PDF-KungFoo Workshop I

bei der ”Ubucon 2013” (Ubuntu Community Event in Heidelberg)

Kurt Pfeifle und Sven Guckes

PDF-KungFoo Workshop I

bei der ”Ubucon 2013” (Ubuntu Community Event in Heidelberg)

Kurt Pfeifle und Sven Guckes

Dieses Buch können Sie hier kaufen http://leanpub.com/pdfkungfoo-ws1-deu

Diese Version wurde auf 2013-10-24 veröffentlicht

Das ist ein Leanpub-Buch. Leanpub bietet Autoren und Verlagen mit Hilfe des Lean-Publishing-Prozesses ganz neue Möglichkeiten des Publizierens. Lean Publishing bedeutet die permanente, iterative Veröffentlichung neuer Beta-Versionen eines E-Books unter der Zuhilfenahme schlanker Werkzeuge. Das Feedback der Erstleser hilft dem Autor bei der Finalisierung und der anschließenden Vermarktung des Buches. Lean Publishing unterstützt de Autor darin ein Buch zu schreiben, das auch gelesen wird.

Twittern Sie dieses Buch!

Bitte unterstützen Sie Kurt Pfeifle und Sven Guckes, indem Sie dieses Buch auf Twitter weiterempfehlen!

Hier ein Vorschlag für einen Tweet:

Habe gerade das Protokoll des #ubucon13 Workshops zu #pdfkungfoo als E-Book gekauft (http://leanpub.com/pdfkungfoo-ws1-deu). Bin gespannt auf den Inhalt...

Vorschlag: Verwenden Sie den folgenden Hashtag, wenn Sie über dieses Buch twittern: #pdfkungfoo.

Was sagen Andere über dieses Buch? Klicken Sie hier, um nach diesem Hashtag auf Twitter zu suchen:

https://twitter.com/search?q=#pdfkungfoo

Ebenfalls von Kurt Pfeifle

PDF-KungFoo with Ghostscript & Co.

Inhaltsverzeichnis

Einführung

Dieses Dokument ist gemeinschaftlich erstellt. Es stellt das Protokoll eines Workshops zum Thema *"PDF-KungFoo mit Ghostscript & Co."* dar.

Die Arbeit am Protokoll erfolgte während des Workshop in "Echtzeit", mit Hilfe eines *TitanPads*[1]. In der Woche nach dem Workshop fanden weitere Ergänzungen des Protokolls statt.

Ein Teil der Teilnehmer bei der Ubucon 2013 in Heidelberg.

Lizenz

Es wird als eBook veröffentlicht in den Formaten PDF, MOBI und EPUB. Man kann es über Leanpub[3] erwerben, und zwar wahlweise…

- …zu einem Mindestpreis von 0,00 $US (also kostenlos),
- …zum (empfohlenen) voreingestellten Preis von 4,99 $US,
- …oder zu einem (selbstgewählten) beliebigen höheren Preis.

Um einen anderen als den voreingestellten Preis zu bezahlen, genügt es, den Schieberegler auf der Verkaufsseite[4] auf den gewünschten Wert zu stellen.

> **Alle Autoren-Honorare aus den Verkäufen dieses eBooks gehen zu 100% direkt an die *Electronic Frontier Foundation (EFF)***[a]. Das Autorenhonorar aus jedem Leanpub-Verkauf beträgt 90% des Verkaufs-Erlöses, minus 50 Cent pro Buch. Das ist höher als bei jedem anderen Publisher! Bei einem Kaufpreis von 5.- $US erhält die EFF somit 4.- $US (80% des Erlöses), bei einem Kaufpreis von 10.- $US erhält sie 8,50 $US (85% des Erlöses).
>
> [a] http://www.eff.org/

Die Lizenz dieses Dokuments erlaubt es u.a., Kopien beliebig oft und kostenlos weiterzugeben. Machen Sie Gebrauch davon! Aber wir bitten Sie dennoch: falls Sie dieses Buch kostenlos erhalten haben, ziehen Sie bitte in Betracht, eine weitere Kopie zu kaufen und dadurch eine entsprechende Spende an die Electronic Frontier Foundation zu bewirken!

