

API pro práci s XML

Jirka Kosek

Poslední modifikace: \$Date: 2008/05/08 10:37:55 \$

Copyright © 2001-2003 Jiří Kosek

Obsah

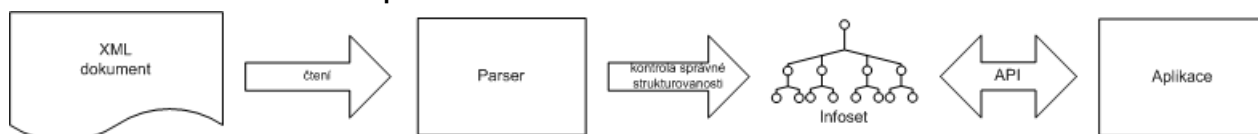
Úvod	3
Parseery XML	4
Rozhraní pro přístup k dokumentu XML	5
Další charakteristiky parseru	6
Sekvenční čtení	7
SAX	8
Pull parseery	9
Stromová reprezentace	10
DOM	11
Hierarchie tříd použitých v DOM	12
Nejběžnější metody DOM	13
Další směr vývoje DOM	14
Rozšíření a varianty DOM	15
Generátory tříd	16
Úvod	17
Příklad schématu	18
Ukázka rozhraní vygenerované třídy	19
Práce s dokumentem	20
Další API	21
XSLT, XPath	22
Další vývoj	23
Podpora XML v jednotlivých jazycích	24
Java	25
.Net	26
Další informace	27
Další informace	28

Úvod

Parsery XML	4
Rozhraní pro přístup k dokumentu XML	5
Další charakteristiky parseru	6

Parseery XML

- mnoho aplikací potřebuje číst (generovat) dokumenty XML
- dokumenty XML lze z aplikace číst jako „Unicode text with angle brackets“
 - pracné a nepohodlné
 - pomalé na implementaci
- aplikaci většinou nezajímá XML jako text, ale infoset odpovídající dokumentu
- parser je komponenta (knihovna), která umí přes rozhraní nabízet infoset dokumentu aplikaci



- jednotlivé parseery se liší podporovanými rozhraními a dalšími charakteristikami

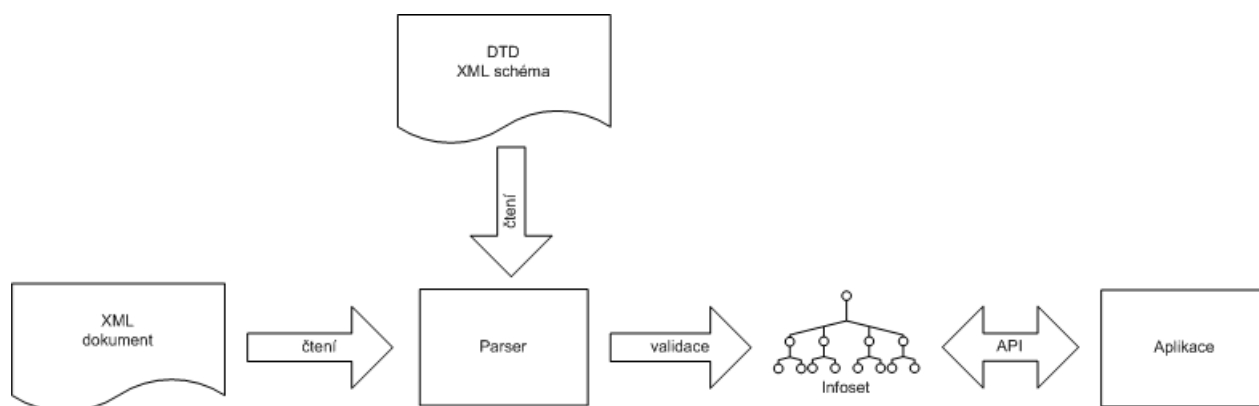
Rozhraní pro přístup k dokumentu XML

- sekvenční (proudové)
 - velmi rychlé a paměťově nenáročné
 - dokument musíme zpracovat během jednoho průchodu
 - standardní rozhraní – SAX, pull-parsery, ...
- stromová reprezentace:
 - celý dokument je zpřístupněn jako hierarchie objektů
 - dokument můžeme opakovaně a nelineárně procházet
 - velká paměťová náročnost, pomalejší než sekvenční parsery
 - pro chybný dokument se stromová reprezentace nevytvoří
 - standardní rozhraní – DOM
 - novější, specializovaná rozhraní – JDOM, DOM4J, XOM, ...
 - generátory tříd (data-binding)

