Java XML i web servisi

Uvod u XML

Uvod u Java XML i web servise

- Aplikacije veoma često koriste tekst dokumente za smeštanje i razmenu podataka. Postoje razni načini kojima se ovo može postići, a jedan od najčešće korišćenih načina danas jeste upotreba XML-a. Java omogućava podršku za XML korišćenjem nekoliko API-ja: SAX, DOM, StAX, Xpath, XSLT, JAXB... Razumevanje ovih API-ja je preduslov za upoznavanje sa nekim drugim Java tehnologijama koje koriste ili zavise od XML-a, kao što su na primer veb servisi
- Pre nego što počnemo da razmatramo primenu XML-a u Javi potrebno je da se upoznamo sa samim pojmom XML-a

Šta je XML?

- XML je skraćenica od Extensible Markup Language, što predstavlja jedan od čestih pojmova upotrebljavanih u kontekstu današnjeg programiranja. Ovaj pojam se često „bespravno" upotrebljava za prezentaciju programskog jezika, iako XML to, u stvari, nije.

- Programski jezik se sastoji iz gramatičkih pravila i sopstvene sintakse i on se koristi za kreiranje kompjuterskih programa. Takvi programi daju instrukcije računaru da obavi određene zadatke. Zbog ovoga se XML ne može nazvati programskim jezikom, s obzirom na to da ne obavlja nikakva računanja i ne izvršava logaritamsku logiku. Jednostavno on se čuva u običnim tekstualnim fajlovima, a koriste ga specijalni programi koji su u stanju da interpretiraju XML.

Šta je XML?

- XML je samo način za serijalizaciju odnosno strukturiranje podataka, odnosno, način na koji ćemo jednostavno i brzo moći da zapamtimo podatke (a da pri tome to nije baza podataka) i prosledimo ih nekome ko će ih takođe razumeti, jer poštujemo iste konvencije.
- Neki od primera strukturiranih podataka mogu biti tabele, adresari, konfiguracioni parametri, finansijske transakcije, tehnički crteži. XML je zapravo način kojim se kreira set pravila ili konvencija za kreiranje tekstualnog formata koji omogućava strukturiranje podataka. XML omogućava računarima da na lak način generišu i pročitaju podatke.
- XML je kako i samo ime kaže, markap jezik.

Šta je markap jezik?

- Markap jezik definiše skup pravila za predstavljanje dokumenata u obliku koji je čitljiv kako za ljude tako i za mašine. Markup je drugim rečima oznaka ili simbol koji se može postaviti u dokument kako bi označio njegove različite delove:

```
<message>
  <text>Hello, world!</text>
</message>
```

- U datom primeru, markap simboli su zapravo tagovi <message>...</message> i <text>... </text>.

Razvoj XML-a

- Istorija XML-a počinje još šezdesetih godina prošlog veka kada je u IBM-u u konstruisan prvi višenamenski jezik za serijalizaciju podataka. Ovaj jezik se zvao GML (Generalized Markup Language). Uspeh ovog jezika doveo je do nastavka istraživanja na ovom polju i od njega je nastao i jezik SGML (Standard Generalize Markup Language).
- Deo SGML jezika iskorišćen je za HTML (HyperText Markup Language), dok je ostatak korišćen za kompleksnije internet aplikacije.
- 1996. godine počeo je rad na uprošćenoj verziji SGML-a čija je dotadašnja komplikovanost učinila da bude upotrebljavan samo u vrlo velikim institucijama. Rezultat procesa uprošćavanja omogućio je stvaranje XML-a, za čiju se godinu nastanka smatra 1998. Do danas poznajemo dve verzije (XML 1.0 ili Fifth Edition i XML 1.1 ili Second Edition) i nekoliko međuverzija XML-a. Osnovna razlika je pre svega u rukovanju novim Unicode setovima, s obzirom na to da razvoj XML-a ne prati paralelno proširenje Unicodea.
- XML je već odavno standardni serijalizacioni jezik i na njemu su čak zasnovani i neki drugi, specifični jezici serijalizacije: XHTML, WSDL, WML (WAP XML), RSS, RDF, OWL, SMIL, XSLT...

