Konstruktion av datorspråk

Fö3: Uppmärkningsspråk

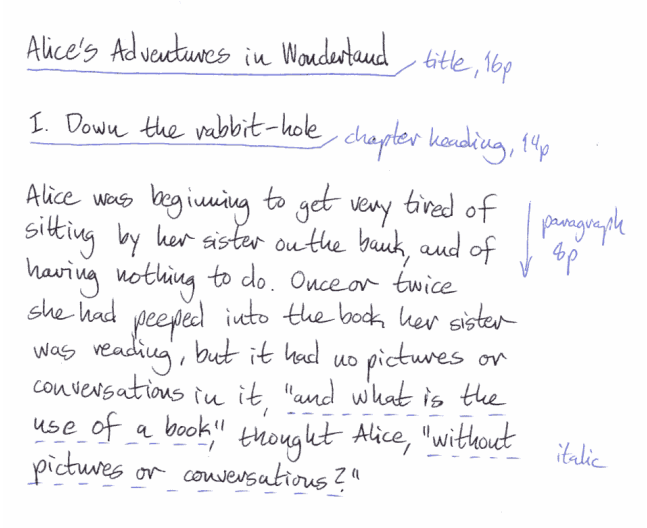
Peter Dalenius
petda@ida.liu.se
Institutionen för datavetenskap
Linköpings universitet

2009-01-29

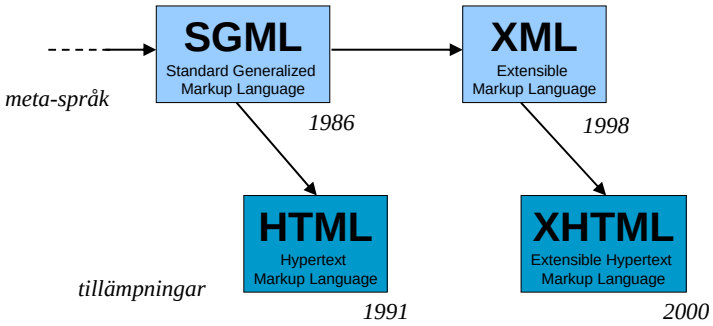
Översikt

- Uppmärkningsspråk
- Struktur och specifikation av XML-dokument
- Parsning av XML-dokument (SAX, DOM, XPath)
- Mikroformat
- Inför seminariet

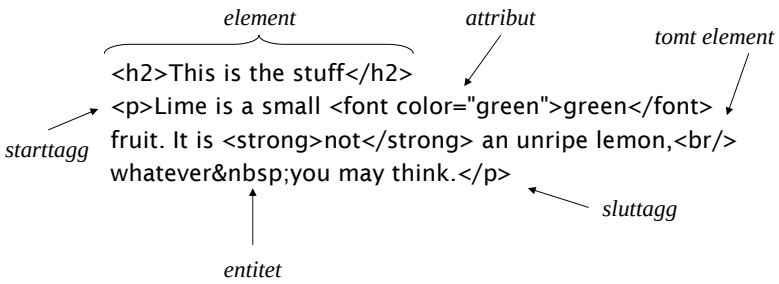
Uppmärkningsspråk



Kort historisk översikt



Begrepp inom uppmärkning



This is the stuff

Lime is a small green fruit. It is **not** an unripe lemon, whatever you may think.

Exempel på XML

```
<?xml version="1.0"?>
<note>
  <to>Ola</to>
  <from>Peter</from>
  <heading>Öl på fredag?</heading>
  <body>
    <paragraph>Det har öppnat ett nytt holländskt ställe
    på S:t Larsgatan. De lär ha en massa sorters holländsk
    öl på fat. Vad sägs om att vi testar det på fredag
    kväll?</paragraph>
    <paragraph>Förresten, stället heter De Klomp.</paragraph>
  </body>
</note>
```

”Well-formed”

Exempel på DTD

```
<!DOCTYPE note [
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to (#PCDATA)>
  <!ELEMENT from (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body (paragraph+)>
  <!ELEMENT paragraph (#PCDATA)>
]>
```

DTD (Document Type Defintion)

”Valid”

Exempel på DTD (2)