Další charakteristiky parseru

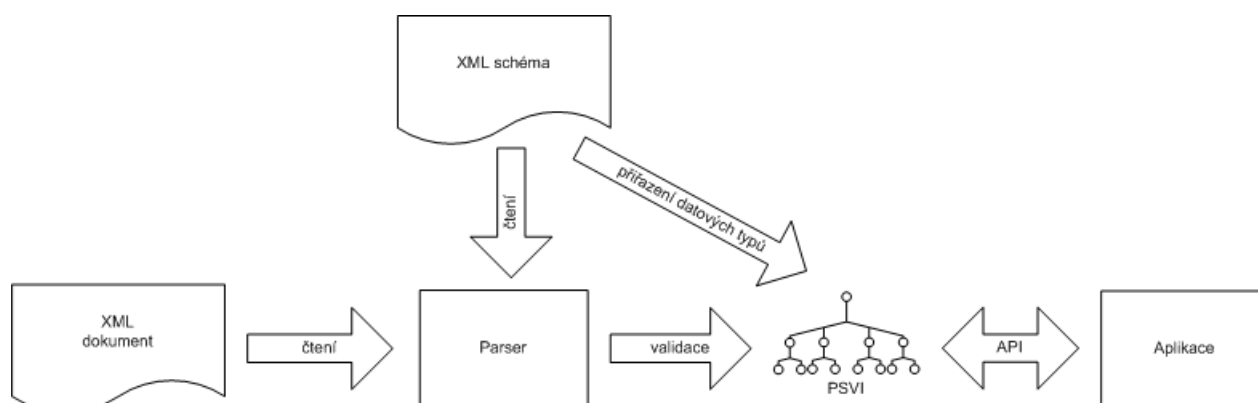
- validující × nevalidující
 - umí kontrolovat dokument oproti DTD, XML Schema, ...

Obrázek 1. Validující parser



- validace je pomalejší než jen kontrola well-formedness
- podpora jmenných prostorů
- podpora XML katalogů
- podpora XInclude
- podporované jazyky a platformy
- podpora PSVI

Obrázek 2. Parser zpřístupňující PSVI

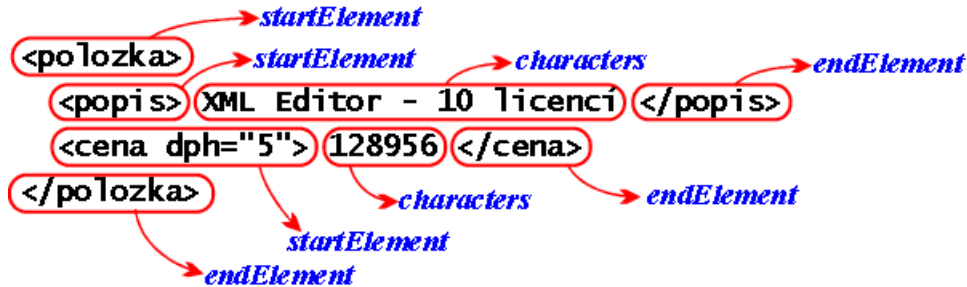


Sekvenční čtení

SAX	8
Pull parsery	9

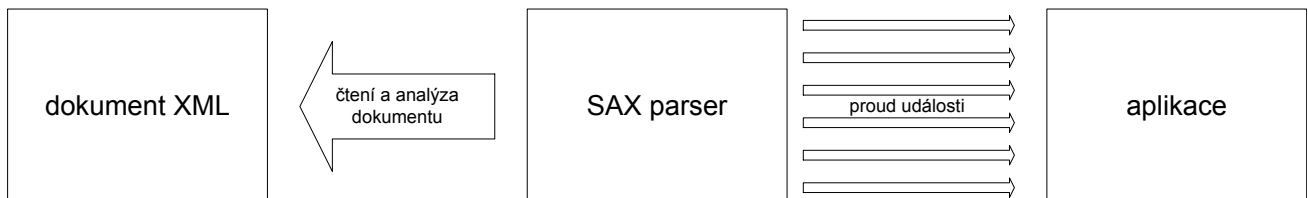
SAX

- SAX = Simple API for XML
- každý důležitý prvek dokumentu je aplikaci předáván ve formě události
- události:



- startDocument, endDocument
- startElement, endElement
- characters
- processingInstructions
- pro každou událost můžeme definovat vlastní obsluhu
- obsluha událostí má podobu třídy implementující rozhraní `ContentHandler`
- SAX je de facto standard podporovaný v mnoha aplikacích

Obrázek 3. Princip push-parseru



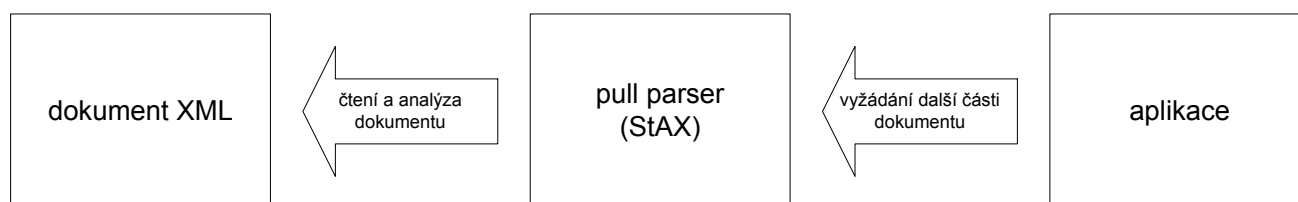
Pull parsers

- části XML dokumentu čteme sekvenčně podle svých potřeb
- přehlednější kód než v případě použití push-parserů (SAX)
- implementace – např. XmlReader v .Net
- postupně přidáváno i do dalších jazyků a knihoven – XmlReader v libxml2, StAX pro Javu, ...

```
// vytvoření pull parseru pro dokument
XmlTextReader reader = new XmlTextReader("faktura.xml");

// průchod celým dokumentem
while (reader.Read())
{
    if (reader.NodeType == XmlNodeType.Element
        && reader.Name == "cena")
    {
        double sazbaDPH = Double.Parse(reader.GetAttribute("dph")) / 100;
        double castka = Double.Parse(reader.ReadString());
        suma += castka;
        sumaDPH += sazbaDPH * castka;
    }
}
```

