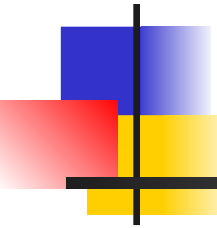
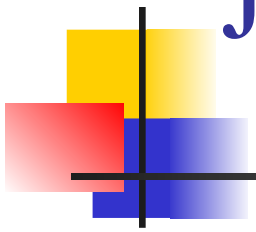


Java & XML



JAXB

- Od Java SE 6 je možné pro práci s XML použít JAXB (Java API for XML Binding).
- JAXB umožňuje načtení XML do Java objektů a obráceně, čili prakticky se jedná o objektovou práci s XML.
- Při práci s JAXB existují dva přístupy:
  - Máte k dispozici XSD – poté je možné v Eclipse kliknout na XSD soubor pravým tlačítkem, vybrat Generate → JAXB classes.
    - Popsáno zde: http://www.javavids.com/tutorial/java-xml.html
  - Nebo můžete třídy, na které se bude XML mapovat, ručně vytvořit.
    - Viz. další snímek.

Test. XML

- Máme tento XML soubor:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<seznam>
   <nazev>Kurz Java</nazev>
   <ucastnik id="1">
      <jmeno>Jirka</jmeno>
   </ucastnik>
   <ucastnik id="2">
      <jmeno>Michal</jmeno>
   </ucastnik>
</seznam>
```

- Chceme ho vytvořit / načíst pomocí JAXB. Jak na to? Nejprve vytvoříme JAXB třídy.

Třída UcastniciSkoleni

Jedná se o root tag v XML.
Toto je jediná povinná anotace.

```
@XmlRootElement(name = "seznam")
@XmlAccessorType(XmlAccessType.FIELD)
public class UcastniciSkoleni {

    @XmlElement(name = "ucastnik") private List<Ucastnik> ucastnici;

    @XmlElement(name = "nazev") private String nazevSeznamu;

    public List<Ucastnik> getUcastnici() { return ucastnici; }

    public void setUcastnici(List<Ucastnik> ucastnici) { this.ucastnici = ucastnici; }

    public String getNazevSeznamu() { return nazevSeznamu; }

    public void setNazevSeznamu(String nazevSeznamu) { this.nazevSeznamu = nazevSeznamu; }
}
```

Anotace uvnitř třídy definuji na atributech

Název tagu

Třída Ucastnik

```
@XmlAccessorType(XmlAccessType.FIELD)
public class Ucastnik {

  @XmlAttribute
  private int id;

  private String jmeno;

  public String getJmeno() { return jmeno; }

  public void setJmeno(String jmeno) { this.jmeno = jmeno; }

  public int getId() { return id; }

  public void setId(int id) { this.id = id; }
}
```

Tento atribut se nebude mapovat jako XML tag, ale jako XML atribut.

Vytvoření XML (Marshalling) I.

- Mějme následující testovací data:

```
UcastniciSkoleni ucastniciSkoleni = new UcastniciSkoleni();
ucastniciSkoleni.setUcastnici(new ArrayList<Ucastnik>());

ucastniciSkoleni.setNazevSeznamu("Kurz Java");

Ucastnik ucastnik1 = new Ucastnik();
ucastnik1.setJmeno("Jirka");
ucastnik1.setId(1);
ucastniciSkoleni.getUcastnici().add(ucastnik1);

Ucastnik ucastnik2 = new Ucastnik();
ucastnik2.setJmeno("Michal");
ucastnik2.setId(2);
ucastniciSkoleni.getUcastnici().add(ucastnik2);
```

Vytvoření XML (Marshalling) II.

- Vytvoření XML souboru:

```
JAXBContext jaxbContext = JAXBContext.newInstance(UcastniciSkoleni.class);
Marshaller marshaller = jaxbContext.createMarshaller();
marshaller.marshal(ucastniciSkoleni, new File("ucastnici.xml"));
```

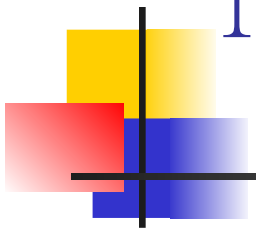
Tento objekt je vhodné mít v celé aplikaci max. jednou (jinak byste se časem dostali do `OutOfMemoryError`)

Uložení objektu `ucastniciSkoleni` do souboru `ucastnici.xml`

Poznámka: tímto nastavíte formátování výstupu:

```
marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
```

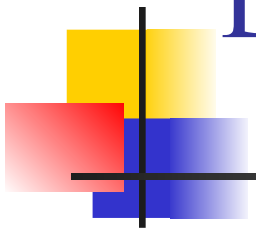
Načtení XML (Unmarshalling)



Vytvoření tohoto objektu je na předcházejícím snímku

```
Unmarshaller unmarshaller = jaxbContext.createUnmarshaller();
UcastniciSkoleni ucastnici2 = (UcastniciSkoleni) unmarshaller
    .unmarshal(new File("ucastnici.xml"));

for (Ucastnik ucastnik : ucastnici2.getUcastnici()) {
  System.out.println(ucastnik.getJmeno());
}
```

Další JAXB anotace



- JAXB obsahuje spoustu anotací. Na jednom blogu je vše moc hezky popsané:
  - http://blog.bdoughan.com/
- Na následujících snímcích jsou popsány nejčastěji používané anotace.

XML Namespace I.

- Když Vaše XML obsahuje namespace, pak je možné ho přidat pomocí souboru `package-info.java`, (který je ve stejném balíčku jako ostatní třídy s JAXB anotacemi):

```
@javax.xml.bind.annotation.XmlSchema(
    namespace = "http://www.sitemaps.org/schemas/sitemap/0.9",
    elementFormDefault = javax.xml.bind.annotation.XmlNsForm.QUALIFIED)
package com.test.sitemap;
```

```
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
    <url>
        <loc>http://www.javavids.com/</loc>
        <priority>1.0</priority>
    </url>
</urlset>
```

XML Namespace II.

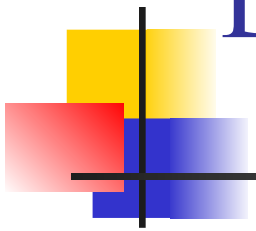
- Více různých namespace v jednom dokumentu:

```
@XmlSchema(namespace = "", elementFormDefault = XmlNsForm.QUALIFIED,
           xmlns = { @XmlNs(prefix = "feedburner",
        namespaceURI = "http://rssnamespace.org/feedburner/ext/1.0") })
package cz.jiripinkas.jba.rss;

import javax.xml.bind.annotation.XmlNs;
import javax.xml.bind.annotation.XmlNsForm;
import javax.xml.bind.annotation.XmlSchema;
```

Poté u jednotlivých elementů, které jsou z namespace „feedburner“ je nutné přidat:

