

# Photosort

Programmieren in Clojure

Prof. Dr. Burkhardt Renz

# Agenda

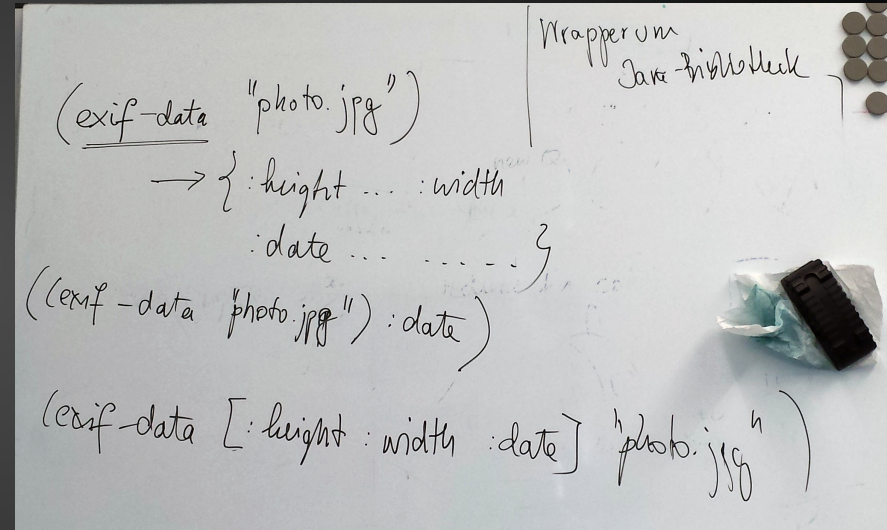
1. Idee und Anforderungen
2. Exif
3. Exif: Umsetzung
4. Abhängigkeiten im Projekt
5. Leiningen und lokale JARs
6. Main Funktion in Clojure
7. Live Präsentation des Projektes

# Idee

- Fotografien aus einem Ordner (z.B. Urlaubsfotos von verschiedenen Kameras) nach Datum sortieren
- Sollte Command-Line fähig sein und es sollte einfach in der Bedienung sein
- Parameterangaben sind optional

# Anforderung

- Fotos aus einem Ordner auslesen und nach bestimmten Kriterien umbenennen/kopieren
- Tags als Kriterien angeben
- Optionale Angaben von Quellort und Speicherort
- Erstellen eines Exif Wrapper für Clojurer

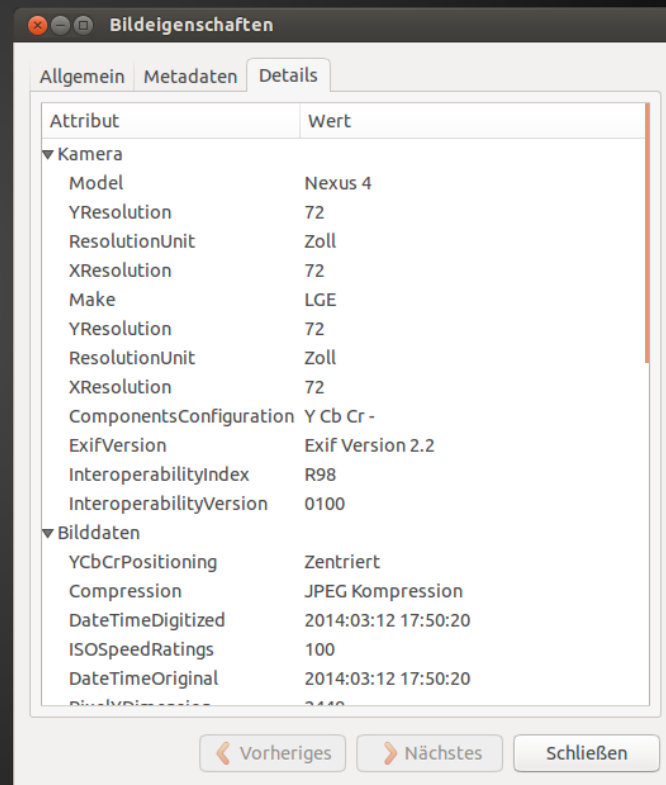


# Exif

- Exchangeable Image File Format
- Standard der *Japan Electronic and Information Technology Industries Association* (JEITA)
- Ermöglicht das Speichern von Metadaten in digitalen Bildern
- Aktuelle Version 2.3 in Verbindung mit *Camera & Imaging Products Association* (CIPA)
- Spezifiziert in [CIPA DC-008-2012](#)

# Exif: Daten

- Datum und Uhrzeit
- Kameraeinstellungen
  - Blende
  - ISO
  - Marke und Modell
  - ...
- Thumbnail
- GPS
- Photoshop
- ...



