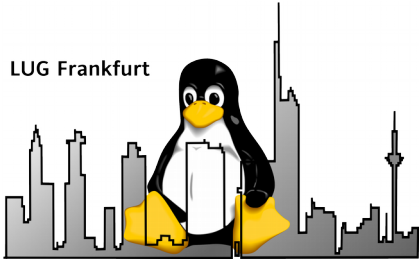


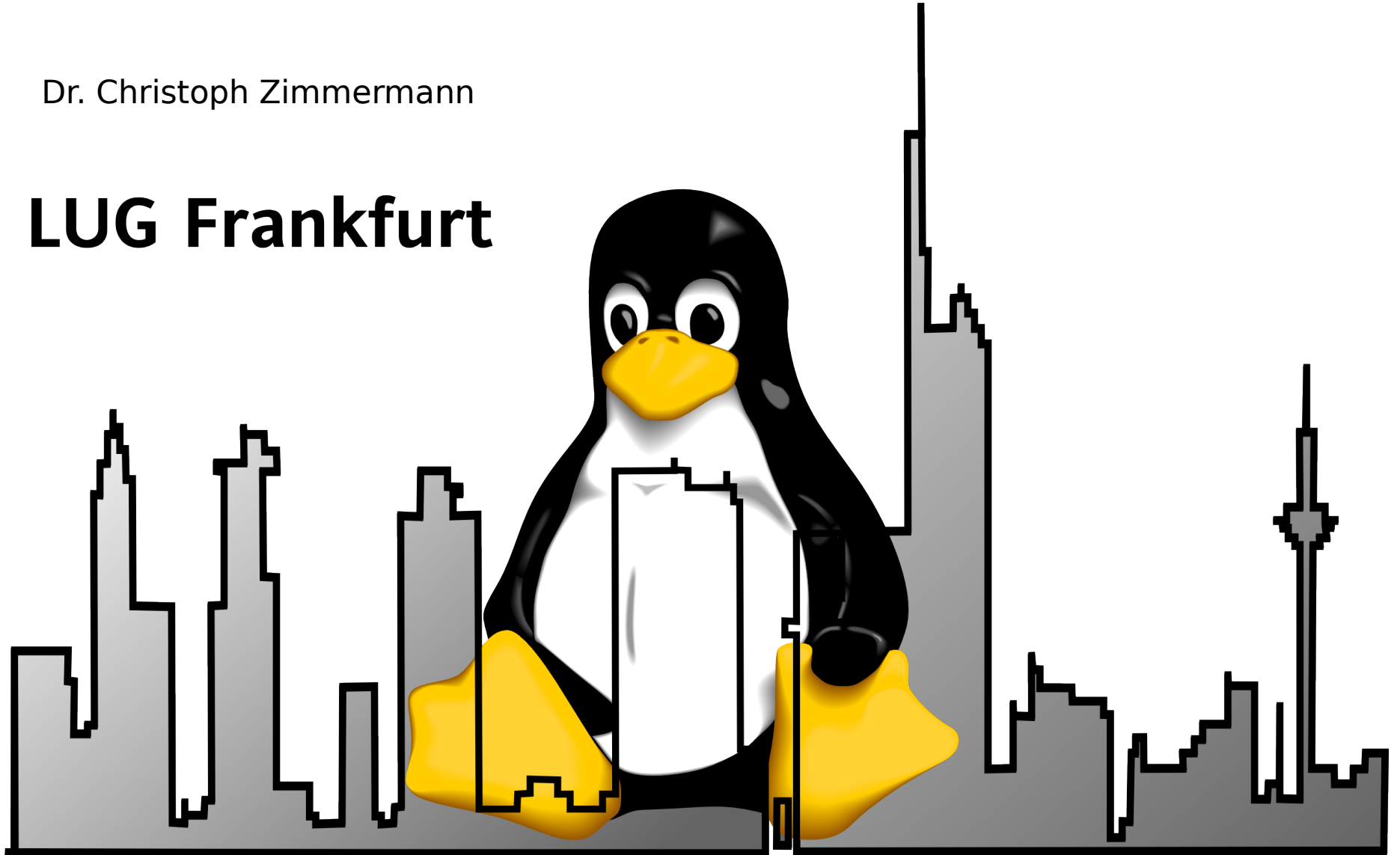
LUG Frankfurt



Entwickeln mit LibreOffice und Google APIs

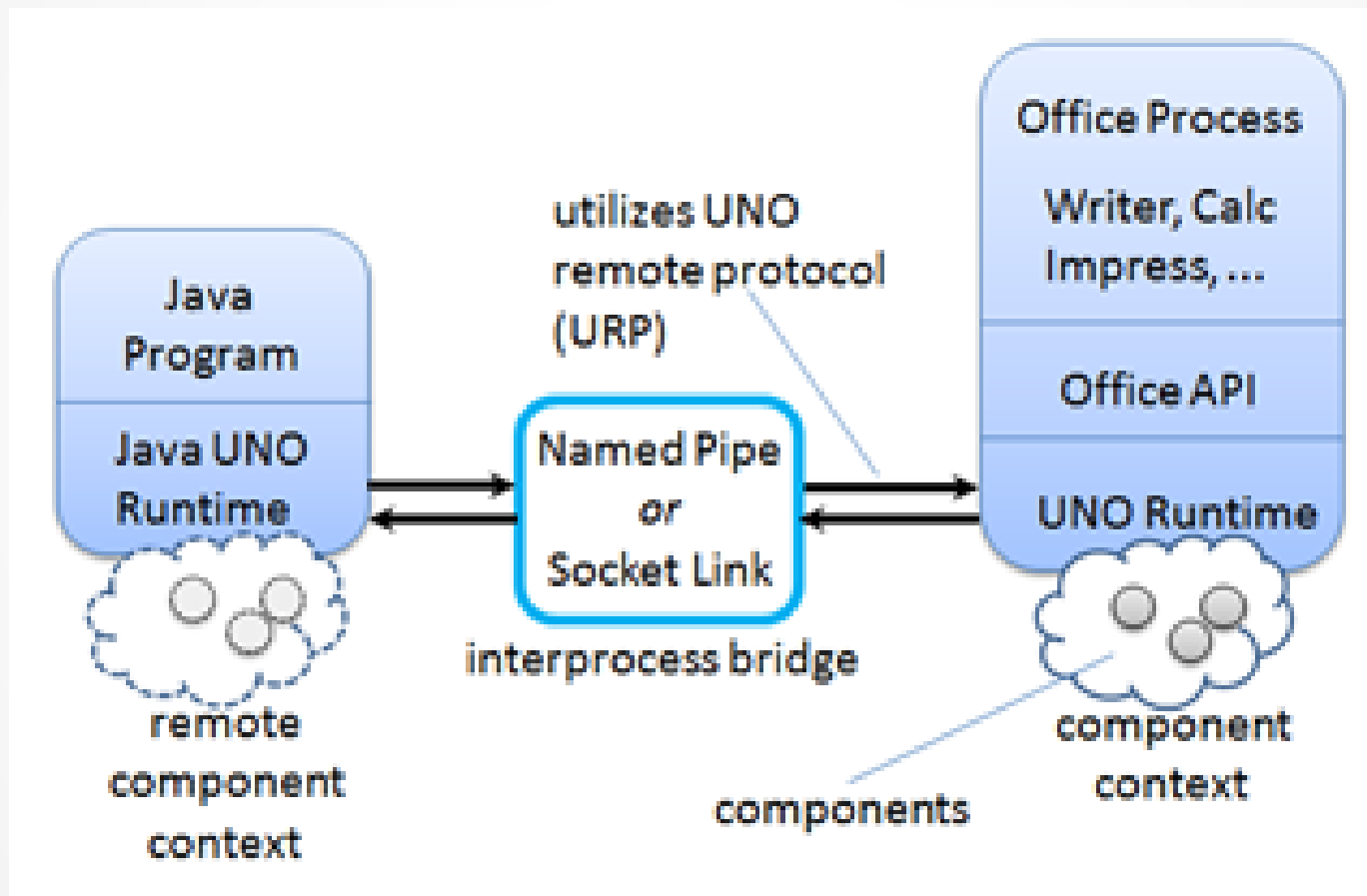
Dr. Christoph Zimmermann

LUG Frankfurt



Teil 1	Teil 2
1. Libreoffice Architektur	5.Rekap Teil 1
2. Python-Übersicht	6.Google APIs
3. Applikations-Design	7.Python Schnittstelle
4. Libreoffice Code	8.Google Code
	9.Zusammenfassung
	10.Diskussion

Übersicht LibreOffice UNO



- Universal Network Objects:
Komponenten-Modell für LO-Zugriff für
verschiedene Programmiersprachen
- Bindings existieren für: C++, Java,
Python, Tcl
- LO Basic: benutzt selbst intern UNO 😊
- Weitere Bindings durch Implementierung
der IDL für Programmiersprache möglich

Python-Übersicht

- Python: OOPL mit funktionalen und anderen Erweiterungen
- Umfangreiche Funktionalität durch externe Module
- Genereller Aufbau
 - Blockbildung durch Einrückung und ':'
 - Datentypen:
 - Integer, Fließkomma, Charakter
 - Arrays, Dictionaries, Sets: veränderlich
 - Sequences, Strings: nicht veränderlich

Python-Übersicht (Fort.)