Struktura i sintaksa

- Sa sintaksom XML-a smo se sreli pri pisanju bilo kog HTML taga, jer XML poštuje konvencije tagova, kao i HTML. Štaviše, sama struktura tih tagova identična je HTML-u jer se poštuje isti princip elemenata i atributa. Ipak, postoje neke osobenosti karakteristične isključivo za XML

- U primeru desno, može se videti XML dokument koji opisuje jedan recept. Primećuje se sličnost sa HTML dokumentima, (korišćenje tagova, atributa i sadržaja). Za razliku od HTML-a, kod koga se koriste tagovi kao što su <html>, <head>, <img> i drugi, prilikom pisanja predstavljenog XML dokumenta korišćeni su tagovi koji su specijalno prilagođeni opisivanju recepta: <recipe>, < ingredients>

```
<recipe>
  <title>
    Grilled Cheese Sandwich
  </title>
    <ingredients>
      <ingredient qty="2">
        bread slice
      </ingredient>
      <ingredient>
        cheese slice
      </ingredient>
      <ingredient qty="2">
        margarine pat
      </ingredient>
    </ingredients>
</recipe>
```

Komponente XML dokumenta

- XML deklaracije,
- elementi,
- atributi,
- podaci,
- CDATA sekcije,
- prostori imena,
- komentari i
- instrukcije za procesuiranje

Tagovi i elementi

- Tag je selekcija u nekom dokumentu markirana određenim oznakama i nazivom. Ove oznake su predstavljene trougaonim zagradama (< >), a naziv taga može biti bilo koji tekst, sve dok se poštuju konvencije aktuelnog tag jezika.

```
<ingredient>
```

- Ovo je pravilno napisan tag, ali nedovoljan da zaokruži jednu celinu u XML dokumentu. Da bi neka celina bila zaokružena u XML-u, potrebna su bar dva taga: jedan otvarajući i jedan zatvarajući, ili eventualno, jedan „samozatvarajući" tag:

```
<ingredient> </ingredient>

<ingredient/>
```

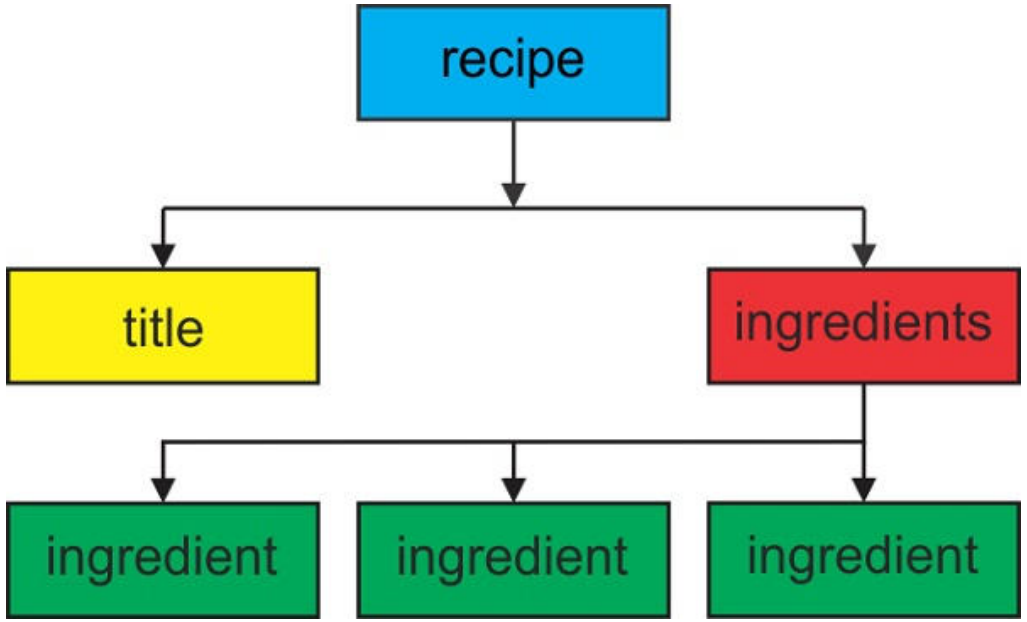
Tagovi i elementi

- Zatvarajući tag izgleda isto kao i otvarajući, osim oznake / koja se nalazi odmah na njegovom početku. Kada je tag samozatvarajući, oznaka / nalazi se pre njegovog završetka. Pravilno postavljena dva istoimena taga čine jedan element i sve unutar njih čini sadržaj tog elementa:

```
<ingredient>cheese slice</ingredient>
```

