
Kapitola 1. Rozhraní pro práci s XML (SAX, DOM, pull)

Obsah

Základní pojmy	1
Cílem rozhraní je	1
Hlavní typy rozhraní pro zpracování XML dat:	2
Stromově orientovaná rozhraní (Tree-based API)	2
Mapují XML dokument na stromovou strukturu v paměti	2
Modely specifické pro konkrétní prostředí	2
Rozhraní založená na událostech (Event-based API)	2
Při analýze ("parsing") dokumentu "vysílají" zpracovávající aplikaci <i>sled událostí</i>	2
Událostmi je např.:	3
SAX - příklad analýzy dokumentu	3
Kdy zvolit událostmi řízené rozhraní?	3
Vlastnosti (features) nastavitelné pro analýzu - parsing	4
SAX filtry	4
Další odkazy k SAX	4
Rozhraní založená na technice "pull"	4
Rozhraní založená na technice "pull"	4
Streaming API for XML (StAX)	5
StAX - příklad s iterátorem	5
StAX - příklad s kurzorem	6
Document Object Model (DOM)	8
Základní rozhraní pro tvorbu a přístup ke stromové reprezentaci XML dat.	8
Specifický DOM pro HTML dokumenty	8
Odkazy k DOM	9
Implementace DOM	9
Alternativní stromové modely - XOM	9
Alternativní parsery a stromové modely - NanoXML	9
Prakticky dobře použitelný stromový model: dom4j	10
Kombinace stromových a událostmi řízených přístupů	10
Události -> strom	10
Strom -> události	10
Virtuální objektové modely	10

Základní pojmy

Cílem rozhraní je

- poskytnout jednoduchý standardizovaný přístup ke XML datům
- "napojit" analyzátor (parser) na aplikaci a aplikace navzájem
- odstítnit aplikaci od fyzické struktury dokumentu (entity)
- zefektivnit zpracování XML dat

Hlavní typy rozhraní pro zpracování XML dat:


- Stromově orientovaná rozhraní (Tree-based API)
- Rozhraní založená na událostech (Event-based API)
- Rozhraní založená na "vytahování" událostí/prvků z dokumentu (Pull API)

Stromově orientovaná rozhraní (Tree-based API)

Mapují XML dokument na stromovou strukturu v paměti

- dovolují libovolně procházet ("traverse") vzniklý strom;
- nejznámější je *Document Object Model* (DOM) konsorcia W3C, viz <http://www.w3.org/DOM> [<http://www.w3.org/DOM/>]

Modely specifické pro konkrétní prostředí

- pro Javu: JDOM - <http://jdom.org>
- pro Javu: dom4j  [<http://www.instantweb.com/foldoc/foldoc.cgi?dom4j>] - <http://dom4j.org>
- pro Python: 4Suite - <http://4suite.org>

Rozhraní založená na událostech (Event-based API)

Při analýze ("parsing") dokumentu "vysílají" zpracovávající aplikaci *sled událostí*.

- technicky realizováno jako *volání metod* ("callback")
- aplikace poskytuje *handlers*, které volání zachytávají a zpracovávají
- událostmi řízená rozhraní jsou "nižší úrovně" než stromová, protože
- pro aplikaci zůstává "více práce"
- jsou však úspornější na paměť (většinou i čas), samotná analýza totiž nevytváří žádné „trvalé“ objekty

Událostmi je např.:

- začátek a konec dokumentu (start document, end document)
- začátek a konec elementu (start element, end element) - předá současně i atributy
- instrukce pro zpracování (processing instruction)
- komentář (comment)
- odkaz na entitu (entity reference)
- Nejznámějším takovým rozhraním je SAX <http://www.saxproject.org>

SAX - příklad analýzy dokumentu

```
<?xml version="1.0"?>
<doc>
  <para>Hello, world!</para>
</doc>
```

vyprodukuje při analýze (parsingu) sled událostí:

```
start document
start element: doc {seznam atributů: prázdný}
start element: para {seznam atributů: prázdný}
characters: Hello, world!
end element: para
end element: doc
end document
```

Kdy zvolit událostmi řízené rozhraní?