Obrázek 4. Princip pull-parseru

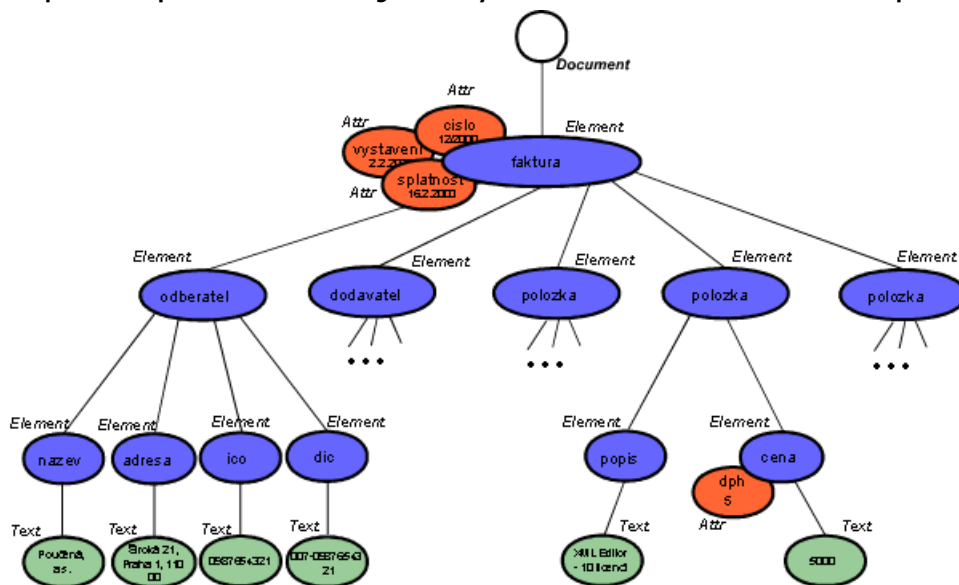


Stromová reprezentace

DOM	11
Hierarchie tříd použitých v DOM	12
Nejběžnější metody DOM	13
Další směr vývoje DOM	14
Rozšíření a varianty DOM	15

DOM

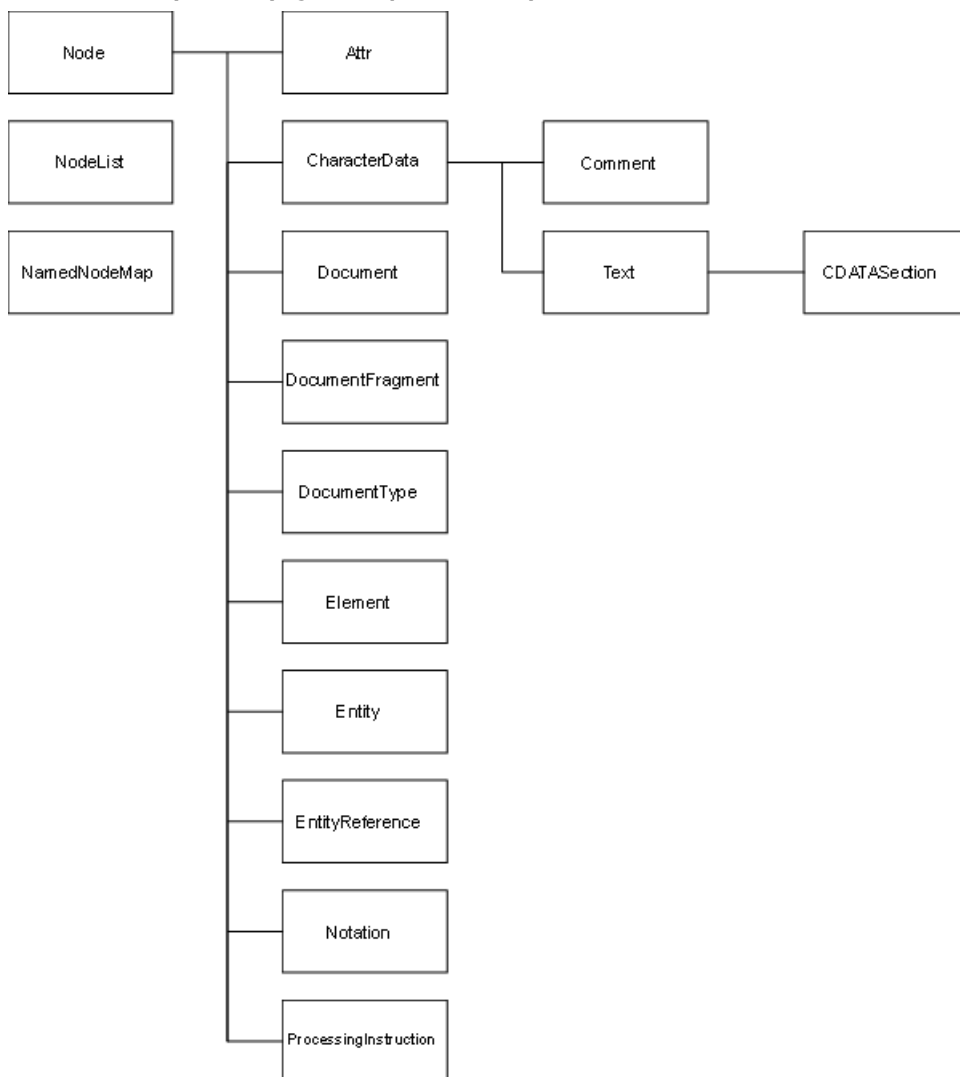
- DOM = Document Object Model
- standard W3C
- verze: DOM1, DOM2 a DOM3
- před zpracováním je celý dokument načten do paměti