- Genereller Aufbau (Fort.):
 - if, then, else, for, while wie gewohnt
 - Kein do .. until
 - Aufbau Prozedur / Funktion
 - `def name(parameter, ...):`
 - `""" doc string """`
 - Blöcke ...
 - `<return>`
 - Blöcke ...
 -

Beispiel-Anwendung

- Ziel: Konvertierung von vorhandenen Schichtplänen (Excel-Dateien) in GCAL-Ereignisse
- Anforderung:
 - Skalierbar (bis zu mehreren 1.000 Schichtplänen)
 - Einfach erweiterbar
 - Idealerweise: Plattform-unabhängig

Applikations-Design

- Format der Schichtpläne (Excel-Sheet):

			Jan			Feb			Mrz
1	1	So	S0	1	Mi	S1	1	Mi	S1
2	2	Mo	S0	2	Do	S0	2	Do	S0
3	3	Di	S0	3	Fr	S1	3	Fr	S1
4	4	Mi	S0	4	Sa	S1	4	Sa	S1
5	5	Do	S0	5	So	S1	5	So	S1
6	6	Fr	S0	6	Mo	S1	6	Mo	S1
7	7	Sa	S0	7	Di	S1	7	Di	S1
8	8	So	S0	8	Mi	S1	8	Mi	S1
9	9	Mo	S1	9	Do	S0	9	Do	S0
10	10	Di	S1	10	Fr	S0	10	Fr	S0
11	11	Mi	S1	11	Sa	S0	11	Sa	S0
12	12	Do	S0	12	So	S0	12	So	S0

Applikations-Design (Fort.)

- Format der Schichtpläne in der Ausgabe:
 - GCAL-Einträge
 - Datum-Beginn + Länge
 - Warnung:
 - Letzter Tag = Länge + 1
 - D. h. letzter Tag liegt nach dem eigentlichen Ende des Eintrags!
 - Google-PM: „It's a feature, not a bug!“ ☹

Applikations-Design (Forts.)

- Idee:
 - Kommunikation via Bitmap über Tage / Monate da nur zwei Schichten (Erweiterung: Ersetzung der Bitmap durch normales 2D-Feld)
 - Teil 1: Einlesen der Spreadsheets und Konvertierung in zweidimensionale Bitmap (Tage / Monate)
 - Teil 2: Auslesen der Bitmap und Generation von GCAL-Einträge
 - Performanz

Python / LO

- Zugriff über pyoo Modul:
 - UNO-Abstraktion für speziell für LO Calc
 - Zugriff über Metaobject-Protokoll
 - Installation via pip
- Voraussetzung: python[3]-uno für die Kommunikation mit LO
- LO Start:
 - `soffice`
`--accept="socket,host=localhost,port=2002;urp;"`
`--norestore --nologo --nodefault --headless`

Code (LO-Teil)

```
1 def getCalc(fileN, year):
2     if not (type(year) is int and year > 0):
3         raise('getCalc: takes year as UINT')
4     if not isinstance(fileN, str):
5         raise('getCalc: takes filename as string')
6     try:
7         desk = pyoo.Desktop('localhost', 2002)
8         doc = desk.open_spreadsheet(fileN)
9     except:
10        raise('getCalc: cannot open file ' + fileN)
11    sheet = doc.sheets ['Kalender']
12    yearM = []
13    for mon in range(12):
14        eom = calendar.monthrange(year, mon + 1)[1]
15        yearM.append(bitarray(eom))
16        for day in range(eom):
17            yearM [mon][day] = sheet [day + 3, mon * 3 + 3].value == 'S1'
18 # Return set bit array
19    return yearM
```

Teil 1	Teil 2
1. Libreoffice Architektur	5.Rekap Teil 1
2. Python-Übersicht	6.Google APIs
3. Applikations-Design	7.Python Schnittstelle
4. Libreoffice Code	8.Google Code
	9.Zusammenfassung
	10.Diskussion

Rekap. Teil 1

- Python
- LibreOffice APIs
- Design der Applikation
 - Teil 1: Einlesen der Spreadsheets und Konvertierung in zweidimensionale Bitmap (Tage / Monate)
 - Teil 2: Auslesen der Bitmap und Generation von GCAL-Einträge

Google APIs

- GSuite APIs:
 - Zugriff Applikation via Applikationen
 - APIs: Python, Java, PHP, Go
- Vorgehen:
 - Anlage eines Projektes in Google Console
 - Definition der Authentifizierung (OAuth2)
 - Dann Zugriff via API-Objekt nach erfolgreicher Authentifizierung

