**Java ve XML Örnekleri**

**Database’den xml oluşturma**

|  |
| --- |
| **import** java.io.FileNotFoundException;  **import** java.io.FileOutputStream;  **import** java.io.PrintStream;  **import** java.sql.Connection;  **import** java.sql.DriverManager;  **import** java.sql.ResultSet;  **import** java.sql.ResultSetMetaData;  **import** java.sql.SQLException;  **import** java.sql.Statement;  **public** **class** DBtoXML {    **public** **static** **void** main(String[] args) {  **try** {  Class.*forName*("sun.jdbc.odbc.JdbcOdbcDriver");  String user = "";  String password = "";  String dbPath = "C:\\java\\IleriJava\\sql\\ornek.mdb";  String urlDB = "jdbc:odbc:driver={Microsoft Access Driver (\*.mdb)};DBQ=" + dbPath;  // Connection con = DriverManager.getConnection("jdbc:odbc:test",  // user, password);  Connection con = DriverManager.*getConnection*(urlDB, user, password);  Statement stmt = con.createStatement();  String sql = "select \* from kimlik";  ResultSet rs = stmt.executeQuery(sql);  ResultSetMetaData rsmd = rs.getMetaData();  **int** sutunSayisi = rsmd.getColumnCount();  FileOutputStream fo = **new** FileOutputStream("personel.xml");  PrintStream pr = **new** PrintStream(fo);  pr.println("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");  pr.println("<PersonelBilgi>");  **while** (rs.next()) {  pr.println(" <Personel>");  **for** (**int** i = 1; i <= sutunSayisi; i++)  pr.println(" <" + rsmd.getColumnName(i) + ">"  + rs.getString(i) + "</" + rsmd.getColumnName(i)  + ">");  pr.println(" </Personel>");  }  pr.println("</PersonelBilgi>");  } **catch** (ClassNotFoundException e) {  System.*out*.println("surucu bulunamadi..");  } **catch** (SQLException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  } **catch** (FileNotFoundException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  }  } |

**xml oluşturma**

|  |
| --- |
| **import** javax.xml.parsers.DocumentBuilder;  **import** javax.xml.parsers.DocumentBuilderFactory;  **import** javax.xml.parsers.ParserConfigurationException;  **import** javax.xml.transform.Transformer;  **import** javax.xml.transform.TransformerConfigurationException;  **import** javax.xml.transform.TransformerException;  **import** javax.xml.transform.TransformerFactory;  **import** javax.xml.transform.dom.DOMSource;  **import** javax.xml.transform.stream.StreamResult;  **import** org.w3c.dom.Attr;  **import** org.w3c.dom.Document;  **import** org.w3c.dom.Element;  **public** **class** XMLOlustur {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  DocumentBuilderFactory docFactory = DocumentBuilderFactory  .*newInstance*();  **try** {  DocumentBuilder docBuilder = docFactory.newDocumentBuilder();  Document doc = docBuilder.newDocument();  Element kokEleman = doc.createElement("ogrenci-kayit");  doc.appendChild(kokEleman);  //ogrenci elemani olusturuluyor(root'a eklenecek)  Element ogrenci = doc.createElement("ogrenci");  // ogrenci numarasi ogrenci elemanina attribute olarak eklenecek  Attr ogrenciNo = doc.createAttribute("no");  ogrenciNo.setValue("1");  ogrenci.setAttributeNode(ogrenciNo);//ogrenci elemanina eklendi  //ogrenci elemanina gelecek ad olusturuluyor  Element ad=doc.createElement("ad");  ad.appendChild(doc.createTextNode("Ahmet"));//ad elemanina deger atandi(text node)  ogrenci.appendChild(ad);//ad elemani ogrenci elemaninin altina ekleniyors  kokEleman.appendChild(ogrenci);//ogrenci elemani root'a ekleniyor    //Elde edilen dom ekran veya dosyaya yaziliyor  TransformerFactory transformerFactory = TransformerFactory  .*newInstance*();  Transformer transformer = transformerFactory.newTransformer();  DOMSource source = **new** DOMSource(doc);  StreamResult result = **new** StreamResult(System.*out*);  transformer.transform(source, result);  } **catch** (ParserConfigurationException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  } **catch** (TransformerConfigurationException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  } **catch** (TransformerException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  }  } |

