7437 - EDI und E-Business Standards

Praktikumsaufgabe 07
Bestellungen und Lieferavise mit UN/EDIFACT und EANCOM,
Mapping per EDI-Bibliothek

Das Szenario

- Szenario
  - Die Praktikumsteilnehmer setzen ihre Handelspartner-Rolle (Lieferanten, Händlern) der Konsumgüterbranche aus Praktikum 01/02 fort.
  - Im Unterschied zur Übung 05/06 steht nun die Verarbeitung eingehender UN/EDIFACT-Daten im Mittelpunkt.
- Ziele der Übung
  - Umgang mit eingehenden UN/EDIFACT-Daten
  - Selektion von Segmenten / Segmentgruppen
  - OO-Erzeugung von Inhouse-Formaten mit fester Satzlänge
  - Mappingtechnik: Vertiefung

Technische Hinweise

- Dateikopf (Einbindung der Bibliothek):

```
#!/usr/bin/env ruby
require "r-edi/base"
```

- Einlesen eines EDIFACT Interchange:

```
ic = EDI::Interchange_E.parse(File.open(fname,'r'))
mapping = Inbound_maps.new # Controllerklasse
```

- EDIFACT Interchange nachrichtenweise verarbeiten:

```
ic.each do |msg|
  mapping(msg) # schreibt nach stdout
end
```


Technische Hinweise

- Selektion der eigentlichen Konverter-Methode:
  - Die Methode mapping wertet aus:
    - msg.root.header.cS002 (Interchange Sender)
    - msg.root.header.cS003 (Interchange Recipient)
    - msg.header.cS009 (Message Identifier)
  - Aus diesen Angaben wird die zuständige Konverter-Methode bzw. -Variante ermittelt. Diese wird dann aufgerufen.
  - Hier kann dieses Verfahren stark vereinfacht werden.

- Kern einer Konvertermethode:

```
msg.each do |seg|
  seg_name = seg.name
  seg_name += ' ' + seg.sg_name unless seg.sg_name.empty?
  case seg_name
   # ... Viele when-Blöcke
  end
end
```

Technische Hinweise

- when-Konstrukte einer Konvertermethode:

```
when 'DTM'      # Selektiert DTM außerhalb einer SG
  cde = seg.cC507    # Hilfsvariable, optional
  # Std-Code mit Hash auf hauseigene Codes mappen:
  a = myDTMcode[cde.d2005]
  # Datum und ggf. Uhrzeit auslesen
  case cde.d2379
  when '203', '102'
   cde.d2380 =~ /^(\d{8})(\d{4})?/
   mydate = $1
   mytime = $2 unless $2.nil?
  when ... # etc.
  else
   raise "Falsches Format in DTM, DE=C502/2379"
end
```


Technische Hinweise

- Selektion eines Segments einer Segmentgruppe:

```
when 'NAD SG1' # Selektiert NAD der SG1
 # ...
when 'CTA SG2' # Selektiert CTA in SG2 (in SG1)
 # ...
when 'NAD SG4' # Selektiert NAD der SG4
 # ...
when 'NAD'     # FEHLER: NAD nie außerhalb einer SG!
# (Kein Abbruch, aber: Trifft nie ein!)
```

Technische Hinweise

- Umgang mit (großen) Segmentgruppen:

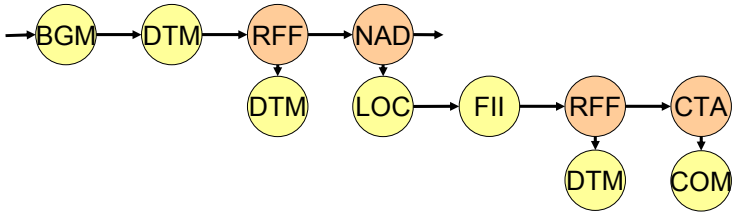
```
when 'LIN SG25'       # Trigger-Segment?
  item_mapper(seg)    # an spezielle Methode geben
# ...

# In item_mapper-Methode:
tseg["descendant-or-self::*"].each do |seg|
  # case aufsetzen, wie in Hauptmethode
end
```

- Bemerkungen
  - Selektion einer Teilmenge der Segmente: XPath-artig!
  - Verteilung auf mehrere Methoden fördert Modularisierung.