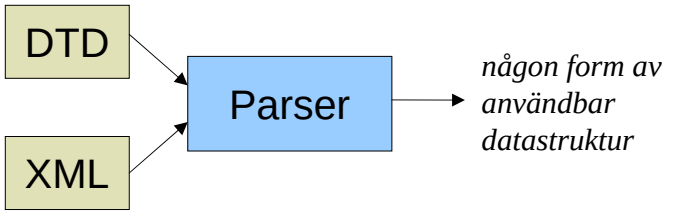
```
<!DOCTYPE RESULTS [
  <!ELEMENT RESULTS (ARTICLE+)>
  <!ELEMENT ARTICLE (HEADLINE,BYLINE,LEAD,BODY,NOTES)>
  <!ELEMENT HEADLINE (#PCDATA)>
  <!ELEMENT BYLINE (#PCDATA)>
  <!ELEMENT LEAD (#PCDATA)>
  <!ELEMENT BODY (#PCDATA)>
  <!ELEMENT NOTES (#PCDATA)>
  <!ATTLIST ARTICLE AUTHOR CDATA #REQUIRED>
  <!ATTLIST ARTICLE EDITOR CDATA #IMPLIED>
  <!ATTLIST ARTICLE DATE CDATA #IMPLIED>
  <!ATTLIST ARTICLE EDITION CDATA #IMPLIED>
  <!ENTITY NEWSPAPER "Dagens Nyheter">
  <!ENTITY COPYRIGHT "Copyright 2008 Dagens Nyheter">
]>
```

Övning

- Kopiera DTD:n från föregående bild från http://www.ida.liu.se/~TDP007/material/seminarie2/newspaper.dtd
- Skapa en XML-fil som använder denna DTD och hitta på lite innehåll.
- Validera filen mot http://xmlvalidation.com/ när du är klar.

Användning av XML-dokument

- Tolka med reguljära uttryck: jobbigt och dåligt
- Använda en *parser*: lätt och bra



1. SAX (Simple API for XML)

- Läser XML-filen som en ström, från början till slut.
- Varje gång det ”händer något” (t.ex. en starttagg) anropas en viss funktion.

```
<?xml version="1.0"?>
<note>
  <to>Ola</to>
  <from>Peter</from>
  <heading>Öl på fredag?</heading>
  …
</note>
```

Exempel på användning av SAX

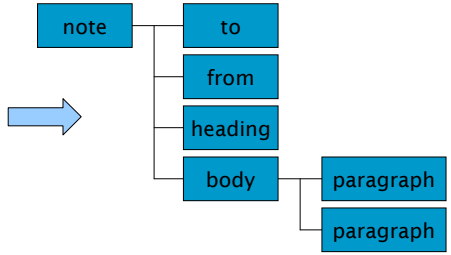
```
require 'rexml/streamlistener'

class MyListener
  include REXML::StreamListener

  def tag_start(name, attrs)
    puts "Start of #{name}."
  end

  def tag_end(name)
    puts "End of #{name
  end

  def text(text)
    puts "Tag contains t
  end
end
```

```
>> lst = MyListener.new
=> #<MyListener:0x2f8d188>
>> src = File.new "c:/lab/ruby/code/note2.xml"
=> #<File:c:/lab/ruby/code/note2.xml>
>> REXML::Document.parse_stream(src,lst)
…
```

2. DOM (Document Object Model)

- Läser hela XML-filen på en gång och returnerar ett objekt som motsvarar hela innehållet.

```
<?xml version="1.0"?>
<note>
  <to>Ola</to>
  <from>Peter</from>
  <heading>Öl på fredag?</heading>
  ...
</note>
```