**Database’den okuyup DOM ile yazma**

|  |
| --- |
| **import** java.io.FileNotFoundException;  **import** java.io.FileOutputStream;  **import** java.sql.Connection;  **import** java.sql.DriverManager;  **import** java.sql.ResultSet;  **import** java.sql.ResultSetMetaData;  **import** java.sql.SQLException;  **import** java.sql.Statement;  **import** javax.xml.parsers.DocumentBuilder;  **import** javax.xml.parsers.DocumentBuilderFactory;  **import** javax.xml.parsers.ParserConfigurationException;  **import** javax.xml.transform.Transformer;  **import** javax.xml.transform.TransformerConfigurationException;  **import** javax.xml.transform.TransformerException;  **import** javax.xml.transform.TransformerFactory;  **import** javax.xml.transform.dom.DOMSource;  **import** javax.xml.transform.stream.StreamResult;  **import** org.w3c.dom.Document;  **import** org.w3c.dom.Element;  **public** **class** DBtoXMLDOM {  **public** **static** **void** main(String[] args) {  **try** {  Class.*forName*("sun.jdbc.odbc.JdbcOdbcDriver");  String user = "";  String password = "";  String dbPath = "D:\\java\\ileri\\xml\\ornek.mdb";  String urlDB = "jdbc:odbc:driver={Microsoft Access Driver (\*.mdb)};DBQ="  + dbPath;  // Connection con = DriverManager.getConnection("jdbc:odbc:test",  // user, password);  Connection con = DriverManager.*getConnection*(urlDB, user, password);  Statement stmt = con.createStatement();  String sql = "select \* from kimlik";  ResultSet rs = stmt.executeQuery(sql);  ResultSetMetaData rsmd = rs.getMetaData();  **int** sutunSayisi = rsmd.getColumnCount();  FileOutputStream fo = **new** FileOutputStream("personel1.xml");  DocumentBuilderFactory docFactory = DocumentBuilderFactory  .*newInstance*();  DocumentBuilder docBuilder = docFactory.newDocumentBuilder();  Document doc = docBuilder.newDocument();  Element kokEleman = doc.createElement("PersonelBilgi");  doc.appendChild(kokEleman);  // Personel elemani olusturuluyor(root'a eklenecek)    // Veri tabani satir satir okunuyor  **while** (rs.next()) {  Element personel = doc.createElement("Personel");  **for** (**int** i = 1; i <= sutunSayisi; i++) {  // sutun elemani olusuyor  Element sutunAd = doc.createElement(rsmd.getColumnName(i));  // sutun elemaninin icine text konuyor  sutunAd.appendChild(doc.createTextNode(rs.getString(i)));  // sutun personele ekleniyor  personel.appendChild(sutunAd);  }  // personel elemani root'a ekleniyor    kokEleman.appendChild(personel);  // Elde edilen dom ekran veya dosyaya yaziliyor  }  TransformerFactory transformerFactory = TransformerFactory  .*newInstance*();  Transformer transformer = transformerFactory.newTransformer();  DOMSource source = **new** DOMSource(doc);  StreamResult result = **new** StreamResult(fo);// dosyaya  transformer.transform(source, result);  System.*out*.println("kaydedildi");  } **catch** (ClassNotFoundException e) {  System.*out*.println("surucu bulunamadi..");  } **catch** (SQLException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  } **catch** (FileNotFoundException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  } **catch** (ParserConfigurationException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  } **catch** (TransformerConfigurationException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  } **catch** (TransformerException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  }  } |

|  |
| --- |
| import java.io.FileNotFoundException;  import java.io.FileOutputStream;  import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.ResultSet;  import java.sql.ResultSetMetaData;  import java.sql.SQLException;  import java.sql.Statement;  import javax.xml.parsers.DocumentBuilder;  import javax.xml.parsers.DocumentBuilderFactory;  import javax.xml.parsers.ParserConfigurationException;  import javax.xml.transform.OutputKeys;  import javax.xml.transform.Transformer;  import javax.xml.transform.TransformerConfigurationException;  import javax.xml.transform.TransformerException;  import javax.xml.transform.TransformerFactory;  import javax.xml.transform.dom.DOMSource;  import javax.xml.transform.stream.StreamResult;  import org.w3c.dom.Document;  import org.w3c.dom.Element;  public class DBtoXMLDOM1 {  public static void main(String[] args) {  try {  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");  String user = "";  String password = "";  String dbPath = "D:\\ileriJavaKasim2013\\xml\\ornek.mdb";  String urlDB = "jdbc:odbc:driver={Microsoft Access Driver (\*.mdb)};DBQ="  + dbPath;  // Connection con = DriverManager.getConnection("jdbc:odbc:test",  // user, password);  Connection con = DriverManager.getConnection(urlDB, user, password);  Statement stmt = con.createStatement();  String sql = "select \* from kimlik";  ResultSet rs = stmt.executeQuery(sql);  ResultSetMetaData rsmd = rs.getMetaData();  int sutunSayisi = rsmd.getColumnCount();  FileOutputStream fo = new FileOutputStream("personel2.xml");  DocumentBuilderFactory docFactory = DocumentBuilderFactory  .newInstance();  DocumentBuilder docBuilder = docFactory.newDocumentBuilder();  Document doc = docBuilder.newDocument();  Element kokEleman = doc.createElement("PersonelBilgi");  doc.appendChild(kokEleman);  // Personel elemani olusturuluyor(root'a eklenecek)  // Veri tabani satir satir okunuyor  while (rs.next()) {  Element personel = doc.createElement("Personel");  for (int i = 1; i <= sutunSayisi; i++) {  // sutun elemani olusuyor  String sutAd = rsmd.getColumnName(i);  if (sutAd.equalsIgnoreCase("Dogum")) {  personel.setAttribute(sutAd.toLowerCase(), rs.getString(i));  } else {  Element sutunAd = doc.createElement(sutAd);  // sutun elemaninin icine text konuyor  sutunAd.appendChild(doc.createTextNode(rs.getString(i)));  // sutun personele ekleniyor  personel.appendChild(sutunAd);  }  }  // personel elemani root'a ekleniyor  kokEleman.appendChild(personel);  // Elde edilen dom ekran veya dosyaya yaziliyor  }//dongu sonu(rs.next)  TransformerFactory transformerFactory = TransformerFactory  .newInstance();  Transformer transformer = transformerFactory.newTransformer();  transformer.setOutputProperty(OutputKeys.METHOD, "xml");  transformer.setOutputProperty(OutputKeys.INDENT, "yes");  transformer.setOutputProperty(  "{http://xml.apache.org/xslt}indent-amount", "5");  DOMSource source = new DOMSource(doc);  StreamResult result = new StreamResult(fo);// dosyaya  transformer.transform(source, result);  System.out.println("kaydedildi");  } catch (ClassNotFoundException e) {  System.out.println("surucu bulunamadi..");  } catch (SQLException e) {  // TODO Auto-generated catch block  e.printStackTrace();  } catch (FileNotFoundException e) {  // TODO Auto-generated catch block  e.printStackTrace();  } catch (ParserConfigurationException e) {  // TODO Auto-generated catch block  e.printStackTrace();  } catch (TransformerConfigurationException e) {  // TODO Auto-generated catch block  e.printStackTrace();  } catch (TransformerException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  }  } |