Technische Hinweise

- Umgang mit (großen) Segmentgruppen:
  - Selektion einer Teilmenge der Segmente: XPath-artig!
  - Verteilung auf mehrere Methoden fördert Modularisierung.
  - Zerlegung der "Seitenketten" in Einzelketten, je eine Methode pro Seitenkette.
  - Konzept:
    - Es gibt einfache Knoten und T-Knoten (Knoten = Segment hier)
    - Vernetzung über die (realen) T-Knoten
    - Segmentgruppen bleiben ein virtuelles Konstrukt (nur Attribut-Info)



Technische Hinweise

- Fixrecord-Formate mit Ruby-Klassen implementieren:

```
# Anregung:
def MyFixRec01
  @@field_descr = {
   :recID =>   [0,  2,  0, 'l', 'n2'],
   :field01 => [1, 20,  2, 'l', 'an..20'],
   :field02 => [2, 16, 22, 'r', 'n..16'],
   # usw.
  }
  @@field_desc.each_key {|key| attr_accessor key}

  def to_s
   ...
  end
end
```


Technische Hinweise

- Fixrecord-Formate mit Ruby-Klassen implementieren:

```
# Anregung, spätere Verwendung:

myrec = MyFixRec01.new      # Initialisierung
# ...
# In Mapping-Schleife:
myrec.field01 = seg.d1001   # Zuweisung eines Feldes
# ...
puts myrec      # Schließlich Ausgabe, via "to_s"
```

Mapping

- Die Aufgabe
  - Mappen Sie Ihre Belege aus Übung 06 (Händler: ein eingehendes Lieferavis, Lieferanten: eine Bestellung)
    - von EANCOM '02 gemäß des Nachrichtenaufbaus Ihres Handelspartners
    - in Ihre Inhouse-Struktur (s.u.)
  - Programm-Name:

    `mapper07.rb`
  - Gewünschter Aufruf:

    `mapper07.rb srcdata.edi > fixrecdata.msg`

Mapping

- Hinweise zum Ablauf
  - Besorgen Sie zunächst das *Interchange*
    - Das *Interchange* stammt von Ihrem Handelspartner. Es stellt das Ergebnis von Übung 05/06 dar.
  - Definieren Sie eine geeignete *Inhouse*-Struktur. Vorgaben:
    - Die ersten 2 Zeichen sind für die Record-ID reserviert
    - Felder haben feste Länge und Position und sind entweder rechts- oder linksbündig zu füllen.
    - Endständiger *white space* darf ausgelassen werden
    - Ein *record* / Segment = eine Textzeile

Mapping

- Hinweise zum Ablauf (Forts.)
  - Weitere Vorgaben:
    - Die Anzahl verschiedener *records* / Segmente können Sie festlegen. Sie werden mindestens 2 benötigen (für Kopf- und Positionsdaten).
    - Das *Inhouse*-Format ist zu dokumentieren mittels Kommentaren im Quellcode. Beschreiben Sie die Bedeutung jedes Feldes stichwortartig.
  - **Bedingung**:
    - Alle eingehenden Nutzdaten sind zu mappen,
    - nicht mehr benötigte Qualifier sowie Inhalte der Service-Segmente müssen Sie nicht in Ihr Inhouse-Format übernehmen.

Abgaben

- Abzugeben
  - `mapper07.rb  # Ihr Programm-Code`
  - `srcdata.edi  # Ihre verwendeten Quelldaten`
  - `fixrecdata.msg # Ihre Ergebnisse`

  (srcdata.edi sollte von Ihrem Handelspartner stammen, fixrecdata.msg erzeugen Sie nach eigener Definition)
- Abgabeordner
  - Wie üblich, unter `~werntges/kurse/edi/abgaben/a/<matrnr>`
- Annahmeschluss
  - 5 Minuten nach Beginn der **über**nächsten Übung (8. Juni).