```
@XmlElement(namespace="http://rssnamespace.org/feedburner/ext/1.0")
```

Další anotace I.

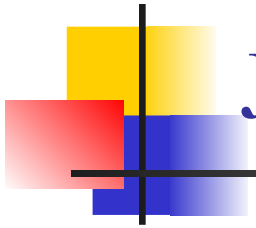


- Pomocí anotace `@XmlTransient` vyloučíte atributy, které nemají být mapovány na XML elementy.
- Pomocí anotace `@XmlElementWrapper` obalíte všechny prvky kolekce do vnějšího elementu, který by jinak nebyl generován.
  - http://blog.bdoughan.com/2010/09/jaxb-collection-properties.html
- Pomocí anotace `@XmlType` můžete namapujete třídu přesně na typ v generovaném XML schématu, pomocí atributu `propOrder` určíte pořadí atributů.
  - http://blog.bdoughan.com/2012/02/jaxbs-xmltype-and-proporder.html
- Pomocí anotace `@XmlAccessorType` specifikujete způsoby mapování atributů:
  - http://blog.bdoughan.com/2011/06/using-jaxbs-xmlaccessortype-to.html

Další anotace II.

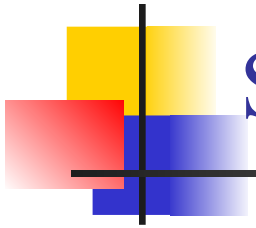
- Pomocí anotace `@XmlList` reprezentujete kolekci dat jako text oddělený mezerou.
  - http://blog.bdoughan.com/2010/09/jaxb-collection-properties.html

Starší způsob práce s XML (JAXP – SAX, DOM)



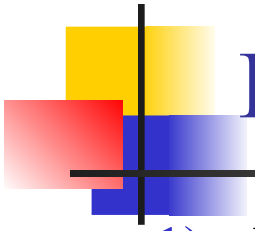
JAXP

- V balíčku javax.xml.parsers, od Javy 5 je součástí JDK.
- Umožňuje parsovat, validovat a transformovat XML,
- využívá již existující parsery, nenabízí nic nového, pouze usnadňuje parsování pomocí těchto API:
  - DOM (Document Object Model),
  - SAX (Simple API for XML),
- Interně lze používat různé již implementované parsery (Crimson, Apache Xerces, Sun XML parser, Oracle parser aj.). S JAXP je distribuován jeden z parserů (Apache Xerces), aby bylo API plně použitelné.
- Parser je možné díky JAXP snadno vyměnit bez rekompilace kódu.



SAX

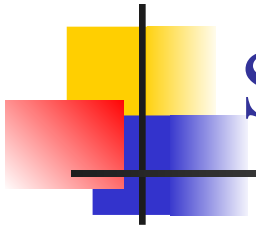
- Událostně orientované API pro čtení XML. Sestává z tzv. callbacků – metod, ve kterých můžeme psát kód obsluhující určitou událost. SAX tyto metody automaticky volá při výskytu události:
  - startElement() - voláno při zjištění začátku nového elementu,
  - characters() - voláno při zjištění obsahu v podobě řetězce,
  - endElement() - voláno při zjištění konce elementu,
  - startDocument() - voláno na začátku XML dokumentu (root node),
  - entDocument() - voláno na konci XML dokumentu, ...
- Implementuje se potomek třídy org.xml.sax.helpers.DefaultHandler, která obsahuje prozatím prázdné, nic nedělající „callback“ metody.



Postup použití SAX

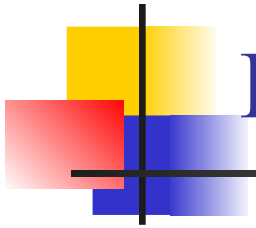
1) Vytvoření SAXParserFactory (SAXParserFactory.newInstance()),
2) nastavení konfiguračních voleb pro parser,
3) vytvoření instance třídy SAXParser (factory.newSAXParser()), která používá XML parser některého z poskytovatelů,
4) spuštění parsování (se současným zaregistrováním třídy, která implementuje callback metody), SAX parser čte XML a automaticky volá callback metody.

- JAX umožňuje specifikovat konkrétní parser (např. org.apache.xerces.parsers.SAXParser) pomocí systémové vlastnosti javax.xml.parsers.SAXParserFactory. Při změně parseru není potřeba překompilovávat již existující kód.



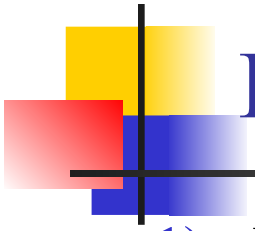
SAXParserFactory

- Umožňuje nastavovat konfigurační volby parseru:
  - setNamespaceAware(boolean awareness) – zda parser bude zpracovávat jmenné prostory v XML dokumentu,
  - setValidating(boolean validating) – zda parser bude provádět validaci podle schématu.
- Vrací SAXParser (metodou newSAXParser()).
- SAX neumožňuje modifikovat XML dokument, pouze jej umí číst.



DOM

- DOM vytváří v paměti stromovou strukturu XML dokumentu. SAX žádnou strukturu v paměti nevytváří, proto je rychlejší, ale oproti DOMu se zase obtížněji zpracovávají konkrétní požadované elementy.
- DOM umožňuje také modifikovat stromovou strukturu v paměti a výsledek promítnout zpět do XML dokumentu.
- XML dokument v paměti je reprezentován instancí třídy org.w3c.dom.Document, která eviduje hierarchickou strukturu DOM uzlů, představujících elementy, atributy a vnitřní řetězcové hodnoty elementů.
- Výsledkem čtení XML je instance třídy Document.

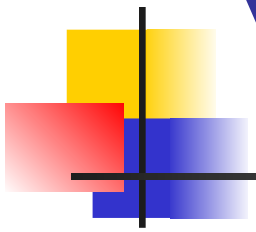


Postup použití DOM

1) Vytvoření DocumentBuilderFactory (DocumentBuilderFactory.newInstance()),
2) nastavení konfiguračních voleb pro parser,
3) vytvoření instance třídy DocumentBuilder (factory.newDocumentBuilder()), která používá XML parser některého z poskytovatelů,
4) spuštění parsování (získání instance třídy Document metodou parse()).

- JAX umožňuje specifikovat konkrétní parser (např. org.apache.xerces.jaxp.DocumentBuilderFactoryImpl) pomocí systémové vlastnosti javax.xml.parsers.DocumentBuilderFactory. Při změně parseru není potřeba překompilovávat již existující kód.

Validace XML I.



- Stáhněte tyto soubory a uložte je k projektu:
  - http://www.sitemaps.org/schemas/sitemap/0.9/sitemap.xsd
  - http://www.sqlvids.com/sitemap.xml
- Na následujících dvou stránkách je příklad na jejich validaci.
- Poznámka: Vnitřně se použije SAX, tudíž je to velice efektivní (v paměti se nevytváří strom celého XML souboru)

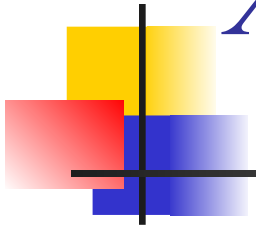
Validace XML II.