**Transformation**

cdcatalog.xml

|  |
| --- |
| <?xml version="1.0" encoding="ISO-8859-1"?>  <?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>  <catalog>  <cd>  <title>Empire Burlesque</title>  <artist>Bob Dylan</artist>  <country>USA</country>  <company>Columbia</company>  <price>10.90</price>  <year>1985</year>  </cd>  <cd>  <title>Hide your heart</title>  <artist>Bonnie Tyler</artist>  <country>UK</country>  <company>CBS Records</company>  <price>9.90</price>  <year>1988</year>  </cd>  <cd>  <title>Greatest Hits</title>  <artist>Dolly Parton</artist>  <country>USA</country>  <company>RCA</company>  <price>9.90</price>  <year>1982</year>  </cd>  <cd>  <title>Still got the blues</title>  <artist>Gary Moore</artist>  <country>UK</country>  <company>Virgin records</company>  <price>10.20</price>  <year>1990</year>  </cd>  <cd>  <title>Eros</title>  <artist>Eros Ramazzotti</artist>  <country>EU</country>  <company>BMG</company>  <price>9.90</price>  <year>1997</year>  </cd>  <cd>  <title>One night only</title>  <artist>Bee Gees</artist>  <country>UK</country>  <company>Polydor</company>  <price>10.90</price>  <year>1998</year>  </cd>  <cd>  <title>Sylvias Mother</title>  <artist>Dr.Hook</artist>  <country>UK</country>  <company>CBS</company>  <price>8.10</price>  <year>1973</year>  </cd>  <cd>  <title>Maggie May</title>  <artist>Rod Stewart</artist>  <country>UK</country>  <company>Pickwick</company>  <price>8.50</price>  <year>1990</year>  </cd>  <cd>  <title>Romanza</title>  <artist>Andrea Bocelli</artist>  <country>EU</country>  <company>Polydor</company>  <price>10.80</price>  <year>1996</year>  </cd>  <cd>  <title>When a man loves a woman</title>  <artist>Percy Sledge</artist>  <country>USA</country>  <company>Atlantic</company>  <price>8.70</price>  <year>1987</year>  </cd>  <cd>  <title>Black angel</title>  <artist>Savage Rose</artist>  <country>EU</country>  <company>Mega</company>  <price>10.90</price>  <year>1995</year>  </cd>  <cd>  <title>1999 Grammy Nominees</title>  <artist>Many</artist>  <country>USA</country>  <company>Grammy</company>  <price>10.20</price>  <year>1999</year>  </cd>  <cd>  <title>For the good times</title>  <artist>Kenny Rogers</artist>  <country>UK</country>  <company>Mucik Master</company>  <price>8.70</price>  <year>1995</year>  </cd>  <cd>  <title>Big Willie style</title>  <artist>Will Smith</artist>  <country>USA</country>  <company>Columbia</company>  <price>9.90</price>  <year>1997</year>  </cd>  <cd>  <title>Tupelo Honey</title>  <artist>Van Morrison</artist>  <country>UK</country>  <company>Polydor</company>  <price>8.20</price>  <year>1971</year>  </cd>  <cd>  <title>Soulsville</title>  <artist>Jorn Hoel</artist>  <country>Norway</country>  <company>WEA</company>  <price>7.90</price>  <year>1996</year>  </cd>  <cd>  <title>The very best of</title>  <artist>Cat Stevens</artist>  <country>UK</country>  <company>Island</company>  <price>8.90</price>  <year>1990</year>  </cd>  <cd>  <title>Stop</title>  <artist>Sam Brown</artist>  <country>UK</country>  <company>A and M</company>  <price>8.90</price>  <year>1988</year>  </cd>  <cd>  <title>Bridge of Spies</title>  <artist>T`Pau</artist>  <country>UK</country>  <company>Siren</company>  <price>7.90</price>  <year>1987</year>  </cd>  <cd>  <title>Private Dancer</title>  <artist>Tina Turner</artist>  <country>UK</country>  <company>Capitol</company>  <price>8.90</price>  <year>1983</year>  </cd>  <cd>  <title>Midt om natten</title>  <artist>Kim Larsen</artist>  <country>EU</country>  <company>Medley</company>  <price>7.80</price>  <year>1983</year>  </cd>  <cd>  <title>Pavarotti Gala Concert</title>  <artist>Luciano Pavarotti</artist>  <country>UK</country>  <company>DECCA</company>  <price>9.90</price>  <year>1991</year>  </cd>  <cd>  <title>The dock of the bay</title>  <artist>Otis Redding</artist>  <country>USA</country>  <company>Atlantic</company>  <price>7.90</price>  <year>1987</year>  </cd>  <cd>  <title>Picture book</title>  <artist>Simply Red</artist>  <country>EU</country>  <company>Elektra</company>  <price>7.20</price>  <year>1985</year>  </cd>  <cd>  <title>Red</title>  <artist>The Communards</artist>  <country>UK</country>  <company>London</company>  <price>7.80</price>  <year>1987</year>  </cd>  <cd>  <title>Unchain my heart</title>  <artist>Joe Cocker</artist>  <country>USA</country>  <company>EMI</company>  <price>8.20</price>  <year>1987</year>  </cd>  </catalog> |

