

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 specified 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

The screenshot shows the PDFUnit Monitor application window. It is divided into several sections:

- Watched Folders:** A tree view on the left showing a folder named 'din5008' containing sub-folders 'check' and 'compare'. The 'check' folder contains 'CheckDIN5008_FormA.xls', 'din5008_letter-formA_1-page_content-with-errors.pdf', and 'din5008_letter-formA_5-pages.pdf'. The 'compare' folder contains 'CompareConstraints.xls', 'document-under-test.pdf', and 'document-under-test_different-text.pdf'.
- Filter:** A section on the right with a dropdown menu set to 'ALL'. Below it are input fields for 'Filter Error Level', 'Filter PDF', 'Filter Folder' (set to 'compare'), 'Filter Constraints', and 'Filter Message', each with a 'Clear' button.
- Validation Results:** A table with columns 'Time', 'Error Level', 'PDF Document', and 'Constraint'.

Time	Error Level	PDF Document	Constraint
2016-05-25 12:40:57	ERROR	document-under-test_different-text.pdf	CompareConstraints.xls
2016-05-25 12:40:57	ERROR	document-under-test_different-text.pdf	CompareConstraints.xls
2016-05-25 12:40:55	OK	document-under-test.pdf	CompareConstraints.xls
- Details:** A section at the bottom showing information for the selected error:
 - Error Level:** ERROR
 - Message:** Unterschiede beim Vergleich der Dokumente als Text. ID: 'TexteSeite2ff_AlsText'.
 - Message from PDFUnit:** Different text in 'C:\tmp\new...008\compare\document-under-test_different-text.pdf' and 'C:\tmp\new...e\reference\document-under-test_different-text.pdf' on page [2]. Found: 'tent on page 2' and: 'tent on page 2 is the dif...'
 - PDF File:** document-under-test_different-text.pdf
 - Folder:** C:\tmp\new-release_pdfunit-monitor\folder-to-watch\din5008\compare
 - Constraint:** CompareConstraints.xls
 - Validation Time:** 2016-05-25 12:40:57
 - Document Creation:** 2013-10-27 17:49:17
 - Document Title:** PDFUnit sample - compare to a master PDF

Callouts in yellow boxes point to: 'Watched directories' (pointing to the folder tree), 'Message filter' (pointing to the Filter section), 'Validation result list' (pointing to the table), and 'Message details' (pointing to the Details section).

PDFUnit-Monitor - in German

User-Interface and error messages are also available in German

German labels

German message details

Filter:

Filter Ergebnis: ALL

Filter PDF: Zurücksetzen

Filter Verzeichnis: Zurücksetzen compare

Filter Regeldatei: Zurücksetzen

Filter Fehlermeldung: Zurücksetzen

Ergebnisse:

Zeit	Ergebnis	PDF-Dokument	Regeldatei
28.05.2016 14:28:38	ERROR	document-under-test_different-text.pdf	CompareConstraints.xls
28.05.2016 14:28:38	ERROR	document-under-test_different-text.pdf	CompareConstraints.xls
28.05.2016 14:28:37	OK	document-under-test.pdf	CompareConstraints.xls

Details:

Ergebnis: ERROR

Beschreibung: Unterschiede beim Vergleich der Dokumente als Text. ID: 'TexteSeite2ff_AlsText'.

Fehlermeldung von PDFUnit:
 Unterschiedlicher Text in den Dokumenten
 'C:\tmp\new...008\compare\document-under-test_different-text.pdf' und
 'C:\tmp\new...e\reference\document-under-test_different-text.pdf'. Gefunden: 'tent on page 2'
 und: 'tent on page 2 is the dif...'. Betroffene Seiten: [2].

PDF-Dokument: document-under-test_different-text.pdf

Verzeichnis: C:\tmp\new-release_pdfunit-monitor\folder-to-watch\din5008\compare

Regeldatei: CompareConstraints.xls

Geprüft am: 28.05.2016 14:28:38

Dokument erstellt am: 27.10.2013 17:49:17

Dokumententitel: PDFUnit sample - compare to a master PDF

Buttons: Alles validieren, Filter zurücksetzen, Ergebnisse zurücksetzen, Alles zurücksetzen, Monitoring anhalten, Importieren, Exportieren, Exit

Utilities

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

- Extract acrofield infos to XML
- Extract bookmarks to XML
- Extract embedded 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(filename)
    .hasSignatureField("Signature2")
    .signedBy("John B Harris")
    .signedOn(signingDate)
;
```

Test for signatures

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

Comparing document info

```
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")
;
}
```

The DLL is created from JAR files

So:

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

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

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

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";
```