Hijerarhija elemenata

- Jedan element može imati podelemente koji mogu takođe biti elementi. To znači da jedan XML dokument ima strukturu stabla. U to se možemo uveriti i u primeru XML dokumenta za definisanje recepta u ovoj lekciji. <recipe> je koreni element, dok je na primer <ingredients> podelement ovog elementa.
- Ovo se može zaključiti i po formatiranju XML dokumenta.



Atributi

- Atributi su još jedan odeljak u kome XML element može nositi podatke. Nalaze se u otvarajućem tagu i označavaju se po sistemu ključ=vrednost:

```
<ingredient qty="2">
  bread slice
</ingredient>

<ingredient qty="2"/>
```

- Otvarajući tagovi u XML mogu sadržati jedan ili više atributa. U primeru recepta u ovoj lekciji element (tag) <ingredient> poseduje atribut qty, što je skraćenica od reči quantity (količina prim. aut). Atributi omogućavaju mehanizam za navođenje dodatnih informacija o elementima. Tako i u primeru recepta u ovoj lekciji, atribut qty, definiše količinu sastojka koji je potrebno dodati.
- Atributi su opcioni delovi XML-a

XML deklaracija

- XML dokument uglavnom počinje XML deklaracijom, specijalnom oznakom koja informiše XML parser da je reč o XML dokumentu. Odsustvo ove oznake na početku XML dokumenta za opis recepta koji je dat u ovoj lekciji govori o tome da XML deklaracija nije obavezna. Ipak, kada je navedena, ništa se ne može pojaviti pre nje (zapravo može, ali neće biti uzeto u obzir).
- Minimalni oblik XML deklaracije izgleda ovako:

```
<?xml version="1.0"?>
```

XML deklaracija

- Iz ovoga se može zaključiti da je atribut version obavezan i on se koristi za identifikovanje verzije XML specifikacije koja se koristi na dokumentu.
- XML podražava UNICODE, što znači da se XML dokumenti sastoje od karaktera preuzetih iz UNICODE seta karaktera. Karakteri dokumenta se prevode u bajtove prilikom smeštanja ili prenosa, a tip enkodinga koji se tom prilikom koristi definiše se korišćenjem atributa encoding, XML deklaracije.
- Primer deklaracije sa navedenim atributom encoding:

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

Komentari i instrukcije za procesuiranje

- XML dokumenti mogu sadržati komentare. Komentarisani kod znači da parser neće uzeti u obzir taj deo koda. U XML-u komentar se piše na sledeći način:

```
<!-- comment -->
```

- Komentari se mogu pojaviti bilo gde nakon XML deklaracije, osim unutar taga. Komentari takođe ne mogu biti ugneždeni, ne mogu sadržati dve uzastopne crtice (--), niti crticu. Komentari se ne smatraju sadržajem.

```
<!-- myElement tag in use -->
<myElement atribut="my atribut">Content</myElement>
```

- Pored komentara, jedan XML dokument može da sadrži i instrukcije za procesuiranje. To su instrukcije koje su namenjene aplikacijama koje će vršiti parsiranje XML dokumenta. Instrukcije su obično specifične, odnosno razumljive samo aplikacijama koje će parsirati sadržaj takvih instrukcija. Ovakve instrukcije se pišu između **<?** i **?>** tagova