- O co snazší pro autora parseru, o to náročnější pro aplikačního programátora...

- Aplikace si musí (někdy složitě) pamatovat stav analýzy, nemá nikdy "celý dokument pohromadě".
- Na úlohy, které lze řešit "lokálně", bez kontextu celého dokumentu, je to vhodné rozhraní.
- Obvykle poskytuje nejrychlejší možné zpracování.
- Aplikační nepříjemnosti lze obejít použitím nadstaveb, např. Streaming Transformations for XML (STX) [<http://stx.sourceforge.net>]

Vlastnosti (features) nastavitelné pro analýzu - parsing

Chování parseru produkujícího SAX události je možné ovlivnit nastavením tzv. *features* a *properties*.

- *Vlastnosti (features)* nastavitelné pro analýzu (parsing) <http://www.saxproject.org/?selected=get-set>
- Blíže k jednotlivým *properties* a *features* v článku *Use properties and features in SAX parsers* [???] (IBM DeveloperWorks/XML).

SAX filtry

SAX rozhraní nabízí možnost napsat třídu jako tzv. SAX filtr (přesněji implementaci rozhraní `org.xml.sax.XMLFilter`).

Objekt takové třídy na jedné straně události přijímá, zpracuje je a posílá dále.

Další informace k filtrování událostí naleznete např. v článku *Change the events output by a SAX stream* [<http://www.ibm.com/developerworks/xml/library/x-tipsaxfilter/>] (IBM DeveloperWorks/XML).

Další odkazy k SAX

- "Přímo od zdroje" <http://www.saxproject.org>
- SAX Tutorial k JAXP - <http://java.sun.com/webservices/docs/ea1/tutorial/doc/JAXPSAX.html>

Rozhraní založená na technice "pull"

Rozhraní založená na technice "pull"

- Aplikace "nečeká na události", ale "vytahuje si" příslušná data ze vstupního parsovaného souboru.
- Využíváme tam, kde "víme, co ve zdroji očekávat" a "postupně si to bereme"
- ... vlastně opak API řízeného událostmi.

- Z hlediska aplikačního programátora velmi pohodlné, ale implementace bývají o něco pomalejší než klasická "push" událostmi řízená rozhraní.
- Pro Javu existuje *XML-PULL parser API* - viz *Common API for XML Pull Parsing* [<http://www.xmlpull.org/>] a také
- nově vyvíjené rozhraní *Streaming API for XML (StAX)* [<http://www.jcp.org/en/jsr/detail?id=173>] vznikající "shora i zdola" jako produkt JCP (Java Community Process).

Streaming API for XML (StAX)

Toto API se později může stát standardní součástí javového prostředí pro práci s XML, tzv. JAXP.

Nabízí dva přístupy k "pull" zpracování:

- přístup k "vytahovaným" událostem prostřednictvím iterátoru - pohodlnější
- nízkourovňový přístup přes tzv. kurzor - rychlejší

StAX - příklad s iterátorem

Příklad 1.1. StAX - přístup iterátorem

```
import java.io.*;
import java.util.Iterator;
import javax.xml.namespace.QName;
import javax.xml.stream.*;
import javax.xml.stream.events.*;
public class ParseByEvent {
    public static void main(String[] args)
        throws FileNotFoundException, XMLStreamException {
        // Use the reference implementation for the XML input factory
        System.setProperty("javax.xml.stream.XMLInputFactory",
            "com.bea.xml.stream.MXParserFactory");
        // Create the XML input factory
        XMLInputFactory factory = XMLInputFactory.newInstance();
        // Create the XML event reader
        FileReader reader = new FileReader("somefile.xml");
        XMLEventReader r =
            factory.createXMLEventReader(reader);
        // Loop over XML input stream and process events
        while(r.hasNext()) {
            XMLEvent e = r.next();
            processEvent(e);
        }
    }
}
```

```
    }
  }
  /**
   * Process a single XML event
   * @param e - the event to be processed
   */
  private static void processEvent(XMLEvent e) {
    if (e.isStartElement()) {
      QName qname = ((StartElement) e).getName();
      String namespaceURI = qname.getNamespaceURI();
      String localName = qname.getLocalPart();
      Iterator iter = ((StartElement) e).getAttributes();
      while (iter.hasNext()) {
        Attribute attr = (Attribute) iter.next();
        QName attributeName = attr.getName();
        String attributeValue = attr.getValue();
      }
    }
    if (e.isEndElement()) {
      QName qname = ((EndElement) e).getName();
    }
    if (e.isCharacters()) {
      String text = ((Characters) e).getData();
    }
    if (e.isStartDocument()) {
      String version = ((StartDocument) e).getVersion();
      String encoding = ((StartDocument) e).getCharacterEncodingScheme();
      boolean isStandAlone = ((StartDocument) e).isStandalone();
    }
  }
}
```