[1] http://titanpad.com/
[2] http://creativecommons.org/licenses/by-nc-sa/3.0/
[3] http://leanpub.com/
[4]

Ubucon 2013

Der in diesem Dokument protokollierte Workshop fand im Rahmen der *Ubucon 2013*[5] in Heidelberg statt. Die Ubucon ist das zentrale Event aller *Ubuntu-Benutzer in Deutschland. Workshop-Leiter war Kurt Pfeifle.

Event-Details:

- Ubuntu Conference Ubucon 2013
- 2013-10-11-13 Freitag - Sonntag

Veranstaltungs-Ort:

- SRH Berufsförderungswerk Heidelberg
- Bonhoefferstr. 6, 69123 Heidelberg
- Veranstaltungs-Ort in OpenStreetMap[6]

Fahrplan der Ubucon 2013

Tag	Uhrzeit	Was?
Fri 2013-10-11	14:00-16:00	Aufbau,
	16:00-16:30	Anmeldung
	16:30-17:00	Eroeffnung
	17:00-18:00	Vortraege
	18:00-10:00	Stadtfuehrung
	19:00-23:00	Social Event (im Schnookeloch)
Sat 2013-10-12	09:00-10:00	Anmeldung
	10:00-13:00	Vortraege
	13:00-14:00	PAUSE
	14:00-17:00	Vortraege
	17:00-19:00	Linux Jeopardy (Quiz)
	19:00-20:00	Stadtfuehrung
	20:00-23:00	Social Event (im Cafe Villa)
Sun 2013-10-13	09:00-10:00	Anmeldung
	10:00-12:00	Vortraege
	12:00-13:00	PAUSE
	13:00-15:00	Vortraege
	15:00-15:30	Verabschiedung
	15:30-18:00	Abbau und Aufraeumen
Tag	**Uhrzeit**	**Was?**

hier: Workshop zu “PDF-KungFoo mit Ghostscript & Co.”

- Raum: Precise
- Zeit: Samstag, 11-13 h
- Referenten: Kurt Pfeifle + Sven Guckes

[5] http://ubucon.de/2013/programm/
[6] http://osm.org/go/0DwYPO22k--?m=

Hinweise zum Workshop

Wer die im Workshop vorgeführten Kommandos nachvollziehen möchte, benötigt die folgenden Kommandozeilen-Tools. Diese sind ausnahmslos für alle Linux-Distributionen verfügbar (ebenso für Mac OS X und für Windows):

- Ghostscript, `gs` oder `gsx`. Auf Windows: `gswin32c.exe` oder `gswin64c.exe`.
- `pdfinfo`
- `pdffonts` (die neueste Poppler-Version, nicht die XPDF-Version!)
- `pdfimages` (die neueste Poppler-Version, nicht die XPDF-Version!)
- `pdfid.py` (von Didier Stevens, ein Python-Skript, Link siehe unten)
- `pdf-parser.py` (von Didier Stevens, ein Python-Skript, Link siehe unten)
- `mutool` (von MuPDF)
- `fontforge`
- `compare` (von ImageMagick)

Um diese Tools zu installieren, bitte in der Konsole folgendes Kommando ausführen:

```
1  sudo apt-get update;
2  sudo apt-get install fontforge ghostscript poppler-utils mupdf
```

Die Python-Skripte gibt es bei blog.didierstevens.com[7].

Workshop-Beschreibung, wie im 'Call for Papers' eingereicht:

"Der Workshop demonstriert einige der Top-10-Probleme (Reihung nach der subjektiven Erfahrung des Workshop-Leiters), die bei der Verarbeitung oder Erstellung von PDF-Dateien in der Praxis auftreten können. Dazu gehören u.a. Darstellung von Schriften auf dem Bildschirm oder im Druckbild, Darstellung von transparenten Grafik-Elementen im Druckbild, Extraktion von Text-Stellen oder ganzen Texten, Extraktion von Bildern, Reduzierung der Dateigröße und vieles mehr. Neben Ghostscript werden noch viele andere PDF-Werkzeuge vorgestellt."

"Der Workshop kommt ohne viele Folien aus, dafür werden alle Beispiele live vorgeführt und die Teilnehmer können auf den eigenen Notebooks mitmachen."

"Zusätzlich werden alle Ergebnisse des Workshops gemeinsam von den Teilnehmer auf einem Titanpad bearbeitet und sollen noch vor Ort auf Leanpup hochgeladen werden."