Code (Authentifizierung)

```
1  SCOPES = 'https://www.googleapis.com/auth/calendar'
2  CLIENT_SECRET_FILE = 'client_id1.json'
3  APPLICATION_NAME = 'Google Calendar API Python'
4  def get_credentials():
5      """Gets valid user credentials from storage.
6      If nothing has been stored, or if the stored credentials are invalid,
7      the OAuth2 flow is completed to obtain the new credentials.
8
9      Returns: Credentials, the obtained credential.
10     """
11     home_dir = os.path.expanduser('~')
12     credential_dir = os.path.join(home_dir, '.credentials')
13     if not os.path.exists(credential_dir):
14         os.makedirs(credential_dir)
15     credential_path = os.path.join(credential_dir,
16                                     'calendar-python-quickstart1.json')
17     store = Storage(credential_path)
18     credentials = store.get()
```


Code (Authentifizierung, Fort.)

```
17         if not credentials or credentials.invalid:
18             flow =
19             client.flow_from_clientsecrets(CLIENT_SECRET_FILE, SCOPES)
20             flow.user_agent = APPLICATION_NAME
21             # Needed only for compatibility with Python 2.6
22             credentials = tools.run(flow, store)
23             print('Storing credentials to ' + credential_path)
24             return credentials
25
26 def main():
27     credentials = get_credentials()
28     http = credentials.authorize(httplib2.Http())
29     calendarId='XXXX@group.calendar.google.com'
30     service = discovery.build('calendar', 'v3', http=http)
```

Code (GCal-Einträge u. m.)

```
1 def buildDates(year, yearM):
2     if not (type(year) is int and year > 0):
3         raise('getCalc: takes year as UINT')
4     if not (isinstance(yearM, list) and len(yearM) == 12):
5         raise('getCalc: takes yearM as list of bitarrays')
6     mon = 0
7     res = {}
8     counting = False
9     while mon < 12:
10        day = 0
11        while day < len(yearM[mon]):
12            if not counting:
13                if yearM [mon][day]:
14                    start = datetime.datetime(year, mon+1, day +
15)
15                    counting = True
```

Code (GCal-Einträge u. m.)

```
16         else:
17             if not yearM [mon][day]:
18 # Note that end date is already the following date, but due to
   GCAL's braindead
   # end date rule (for all day events, the end date isn't included
   in the even)
19 # this turns out to be OK
           counting = False
20           end = datetime.datetime(year, mon + 1, day +
   1)
21           res [start.strftime('%Y-%m-%d')] =
   end.strftime('%Y-%m-%d')
22           day += 1
23           mon += 1
24 # Hit year-end while counting?
25 if counting:
26     end = datetime.datetime(year, mon, day)
27     res [start.strftime('%Y-%m-%d')] = end.strftime('%Y-%m-
   %d')
28 return res
```

Code (GCal-Einträge u. m.)

```
1 def storeInGCAL(service, calendarId, dates):
2     """
3     Shows basic usage of the Google Calendar API.
4     Store dict DATES in GCAL object
5
6     """
7     if not isinstance(dates, dict):
8         raise('storeInGCAL: takes dict as sole argument')
9     if len(dates) == 0:
10        raise('dict cannot be empty')
11    for i in dates.keys():
12        startD = { 'date' : i, 'timeZone' : 'Europe/Berlin' }
13        endD = { 'date' : dates [i], 'timeZone' :
14            'Europe/Berlin' }
15        event = { 'summary' : 'Schicht 1', 'start' : startD,
16            'end' : endD }
17        result =
18        service.events().insert(calendarId=calendarId,
19            body=event).execute()
```

Zusammenfassung

- LibreOffice:
 - Vielseitige Office-Suite inkl. mächtiger API
 - Vielzahl von unterstützten Sprachen mit unterschiedlichen Abstraktionsebenen bzgl. API-Kapselung
 - Einfacher Zugriff auf anwendungsspezifische Daten durch UNO
 - Abstraktion von UNO-APIs durch spezielle Module
- Googles GSuite APIs:
 - Autorisierter Zugriff auf Anwendungsdaten
 - Vielzahl von unterstützten Sprachen

Weiterführendes

- pyoo: <https://pypi.python.org/pypi/pyoo>
- GCAL Python API:
<https://developers.google.com/google-apps/calendar/quickstart/python>
- UNO:
https://en.wikipedia.org/wiki/Universal_Network_Objects
- LO API Referenz (inkl. UNO):
<https://api.libreoffice.org>

Diskussion / Fragen

Vielen Dank!

Dr. Christoph Zimmermann
monochrome@gmail.com

Backup