- jednotlivé uzly stromu jsou reprezentovány objekty
- rozhraní objektů (použitelné metody) je dáno specifikací DOM

Hierarchie tříd použitých v DOM

- všechny třídy jsou potomky Node



Nejběžnější metody DOM

- informace o uzlu:

```
getNodeName()  
getNodeValue()  
getNodeType()
```

- pohyb po stromu dokumentu:

```
getChildNodes()  
getParentNode()  
getFirstChild()  
getLastChild()  
getPreviousSibling()  
getNextSibling()  
hasChildNodes()
```

- vytváření nových uzlů v paměti:

```
createElement()  
createElementNS()  
createDocumentFragment()  
createTextNode()  
createComment()  
createCDATASection()  
createProcessingInstruction()  
createAttribute()  
createAttributeNS()  
createEntityReference()
```

- modifikace stromu:

```
insertBefore()  
replaceChild()  
removeChild()  
appendChild()
```

- pomocí těchto metod lze udělat cokoliv, ale někdy je to opravdu hodně pracné

Další směr vývoje DOM

- DOM1
 - DOM HTML – práce s HTML stránkou
 - DOM Core – práce s obecným dokumentem XML
- DOM2
 - DOM Events – obsluha událostí
 - DOM Style – manipulace s připojenými kaskádovými styly
 - DOM Traversal and Range – průchod dokumentem (iterátory) a práce s bloky dokumentu
 - Core, HTML – dále rozšířeny
- DOM3
 - DOM Load and Save – rozhraní pro čtení/ukládání do souboru
 - DOM Validation – validace dokumentu, práce se schématem, zjištění elementů přípustných v daném místě apod.
- další připravované části DOM3
 - DOM XPath – vyhodnocování XPath výrazů nad DOM stromem

Rozšíření a varianty DOM

- DOM je na mnoho úloh dost neohrabaný
 - některé problémy jsou postupně odstraňovány (DOM3, DOM3 XPath)
- rozšíření DOM o užitečné metody
 - např. knihovny MSXML a System.Xml umožňují pomocí vlastnosti `Text` zjistit textový obsah libovolného uzlu (včetně poduzlů)
- knihovny optimalizované pro určitý jazyk nebo způsob použití
 - JDOM, DOM4J, XOM
 - snazší navigace po dokumentu, snadné zjištění obsahu elementu
- „lepší“ knihovny umožňují proti DOM stromu pokládat XPath dotaz

Generátory tříd

Úvod	17
Příklad schématu	18
Ukázka rozhraní vygenerované třídy	19
Práce s dokumentem	20

Úvod

- z DTD nebo XML Schématu se vytvoří jednoúčelový parser
- pro každý element se vytvoří samostatná třída
- parser při načtení dokumentu vytvoří v paměti strukturu objektů
- psaní kódu pro čtení a manipulaci s XML dokumentem je velmi jednoduché
- někdy se nazývá i „data-binding“
- lze použít pouze v případě, že schéma je pevně dané a dopředu známé (což většinou je)

Příklad schématu

```
<xsd:element name="faktura">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="odberatel" type="subjektInfoTyp" />
      <xsd:element name="dodavatel" type="subjektInfoTyp" />
      <xsd:element ref="polozka" minOccurs="1"
        maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="cislo" type="xsd:string"
      use="required" />
    <xsd:attribute name="vystaveni" type="xsd:date"
      use="required" />
    <xsd:attribute name="splatnost" type="xsd:date"
      use="required" />
    <xsd:attribute name="vystavil" type="xsd:string" />
  </xsd:complexType>
</xsd:element>

<xsd:complexType name="subjektInfoTyp">
  <xsd:sequence>
    <xsd:element name="nazev" type="xsd:string" />
    <xsd:element name="adresa" type="xsd:string" />
    <xsd:element name="ico" type="xsd:string" />
    <xsd:element name="dic" type="xsd:string" />
  </xsd:sequence>
</xsd:complexType>

<xsd:element name="polozka">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="popis" type="xsd:string"
        minOccurs="0" maxOccurs="1" />
      <xsd:element name="cena" type="xsd:decimal" />
      <xsd:element name="dph" type="xsd:decimal" />
      <xsd:element name="ks" type="xsd:positiveInteger"
        minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Ukázka rozhraní vygenerované třídy

- čím generovat
 - Castor, JAXB – Java
 - **xsd** – .NET

