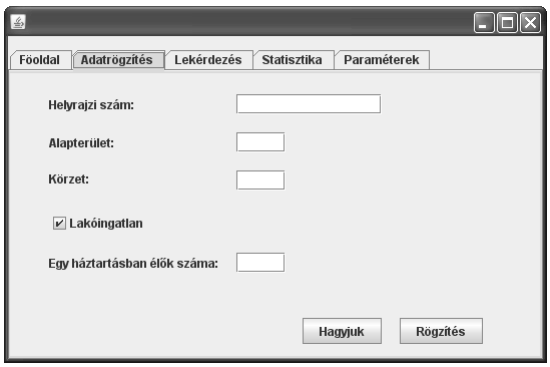
Programozás III

STRING-XML-JSON

BEVEZETÉS



Sok adatkezeléssel kapcsolatos feladat van.

a/ Adatok fájlban

b/ Adatbázisban

c/ Egyéb

KITÉRŐ (KIEGÉSZÍTÉS)

Kiegészítés: a precíz DAO minta:

```
public interface Dao<T, PK extends Serializable> {

    public void create(T t);

    public T read(PK id);

    public void update(T t);

    public void delete(T t);
}
```

ADATKEZELÉSBEN (IS) HASZNOS OSZTÁLYOK: STRING, STRINGBUFFER, STRINGTOKENIZER

Szövegek tárolására, manipulálására alkalmas osztályok. Mindkettő a java.lang csomag része.

A **String** típusú objektum állapota nem, illetve alig változtatható,

a **StringBuffer** típusú objektumok állapota változtatható.

Konkatenáció helyett jobb a **StringBuilder** vagy StringBuffer osztály használata.

vagy:

http://stackoverflow.com/questions/370818/cleanest-way-to-build-an-sql-string-in-java

STRING OSZTÁLY

Mint minden osztályt, a String osztályt is példányosítással kellene létrehoznunk, vagyis így:

String szoveg = new String(” ez egy szöveg”);

De megadhatjuk az eddig megszokott módon is:

String szoveg = ” ez egy szöveg”;

ekkor a rendszer automatikusan elvégzi a példányosítást.

A STRING OSZTÁLY NÉHÁNY METÓDUSA

int length() – visszaadja a szöveg hosszát.

char charAt(int index) – visszaadja az index indexű karaktert.

String toLowerCase(); String toUpperCase()
– visszaadja a szöveg csupa kis-, ill. nagybetűs változatát

String replace(char regi, char uj) – visszaad egy olyan szöveget, amelyben minden régi karaktert újra cserélt.

boolean equals(Object valami)
– összehasonlítja az objektumot a paraméterként megadott másik objektummal.

stb.

A STRINGBUFFER OSZTÁLY

Egy String objektumot nem tudunk lényegesen megváltoztatni.
Ha ilyen feladatunk van, akkor a StringBuffer osztályt kell példányosítanunk.

Az így deklarált szövegek
- bővíthetők (append metódus),
- ki lehet törölni belőlük részleteket (delete, deleteCharAt metódusok),
- ki lehet cserélni rész-szöveget (replace metódus),
- meg lehet fordítani az objektum szövegét (reverse metódus),
- stb. – ld. help

STRINGBUFFER OSZTÁLY – PÉLDA

```
import input.Input;

public class Forditas{

   public static void main(String args[]){

        StringBuffer be= new StringBuffer("");
        String s;

        System.out.println("Írjon ide egy mondatot: ");
        s = Input.readln();
        System.out.println("A mondat megfordítva: ");
        System.out.println(be.append(s).reverse());
    }
}
```

STRINGTOKENIZER OSZTÁLY

A java.util csomag osztálya.

Segítségével egy szöveg könnyen részekre bontható.

Alapértelmezett elválasztójelek: fehér szóköz
(szóköz, tabulátor, sor- és lapvég-jelek)

De mi magunk is definiálhatunk elválasztójeleket.