cdcatalog.xsl

|  |
| --- |
| <?xml version="1.0" encoding="ISO-8859-1"?>  <!-- Edited with XML Spy v2007 (http://www.altova.com) -->  <xsl:stylesheet version="1.0"  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  <xsl:output method='html' version='1.0' encoding='UTF-8' indent='yes'/>  <xsl:template match="/">  <html>  <body>  <h2>My CD Collection</h2>  <table border="1">  <tr bgcolor="#9acd32">  <th align="left">Title</th>  <th align="left">Artist</th>  </tr>  <xsl:for-each select="catalog/cd">  <tr>  <td><xsl:value-of select="title"/></td>  <td><xsl:value-of select="artist"/></td>  </tr>  </xsl:for-each>  </table>  </body>  </html>  </xsl:template>  </xsl:stylesheet> |

|  |
| --- |
| import javax.xml.transform.Result;  import javax.xml.transform.Source;  import javax.xml.transform.Transformer;  import javax.xml.transform.TransformerFactory;  import javax.xml.transform.stream.StreamResult;  import javax.xml.transform.stream.StreamSource;  public class XMLTransform {  public static void main(String[] args) {  try  {  TransformerFactory tf = TransformerFactory.newInstance();  Source xslfile = new StreamSource("cdcatalog.xsl");  Transformer transformer = tf.newTransformer(xslfile);  Source xmlfile = new StreamSource("cdcatalog.xml");  Result output = new StreamResult("abc.html");    transformer.transform(xmlfile, output);  System.out.println("bitti");    }  catch (Exception e) {  e.printStackTrace();  }  }  } |

**SAX Örneği**

|  |
| --- |
| package com.csystem;  import javax.xml.parsers.SAXParser;  import javax.xml.parsers.SAXParserFactory;  import org.xml.sax.helpers.DefaultHandler;  public class SAXExample {  public static void main(String[] args) {  try {  SAXParserFactory factory = SAXParserFactory.newInstance();  SAXParser parser = factory.newSAXParser();  SAXHandler tsh = new SAXHandler();  parser.parse("siparis.xml",tsh);  } catch (Exception e) {  e.printStackTrace();  }  }  } |

|  |
| --- |
| package com.csystem;  import org.xml.sax.Attributes;  import org.xml.sax.SAXException;  import org.xml.sax.helpers.DefaultHandler;  public class SAXHandler extends DefaultHandler{  @Override  public void startDocument() throws SAXException {  System.out.println("Dokuman okunmaya basladi");  }  @Override  public void endDocument() throws SAXException {  System.out.println("Dokuman okumasi bitti");  }    @Override  public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException {  if(qName.equals("urunadi"))  {  System.out.println("<" +qName + ">");  }  }    @Override  public void endElement(String uri, String localName, String qName) throws SAXException {  //System.out.println("</" +qName + ">");  }  @Override  public void characters(char[] ch, int start, int length) throws SAXException {  String s = new String(ch,start,length);  System.out.println(s.trim());  }  } |

siparis1.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <siparisler xmlns=*"http://www.example.org/siparis"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://www.example.org/siparis siparis.xsd "*>  <siparis>  <urunadi>Elma</urunadi>  <miktar>5</miktar>  </siparis>  <siparis>  <urunadi>Portakal</urunadi>  <miktar>4</miktar>  </siparis>  <siparis>  <urunadi>Ayva</urunadi>  <miktar>2</miktar>  </siparis>  </siparisler> |

**DOM**

**Validation (Geçerleme)**

|  |
| --- |
| import javax.xml.XMLConstants;  import javax.xml.transform.Source;  import javax.xml.transform.stream.StreamSource;  import javax.xml.validation.Schema;  import javax.xml.validation.SchemaFactory;  import javax.xml.validation.Validator;  public class XMLSchemaValidation {  public static void main(String[] args) {  try {  SchemaFactory sf = SchemaFactory.newInstance(XMLConstants.W3C\_XML\_SCHEMA\_NS\_URI);  Source schemasource = new StreamSource("Okul.xsd");  Source xmlsource = new StreamSource("Okul.xml");  Schema schema = sf.newSchema(schemasource);  Validator vldtr = schema.newValidator();  vldtr.validate(xmlsource);  System.out.println("valide edildi");    } catch (Exception e) {  System.out.println("Hata");  System.out.println(e.getMessage());  }  }  } |

Okul.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <!-- edited with XMLSpy v2011 rel. 2 (http://www.altova.com) by cdernek (cdernek) -->  <Okul>  <Ogrenci>  <AdSoyad>Mehmet Sahin</AdSoyad>  <Yas>15</Yas>  </Ogrenci>  <Ogrenci>  <AdSoyad>Ali Kaya</AdSoyad>  <Yas>14</Yas>  </Ogrenci>  </Okul> |