```
import java.io.File;
import javax.xml.XMLConstants;
import javax.xml.transform.Source;
import javax.xml.transform.stream.StreamSource;
import javax.xml.validation.*;

public class Main {

   public static void main(String[] args) throws Exception {
```

Validace XML III.

```
        Source xmlFile = new StreamSource(new File("sitemap.xml"));
        SchemaFactory schemaFactory =
            SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
        Schema schema = schemaFactory.newSchema(new File("sitemap.xsd"));
        Validator validator = schema.newValidator();
        try {
            validator.validate(xmlFile);
            System.out.println(xmlFile.getSystemId() + " is valid");
        } catch (SAXException e) {
            System.out.println(xmlFile.getSystemId() + " is NOT valid");
            System.out.println("Reason: " + e.getLocalizedMessage());
        }
    }
}
```

XPath I.



- Java out-of-the-box podporuje pouze XPath pomocí DOM (čili celý XML soubor se nahraje do operační paměti). V následujících stránkách bude tento přístup popsán.
- Pokud byste potřebovali pracovat s velkými XML soubory (o velikosti v řádu stovek MB nebo GB), pak se musíte poohlédnout po externí knihovně:
  - http://stackoverflow.com/a/16416796/894643
- Syntaxe XPath je velice pěkně popsána zde:
  - http://www.w3schools.com/xpath/default.asp
- Online XPath tester:
  - http://www.freeformatter.com/xpath-tester.html

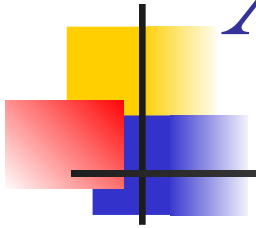
XPath II.

```
import java.io.File;
import javax.xml.parsers.*;
import javax.xml.xpath.*;

import org.w3c.dom.*;

public class Main {
   public static void main(String[] args) throws Exception {
```

XPath III.

```
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = factory.newDocumentBuilder();
        Document doc = builder.parse(new File("sitemap.xml"));
        XPathFactory xPathfactory = XPathFactory.newInstance();
        XPath xpath = xPathfactory.newXPath();
        XPathExpression expr =
           xpath.compile("/urlset/url[priority >= 0.8]/loc/node()");
        NodeList nodeList =
           (NodeList) expr.evaluate(doc, XPathConstants.NODESET);
        for(int i = 0; i < nodeList.getLength(); i++) {
           System.out.println(nodeList.item(i).getNodeValue());
        }
    }
}
```

XSLT I.



- XSLT (XSL Transformations) je XML jazyk pro transformaci jednoho XML souboru na druhý. Jednu chvíli se dokonce XSLT používalo pro konverzi z XML na XHTML … něco takového je ale zbytečně složité, proto se to obvykle nedělá.
- Tutoriál:
  - http://www.w3schools.com/xsl/
- Online nástroj pro XSLT transformace:
  - http://www.w3schools.com/xsl/tryxslt.asp?xmlfile=cdcatalog&xsltfile=cdcatalog

XSLT II.

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

public class Main {
   public static void main(String[] args) throws Exception {
        TransformerFactory factory = TransformerFactory.newInstance();
        Source xslt = new StreamSource(new File("catalog2html.xslt"));
        Transformer transformer = factory.newTransformer(xslt);
        Source text = new StreamSource(new File("catalog.xml"));
        transformer.transform(text,
           new StreamResult(new File("catalog.html")));
   }
}
```