STRINGTOKENIZER OSZTÁLY

Néhány metódus:

boolean hasMoreTokens()

String nextToken()

String nextToken(String sajat_elvalasztojel)

int countTokens()

STRINGTOKENIZER OSZTÁLY – PÉLDA

```
import input.Input;
import java.util.*;

public class Token{

   public static void main(String args[]){

       System.out.println("Írjon egy mondatot: ");
       String mondat = Input.readln();
       StringTokenizer st = new StringTokenizer(mondat);

       System.out.println("A szavak száma: " + st.countTokens());

       System.out.println("A szavak: ");
       int sorszam = 1;
       while(st.hasMoreTokens())
           System.out.println(sorszam++ + ".:" + st.nextToken());
   }
}
```

STRINGUTILS OSZTÁLY

Nem része a JDK-nak, Apache Commons termék

org.apache.commons.lang3

Class StringUtils

java.lang.Object
    org.apache.commons.lang3.StringUtils

```
public class StringUtils
extends Object
```

Operations on `String` that are `null` safe.

- **IsEmpty/IsBlank** - checks if a String contains text
- **Trim/Strip** - removes leading and trailing whitespace
- **Equals** - compares two strings null-safe

https://commons.apache.org/proper/commons-lang/javadocs/api-release/index.html

STRINGUTILS OSZTÁLY

Külön hozzá kell csatolnunk a programunkhoz, de célszerű Maven projektben függőségként megadni:

```
<dependency>
   <groupId>org.apache.commons</groupId>
   <artifactId>commons-lang3</artifactId>
   <version>3.0</version>
</dependency>
```

Érdemes más csomagokat is megnézni!

https://commons.apache.org/

EZ MI?

```
<?xml version="1.0" encoding="UTF-8" ?>

<Form version="1.3" type="org.netbeans.modules.form.forminfo.JFrameFormInfo">
  <Properties>
    <Property name="defaultCloseOperation" type="int" value="3"/>
  </Properties>
  <SyntheticProperties>
    <SyntheticProperty name="formSizePolicy" type="int" value="1"/>
  </SyntheticProperties>

  <Layout>
    <DimensionLayout dim="0">
      <Group type="103" groupAlignment="0" attributes="0">
          <EmptySpace min="0" pref="400" max="32767" attributes="0"/>
      </Group>
    </DimensionLayout>
    <DimensionLayout dim="1">
      <Group type="103" groupAlignment="0" attributes="0">
          <EmptySpace min="0" pref="300" max="32767" attributes="0"/>
      </Group>
    </DimensionLayout>
  </Layout>
</Form>
```

Egy üres, frissen generált JFrame form.

EZ MI?

Több XML fájl is szerepel a projektben. Mi lehet a szerepük?