Reference na karaktere

- Određeni karakteri se ne mogu pojaviti kao tekst između otvarajućeg i zatvarajućeg taga ili kao vrednost atributa. Na primer, ne može se postaviti karakter < između otvarajućeg i zatvarajućeg taga, jer bi na taj način došlo do zbunjivanja XML parsera, koji bi pomislio da je reč o novom tagu.
- Rešenje ovog problema ogleda se u upotrebi reference na karaktere, što je u suštini kod koji predstavlja reprezentaciju karaktera. Reference na karaktere se mogu podeliti na dve grupe:
- numeričke reference na karaktere kojima se karakteri predstavljaju preko koda koji im je dodeljen u Unicodeu; na primer &#0931 predstavlja grčko slovo sigma,
- unapred definisani entiteti koji predstavljaju zamenu za neke od najčešće korišćenih specijalnih karaktera; ovakvih predefinisanih entiteta postoji pet i oni su prikazani u tabeli.

entitet	referencira
&lt;	<
&gt;	>
&amp;	&
&apos;	'
&quot;	"

CDATA sekcije

- CDATA predstavlja sekciju, odnosno blok HTML ili XML koda koji je oivičen specijalnim prefiksom i sufiksom:
- **<![CDATA[** content **]]>**
- Sve što se nađe između ovih „tagova", odnosno content u gornjem primeru, može sadržati specijalne karaktere koji se inače ne mogu naći u izvornom obliku u XML dokumentu.

```
<expression>
<![CDATA[
    <div id="wrapper">
        <a href="#"></a>
    </div>
]]>
</expression>
```

Prostori imena

- U XML dokumentima nazive elemenata definiše developer. Zbog toga je veoma česta situacija kreiranje XML dokumenata koji kombinuju više različitih XML jezika. U takvim situacijama se mogu pojaviti dva elementa sa istim nazivima, koji pripadaju različitim jezicima, pa može doći do konflikata. U ovakvim situacijama se koriste prostori imena (namespaces)

```
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

- Konflikti poput onoga na prethodnom slajdu, lako se mogu predupрediti korišćenjem prostora imena:

```
<root>
  <h:table xmlns:h="http://www.w3.org/TR/html5/">
    <h:tr>
      <h:td>Apples</h:td>
      <h:td>Bananas</h:td>
    </h:tr>
  </h:table>