Zielgruppe des Workshops

Alle Computerbenutzer, denen immer wieder mal PDFs in die Haende fallen. Und jene, die im Vortrag gerne Fragen stellen und auch mal den einen oder anderen Kommentar am liebsten direkt abgeben wuerden oder auch 'nen Link wissen moechte...

[7] http://blog.didierstevens.com/programs/pdf-tools/

Folgende Teilnehmer haben per TitanPad zum Protokoll des Workshops beigetragen:

- Kurt Pfeifle kurt.pfeifle@gmail.com
- Sven Guckes ubucon2013@guckes.net
- Max Schneller cicloalpin@gmail.com
- Torsten Scheck torsten@gmx.org
- Florian Heß <–>

I Workshop-Themen

1 Workshop-Thema (Theorie): Innere Struktur des PDF-Formats

(Hier fängt das Protokoll an….)

Kurt gibt eine Kurz-Einführung in die Syntax von PDFs. (Achtung: diese Darstellung ist äußerst stark vereinfacht und verkürzt. Sie soll lediglich dazu dienen, sich auf die Schnelle orientieren zu können, wenn man eine PDF im Text-Editor öffnet….)

PDFs bestehen im einfachsten Fall aus genau 4 Teilen:

1. Header
2. Objekte
3. `xref`-Tabelle
4. `trailer`

Diese 4 Teile werden in den nächsten Abschnitten etwas genauer erläutert.

(Falls man mit PDFs zu tun kriegt, die *'web-optimiert'* (a.k.a. *'linearisiert'*) sind, oder die ein '*incremental update'* hinter sich haben, reicht diese einfache Struktur nicht mehr!)

1.1 Header

Der *Header* besteht aus genau 2 Zeilen:

1. `PDF-1.x`
2. in der 2. Zeile aus 4 binären Bytes (nicht-ASCII-Zeichen), welche signalisieren sollen, dass PDF kein ASCII-Format ist.

Die erste Zeile stellt die PDF-Version dar. Zulässig sind derzeit die Strings `PDF-1.0`, `PDF-1.1`, … `PDF-1.7`. Es fing vor 20 Jahren an mit `PDF-1.0`. Damals war PDF noch ein rein proprietäres Format. Allerdings hat Adobe von Anfang an die genaue Spezifikation des PDF-Formats veröffentlicht, und jedermann freigestellt, Software zu implementieren, die PDF lesen, darstellen oder generieren kann.

Die Version `PDF-1.7` wurde an die ISO-Standardisierungs-Organisation eingereicht. PDF-1.7 ist von der ISO mit minimalen Änderungen als ISO-Standard 32000-1 veröffentlicht worden. Damit hat Adobe die Kontrolle über die weitere Entwicklung von PDF aufgegeben. Ein ISO-Kommittee arbeitet derzeit an ISO 32000-2, welches demnächst zur Freigabe der Spezifikation *PDF-2.0* führen soll.

1.2 PDF-Objekte

Alle Objekte werden in folgenden ASCII-Strings eingeklammert: `N M obj` und `endobj`.

Dabei sind `N` und `M` ganze Zahlen. `N` stellt die Nummer des Objekts dar. `M` repräsentiert die *Generation* des Objekts. Im Normalfall ist `M` aller-aller-allermeistens `0`.

Die Objekt-Nummern sind in der Praxis nie fortlaufend durchnummeriert. Allerdings darf pro PDF-Datei jede Objekt-Nummer höchstens 1mal vorkommen. Die Nummern müssen nicht unbedingt lückenlos sein, und auch nicht aufsteigend.

Ein Objekt kann einen *Stream* enthalten. Dieser kann komprimiert sein, und er kann deshalb auch durch binäre, nicht-ASCII-Zeichen repräsentiert werden.

Streams werden ebenfalls durch zwei ASCII-Strings eingeklammert: `stream` und `endstream`.

Hier ist ein Beispiel-Objekt *ohne* Stream:

```
1 0 obj
<<
 /Type /Catalog
 /Outlines 2 0 R
 /Pages 3 0 R
>>
endobj
```

Hier ist ein Beispiel-Objekt *mit* Stream (unkomprimierter Stream):

```
5 0 obj
<< /Length 48 >>
  stream
    BT
      /F1 24 Tf
      100 700 Td
      (Hello World)Tj
    ET
  endstream
endobj
```

Wenn im Objekt ein Stream vorkommt, muß seine Größe als `/Length` angegeben sein. Diese ‘Länge’ ist eine Zahl, welche die Anzahl der Bytes im Streams darstellt.