Pl.: a project.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://www.netbeans.org/ns/project/1">
    <type>org.netbeans.modules.java.j2seproject</type>
    <configuration>
        <data xmlns="http://www.netbeans.org/ns/j2se-project/3">
            <name>oras</name>
            <source-roots>
                <root id="src.dir"/>
            </source-roots>
            <test-roots>
                <root id="test.src.dir"/>
            </test-roots>
        </data>
    </configuration>
</project>
```

MI AZ AZ XML?

Az XML általános célú platform-független nyelv, amely az adat-formátumok használatánál biztosítja a teljes hordoz-hatóságot.

Nem határozza meg
– a nyelv jelölőelem-készletét (tag set) és
– a nyelvtanát sem.

ezért teljes mértékben kiterjeszthető (innen ered a neve is)
Ez a lényeges különbség közte és a HTML között.

MI AZ AZ XML?

Létrehozását a nyílt rendszerek térhódítása és az Internet tette szükségessé.

A Java nyelv és az XML közötti érdekes hasonlóság:
- a Java a futtatható formában hordozható programok nyelve,
- az XML a hordozható adat készítésének az eszköze.

Főként adatok rugalmas kezelésére használható.

Ezért válhat fontossá Java programok esetén is.

MI AZ AZ XML?

Az XML dokumentumokban az adatok értékein túl olyan további címkéket és hivatkozásokat helyezhetünk el, amelyek utalnak az adat természetére, a dokumentum szerkezeti és tartalmi felépítésére, továbbá ezek az információk felhasználhatóak a dokumentum érvényességének vizsgálatához is.

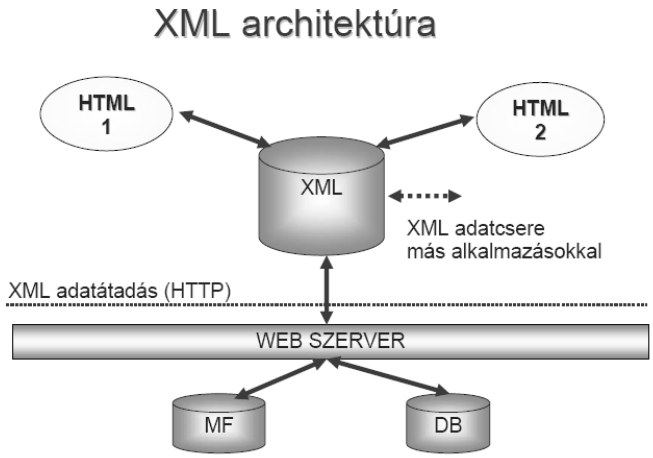
XML ALKALMAZÁSOK

XML: egyszerű adathordozhatóság

A programok jobban tudnak egymással kommunikálni

A lehető legtávolabbi platformon levő gépek rendszerei között is ki tud alakulni egy kommunikáció.

XML



AZ XML FOGALMA

XML: E**X**tensible **M**arkup **L**anguage (kiterjeszthető leíró nyelv)

a W3C (World Wide Web Consortium – http://www.w3c.hu/) által ajánlott általános célú leíró nyelv speciális célú leíró nyelvek létrehozására**.**

Egy XML dokumentum elemekből áll – ***szókincs***
az elemek egymáshoz való kapcsolata és tartalma szabályokkal rögzíthető – ***nyelvtan***

Szintaxis:
– az XML dokumentumok (vagyis az elemek) jelölésére
– a szabályok leírására (DTD–Documentum Type Definition).

⟹ bárki saját nyelvet (*dokumentum-típust*) készíthet.

AZ XML FOGALMA

Az XML tehát:

• *bővíthető* (eXtensible), mert saját elemeket lehet deklarálni;

• *jelölő* (Markup), mert az elemek – egy megadott módon – jelöléssel különböztethetőek meg egymástól;

• *nyelv* (Language), mert rögzíthető a szókincs és a szabályrendszer

De ugyanakkor rugalmas:

Tetszőleges elemkészlet, tetszőleges nyelvtan alapján saját jelölőnyelv készíthető.

AZ XML FOGALMA

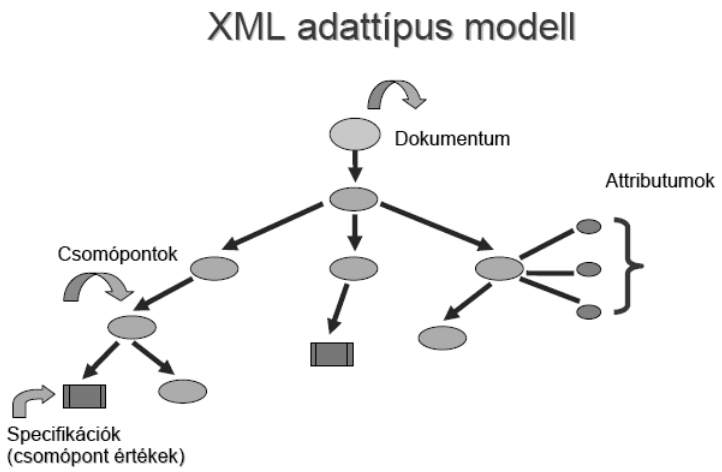
Az XML :

- (általában) *hierarchikus-felépítésű,*
- *platform-független,*
- (ember és gép számára) *könnyen értelmezhető,*
- *egyszerűen verifikálható* (ellenőrizhető)

adatleíró nyelv.

AZ XML FOGALMA

Az XML (hierarchikus) **struktúrája:**



AZ XML FOGALMA

Az XML (hierarchikus) **struktúrája:**

A struktúra legfelső szintje két részből áll:

– Fejrész (XML deklaráció) – pl.:
<?xml version="1.0" encoding="UTF-8"?>

– Dokumentum-elem (gyökér) – pl.:
<uzenet>
az üzenet törzse
</uzenet>

XML PÉLDÁK

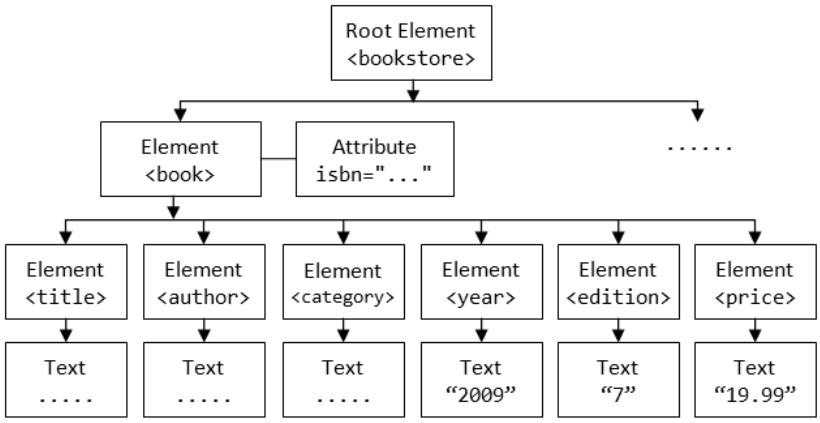
```
– <Personnel>
  – <Employee type="állandósított">
      <Name>Baranyai Endre</Name>
      <Id>1234</Id>
      <BirthYear>1980</BirthYear>
    </Employee>
  – <Employee type="szerződéses">
      <Name>Tolnai Pál</Name>
      <Id>2345</Id>
      <BirthYear>1990</BirthYear>
    </Employee>
  – <Employee type="állandósított">
      <Name>Somogyi Péter</Name>
      <Id>4567</Id>
      <BirthYear>1985</BirthYear>
    </Employee>
</Personnel>
```