  <f:table xmlns:f="http://www.link.co.rs/furniture">
    <f:name>African Coffee Table</f:name>
    <f:width>80</f:width>
    <f:length>120</f:length>
  </f:table>
</root>
```

Pregled XML dokumenata

- Sadržaj XML dokumenta je zapravo običan tekst i može se pregledati na više načina.
- Svaki pretraživač poseduje svoj mehanizam za pregled XML dokumenata, ali generalno, većina izgleda isto.
- Svakodnevno rukujemo mnoštvom XML dokumenata, a da to i ne znamo. Većina programa čuva setovanja i ostale podatke u XML formatu, a onda ih koristi prerađene i deserijalizovane.
- Takođe XML se može pregledati u običnom tekstualnom pregledaču (Notepad i slično) u izvornom obliku – kao čist tekst.
- XML dokument se može i stilizovati na klijentu, CSS i XSLT stilovima.

Formatiranje i validacija XML-a

- Kako bi se proces parsiranja XML dokumenta učinio lakšim, definisana su određena pravila, koja svaki XML dokument mora da poštuje, kako bi mogao da se nazove ispravno formatiranim dokumentom (Well-Formed Document). Ta pravila su sledeća:
  - Svi elementi moraju imati otvarajući i zatvarajući tag ili samozatvarajući tag; za razliku od, na primer, HTML paragraf taga , koji se obično navodi bez zatvarajućeg taga, da bi se XML dokument nazvao ispravno formatiranim u takvim situacijama je neophodno postojanje ,
  - Tagovi moraju da budu pravilno ugneždeni; na primer ovo bi bio neispravno formatiran XML dokument: JavaFX, dok bi ovo bio ispravno formatiran: JavaFX,
  - Sve vrednosti atributa moraju biti navedene između navodnika: mogu se koristiti kako jednostruki ('), tako i dvostruki (") navodnici, iako su dvostruki navodnici u široj upotrebi; smatra se greškom izostaviti navodnike,
  - Prazni tagovi moraju biti zatvoreni; na primer ne može se navesti samo <br>, već se ovakav tag mora navesti kao samozatvarajući <br/> i
  - Velika i mala slova moraju biti adekvatno korišćena; XML je osetljiv na velika i mala slova, tako da se tagovi <name> i <Name> ne smatraju istim tagovima.
- XML parseri koji poznaju i prostore imena definišu i dva dodatna pravila:
  - imena elemenata i atributa ne smeju da sadrže više od jednog karaktera : (dve tačke) i
  - imena entiteta, instrukcije za procesuiranje i nazivi notacija ne smeju sadržati karakter : (dve tačke).
- XML dokument koji zadovolji navedena pravila može se nazvati ispravno formatiranim ili well-formed. XML parseri će parsirati jedino ovakve ispravno formatirane dokumente.

Validni dokument (Valid Document)

- U najvećem broju situacija nije dovoljno da XML dokument bude ispravno formatiran. U mnogim situacijama dokument mora biti i validan. Validnost se pre svega odnosi na logičke greške koje se mogu javiti u jednom XML dokumentu. Logičke greške se mogu javiti ukoliko ima odstupanja od nekog definisanog pravila. U našem primeru XML dokumenta sa receptom, jedno takvo pravilo bi moglo da definiše da je obavezno pojavljivanje elementa ingredients, tek nakon pojavljivanja elementa title.
- Neki XML parseri sprovode ovakav tip validacije, a neki, najviše zbog dodatne kompleksnosti, ne sprovode proveru validacije.
- Potpuno je logično sada se zapitati na koji način će parser znati koja su validaciona pravila definisana za određeni XML dokument. U tu svrhu se koriste specijalni dokumenti u kojima su navedena ovakva pravila. Parser jednostavno poredi XML dokument i dokument sa definisanim pravilima i ukoliko utvrdi bilo kakvo odstupanje, dokument smatra nevalidnim. Za razliku od grešaka u formatiranju, ovakve greške ne moraju biti fatalne i parser može nastaviti sa obradom dokumenta.
- Dokumenti u kojima se čuvaju validaciona pravila napisani su specijalnim jezikom. Dva najčešće korišćena jezika su Document Type Definition i XML Schema

DTD (Document Type Definition)

- Document Type Definition (DTD) je najstariji jezik za definisanje validacionih ili drugim rečima gramatičkih pravila XML dokumenta. Ovaj jezik karakterišu stroga sintaksna pravila, koja definišu koji elementi mogu biti deo dokumenta, šta oni mogu sadržati i kakve atribute mogu imati
- Dokument za definisanje gramatičkih pravila XML dokumenta sa leve strane mogao bi da se predstavi dokumentom sa desne strane

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <country countryCode="sr">
      <name>Serbia</name>
      <capital>Belgrade</capital>
      <description>Serbia is.....</description>
  </country>
  <country countryCode="fr">
      <name>France</name>
      <capital>Paris</capital>
      <description>France is.....</description>
  </country>
</root>
```

```
<!ELEMENT root (country)+>
<!ELEMENT country (name, capital, description)>
<!ATTLIST country countryCode CDATA #IMPLIED >

<!ELEMENT name (#PCDATA)>
<!ELEMENT capital (#PCDATA)>
<!ELEMENT description (#PCDATA)>
```

Korišćenje DTD-a

- Validacija korišćenjem DTD-a zahteva postojanje document type deklaracije, za identifikovanje DTD dokumenta koji sadrži gramatička pravila za validiranje. Obratite pažnju na to da su Document Type Definition i document type declaration dva različita pojma. Skraćenica DTD se odnosi na pojam Document Type Definition, dok pojam document type declaration označava deklaraciju koju je potrebno dodati u XML dokument kako bi se dokument validirao.

Korišćenje DTD-a

- U okviru XML dokumenta se može referencirati interni DTD. To se postiže na sledeći način:

```
<?xml version="1.0"?>

<!DOCTYPE root [
<!ELEMENT root (country)+>
<!ELEMENT country (name, capital, description)>
<!ATTLIST country countryCode CDATA #IMPLIED >
<!ELEMENT name (#PCDATA)>
<!ELEMENT capital (#PCDATA)>
<!ELEMENT description (#PCDATA)>
]>
<root>
  <country countryCode="sr">
      <name>Serbia</name>
      <capital>Belgrade</capital>
      <description>Serbia is.....</description>
  </country>
  <country countryCode="fr">
      <name>France</name>
      <capital>Paris</capital>
      <description>France is.....</description>
  </country>
</root>
```