Hier ist ein Beispiel-Objekt (unvollständig) mit *komprimiertem* Stream:

```
15 0 obj
<<
  /Length1 778552
  /Length 1581435
  /Filter /ASCIIHexDecode
>>
  stream
    000100000018010000040080445349470 31a0916000bc9bc0000177c47444546
    89f98d49000aff24000002c247504f537b56ab9f000b01e80000ac1a47535542
    de426051000bae0400001b984a5354466d2a6906000bc99c0000001e4c545348
    [...gekuerzt...]
    c7c256ede5a5672d87a7d212532c8efc3d587763533d1cb729157efb1376d605
    e5acf05f3ad6e1379020d1c802d386e0b5328363c8e9fedf28401583f40ee35d
```

```
    bb5ca148ccc9d6ab38313b7038564332acd1a42cfa3b0000>
  endstream
endobj
```

Wenn ein Stream komprimiert ist, muß seine (De-)Kompressions-Methode angeben sein. Dies taucht in dem Wort `/Filter` samt dem darauffolgenden Wert auf (im vorliegenden Fall `/ASCIIHexDecode`):

(**Anmerkung**: eine ASCII-Hex-Codierung führt zu einer Darstellung, welche alle binären Zeichen in reines ASCII übersetzt hat. Die Anwendung der De-codierung `/ASCIIHexDecode` führt zurück zur rein binären Darstellung…)

1.3 Objekt-Verweise

Objekte können andere Objekte enthalten. In diesem Fall tritt die Syntax wie folgt auf:

```
N M R
```

wobei `N` und `M` ganze Zahlen sind: `N` ist wieder die Objekt-Nummer ist, `M` die entsprechende Generation (in der Praxis fast immer `0`) und der Buchstabe `R` steht für 'Referenz'.

Will man den Inhalt der Referenz `3 0 R` kennen lernen, muss man das Objekt Nr. `3` anschauen: also alles, was in `3 0 obj [....] endobj` vorkommt.

1.4 Oben verwendete Syntax-Elemente

Die obigen Beispiele verwenden die folgenden Syntax-Elemente (hauptsächlich im *Content Stream*):

<< … >>

> *Spitze* Klammern enthalten ein "Dictionary". Ein Dictionary definiert Paare von Variablen-Namen und Variablen-Werten. (Dictionaries können verschachtelt sein, denn Variablen-Werte können auch weitere Dictionaries enthalten.)

(…)

> Auf der Seite darzustellende Texte und Buchstaben stehen in *runden* Klammern, sofern sie als *literale* ASCII-Zeichen dargestellt werden. (Es gibt auch die Möglichkeit, Text-Teile als Hex- oder Oktal-Daten darzustellen.)

[…]

> In *eckigen* Klammern stehen *Arrays*. Arrays sind eine Sammlung anderer Elemente. (Die Koordinaten der Eckpunkte für die Seitengröße werden beispielsweise in Form eines Arrays angegeben.)

Tabelle weiterer verwendeter Syntax-Elemente:

Operator	PostScript-Äquivalent	Beschreibung
`Tj`	***show***	Zeige Text
`Td`	(keines direkt, ähnlich zu ***moveto**)	Bewege Text-Position
`Tf`	***selectfont***	Setze Font
`BT`	(keines)	Beginne Text-Objekt
`ET`	(keines)	Beende Text-Objekt

Für eine vollständige 'Übersetzungs-Tabelle' von PostScript- nach PDF-Syntax siehe Abschnitt *"Annex A (informative): Operator Summary"* der ISO-PDF-Spezifikation[1].

Weitere Hinweise:

- Die Zeile `/F1 24 Tf` setzt den Font mit dem symbolischen Namen `/F1` auf eine Größe von 24 Punkt und verwendet ihn ab sofort bis auf weiteres. Welcher Font konkret den `/F1` darstellt, ist anderweitig in der PDF definiert.
- Die Zeile `/Outlines 2 0 R` besagt, dass im Objekt Nr. 2 weitere Einzelheiten über *Outlines* definiert sind. (Outlines sind ein optionaler Bestandteil von PDFs. Manchmal werden diese auch als *Bookmarks* bezeichnet. Sie stellen eine Inhaltsverzeichnis-artige Liste von Links dar: beim Klick auf einen Link springt der Viewer an die entsprechende Seite im PDF.)
- Die Zeile `/Pages 3 0 R` besagt, dass das Verzeichnis aller PDF-Seiten im Objekt Nr. 3 nachzuschlagen ist.

