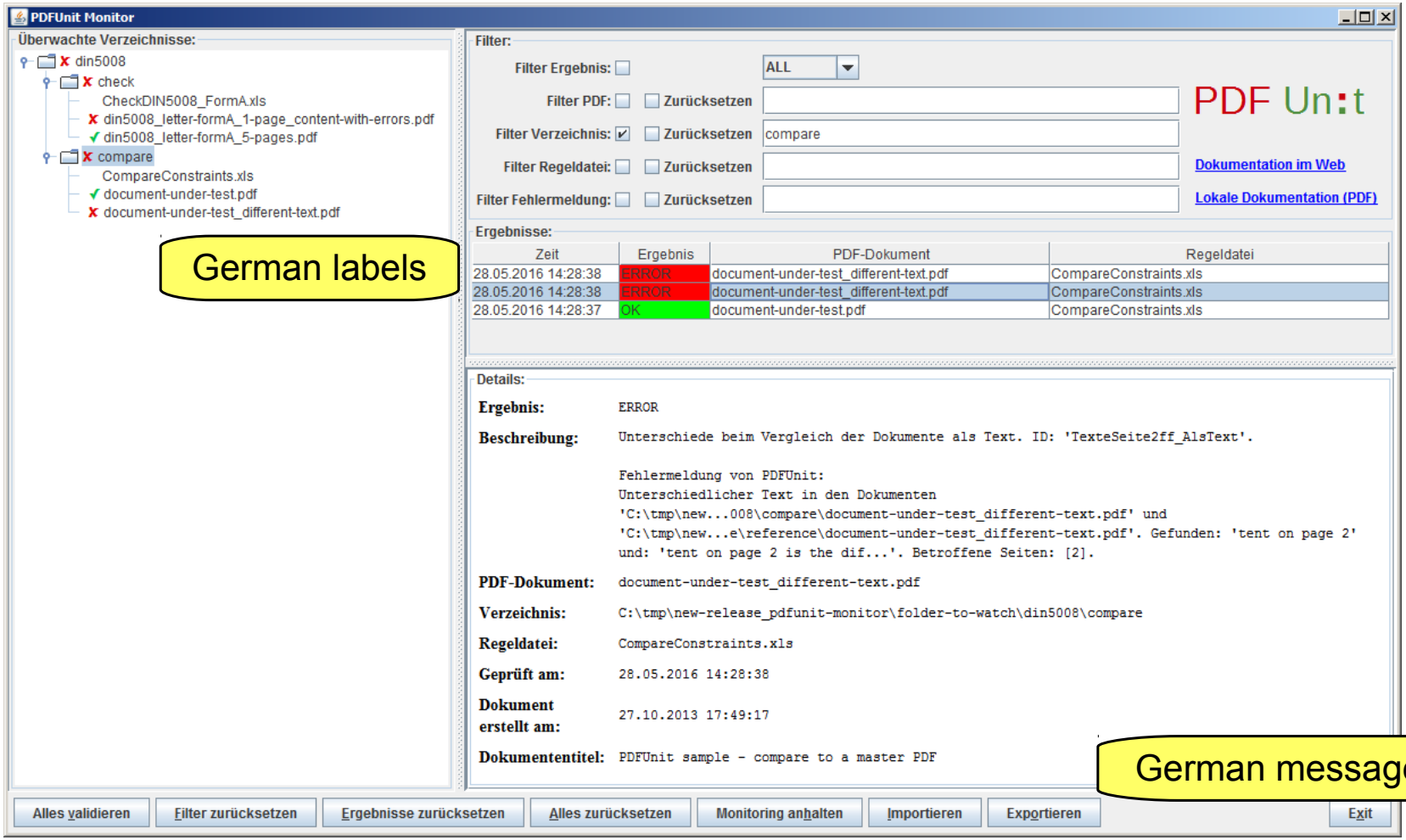
PDFUnit-Monitor - in English

Non-developers can use the PDFUnit-Monitor for PDF tests



PDFUnit-Monitor - in German

User-Interface and error messages are also available in German



Utilities

A couple of utilities are provided to extract data/images from PDF:

- Extract acrofield infos to XML
- Extract bookmarks to XML
- Extract embbedded files
- Extract font infos to XML
- Extract images to PNG
- Extract JavaScript
- Extract named destinations to XML
- Extract signature infos to XML
- Extract XFA-data to XML
- Extract XMP-data to XML
- Extract ZUGFeRD data to XML
- Render PDF pages into PNG
- Render sections of a PDF page into PNG
- Convert any Unicode string into Unicode-Hex-Values (\uXXXX)

Examples Java (I)

```
AssertThat.document(filename)
          .restrictedTo(page2)
          .hasText()
          .first(titleChapter4)
          .then(titleChapter5)
          .then(titleChapter6)
;
```

Validate ordered text

```
String filename = PATH + "content/diverseContentOnMultiplePages.pdf";
String linkToHomepage = "http://pdfunit.com/";
PagesToUse pagesAfter1 = ON_EVERY_PAGE.after(1);
PageRegion headerRegion = createHeaderRegion();

AssertThat.document(filename)
          .restrictedTo(pagesAfter1)
          .restrictedTo(headerRegion)
          .hasText()
          .containing(linkToHomepage)
;
```

Text in specified regions

```
AssertThat.document(filename)
          .hasField("Textfield-1_left-aligned") .withWidth(200, POINTS).withHeight(25, POINTS)
          .hasField("Textfield-2_right-aligned").withWidth(200, POINTS).withHeight(60, POINTS)
          .hasField("Textfield-3_centered")     .withWidth(150, POINTS).withHeight(60, POINTS)
;
```

Form field validation

Examples Java (II)

```
AssertThat.document(filename)
          .restrictedTo(FIRST_PAGE)
          .restrictedTo(pageRegion)
          .hasImage()
          .withBarcode()
          .containing("hello, world")
;
```

Bar code

```
AssertThat.document(filenameTest)
          .and(filenameMaster)
          .haveSameTitle()
          .haveSameSubject()
          .haveSameKeywords()
          .haveSameAuthor()
;
```

Comparing document info

```
AssertThat.document(filename)
          .hasSignatureField("Signature2")
          .signedBy("John B Harris")
          .signedOn(signingDate)
;
```

Test for signatures

```
String filename = PATH + "zugferd10/ZUGFeRD_1p0_BASIC_Einfach.pdf";
String xpathNumberOfTradeItems = "count(//ram:IncludedSupplyChainTradeLineItem) = 1";
XPathExpression exprNumberOfTradeItems = new XPathExpression(xpathNumberOfTradeItems);
AssertThat.document(filename)
          .hasZugferdData()
          .matchingXPath(exprNumberOfTradeItems)
;
```

Validating ZUGFeRD data

Examples .NET, C#

```
[TestMethod]
public void HasAuthor()
{
  String filename = path + "documentInfos/documentInfos_allInfos.pdf";

  AssertThat.document(filename)
            .hasAuthor()
            .matchingComplete("PDFUnit.com")
  ;
}
```

```
[TestMethod]
[ExpectedException(typeof(PDFUnitError))]
public void HasAuthor_NoAuthorInPDF()
{
  String filename = path + "documentInfos/documentInfos_noAuthor.pdf";

  AssertThat.document(filename)
            .hasAuthor()
  ;
}
```

The DLL is created from JAR files

So:
- 100% compatibel with Java
- Method names are in lower case

Examples Perl

```
lives_ok {
    my $pages1To3 = PagesFromTo->spanningFrom(1)->to(3);
    my $textBody = _createBodyRegion();

    my $chapter2Header = "Text Running Over Two Pages";
    my $chapter3Header = "QR code and Text in Images";
    my $chapter2BodyPart =
        "Huck Finn is drawn from life; "
        . "Tom Sawyer also, but not from an "
        . "individual -- he is a combination "
        . "of the characteristics of three boys";

    AssertThat
        ->document($pdfUnderTest)
        ->restrictedTo($pages1To3)
        ->restrictedTo($textBody)
        ->hasText()
            ->first($chapter2Header)
            ->then($chapter2BodyPart)
            ->then($chapter3Header)
        ;
} "validateOrderedTextInPageBody_SpanningOverMultiplePages";
```