```
public class faktura {  
    public subjektInfoTyp odberatel;  
    public subjektInfoTyp dodavatel;  
    public polozka[] polozka;  
    public string cislo;  
    public System.DateTime vystaveni;  
    public System.DateTime splatnost;  
    public string vystavil;  
}
```

```
public class subjektInfoTyp {  
    public string nazev;  
    public string adresa;  
    public string ico;  
    public string dic;  
}
```

```
public class polozka {  
    public string popis;  
    public System.Decimal cena;  
    public System.Decimal dph;  
    public string ks;  
}
```

Práce s dokumentem

```
// stream pro čtení dat ze souboru
StreamReader reader = new StreamReader("faktura.xml");

// serializátor založený na třídě vygenerované ze schématu
// pomocí: xsd /c faktura.xsd
XmlSerializer serializer = new XmlSerializer(typeof(faktura));

// do objektu f se deserializuje celý dokument faktury
faktura f = (faktura)serializer.Deserialize(reader);

// pomocné proměnné
decimal suma = 0;
decimal sumaDPH = 0;

// sečtení je hračka, nemusíme se starat ani o datové typy
foreach (polozka p in f.polozka)
{
    suma += p.cena;
    sumaDPH += p.cena * (p.dph/100);
}

// výpis statistiky
System.Console.WriteLine("Celkem Kč: " + suma);
System.Console.WriteLine("Celkem DPH: " + sumaDPH);
```

- dokument je samozřejmě R/W, takže jej můžeme v paměti i vytvořit, upravovat a pak uložit

Další API

XSLT, XPath	22
Další vývoj	23

XSLT, XPath

- většina XSLT procesorů má API
- transformace lze spouštět programově
- vstup a výstup transformace nemusí být jen soubor, ale klidně DOM strom, proud událostí SAX apod.
- novější parseery obsahují i procesor XPath a umožňují vyhodnocování výrazů nad dokumentem – velmi pohodlné pro programátora

```
// vytvoření XPath navigátoru pro dokument
XPathDocument doc = new XPathDocument("faktura.xml");
XPathNavigator nav = doc.CreateNavigator();

// sečtení faktury přímo pomocí XPath výrazů
double suma = (double) nav.Evaluate("sum(/faktura/polozka/cena)");
System.Console.WriteLine("Celkem Kč: " + suma);

// vytvoření iterátoru pro všechny elementy cena
XPathNodeIterator iter = nav.Select("/faktura/polozka/cena");

// průchod vybranými uzly
while (iter.MoveNext())
{
    double sazbaDPH = Double.Parse(iter.Current.GetAttribute("dph","")) / ►
100;
    double castka = Double.Parse(iter.Current.Value);
    suma += castka;
    sumaDPH += sazbaDPH * castka;
}