# Exif: Date/Time vs. Date/Time Original vs. Date/Time Digitized

Date/Time	Änderungsdatum
Date/Time Original	Aufnahmedatum des Bildes
Date/Time Digitized	Datum und Uhrzeit, an dem das Bild digitalisiert wurde

Problem: Date/Time ist nicht immer vorhanden. (bspw. LG Nexus 4)

# Exif: Eine Anekdote

“Der in einem Mordfall als Zeuge gesuchte John McAfee hat sich von dem kanadischen Magazin Vice auf der Flucht interviewen lassen. In den Metadaten der Fotos des Gründers des IT-Sicherheitsunternehmens, die mit einem iPhone gemacht wurden, wurde der Aufenthaltsort des Flüchtigen in Río Dulce, Guatemala, klar.”

Quelle: <http://www.golem.de/news/vice-john-mcafee-mit-iphone-geolocation-geortet-1212-96131.html>



<http://www.whoismcafee.com/wp-content/uploads/2013/08/john-mcafee-video-310x352.jpg>



# Exif: Java Bibliothek

- [metadata-extractor](#) von Drew Noakes
- Java-Bibliothek zum komfortablen auslesen der Daten
- versteht unter anderem:

Metadata Formate	Dateiformate
Exif	JPEG
Photoshop	TIFF
PNG	PNG
GIF	RAW (NEF/CR2/ORF/ARW/RW2/...)
...	PSD
	...

# Exif: Wrapper in Clojure

Anforderungen:

- Hauptfunktionalitäten der Java-Bibliothek, in einem eleganten “Clojure-like way” zur Verfügung stellen
- Leichte Erweiterbarkeit

# Exif: Wrapper in Clojure

## Allgemeine Deklaration:

```
(exif-data param1 param2)
```

Param 1	Param 2
<ul style="list-style-type: none"><li>• String</li><li>• File</li><li>• URL</li></ul>	<ul style="list-style-type: none"><li>• nil</li><li>• String</li><li>• Collection</li><li>• Object (com.drew.metadata.Directory.java)</li></ul>

# Exif: Wrapper in Clojure

## Beispiele:

```
=> (exif-data "/home/thomas/Downloads/IMG_20140302_160056.jpg" "Model")
```

```
"Nexus 4"
```

```
=> (exif-data (File. "/home/thomas/Downloads/IMG_20140302_160056.jpg") [:Model :Make])
```

```
{:Make "LGE", :Model "Nexus 4"}
```

```
=> (exif-data (URL. "https://github.com/andrewissner/sorter1.0/blob/master/IMG_20140312_175020.jpg")  
(GpsDirectory.))
```

```
{:GPS Altitude "0 metres", :GPS Longitude "8° 40' 40,73\"", :GPS Processing Method "ASCII", :GPS Img  
Direction Ref "Magnetic direction", :GPS Img Direction "157 degrees", :GPS Latitude "50° 35' 29,4\"", :  
GPS Latitude Ref "N", :GPS Altitude Ref "Sea level", :GPS Longitude Ref "E", :GPS Date Stamp "2014:03:  
12", :GPS Time-Stamp "16:50:15 UTC"}
```

# Exif: Wrapper in Clojure

```
(defn- exif-for-file
  ([file]
   (exif-for-file file nil))

  ([file dir]
   (try
    (if (nil? dir)
      ;then
      (let [metadata (ImageMetadataReader/readMetadata file)
            exif-dirs (filter #(re-find exif-directory-regex (.getName %)) (.getDirectories metadata))
            tags (map #(.getTags %) exif-dirs)]
        (into {} (map extract-from-tag tags)))
      ;else
      ...
    (catch Exception e
      (println (str "caught exception: " (.getMessage e)))
      nil))))
```

# Exif: Wrapper in Clojure

## Umsetzung mit Hilfe von Protocols

```
(defprotocol exif
  (exif-data
    [x]
    [x tag-or-dir]
  )
)
```

```
(extend-protocol exif
  File
    (exif-data
      ([f tag-or-dir]
        (if (instance? String tag-or-dir)
          (exif-tag-for-file f tag-or-dir)
          (if (instance? Directory tag-or-dir)
            (exif-for-file f tag-or-dir)
            (exif-tags-for-file f tag-or-dir))))
      ([f]
        (exif-for-file f)))
    ...
  )
```