1.5 `xref`-Tabelle

Diese Tabelle stellt ein Inhalts-Verzeichnis aller PDF-Objekte dar. Die erste Spalte enthält eine Zahl, welche den *Byte-Offset* darstellt, wo man dieses Objekt innerhalb der PDF-Datei findet.

Daher muss jeder PDF-Reader ein Standard-PDF "von hinten" lesen. Nur durch Auswerten der `xref`-Tabelle (und die befindet sich nun mal am Ende der PDF-Datei) kann er die anderen Objekte finden. Daher muss ein Standard-PDF im Web komplett heruntergeladen sein, bevor der Viewer auch nur die erste Seite anzeigen kann.

Erst die Erweiterung für sogenannte *linearisierte* [a.k.a. *web-optimierte*] PDFs hat diese Beschränkung beseitigt (erstmal in Version PDF-1.2). Linearisierte PDFs verlegen einen Teil der xref-Tabelle an den Anfang der Datei. Der Web-Client kann dadurch bereits die 1. Seite anzeigen, bevor die komplette PDF heruntergeladen ist, und er kann auch mit geringer Verzögerung zu anderen Seiten springen, bevor die restlichen Seiten angekommen sind.

Linearisierte PDFs sind geringfügig grösser als un-linearisierte Pendants; sie rendern jedoch "gefühlt schneller". Allerdings ist ihre Struktur deutlich komplexer zu verstehen.

Wenn man eine PDF (händisch oder programmatisch) editiert, und dabei Bytes löscht oder hinzufügt, muss man die `xref`-Tabelle entsprechend anpassen.

Hier ist das Beispiel einer `xref`-Tabelle, die 8 Objekte verzeichnet:

[1] http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/PDF32000_2008.pdf#page=651

```
xref
0 8
0000000000 65535 f
0000000012 00000 n
0000000089 00000 n
0000000145 00000 n
0000000214 00000 n
0000000381 00000 n
0000000485 00000 n
0000000518 00000 n
```

1.6 Trailer

Zuletzt noch das Beispiel eines Trailers:

```
trailer
<<
  /Size 8
  /Root 7 0 R
>>
startxref
642
%%EOF
```

Der Trailer enthält als wichtigste Information den Byte-Offset, wo man die `xref`-Tabelle findet: und zwar (im obigen Beispiel) als `642` per `startxref`. Auch der Trail benennt (nochmals) die Anzahl der Objekte im PDF (als `/Size 8` – eigentlich benennt er die Anzahl der Elemente, die in der `xref`-Tabelle verzeichnet sind…. Denn in 'weniger einfachen' PDFs können mehrere Trailer und `xref`-Tabellen vorkommen – diese behandeln wir hier aber nicht.)

Der Trailer enthält außerdem noch den Verweis auf das sogennannte `/Root`-Objekt. Im obigen Fall ist es das Objekt Nr. 7. Im Root-Objekt (welches als `/Type /Catalog` firmiert) findet man den Ausgangspunkt für den Seiten-Baum des Dokuments (in obigem Beispiel als Verweis auf `/Pages 3 0 R`, welches auf das Objekt Nr. 3 verweist).

1.7 *'Hello World'*-PDF

Hier ist der vollständige Quellcode eines einfachen, 1-seitigen PDFs:

```
%PDF-1.1

7 0 obj
<<
  /Type /Font
  /Subtype /Type1
  /Name /F1
  /BaseFont /Helvetica
  /Encoding /MacRomanEncoding
>>
endobj

6 0 obj
  [/PDF /Text]
endobj

5 0 obj
<< /Length 48 >>
  stream
    BT
    /F1 24 Tf
    100 700 Td
    (Hello World)Tj
    ET
  endstream
endobj

4 0 obj
<<
  /Type /Page
  /Parent 3 0 R
  /MediaBox  [0 0 842 842]
  /Contents 5 0 R
  /Resources
     << /ProcSet 6 0 R
        /Font << /F1 7 0 R >>
     >>
>>
endobj

3 0 obj
<<
  /Type /Pages
  /Kids [4 0 R]
  /Count 1
>>
endobj

2 0 obj
```