- Needs PDF::PDFUnit and Inline::Java

Validate ordered text
spanning over 3 pages

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>
```

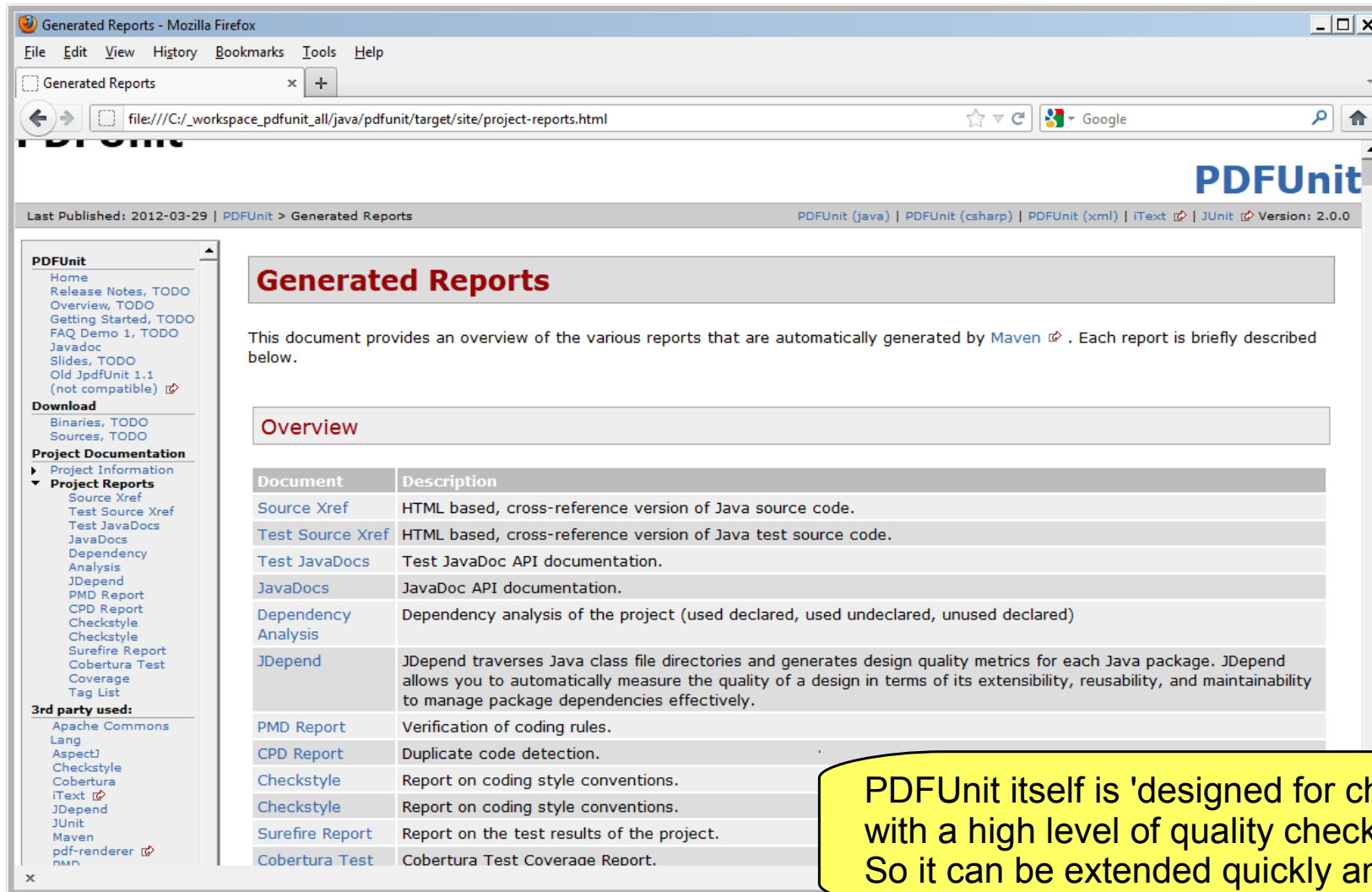
Compare 2 documents as rendered images

- 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



Generated Reports - Mozilla Firefox

file:///C:/_workspace_pdfunit_all/java/pdfunit/target/site/project-reports.html

PDFUnit

Last Published: 2012-03-29 | PDFUnit > Generated Reports

PDFUnit (java) | PDFUnit (csharp) | PDFUnit (xml) | iText | JUnit | Version: 2.0.0

Generated Reports

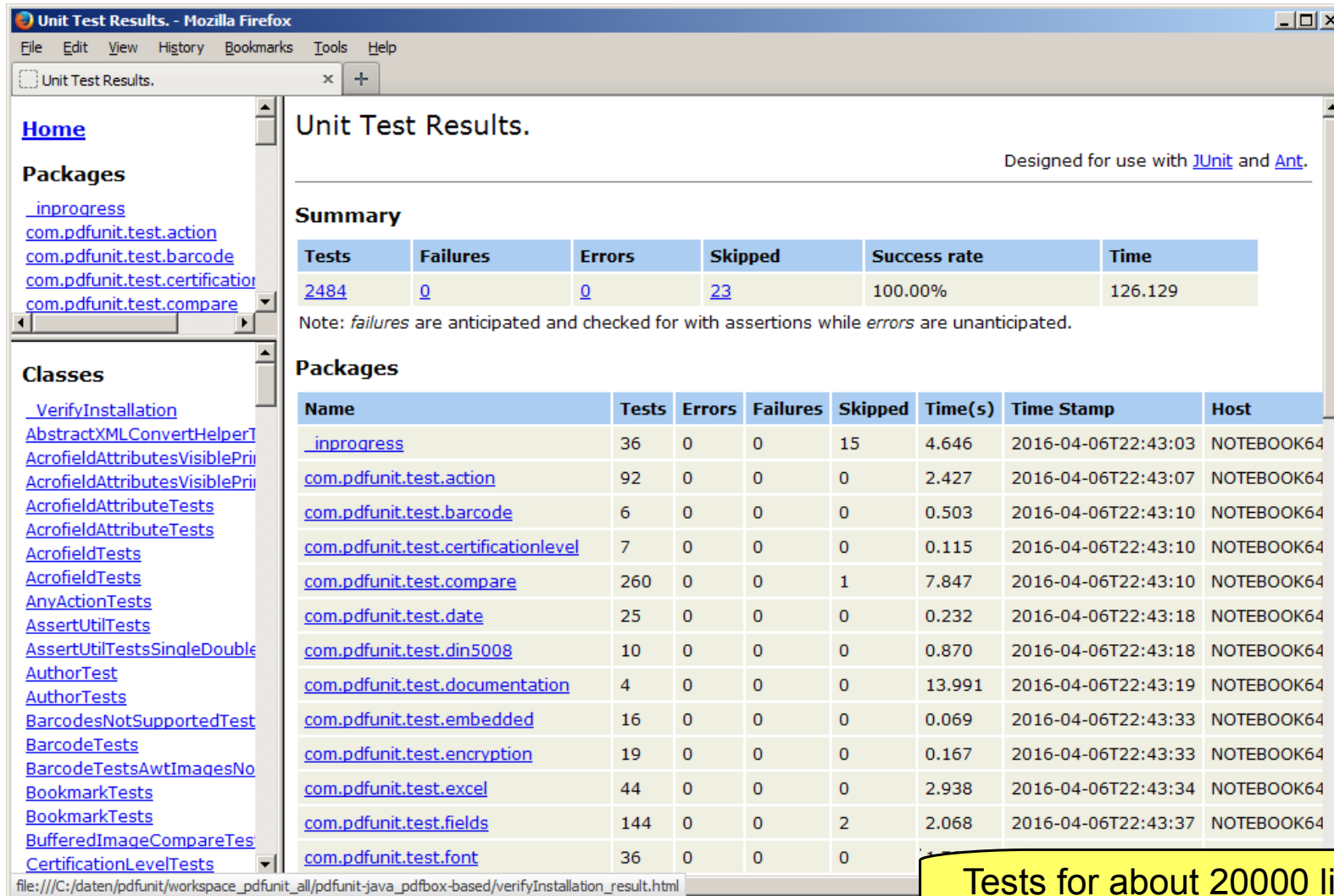
This document provides an overview of the various reports that are automatically generated by [Maven](#). Each report is briefly described below.

Overview

Document	Description
Source Xref	HTML based, cross-reference version of Java source code.
Test Source Xref	HTML based, cross-reference version of Java test source code.
Test JavaDocs	Test JavaDoc API documentation.
JavaDocs	JavaDoc API documentation.
Dependency Analysis	Dependency analysis of the project (used declared, used undeclared, unused declared)
JDepend	JDepend traverses Java class file directories and generates design quality metrics for each Java package. JDepend allows you to automatically measure the quality of a design in terms of its extensibility, reusability, and maintainability to manage package dependencies effectively.
PMD Report	Verification of coding rules.
CPD Report	Duplicate code detection.
Checkstyle	Report on coding style conventions.
Checkstyle	Report on coding style conventions.
Surefire Report	Report on the test results of the project.