# Abhängigkeiten im Projekt

## Aktuelle Abhängigkeit im Projekt

```
:dependencies
```

```
[[org.clojure/clojure "1.5.1"]
```

```
[com.drewnoakes/metadata-extractor "2.7.0-SNAPSHOT"]
```

```
[seesaw "1.4.2"]
```

```
[clj-http "0.7.2"]
```

```
[clargon "1.0.0"]
```

```
..]
```

# Abhängigkeiten im Projekt

- **com.drewnoakes/metadata-extractor**  
Bibliothek zum Auslesen der Metadaten aus einem Foto/Bild
- **clargon "1.0.0"**  
Bibliothek zur Verwendung der Command-Line
- **clj-http "0.7.2"**  
Clojure-Wrapper für die Apache HttpClient Bibliothek
- **seesaw "1.4.2"**  
Bibliothek zur grafischen Oberflächen Erweiterung



# Leiningen und lokale JARs

Einbinden von lokalen JARs in der *project.clj*

Möglichkeit 1:

- `:resource-paths ["resources/Datei.jar"]`  
Lokalen Ordner *resources* erstellen

Möglichkeit 2:

- Einbinden über ein lokales Maven Repository  
`:dependencies`  
`[ .. [com.drewnoakes/metadata-extractor 2.7.0-SNAPSHOT] ..]`

# Main-Funktion in Clojure

- Wie auch in Java, so gibt es auch in Clojure eine Main-Funktion
- Muss in der *project.clj* eingetragen sein
  - :main Sorter.main
- Hilfsbibliothek für die Nutzung der Command-Line
  - Clargon
  - Auch zu finden unter: `Clojure.tools.cli/cli`

# Main-Methode - Command-Line

```
(defn -main [& args]

  (ccl/with-command-line args

    " Some description "

    [

      [help "Shows exactly this menu"]

      [in "This specifies the input directory for the pictures"]

      [out "This specifies the output directory for a new folder"]

      [tag "To sort and rename the pictures by given tags"]

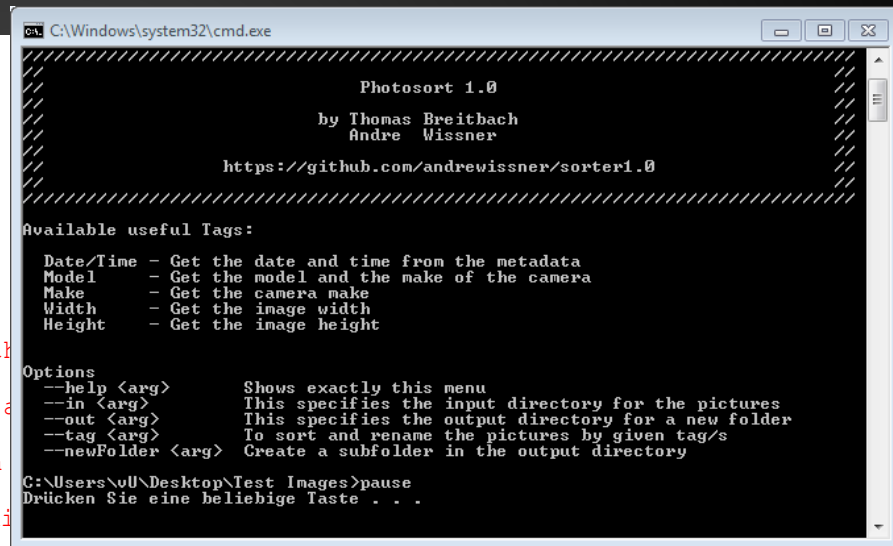
      [newFolder "Create a subfolder in the output directory"]

      extras]

    ...

    ;Create cool stuff!

  ))
```



```
C:\Windows\system32\cmd.exe

Photosort 1.0

by Thomas Breithach
Andre Wissner

https://github.com/andrewwissner/sorter1.0

Available useful Tags:

Date/Time - Get the date and time from the metadata
Model      - Get the model and the make of the camera
Make       - Get the camera make
Width      - Get the image width
Height     - Get the image height

Options
--help <arg>      Shows exactly this menu
--in <arg>        This specifies the input directory for the pictures
--out <arg>       This specifies the output directory for a new folder
--tag <arg>       To sort and rename the pictures by given tag/s
--newFolder <arg> Create a subfolder in the output directory

C:\Users\vu\Desktop\Test Images>pause
Drücken Sie eine beliebige Taste . . .
```

# Photosorter - Das Projekt

Live Präsentation des Projektes

# Fragen?

Vielen Dank!

Quellcode unter

**<https://github.com/andrewissner/sorter1.0>**