- DTD se može postaviti i u okviru eksternog fajla, i u tom slučaju deklaraciju je potrebno formulisati na sledeći način:

```
<?xml version="1.0"?>
<!DOCTYPE root SYSTEM "country.dtd">
<root>
  <country countryCode="sr">
      <name>Serbia</name>
      <capital>Belgrade</capital>
      <description>Serbia is.....</description>
  </country>
  <country countryCode="fr">
      <name>France</name>
      <capital>Paris</capital>
      <description>France is.....</description>
  </country>
</root>
```

XML Schema Definition

- XML Schema je gramatički jezik za deklarisanje strukture, sadržaja i semantičkih pravila za kreiranje XML dokumenta. XML šema se drugačije naziva XML Schema Definition ili skraćeno XSD

XML Schema Definition

- XML šeme su predstavljene od organizacije W3C, kako bi se prevazišli nedostaci DTD-a, na primer podrška za prostore imena. Takođe, XML šeme omogućavaju definisanje validacionih pravila korišćenjem objektno orijentisanih principa i mnogo veći izbor tipova podataka. Naravno, sa XML šemama najbolje je upoznati se na primeru. XML šema za definisanje XML dokumenta za opis država izgleda ovako:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="root">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="country" maxOccurs="unbounded" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="country">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="name" type="xs:string" />
                <xs:element name="capital" type="xs:string" />
                <xs:element name="description" type="xs:string" />
            </xs:sequence>
            <xs:attribute name="countryCode" type="xs:string"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Kreiranje šeme

- XML šema započinje elementom za definisanje gramatičkih pravila:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

- Nakon ovoga sledi definisanje elemenata. XML šema generalno poznaje dva tipa elemenata: proste i kompleksne. U našem primeru su definisana dva kompleksna elementa, koji sadrži podelemente. Element se definiše korišćenjem oznake element, dok oznaka complexType ukazuje na to da je reč o kompleksnom tipu podatka. Kompleksni tip se sastoji iz sekvence prostih tipova. Element prostog tipa se definiše takođe korišćenjem oznake element, pri čemu se navodi naziv i tip elementa. Naravno, reč je o prostim tipovima koje XSD automatski poznaje. Inače, XSD poznaje mnogo širi skup prostih ugrađenih tipova nego što je to slučaj sa DTD-om.

XSD tipovi

- Neki od najčešće korišćenih tipova prikazani su u narednoj tabeli:

Tip	Opis	Primer
string	A character string	New York, NY
int	–2147483648 to 2147483647	+234, –345, 678987
double	A 64-bit floating point number	–345.e-7, NaN, –INF, INF
decimal	A valid decimal number	–42.5, 67, 92.34, +54.345
date	A date in CCYY-MM-DD format	2006-05-05
time	Time in hh:mm:ss-hh:mm format	10:27:34-05:00 (for 10:27:34 EST, which is –5 hours UTC)

LINKgroup

Distance Learning System

XML i web servisi

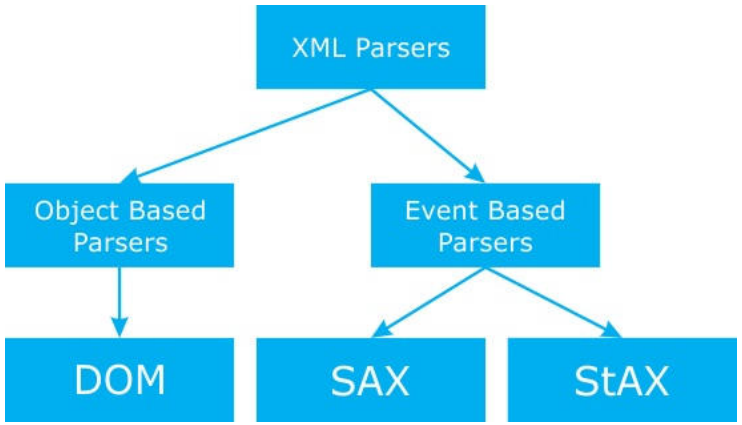
JAXP

Uvod u JAXP

- U Java programskom jeziku postoji mnoštvo biblioteka (paketa) koje olakšavaju rukovanje XML-om. **Java API for XML Processing** ili skraćeno **JAXP** predstavlja arhitekturu Java programskog jezika za parsiranje, kreiranje, adresiranje i transformisanje XML dokumenata.
- JAXP je sastavni deo Java SE i zajedno sa JAXB (Java Architecture for XML Binding) API bibliotekom čini skup alata za rukovanje XML-om u ovoj distribuciji.