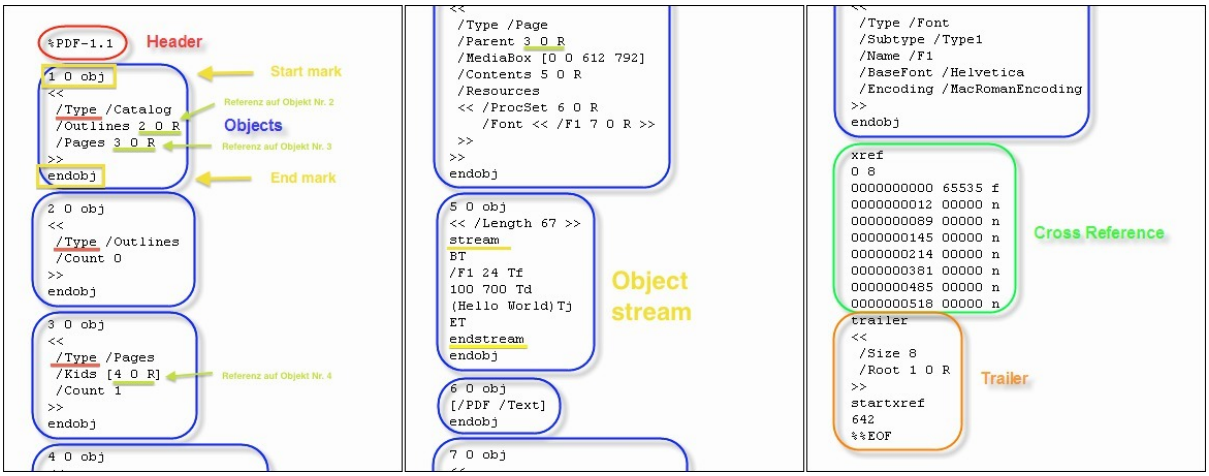
```
<<
  /Type /Outlines
  /Count 0
>>
endobj

1 0 obj
<<
  /Type /Catalog
  /Outlines 2 0 R
  /Pages 3 0 R
>>
endobj

xref
0 8
0000000000 65535 f
0000000565 00000 n
0000000509 00000 n
0000000440 00000 n
0000000273 00000 n
0000000169 00000 n
0000000136 00000 n
0000000012 00000 n
trailer
<<
  /Size 8
  /Root 1 0 R
>>
startxref
642
%%EOF
```

Das obere Viertel dieser PDF-Seite rendert in einem PDF-Viewer wie folgt:

Oberes Viertel einer einfachen PDF-Seite (wie im Quellcode zuvor dargestellt).

1.8 Illustrierte Darstellung der *'Hello World'*-PDF

Die folgende Darstellung entspricht dem o.a. Quellcode. Jedoch verwendet er eine andere Objekt-Reihenfolge, um zu demonstrieren, dass dies ebenfalls möglich ist:



Illustrierte Darstellung des Quellcodes der o.a. 'Hello World'-PDF (jedoch in anderere Reihenfolge der Objekte!)

1.9 Weitere Resourcen zur PDF-Syntax

Selbstverständlich war diese Einführung in die PDF-Syntax extrem verkürzt und vereinfacht. Die offizielle PDF-Spezifikation umfaßt beinahe 800 Seiten.

Hier sind weitere nützliche Resourcen für alle, die sich damit etwas ausgiebiger beschäftigen möchten:

1. PDF-1.7 Spezifikation[2].
2. corkami-Webseite zu PDF-Tricks[3]

[2] http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/PDF32000_2008.pdf
[3] https://code.google.com/p/corkami/wiki/PDFTricks

Appendix

Acknowledgements

Vielen Dank an die Ubuntu-Community in Deutschland für die Organisation der Ubucon 2013 und die Einladung zu diesem Workshop.

Folgende Teilnehmer haben per TitanPad zum Protokoll des Workshops beigetragen. Sie sind deshalb Co-Autoren dieses eBooks:

- Kurt Pfeifle kurt.pfeifle@gmail.com
- Sven Guckes ubucon2013@guckes.net
- Max Schneller cicloalpin@gmail.com
- Torsten Scheck torsten@gmx.org
- Florian Heß <–>