```
– <gui-definition>
  – <colors>
      <background>#808080</background>
      <text>#000000</text>
      <header>#008000</header>
      <link normal="#000080" visited="#800080"/>
      <default>${colors.header}</default>
    </colors>
    <rowsPerPage>15</rowsPerPage>
  – <buttons>
      <name>OK,Cancel,Help</name>
    </buttons>
    <numberFormat pattern="###\,###.##"/>
</gui-definition>
```

Van néhány szintaktikai előírás.

XML SZERKEZET

Jól formált, ha megfelel az xml specifikációnak – ez könnyen ellenőrizhető. A jól formált xml fa struktúrájú:



JAVA – XML

Miért fontos az XML használata?

Újabb absztrakciós szint:
nem kell mindig újraírni a hasonló feladatokat, ha egy jól definiált XML struktúra meg tudja adni a szereplő objektumokat, és ezt az XML-t dolgozza fel egy Java program.

Segítségével pl. konfigurációs fájlokat is készíthetünk, és sok egyéb alkalmazási lehetősége van.

JAVA – XML

XML létrehozása – pl.:

```
public class FibonacciXML {
    public static void main(String[] args) {
        int egyik = 1;
        int masik = 1;
        int uj;
        System.out.println("<?xml version=\"1.0\"?>");
        System.out.println("<Fibonacci_szamok>");
        for (int i = 0; i < 10; i++) {
            System.out.print("  <fibonacci>");
            System.out.print(egyik);
            System.out.println("</fibonacci>");
            uj = egyik + masik;
            egyik = masik;
            masik = uj;     }
        System.out.println("</Fibonacci_szamok>");    }  }
```

JAVA – XML

Eredmény:

```
<?xml version="1.0" ?>
- <Fibonacci_szamok>
    <fibonacci>1</fibonacci>
    <fibonacci>1</fibonacci>
    <fibonacci>2</fibonacci>
    <fibonacci>3</fibonacci>
    <fibonacci>5</fibonacci>
    <fibonacci>8</fibonacci>
    <fibonacci>13</fibonacci>
    <fibonacci>21</fibonacci>
    <fibonacci>34</fibonacci>
    <fibonacci>55</fibonacci>
  </Fibonacci_szamok>
```

JAVA – XML

XML használata:

XML dokumentumot létrehozni viszonylag könnyű,
feldolgozni (olvasni) korántsem az.

Szerencsére a Java-ban van XML elemző (XML parser) – ez fel tudja dolgozni az adott XML fájlt.

Az XML parser
- ellenőrzi a dokumentum (formai) helyességét és érvényességét
- feldolgozza az XML tartalmát

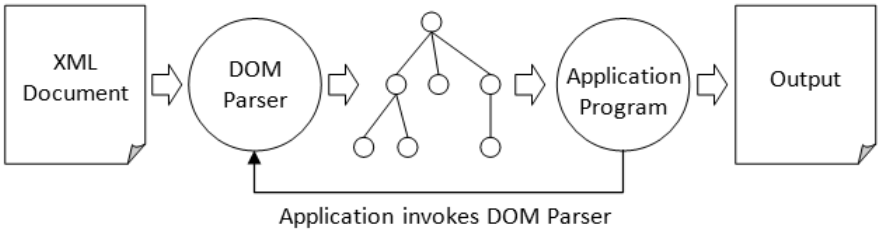
javax.xml; javax.xml. ... csomagok

JAVA – XML-FELDOLGOZÓK (parser)

DOM (Document Object Model):

Végigelemzi az xml fájlt, létrehozza a DOM objektumokat, melyeket egy fa struktúrába fűzve tárol a memóriában.

Ez a fa bejárható, ill. gráf-algoritmusokkal manipulálható.



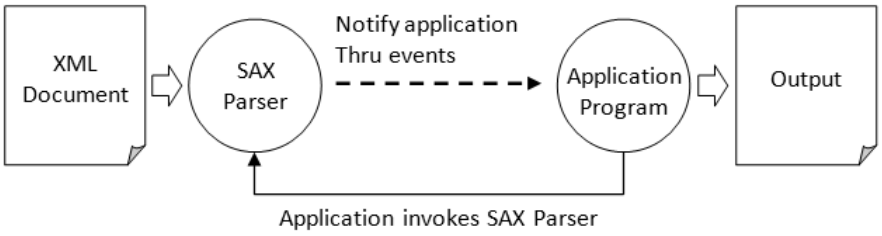
Használatával írható, olvasható az XML fájl.

JAVA – XML-FELDOLGOZÓK (parser)

SAX (Simple API for XML):

Gyorsabb, és kevesebb memóriát igényel, mint a DOM, de kicsit bonyolultabb. Írásra nem alkalmas.

Ez egy esemény alapú modell. Amikor végigelemzi az XML fájlt, akkor az egyes tag-ek hatására kiváltódik egy-egy esemény, és meghívódnak a hozzá kapcsolódó metódusok.



JAVA – XML-FELDOLGOZÓK (parser)

JAXB (Java Architecture for XML Binding) :

modernebb, annotációkkal dolgozik.

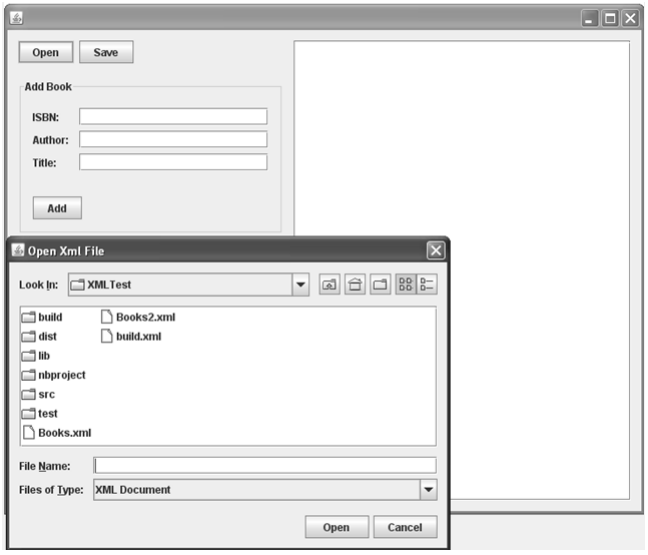
Marshalling: Egy objektumot írható formára konvertál.

http://stackoverflow.com/questions/770474/what-is-the-difference-between-serialization-and-marshaling

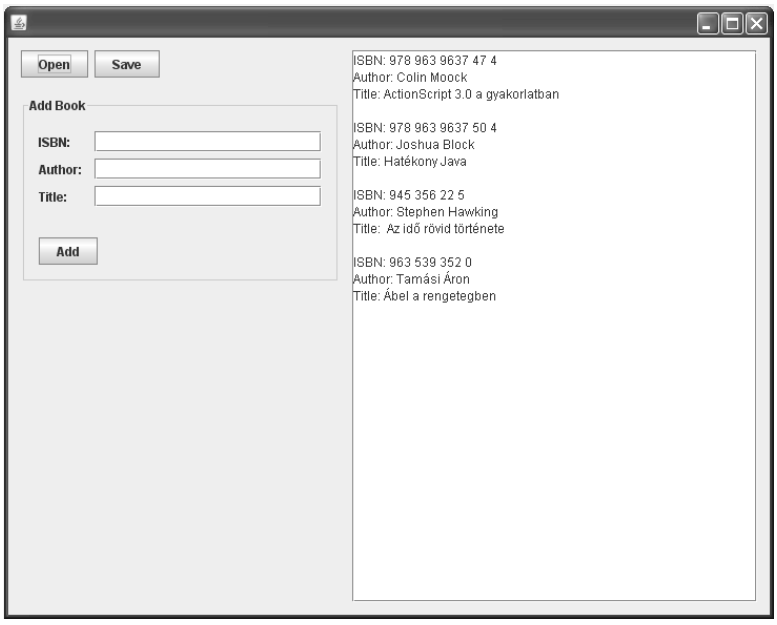
JAVA – XML (ÍRÁS-OLVASÁS PÉLDA)

Példa:

XML fájlból olvassa be az adatokat, majd kiírja a módosított fájlt.



JAVA – XML (ÍRÁS-OLVASÁS PÉLDA)



JAVA – XML (ÍRÁS-OLVASÁS PÉLDA)

Books.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<Books>
        <Book Author="Colin Moock"
              ISBN="978 963 9637 47 4"
             Title="ActionScript 3.0 a gyakorlatban"/>
       <Book Author="Joshua Block"
              ISBN="978 963 9637 50 4"
             Title="Hatékony Java"/>
       <Book Author="Stephen Hawking"
              ISBN="945 356 22 5"
             Title=" Az idő rövid története"/>
       <Book Author="Tamási Áron"
             ISBN="963 539 352 0"
             Title="Ábel a rengetegben"/>
</Books>
```