Okul.xsd

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"* standalone=*"no"*?>  <!--W3C Schema generated by XMLSpy v2011 rel. 2 (http://www.altova.com)-->  <!--Please add namespace attributes, a targetNamespace attribute and import elements according to your requirements-->  <xs:schema xmlns:xs=*"http://www.w3.org/2001/XMLSchema"* elementFormDefault=*"qualified"*>  <xs:import namespace=*"http://www.w3.org/XML/1998/namespace"*/>  <xs:element name=*"Yas"*>  <xs:complexType mixed=*"true"*/>  </xs:element>  <xs:element name=*"Okul"*>  <xs:complexType>  <xs:sequence>  <xs:choice>  <xs:element ref=*"Ogrenci"* maxOccurs=*"unbounded"*/>  </xs:choice>  </xs:sequence>  </xs:complexType>  </xs:element>  <xs:element name=*"Ogrenci"*>  <xs:complexType>  <xs:sequence>  <xs:sequence>  <xs:element ref=*"AdSoyad"*/>  <xs:element ref=*"Yas"*/>  </xs:sequence>  </xs:sequence>  </xs:complexType>  </xs:element>  <xs:element name=*"AdSoyad"*>  <xs:complexType mixed=*"true"*/>  </xs:element>  </xs:schema> |

**Dökümanı Okuma**

|  |
| --- |
| **import** java.io.IOException;  **import** javax.xml.parsers.DocumentBuilder;  **import** javax.xml.parsers.DocumentBuilderFactory;  **import** javax.xml.parsers.ParserConfigurationException;  **import** org.w3c.dom.Document;  **import** org.w3c.dom.Node;  **import** org.w3c.dom.NodeList;  **import** org.w3c.dom.Text;  **import** org.xml.sax.SAXException;  **public** **class** DOMOku {  **public** DOMOku() {  DocumentBuilderFactory dbf = DocumentBuilderFactory.*newInstance*();  DocumentBuilder db;  **try** {  db = dbf.newDocumentBuilder();  Document doc = db.parse("siparis1.xml");  Node node = (Node) doc;  nodeDolas(node);  } **catch** (ParserConfigurationException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  } **catch** (SAXException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  } **catch** (IOException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  }  **private** **void** nodeDolas(Node node) {  **if** (node.getNodeType() == Node.*TEXT\_NODE*) {    String s = ((Text) node).getData();  **if** (s.length() != 0)  System.*out*.println(s);  }  NodeList list = node.getChildNodes();  **for** (**int** i = 0; i < list.getLength(); i++)  nodeDolas(list.item(i));  }  **public** **static** **void** main(String[] args) {  **new** DOMOku();  }  } |

Document doc = db.parse("siparis1.xml");

TreeDumper td = **new** TreeDumper();

td.dump(doc);

|  |
| --- |
| **import** org.w3c.dom.Document;  **import** org.w3c.dom.Node;  **import** org.w3c.dom.NodeList;  **public** **class** TreeDumper {  **public** **void** dump(Document doc) {  dumpLoop((Node) doc, "");  }  **private** **void** dumpLoop(Node node, String indent) {  **switch** (node.getNodeType()) {  **case** Node.*CDATA\_SECTION\_NODE*:  System.*out*.println(indent + "CDATA\_SECTION\_NODE");  **break**;  **case** Node.*COMMENT\_NODE*:  System.*out*.println(indent + "COMMENT\_NODE");  **break**;  **case** Node.*DOCUMENT\_FRAGMENT\_NODE*:  System.*out*.println(indent + "DOCUMENT\_FRAGMENT\_NODE");  **break**;  **case** Node.*DOCUMENT\_NODE*:  System.*out*.println(indent + "DOCUMENT\_NODE");  **break**;  **case** Node.*DOCUMENT\_TYPE\_NODE*:  System.*out*.println(indent + "DOCUMENT\_TYPE\_NODE");  **break**;  **case** Node.*ELEMENT\_NODE*:  System.*out*.println(indent + "ELEMENT\_NODE");  **break**;  **case** Node.*ENTITY\_NODE*:  System.*out*.println(indent + "ENTITY\_NODE");  **break**;  **case** Node.*ENTITY\_REFERENCE\_NODE*:  System.*out*.println(indent + "ENTITY\_REFERENCE\_NODE");  **break**;  **case** Node.*NOTATION\_NODE*:  System.*out*.println(indent + "NOTATION\_NODE");  **break**;  **case** Node.*PROCESSING\_INSTRUCTION\_NODE*:  System.*out*.println(indent + "PROCESSING\_INSTRUCTION\_NODE");  **break**;  **case** Node.*TEXT\_NODE*:  System.*out*.println(indent + "TEXT\_NODE");  **break**;  **default**:  System.*out*.println(indent + "Unknown node");  **break**;  }  NodeList list = node.getChildNodes();  **for** (**int** i = 0; i < list.getLength(); i++)  dumpLoop(list.item(i), indent + " ");  }  } |