Poznámka

příklad převzat z Tip: Use XML streaming parsers [<http://www.ibm.com/developerworks/xml/library/x-tipstx>] (IBM DeveloperWorks, sekce XML).

StAX - příklad s kurzorem

Příklad 1.2. StAX - přístup kurzorem

```
import java.io.*;
import javax.xml.stream.*;
```

```
public class ParseByIterator {
    public static void main(String[] args)
        throws FileNotFoundException, XMLStreamException {
        // Use reference implementation
        System.setProperty(
            "javax.xml.stream.XMLInputFactory",
            "com.bea.xml.stream.MXParserFactory");
        // Create an input factory
        XMLInputFactory xmlif = XMLInputFactory.newInstance();
        // Create an XML stream reader
        XMLStreamReader xmlr =
            xmlif.createXMLStreamReader(new FileReader("somefile.xml"));
        // Loop over XML input stream and process events
        while (xmlr.hasNext()) {
            processEvent(xmlr);
            xmlr.next();
        }
    }
    /**
     * Process a single event
     * @param xmlr - the XML stream reader
     */
    private static void processEvent(XMLStreamReader xmlr) {
        switch (xmlr.getEventType()) {
            case XMLStreamConstants.START_ELEMENT :
                processName(xmlr);
                processAttributes(xmlr);
                break;
            case XMLStreamConstants.END_ELEMENT :
                processName(xmlr);
                break;
            case XMLStreamConstants.SPACE :
            case XMLStreamConstants.CHARACTERS :
                int start = xmlr.getTextStart();
                int length = xmlr.getTextLength();
                String text =
                    new String(xmlr.getTextCharacters(), start, length);
                break;
            case XMLStreamConstants.COMMENT :
            case XMLStreamConstants.PROCESSING_INSTRUCTION :
                if (xmlr.hasText()) {
                    String piOrComment = xmlr.getText();
                }
                break;
        }
    }
    private static void processName(XMLStreamReader xmlr) {
```

```
    if (xmlr.hasName()) {
        String prefix = xmlr.getPrefix();
        String uri = xmlr.getNamespaceURI();
        String localName = xmlr.getLocalName();
    }
}
private static void processAttributes(XMLStreamReader xmlr) {
    for (int i = 0; i < xmlr.getAttributeCount(); i++)
        processAttribute(xmlr, i);
}
private static void processAttribute(XMLStreamReader xmlr, int index) {
    String prefix = xmlr.getAttributePrefix(index);
    String namespace = xmlr.getAttributeNamespace(index);
    String localName = xmlr.getAttributeName(index);
    String value = xmlr.getAttributeValue(index);
}
}
```



Poznámka

příklad převzat z Tip: Use XML streaming parsers [<http://www.ibm.com/developerworks/xml/library/x-tipstx>] (IBM DeveloperWorks, sekce XML).



Document Object Model (DOM)

Základní rozhraní pro tvorbu a přístup ke stromové reprezentaci XML dat.