Validate ordered text spanning over 3 pages

- Needs PDF::PDFUnit and Inline::Java

100% compatible with Java

Examples XML

```
<testcase name="hasTextOnFirstPage_EndingWith">
  <assertThat testDocument="&pdfdir;/content/documentForTextClipping.pdf">
    <hasText on="FIRST_PAGE" >
      <inRegion upperLeftX="18" upperLeftY="45" width="60" height="9" >
        <endingWith>page.</endingWith>
      </inRegion>
    </hasText>
  </assertThat>
</testcase>
```

Text in a specified region

```
<testcase name="comparePDFWithMasterAsRenderedImages_OnFirstPage">
  <assertThat testDocument="&pdfdir;/master/pdfReference.pdf"
              masterDocument="&pdfdir;/master/pdfUnderTest.pdf"
  >
    <haveSameAppearance on="FIRST_PAGE" />
  </assertThat>
</testcase>
```

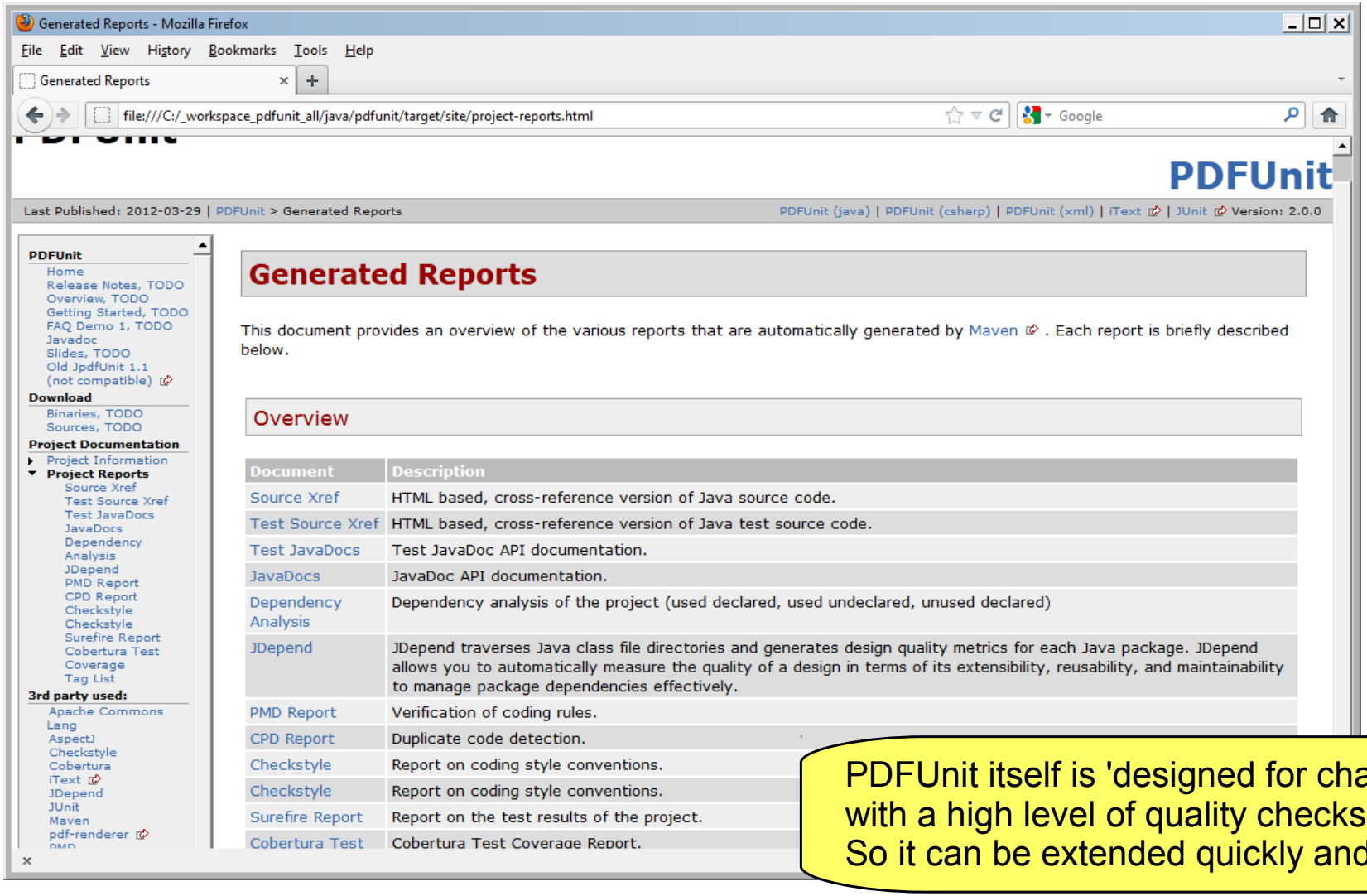
Compare 2 documents as rendered images

- Compatible with the Java-API
- Same reporting as in PDFUnit-Java

History of PDFUnit-Java

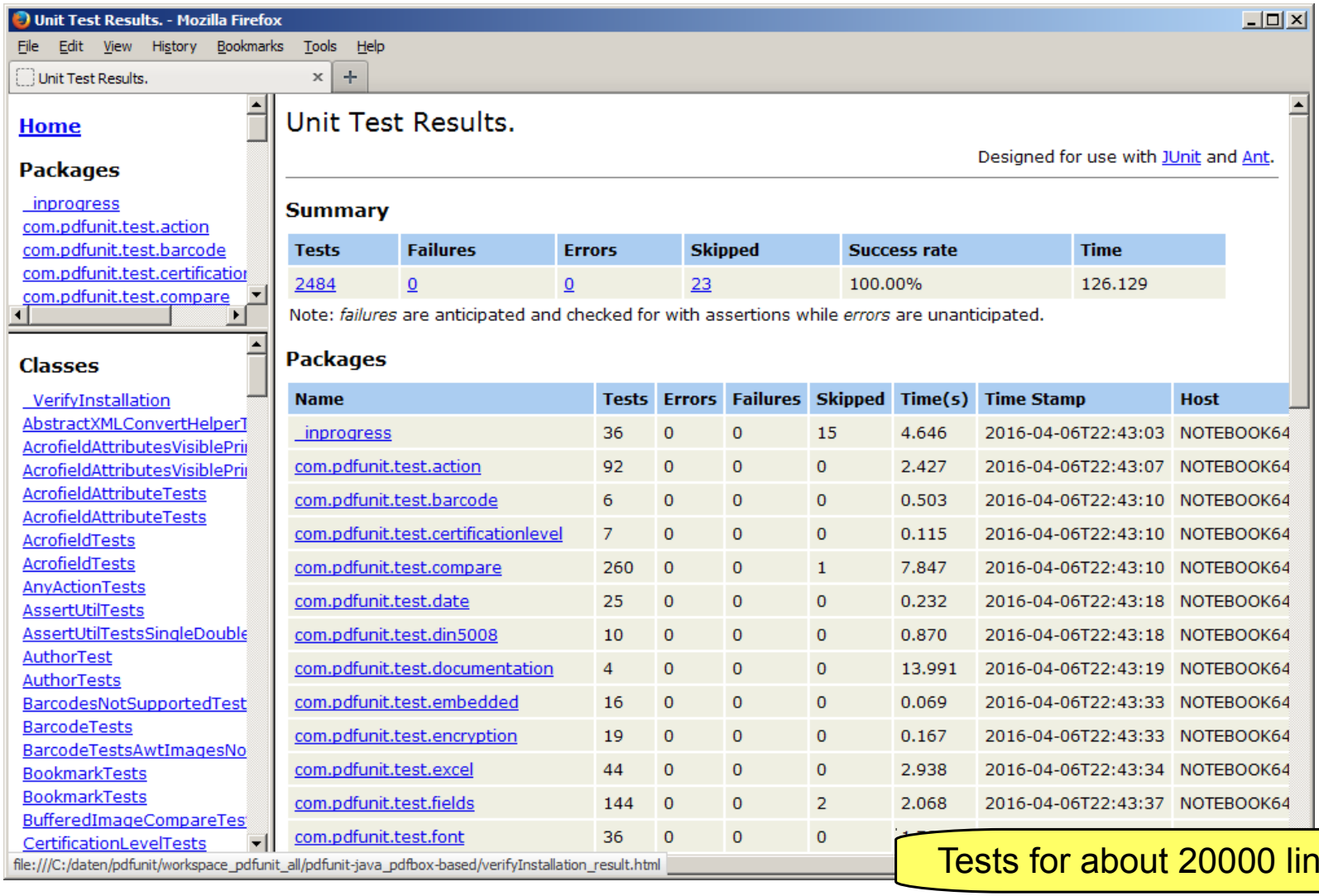
- 01/2011 - Started implementation, using iText as PDF parser.
- 08/2011 - PDFBox dropped off. Total Redesign
- 12/2011 - Proof of concept for an XML-API
- 03/2012 - Proof of concept for a C#-implementation
- 07/2012 - Implementation of the XML-API
- 12/2012 - Finishing the Java-API with a lot of new functions
- 06/2014 - New release with more features and less bugs
- 10/2015 - PDFMonitor added, PDFUnit-Perl released
- 05/2016 - Replaced iText as PDF parser by PDFBox 2.0
  Added features, e.g. OCR, QR, ZUGFeRD, DIN5008