DOCUMENT\_NODE

ELEMENT\_NODE

TEXT\_NODE

ELEMENT\_NODE

TEXT\_NODE

ELEMENT\_NODE

TEXT\_NODE

TEXT\_NODE

ELEMENT\_NODE

TEXT\_NODE

TEXT\_NODE

TEXT\_NODE

ELEMENT\_NODE

TEXT\_NODE

ELEMENT\_NODE

TEXT\_NODE

TEXT\_NODE

ELEMENT\_NODE

TEXT\_NODE

TEXT\_NODE

TEXT\_NODE

ELEMENT\_NODE

TEXT\_NODE

ELEMENT\_NODE

TEXT\_NODE

TEXT\_NODE

ELEMENT\_NODE

TEXT\_NODE

TEXT\_NODE

TEXT\_NODE

bitti

Document doc = db.parse("siparis1.xml");

TreeDumper2 td = **new** TreeDumper2();

td.dump(doc);

|  |
| --- |
| **import** org.w3c.dom.Attr;  **import** org.w3c.dom.CDATASection;  **import** org.w3c.dom.Comment;  **import** org.w3c.dom.Document;  **import** org.w3c.dom.DocumentType;  **import** org.w3c.dom.DocumentFragment;  **import** org.w3c.dom.Element;  **import** org.w3c.dom.Entity;  **import** org.w3c.dom.EntityReference;  **import** org.w3c.dom.NamedNodeMap;  **import** org.w3c.dom.Node;  **import** org.w3c.dom.NodeList;  **import** org.w3c.dom.Notation;  **import** org.w3c.dom.ProcessingInstruction;  **import** org.w3c.dom.Text;  **public** **class** TreeDumper2 {  **public** **void** dump(Document doc) {  dumpLoop((Node) doc, "");  }  **private** **void** dumpLoop(Node node, String indent) {  **switch** (node.getNodeType()) {  **case** Node.*ATTRIBUTE\_NODE*:  dumpAttributeNode((Attr) node, indent);  **break**;  **case** Node.*CDATA\_SECTION\_NODE*:  dumpCDATASectionNode((CDATASection) node, indent);  **break**;  **case** Node.*COMMENT\_NODE*:  dumpCommentNode((Comment) node, indent);  **break**;  **case** Node.*DOCUMENT\_NODE*:  dumpDocument((Document) node, indent);  **break**;  **case** Node.*DOCUMENT\_FRAGMENT\_NODE*:  dumpDocumentFragment((DocumentFragment) node, indent);  **break**;  **case** Node.*DOCUMENT\_TYPE\_NODE*:  dumpDocumentType((DocumentType) node, indent);  **break**;  **case** Node.*ELEMENT\_NODE*:  dumpElement((Element) node, indent);  **break**;  **case** Node.*ENTITY\_NODE*:  dumpEntityNode((Entity) node, indent);  **break**;  **case** Node.*ENTITY\_REFERENCE\_NODE*:  dumpEntityReferenceNode((EntityReference) node, indent);  **break**;  **case** Node.*NOTATION\_NODE*:  dumpNotationNode((Notation) node, indent);  **break**;  **case** Node.*PROCESSING\_INSTRUCTION\_NODE*:  dumpProcessingInstructionNode((ProcessingInstruction) node, indent);  **break**;  **case** Node.*TEXT\_NODE*:  dumpTextNode((Text) node, indent);  **break**;  **default**:  System.*out*.println(indent + "Unknown node");  **break**;  }  NodeList list = node.getChildNodes();  **for** (**int** i = 0; i < list.getLength(); i++)  dumpLoop(list.item(i), indent + " ");  }  /\* Display the contents of a ATTRIBUTE\_NODE \*/  **private** **void** dumpAttributeNode(Attr node, String indent) {  System.*out*.println(indent + "ATTRIBUTE " + node.getName() + "=\""  + node.getValue() + "\"");  }  /\* Display the contents of a CDATA\_SECTION\_NODE \*/  **private** **void** dumpCDATASectionNode(CDATASection node, String indent) {  System.*out*.println(indent + "CDATA SECTION length=" + node.getLength());  System.*out*.println(indent + "\"" + node.getData() + "\"");  }  /\* Display the contents of a COMMENT\_NODE \*/  **private** **void** dumpCommentNode(Comment node, String indent) {  System.*out*.println(indent + "COMMENT length=" + node.getLength());  System.*out*.println(indent + " " + node.getData());  }  /\* Display the contents of a DOCUMENT\_NODE \*/  **private** **void** dumpDocument(Document node, String indent) {  System.*out*.println(indent + "DOCUMENT");  }  /\* Display the contents of a DOCUMENT\_FRAGMENT\_NODE \*/  **private** **void** dumpDocumentFragment(DocumentFragment node, String indent) {  System.*out*.println(indent + "DOCUMENT FRAGMENT");  }  /\* Display the contents of a DOCUMENT\_TYPE\_NODE \*/  **private** **void** dumpDocumentType(DocumentType node, String indent) {  System.*out*.println(indent + "DOCUMENT\_TYPE: " + node.getName());  **if** (node.getPublicId() != **null**)  System.*out*.println(indent + " Public ID: " + node.getPublicId());  **if** (node.getSystemId() != **null**)  System.*out*.println(indent + " System ID: " + node.getSystemId());  NamedNodeMap entities = node.getEntities();  **if** (entities.getLength() > 0) {  **for** (**int** i = 0; i < entities.getLength(); i++) {  dumpLoop(entities.item(i), indent + " ");  }  }  NamedNodeMap notations = node.getNotations();  **if** (notations.getLength() > 0) {  **for** (**int** i = 0; i < notations.getLength(); i++)  dumpLoop(notations.item(i), indent + " ");  }  }  /\* Display the contents of a ELEMENT\_NODE \*/  **private** **void** dumpElement(Element node, String indent) {  System.*out*.println(indent + "ELEMENT: " + node.getTagName());  NamedNodeMap nm = node.getAttributes();  **for** (**int** i = 0; i < nm.getLength(); i++)  dumpLoop(nm.item(i), indent + " ");  }  /\* Display the contents of a ENTITY\_NODE \*/  **private** **void** dumpEntityNode(Entity node, String indent) {  System.*out*.println(indent + "ENTITY: " + node.getNodeName());  }  /\* Display the contents of a ENTITY\_REFERENCE\_NODE \*/  **private** **void** dumpEntityReferenceNode(EntityReference node, String indent) {  System.*out*.println(indent + "ENTITY REFERENCE: " + node.getNodeName());  }  /\* Display the contents of a NOTATION\_NODE \*/  **private** **void** dumpNotationNode(Notation node, String indent) {  System.*out*.println(indent + "NOTATION");  System.*out*.print(indent + " " + node.getNodeName() + "=");  **if** (node.getPublicId() != **null**)  System.*out*.println(node.getPublicId());  **else**  System.*out*.println(node.getSystemId());  }  /\* Display the contents of a PROCESSING\_INSTRUCTION\_NODE \*/  **private** **void** dumpProcessingInstructionNode(ProcessingInstruction node,  String indent) {  System.*out*.println(indent + "PI: target=" + node.getTarget());  System.*out*.println(indent + " " + node.getData());  }  /\* Display the contents of a TEXT\_NODE \*/  **private** **void** dumpTextNode(Text node, String indent) {  System.*out*.println(indent + "TEXT length=" + node.getLength());  System.*out*.println(indent + " " + node.getData());  }  } |