Cobertura Test	Cobertura Test Coverage Report.

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



Unit Test Results. Designed for use with [JUnit](#) and [Ant](#).

Summary

Tests	Failures	Errors	Skipped	Success rate	Time
2484	0	0	23	100.00%	126.129

Note: *failures* are anticipated and checked for with assertions while *errors* are unanticipated.

Packages

Name	Tests	Errors	Failures	Skipped	Time(s)	Time Stamp	Host
_inprogress	36	0	0	15	4.646	2016-04-06T22:43:03	NOTEBOOK64
com.pdfunit.test.action	92	0	0	0	2.427	2016-04-06T22:43:07	NOTEBOOK64
com.pdfunit.test.barcode	6	0	0	0	0.503	2016-04-06T22:43:10	NOTEBOOK64
com.pdfunit.test.certificationlevel	7	0	0	0	0.115	2016-04-06T22:43:10	NOTEBOOK64
com.pdfunit.test.compare	260	0	0	1	7.847	2016-04-06T22:43:10	NOTEBOOK64
com.pdfunit.test.date	25	0	0	0	0.232	2016-04-06T22:43:18	NOTEBOOK64
com.pdfunit.test.din5008	10	0	0	0	0.870	2016-04-06T22:43:18	NOTEBOOK64
com.pdfunit.test.documentation	4	0	0	0	13.991	2016-04-06T22:43:19	NOTEBOOK64
com.pdfunit.test.embedded	16	0	0	0	0.069	2016-04-06T22:43:33	NOTEBOOK64
com.pdfunit.test.encryption	19	0	0	0	0.167	2016-04-06T22:43:33	NOTEBOOK64
com.pdfunit.test.excel	44	0	0	0	2.938	2016-04-06T22:43:34	NOTEBOOK64
com.pdfunit.test.fields	144	0	0	2	2.068	2016-04-06T22:43:37	NOTEBOOK64
com.pdfunit.test.font	36	0	0	0			

file:///C:/daten/pdfunit/workspace_pdfunit_all/pdfunit-java_pdfbox-based/verifyInstallation_result.html

Tests for about 20000 lines of code.

Inside PDFUnit-Java – Good Test Coverage