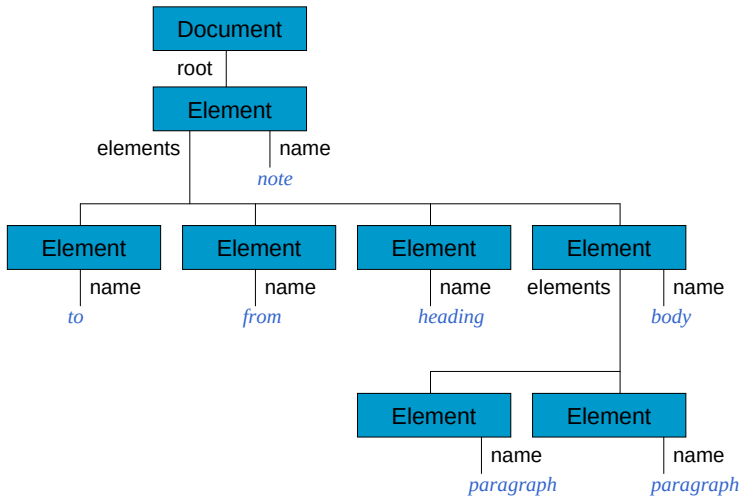


Exempel på användning av DOM

```
>> src = File.open "note2.xml"
=> #<File:note2.xml>
>> require 'rexml/document'
=> true
>> doc = REXML::Document.new src
=> <UNDEFINED> ... </>
```

Detta anrop parsar hela XML-filen och returnerat ett DOM-objekt. Vad finns i detta objekt och hur kommer man åt det?

DOM-struktur för XML-exempel



Exempel på metoder för DOM

```
>> doc = REXML::Document.new src
=> <UNDEFINED> ... </>
>> doc.class
=> REXML::Document
>> r = doc.root
=> <note> ... </>
>> r.class
=> REXML::Element
>> r.name
=> "note"
>> r.elements[1].name
=> "to"
>> r.elements[4].elements[2].text
=> "Förresten, stället heter De Klomp."
```

Dokumentet är ett objekt av klassen `Document`. Det har ett attribut `root` som representerar rotelementet.

Varje element har en array `elements` som innehåller ”barnen”. Attributet `name` innehåller elementets namn.

Övning

- Ladda in filen node2.xml och plocka ut namnen på sändare och mottagare.
- Ladda in följande funktion från print_tree.rb, kör den på det inlästa XML-dokumentet och förklara vad som händer.

```
def print_tree(elem,indent=0)
  elem.elements.each do |subelem|
    puts ' '*2*indent + subelem.name
    print_tree(subelem,indent+1)
  end
end
```

XPath

- Att plocka fram elementen ur dokument-objektet gör att man måste hårdkoda dokumentets specifikation i programmet. Ett lite enklare sätt är att använda XPath.
- XPath (XML Path Language) är ett språk för att välja ut delar av ett XML-dokument.
- Flera av funktionerna i REXML-paketet stödjer XPath-uttryck.
- XPath 1.0 är från 1999. XPath 2.0 är från 2007.

Exempel på XPath-uttryck

Alla *paragraph*-noder inuti en *body*, inuti en *note* som är roten i dokumentet.

```
>> doc.elements.each("/note/body/paragraph") { |n| puts n.text }
Det har öppnat ett nytt holländskt ställe
på S:t Larsgatan. De lär ha en massa sorters holländsk
öl på fat. Vad sägs om att vi testar det på fredag kväll?
Förresten, stället heter De Klomp.
=> …
>> doc.elements.each("//body/*") { |n| puts n.text }
Samma resultat som ovan…
```

Alla noder, vilka som helst (*), som ligger inuti en body som ligger någonstans under roten, oavsett nivå.

Konstruktion av XPath-uttryck

- Alla XPath-uttryck har en *kontext* som tjänar som utgångspunkt (en nod i dokumentet).
- Från utgångspunkten går man med hjälp av tre saker (alla behöver inte vara med):
  - en *riktning*, t.ex. barn eller förälder
  - ett *nodtest*, t.ex. att man söker noder med ett visst namn
  - ett *predikat* som talar om att noderna ska ha ytterligare egenskaper
- Läs på om XPath och experimentera på egen hand. Det finns gott om bra introduktioner på nätet.

Mikroformat

- Microformats är en uppsättning små fria standarder för att representera vissa specifika typer av information.
- Ett typiskt exempel är en överenskommelse om hur man kan bädda in kontaktinformation i en webbsida.

```
<span class="vcard"><span class="fn">Kalle Kula</span></span>
```

- Mer information finns på www.microformats.org

Inför seminariet

- Tre sätt att bearbeta ett XML-dokument:
  - Implementera egen SAX-lyssnare
  - Bygga DOM och använda metoder för element
  - Bygga DOM och använda XPath
- Dagens förkortningar:
  - API, DOM, DTD, HTML, SAX, SGML, XHTML, XML, XPath