DOCUMENT

ELEMENT: siparisler

ATTRIBUTE xmlns="http://www.example.org/siparis"

TEXT length=30

http://www.example.org/siparis

ATTRIBUTE xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

TEXT length=41

http://www.w3.org/2001/XMLSchema-instance

ATTRIBUTE xsi:schemaLocation="http://www.example.org/siparis siparis.xsd"

TEXT length=42

http://www.example.org/siparis siparis.xsd

TEXT length=3

ELEMENT: siparis

TEXT length=5

ELEMENT: urunadi

TEXT length=7

urunadi

TEXT length=5

ELEMENT: miktar

TEXT length=1

0

TEXT length=3

TEXT length=3

ELEMENT: siparis

TEXT length=5

ELEMENT: urunadi

TEXT length=7

urunadi

TEXT length=5

ELEMENT: miktar

TEXT length=1

0

TEXT length=3

TEXT length=3

ELEMENT: siparis

TEXT length=5

ELEMENT: urunadi

TEXT length=7

urunadi

TEXT length=5

ELEMENT: miktar

TEXT length=1

0

TEXT length=3

TEXT length=1

**jdom ile XML Dosyası Oluşturma**

**jdom.jar** projeye eklenmelidir.

|  |
| --- |
| import java.io.FileOutputStream;  import org.jdom.Document;  import org.jdom.Element;  import org.jdom.output.XMLOutputter;  public class JDOMCreateExample {  public static void main(String[] args) {    Document doc = new Document();  Element calisanlar = new Element("calisanlar");  Element calisan = new Element("calisan");  Element ad = new Element("ad");  Element soyad = new Element("soyad");      ad.setText("Ahmet");  soyad.setText("Demirelli");    calisan.addContent(ad);  calisan.addContent(soyad);    calisanlar.addContent(calisan);  doc.addContent(calisanlar);    XMLOutputter out = new XMLOutputter();  try  {  out.output(doc, new FileOutputStream("sonuc.xml"));  }  catch (Exception e) {  e.printStackTrace();  }      }  } |

**Ekran çıktısı:**

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

<calisanlar><calisan><ad>Ahmet</ad><soyad>Demirelli</soyad></calisan></calisanlar>

**jdom ile XML Dosyası Okuma**

**Dikkat:** Kaynak xml dosyasında eğer bir şema veya dtd ile geçerleme (validation) varsa bu durumda xml dosyasi geçerli ise işlem yapar. Bundan dolayi o satirlari silmekte fayda var.

**jdom.jar** projeye eklenmelidir.

|  |
| --- |
| import java.util.ArrayList;  import java.util.List;  import org.jdom.Document;  import org.jdom.Element;  import org.jdom.input.DOMBuilder;  import org.jdom.input.SAXBuilder;  import org.jdom.transform.XSLTransformer;  public class JDOMOku {  public static void main(String[] args) {  try {  SAXBuilder sb = new SAXBuilder();  Document doc = (Document) sb.build("personel.xml");  Element root = doc.getRootElement();  List sp = root.getChildren("Personel");  for (int i = 0; i < sp.size(); i++) {  Element siparis = (Element) sp.get(i);  List<Element> childs = siparis.getChildren();  for (Element elem : childs) {  String elemAd=elem.getName();  System.out.println("<"+elemAd+">"+siparis.getChildText(elem.getName())+"</"+elemAd+">");  }  // System.out.println(siparis.getChildText("miktar"));  }  // System.out.println(root.getQualifiedName());  } catch (Exception e) {  e.printStackTrace();  }  }  } |

|  |
| --- |
| User Variables  PATH  ;D:\java\apache-ant-1.8.0\bin;C:\jaxb-ri-2.2.6\bin  ------------------------------------------------------  System Variables  ANT\_HOME  D:\java\ileri\apache-ant-1.9.2  classpath  D:\java\ileri\xml\jaxb-ri-2.2.6\lib\jaxb-xjc.jar  JAVA\_HOME  C:\Program Files\Java\jdk1.7.0\_11  JAXB\_HOME  C:\jaxb-ri-2.2.6 |