The screenshot shows a 'Coverage Report - All Packages' window. It features a table with columns for Package, # Classes, Line Coverage, Branch Coverage, and Complexity. The 'All Packages' row shows 99% line coverage (10716/10734) and 98% branch coverage (2170/2201). Individual packages like 'com.pdfunit' and 'com.pdfunit.adapter.pdfbox' also show high coverage. A yellow callout box at the bottom states: 'The software which tests your PDF is itself thoroughly tested'.

Package	# Classes	Line Coverage	Branch Coverage	Complexity
All Packages	694	99% 10716/10734	98% 2170/2201	1.694
com.pdfunit	25	99% 429/430	91% 43/47	1.478
com.pdfunit.adapter.pdfbox	16	98% 860/877	90% 188/208	2.349
com.pdfunit.adapter.pdfbox.util	5	100% 130/130	95% 38/40	2.611
com.pdfunit.errors	3	100% 12/12	N/A N/A	1
com.pdfunit.filter.image	1	100% 40/40	N/A N/A	1
com.pdfunit.filter.page	16	100% 155/155	100% 36/36	1.382
com.pdfunit.filter.region	5	100% 108/108	100% 2/2	1.025
com.pdfunit.font	8	100% 55/55	100% 12/12	1.667
com.pdfunit.internal	4	100% 68/68	100% 4/4	1.235
com.pdfunit.internal.action	2	100% 22/22	100% 8/8	2
com.pdfunit.internal.guard	2	100% 162/162	90% 29/32	0
com.pdfunit.internal.matcher	11	N/A N/A	N/A N/A	1
com.pdfunit.internal.matcher.document	173	100% 2559/2559	99% 852/854	2.129
com.pdfunit.internal.matcher.page	47	100% 850/850	100% 262/262	2.069
com.pdfunit.internal.util	18	100% 669/669	100% 136/136	2.126
com.pdfunit.messages	263	100% 1059/1059	100% 4/4	1.011
com.pdfunit.rules	9	100% 512/512	100% 113/113	2.241
com.pdfunit.util	6	100% 276/276	100% 45/45	2.588
com.pdfunit.validators	75	100% 2665/2665	100% 386/386	1.295
com.pdfunit.xml	5	100% 85/85	100% 12/12	1.852

Report generated by Cobertura 2.1.1 on 4/7/16 12:45 AM.

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