- existují verze *DOM Level 1, 2, 3*
- DOM je obecně *nezávislý* na způsobu analýzy (parsingu) vstupního XML
- Je popsán IDL definicemi+popisy rozhraní v jednotlivých jazycích (zejm. C++ a Java)

Specifický DOM pro HTML dokumenty



- Core (základ) DOM pro HTML je nyní "víceméně" sloučen s DOM pro XML
- určen pro styly CSS
- určen pro programování dynamického HTML (skriptování - VB Script, JavaScript)

- kromě samotného dokumentu model zahrnuje i prostředí prohlížeče (např. window 
[<http://www.instantweb.com/foldoc/foldoc.cgi?window>], history 
[<http://www.instantweb.com/foldoc/foldoc.cgi?history>]...)

Odkazy k DOM

- Tutoriál k JAXP, část věnovaná DOMPart III: XML and the Document Object Model (DOM) [<http://java.sun.com/xml/jaxp/dist/1.1/docs/tutorial/dom/index.html>]
- Portál věnovaný DOM <http://www.oasis-open.org/cover/dom.html>
- Vizuální přehled DOM 1 rozhraní <http://www.xml.com/pub/a/1999/07/dom/index.html>
- Tutoriál "Understanding DOM (Level 2)" na <http://ibm.com/developer/xml> [<http://ibm.com/developer/xml>]

Implementace DOM

- v mnoha parserech, např. Xerces [<http://xml.apache.org>]
- jako součást JAXP (Java API for XML Processing) - <http://java.sun.com/xml/jaxp/index.html>
- i jako samostatné, nezávislé na parserech:
 - např. dom4j  [<http://www.instantweb.com/foldoc/foldoc.cgi?dom4j>] - <http://dom4j.org>
 - EXML  [<http://www.instantweb.com/foldoc/foldoc.cgi?EXML>] (Electric XML) - <http://www.theminelectric.net>

Alternativní stromové modely - XOM

- XOM (*XML Object Model*) vznikl jako one-man-show projekt (autor Eliote Rusty Harold) rozhraní, které je "papežtější než papež" a striktně respektuje model XML dat.
- Motivaci a specifikaci najdete na domovské stránce XOM [<http://cafeconleche.org/XOM/>].
- Tam je též k získání open-source implementace XOM [<http://cafeconleche.org/XOM/xom-1.0d24.zip>] a
- dokumentace API [<http://cafeconleche.org/XOM/apidocs/>].

Alternativní parsery a stromové modely - NanoXML

- velmi malé (co do velikosti kódu) stromové rozhraní a parser v jednom
- dostupné jako open-source na <http://nanoxml.n3.net>
- adaptované též pro mobilní zařízení
- z hlediska rychlosti a paměťové efektivity za běhu ale nejlepší *není*

Prakticky dobře použitelný stromový model: dom4j

- pohodlné, rychlé a paměťově efektivní stromově-orientované rozhraní
- psané pro Javu, optimalizované pro Javu...
- dostupné jako open-source na <http://dom4j.org>
- nabízí perfektní přehled díky "kuchařce" [<http://dom4j.org/cookbook/cookbook.html>]
- dom4j je výkonný, viz srovnání efektivity jednotlivých stromových modelů [<http://www.ibm.com/developerworks/xml/library/x-injava/>]

Kombinace stromových a událostmi řízených přístupů


Události -> strom

- Je např. možné "nezajímavou" část dokumentu *přeskočit* nebo odfiltrovat pomocí sledování událostí a pak
- za "zajímavé" části vytvořit strom v paměti a ten zpracovávat.

Strom -> události

- Vytvoříme strom dokumentu (a zpracujeme ho) a
- strom následně procházíme a generujeme události jako bychom četli výchozí soubor.
- Toto umožňuje snadnou integraci obou typů zpracování v jedné aplikaci

Virtuální objektové modely

- DOM model dokumentu není přítomen v paměti, je zprostředkováván "on demand" při přístupu k jednotlivým uzlům
- spojuje výhody událostmi řízeného a stromového modelu zpracování (rychlost + komfort)
- implementován např. u procesoru Sablotron 
[<http://www.instantweb.com/foldoc/foldoc.cgi?Sablotron>] (např. viz
<http://www.xml.com/pub/a/2002/03/13/sablotron.html> nebo
http://www.gingerall.org/charlie/ga/xml/p_sab.xml)