JAVA – XML (ÍRÁS-OLVASÁS PÉLDA)

Megoldás:

Ld. mintapeldak\XMLTest
(szerző: Krasznai Dávid)

JSON

JSON (**JavaScript Object Notation**):
Kis méretű, szöveg alapú szabvány ember által olvasható adatcserére.
A JavaScript nyelvből alakult ki egyszerű adatstruktúrák és asszociatív tömbök reprezentálására.

Legtöbbször egy szerver és egy kliens számítógép közti adatátvitelre használják (főleg AJAX technológiával) az XML egyik alternatívájaként. Általánosságban strukturált adatok tárolására, továbbítására szolgál.

MIÉRT JÓ A JSON?

– Mivel szöveg, ezért könnyen lehet átküldeni egy szerverre vagy szerverről.
– Bármelyik nyelv könnyen használja adatformátumként.

Egy JSON formátumban kapott adat könnyen alakítható Java objektummá (és fordítva).

Használható library: jsonXXX.jar

```
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.json/json -->
    <dependency>
        <groupId>org.json</groupId>
        <artifactId>json</artifactId>
        <version>20171018</version>
    </dependency>
</dependencies>
```

HOL TALÁLKOZOTT VELE? ☺

JSON vs XML

JSON

```
{
  "name": "Gipsz Jakab",
  "eha": "GIJUAAP.PTE",
  "birth_date": "1990.01.02"
}
```

XML

```
<stundent>
    <name>Gipsz Jakab</name>
    <eha>GIJUAAP.PTE</eha>
    <birthdate>1990.01.02</birthdate>
</student>
```

EGYSZERŰ MINTAPÉLDA JSON OLVASÁSÁRA, ÍRÁSÁRA

Ld. mintapeldak\
JSONmaven1\
JSONmaven2

FELHASZNÁLT ÉS AJÁNLOTT IRODALOM

https://www3.ntu.edu.sg/home/ehchua/programming/java/J6d_xml.html

http://tutorials.jenkov.com/java-xml/index.html

http://www.vogella.com/articles/JavaXML/article.html

http://www.vogella.com/tutorials/JAXB/article.html

http://www.mkyong.com/java/how-to-read-xml-file-in-java-dom-parser/

http://www.mkyong.com/java/how-to-read-xml-file-in-java-sax-parser/

http://www.roseindia.net/xml/dom/

http://www.java-samples.com/showtutorial.php?tutorialid=152

http://www.code-thrill.com/2012/05/configuration-that-rocks-with-apache.html

http://stackoverflow.com/questions/370818/cleanest-way-to-build-an-sql-string-in-java

+ google