// výpis statistiky
System.Console.WriteLine("Celkem Kč: " + suma);
System.Console.WriteLine("Celkem DPH: " + sumaDPH);
```

Další vývoj

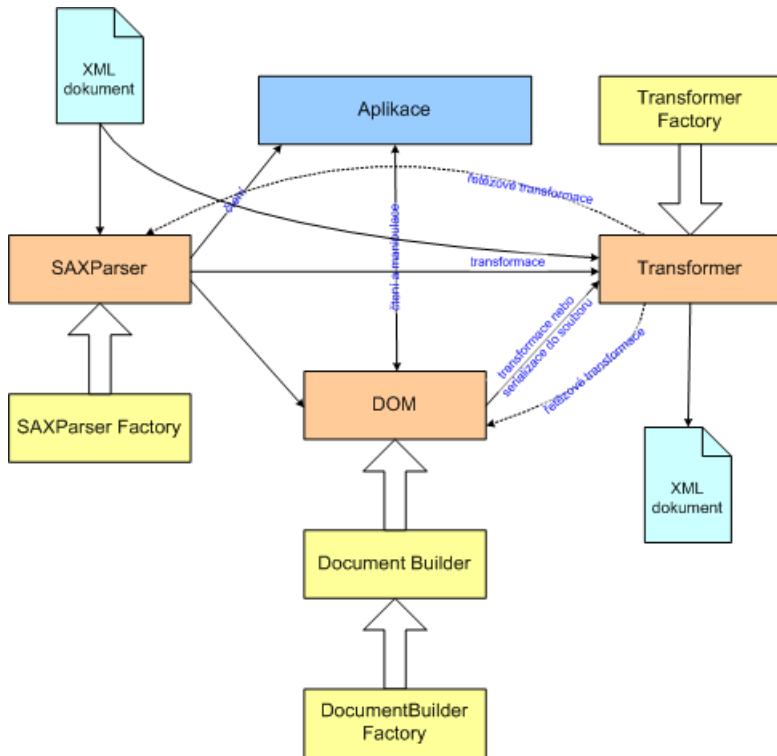
- pro každou novou technologii se dříve nebo později vytvoří standardní API
 - např. XQuery, XML Encryption, XML Signature

Podpora XML v jednotlivých jazycích

Java	25
.Net	26

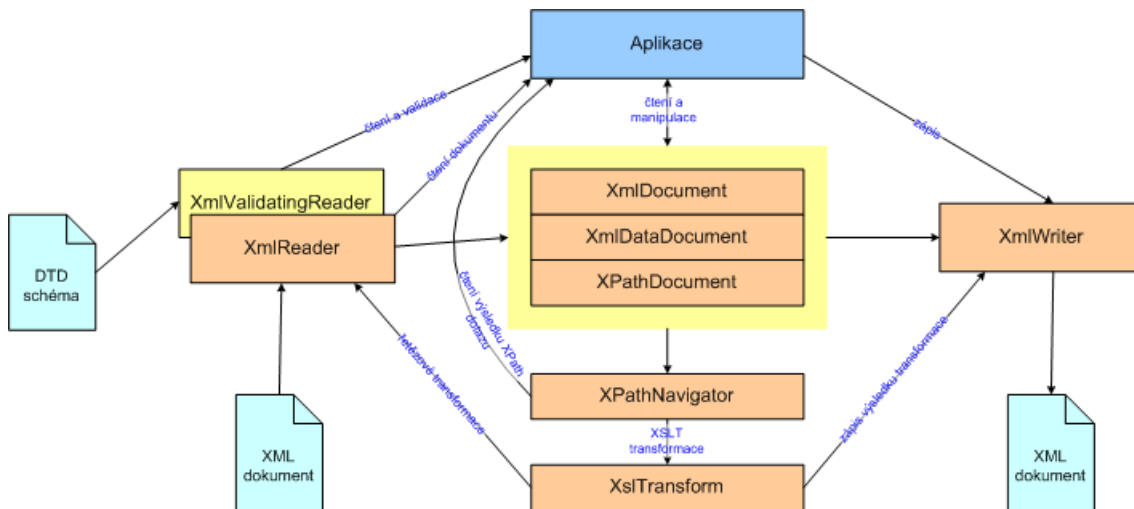
Java

Obrázek 5. Struktura základních tříd rozhraní JAXP a jejich použití



.Net

Obrázek 6. Nejdůležitější XML třídy .NETu a jejich vzájemné vztahy



Další informace

Další informace	28
-----------------------	----

Další informace

- přehled XML API v Javě a .NET¹
- on-line kniha o práci s XML v Javě²
- JAXB – data-binding pro Javu³
- XMLBeans – další data-binding pro Javu⁴
- XML API v .NETu⁵

¹ <http://www.kosek.cz/xml/api/index.html>

² <http://cafeconleche.org/books/xmljava/>

³ <http://java.sun.com/xml/jaxb/index.html>

⁴ <http://java.sun.com/xml/jaxb/index.html>

⁵ <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemxml.asp>