Inside PDFUnit-Java – Analyzing continuously

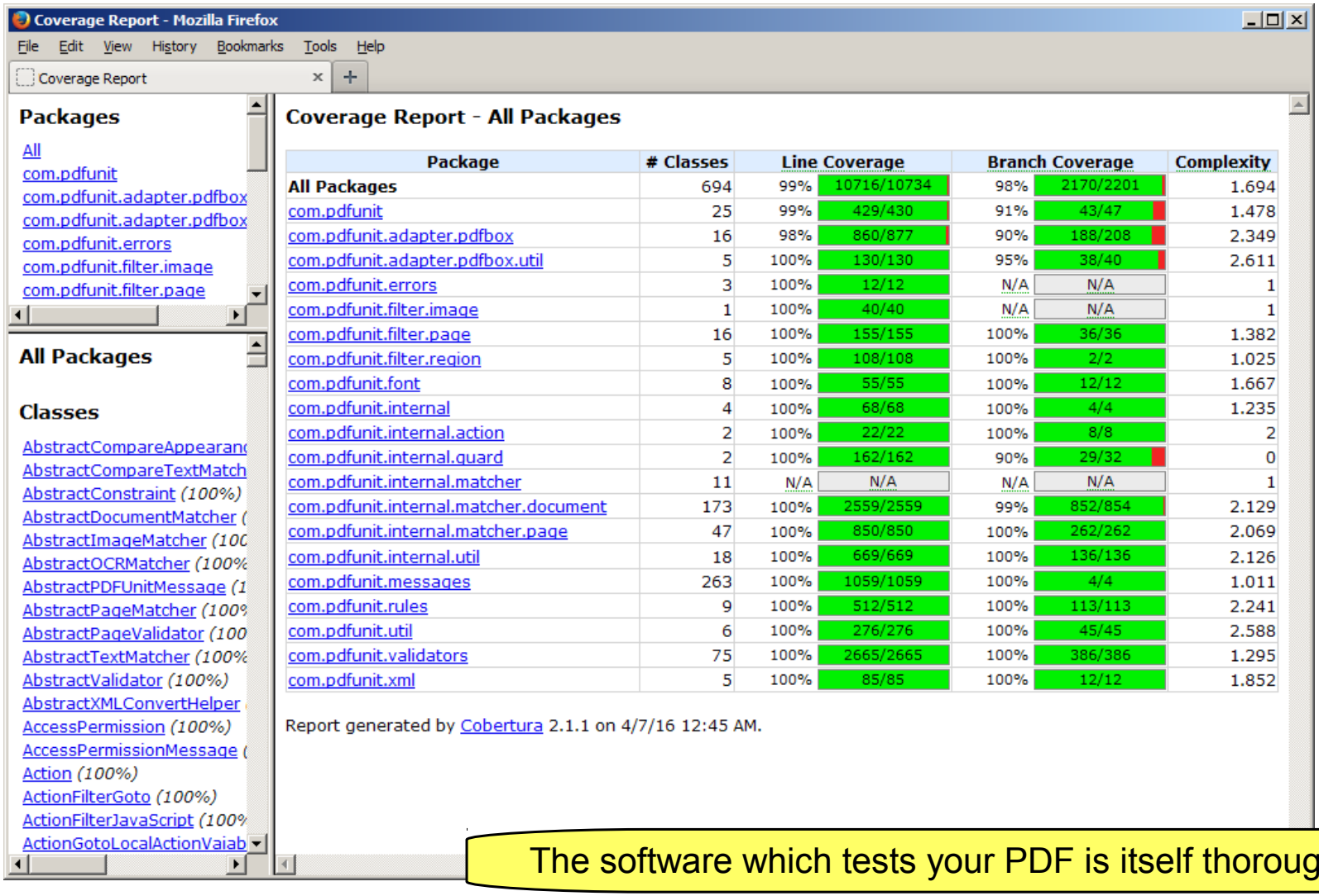


PDFUnit itself is 'designed for change'
with a high level of quality checks.
So it can be extended quickly and safely.

Inside PDFUnit-Java – Many Tests



Inside PDFUnit-Java – Good Test Coverage



The software which tests your PDF is itself thoroughly tested

Inside PDFUnit-Java – Clean Code

DocumentValidator.java

```
297
298  /**
299   * This method verifies that a PDF document has one ore more bookmarks.
300   * It returns a validator class which provides validation methods for bookmarks.
301   */
302  public AllBookmarksValidator hasBookmarks() {
303    AllBookmarksValidator bookmarkValidator = new AllBookmarksValidator(this);
304    bookmarkValidator.hasBookmarks();
305    return bookmarkValidator;
306  }
307
308  /**
309   * This method returns a validator which provides validation methods for dates.
310   */
311  public DateValidator hasCreationDate() {
312    DateValidator validator = new DateValidator(this);
313    validator.hasCreationDate();
314    return validator;
315  }
316
317  /**
318   * This method returns a validator which provides validation methods for the document creator.
319   */
320  public PropertyValidator hasCreator() {
321    PropertyValidator validator = new PropertyValidator(this);
322    validator.hasProperty(PROPERTY_KEY_CREATOR);
323    return validator;
324  }
325
326  /**
327   * This method returns a validator which provides validation methods for embedded files.
328   */
329  public EmbeddedFileValidator hasEmbeddedFile() {
330    EmbeddedFileValidator validator = new EmbeddedFileValidator(this);
331    validator.hasEmbeddedFile();
332    return validator;
333  }
```

Well prepared for extensions and fixes

Contact

- Contact us for further information:

PDFUnit.com
Carsten Siedentop
Leostr. 41
51145 Cologne
Germany

Phone: +49 (0)178 7997141
Mail: info@pdfunit.com

- Website www.pdfunit.com