Parsiranje XML dokumenta

- XML dokument sadrži strukturirane tekstualne informacije.
- Kada određena aplikacija parsira XML dokument, uglavnom sprovodi sledeće korake:
  - Proverava da je dokument ispravno formatiran,
  - Proverava da dokument zadovoljava strukturu koja je definisana DTD i XML šema dokumentom
  - Pristupa i čita dokument i eventualno zamenjuje elemente i atribute definisane u njemu
- Tokom vremena razvilo se nekoliko pristupa parsiranju:
  - **DOM** parsiranje
  - **Push** parsiranje
  - **Pull** parsiranje



DOM parsiranje

- Korišćenjem ovog pristupa XML dokument se parsira u strukturu čvorova u obliku stabla, gde su svi atributi, elementi i ostali članovi XML-a, zapravo čvorovi određenog tipa.
  Najbitnije karakteristike ovakvog pristupa su sledeće:
- Kompletno DOM stablo čvorova mora biti konstruisano i smešteno u memoriju pre nego što mu se može pristupiti,
- Čvorovima se može pristupati nasumično, a ne po nekom unapred utvrđenom redosledu ili po redosledu koji postoji u dokumentu,
- Pristup bilo kojem čvoru je veoma brz i fleksibilan, ali brzina parsiranja kompletnog dokumenta se ne ubraja u jače strane ovog postupka,
- Parsiranje veoma velikih dokumenata, od nekoliko stotina megabajta ili više gigabajta, može biti praktično nemoguće, zbog osobine ovog tipa parsiranja da kompletnu strukturu smešta u radnu memoriju
- Ukoliko postoji potreba za nasumičnom i čestom manipulacijom, DOM parsiranje predstavlja najbolji pristup, upravo zbog činjenica da se kompletna struktura smešta u radnoj memoriji

Push pristup parsiranju

- Push parsiranje funkcioniše tako što parser prilikom parsiranja generiše sinhrone događaje, na koje se može pretplatiti aplikacija koja vrši parsiranje. Praktično kod Push streaminga jedan klijent šalje podatke ka drugom klijentu, dok je drugi klijent pasivan i samo prihvata pristigle podatke. API koji koristi ovaj princip naziva se SAX. SAX se naročito preporučuje ukoliko nisu planirane modifikacije ili učestalo nasumično pristupanje elementima XML dokumenta.

Pull pristup parsiranju

- Za razliku od upravo opisanog push principa, kod kojeg se događaji delegiraju od parsera aplikaciji, kod pull principa situacija je nešto drugačija. Sada je aplikacija ta koja kontroliše isporuku događaja i API koji ovo podržava naziva se StAX. Drugim rečima, ukoliko je klijent u Pull Modeu, znači da sam preuzima podatke od pasivnog klijenta ili servera.
- StAX omogućava dva principa za obradu događaja: korišćenje kursora i iteratora. Oba su, naravno, pod kontrolom aplikacije.
- Za razliku od SAX API-ja, StAX može biti korišćen kako za čitanje, tako i za pisanje XML dokumenata.

Poređenje različitih pristupa parsiranju

U narednoj tabeli moguće je videti uporedni prikaz osobina različitih pristupa parsiranju.