**JAXB Örneği**

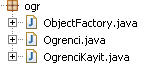
Önce ogrenciKayit.xsd dosyasi kullanilarak gerekli xml sinif eşleşmeleri elde edilir.

JAXB\_HOME ve ANT\_HOME tanıtılmalı. Path’de ant’in ve jaxb’nin bin dizini de eklenmeli

Xjc –p ogr ogrenciKayit.xsd

java -jar D:\java\ileri\xml\jaxb-ri-2.2.6\lib\jaxb-xjc.jar -p ogrenci ogrenciKayit.xsd

Aşağıdaki dosyalar elde edilir.



ogrenci-kayit.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <ogrenci-kayit xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="file:///D:/java/ileri/xml/xslCalisma\_Ocak\_2014/ogrenci-kayit1.xsd">  <ogrenci no="12">  <ad>Ahmet</ad>  <soyad>Akar</soyad>  <dogTar>12-03-1987</dogTar>  <dogYer>Bursa</dogYer>  <veliAd>Kemal </veliAd>  <sinif>4-C</sinif>  </ogrenci>  <ogrenci no="2">  <ad>Mehmet</ad>  <soyad>Bakan</soyad>  <dogTar>10-06-1988</dogTar>  <dogYer>Bolu</dogYer>  <veliAd>Bekir</veliAd>  <sinif>3-C</sinif>  </ogrenci>  <ogrenci no="3">  <ad>Veli</ad>  <soyad>Cosar</soyad>  <dogTar>08-06-1985</dogTar>  <dogYer>Istanbul</dogYer>  <veliAd>Nazim</veliAd>  <sinif>6-A</sinif>  </ogrenci>  <ogrenci no="4">  <ad>Mine</ad>  <soyad>Oztorun</soyad>  <dogTar>10-03-1986</dogTar>  <dogYer>Ankara</dogYer>  <veliAd>Ayse</veliAd>  <sinif>5-B</sinif>  </ogrenci>  <ogrenci no="5">  <ad>Mine</ad>  <soyad>Kaya</soyad>  <dogTar>13-03-1987</dogTar>  <dogYer>Bursa</dogYer>  <veliAd>Fatma</veliAd>  <sinif>4-B</sinif>  </ogrenci>  <ogrenci no="6">  <ad>Hakan</ad>  <soyad>Kisa</soyad>  <dogTar>08-06-1981</dogTar>  <dogYer>Istanbul</dogYer>  <veliAd>Burak</veliAd>  <sinif>8-A</sinif>  </ogrenci>  </ogrenci-kayit> |

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!--W3C Schema generated by XMLSpy v2011 rel. 2 (http://www.altova.com)-->  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">  <xs:element name="veliAd" type="xs:string"/>  <xs:element name="soyad" type="xs:string"/>  <xs:element name="sinif" type="xs:string"/>  <xs:element name="ogrenci-kayit">  <xs:complexType>  <xs:sequence>  <xs:element ref="ogrenci" maxOccurs="unbounded"/>  </xs:sequence>  </xs:complexType>  </xs:element>  <xs:element name="ogrenci">  <xs:complexType>  <xs:sequence>  <xs:element ref="ad"/>  <xs:element ref="soyad"/>  <xs:element ref="dogTar"/>  <xs:element ref="dogYer"/>  <xs:element ref="veliAd"/>  <xs:element ref="sinif"/>  </xs:sequence>  <xs:attribute name="no" type="xs:byte" use="required"/>  </xs:complexType>  </xs:element>  <xs:element name="dogYer" type="xs:string"/>  <xs:element name="dogTar" type="xs:string"/>  <xs:element name="ad" type="xs:string"/>  </xs:schema> |

Test dosyası

|  |
| --- |
| **import** java.io.FileInputStream;  **import** java.io.FileNotFoundException;  **import** java.util.List;  **import** javax.xml.bind.JAXBContext;  **import** javax.xml.bind.JAXBException;  **import** javax.xml.bind.Unmarshaller;  **import** ogr.Ogrenci;  **import** ogr.OgrenciKayit;  **public** **class** OgrenciOkuJAXB {  **public** **static** **void** main(String[] args) {  JAXBContext jc;  **try** {  jc = JAXBContext.*newInstance*("ogr");  // create an Unmarshaller  Unmarshaller u = jc.createUnmarshaller();  OgrenciKayit ogrenciKayit = (OgrenciKayit) u  .unmarshal(**new** FileInputStream("ogrenci-kayit.xml"));    List<Ogrenci> ogrenciler = ogrenciKayit.getOgrenci();  **for** (Ogrenci ogrenci : ogrenciler) {  System.*out*.println(ogrenci.getAd());  }  } **catch** (JAXBException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  } **catch** (FileNotFoundException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  }  } |