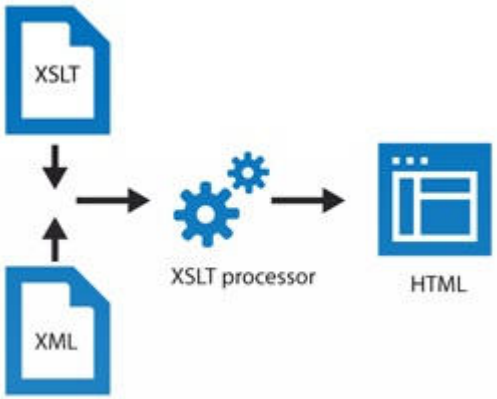
Parsiranje	Prednosti	Mane	Primena
DOM	jednostavnost, navigacija, brzina nasumičnog pristupa	obaveza parsiranja kompletnog dokumenta, zauzeće memorije	XML editori
SAX	malo zauzeće memorije, efikasnost	nedostatak navigacije, nasumičnog pristupa i mogućnosti modifikacije	aplikacije za čitanje i validiranje XML dokumenata
StAX	jednostavnost, malo zauzeće memorije, kontrola parsiranja od aplikacije, filtriranje	nedostatak nasumičnog pristupa	Data binding, SOAP

Adresiranje XML dokumenata

- Još jedan bitan segment JAXP-a je svakako i adresiranje. Pojam adresiranja odnosi se na mogućnost definisanja tačne lokacije svakog čvora, jednog dokumenta. Jezik koji omogućava upravo ovakvo adresiranje je **XPath**.
- **XPath** se bazira na apstraktnom modelu podataka, koji se prevashodno usredsređuje na informacije sadržane u XML dokumentu, pri tome ignorišući bilo kakvu markap sintaksu. Praktično XPath omogućava specificiranje putanje do čvora, tako da na primer čvorovi sa istim nazivom, ali na različitom pozicijama mogu biti razlikovani.
- XPath je veoma značajan i za sam pojam transformacija, s obzirom na to da predstavlja osnovu za njihovo izvršavanje.

Transformisanje XML dokumenata

- Transformacija XML dokumenta predstavlja njegovo konvertovanje u neki drugi oblik, tj. formu. JAXP sadrži standard koji definiše mehanizme za obavljanje pomenutog posla, koji se nazivaju **The Extensible Stylesheet Language Transformations** ili skraćeno **XSLT**.
- XSLT jezik je potpuno zasnovan na XML-u, tako da XSLT transformacije postoje kao validni XML dokumenti. Ovakvi XML dokumenti se često nazivaju stilovi (style sheets). S obzirom na to da ovakvi dokumenti sadrže set transformacija, neophodno je postojanje XSLT procesora koji će pomenute transformacije primeniti na dokumentu. Praktično, XML dokument i XSLT stil predstavljaju ulazne elemente, koje XSLT procesor koristi kako bi izvršio transformaciju. Izlazni tip dokumenta ovakvog procesa je najčešće XML, mada to može biti i HTML dokument.



JAXP paketi

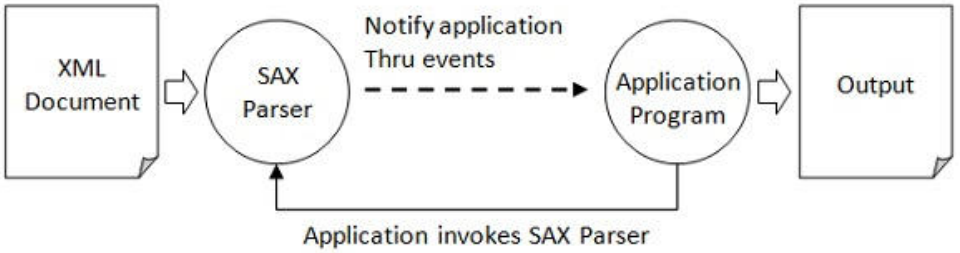
- Da bi se postigle opisane funkcionalnosti, koriste se paketi koji postoje u JAXP-u. Najznačajniji paketi su sledeći:
  - **javax.xml.parsers** paket sadrži četiri klase: DocumentBuilder, DocumentBuilderFactory, SAXParser, SAXParserFactory,
  - **org.w3c.dom** paket sadrži alate za rukovanje i manipulaciju XML-om putem DOM tehnologije,
  - **org.xml.sax** paket sadrži alate za rukovanje i manipulaciju XML-om putem SAX tehnologije,
  - **javax.xml.transform** sadrži četiri potpaketa (DOM, SAX, StAX i STREAM) u kojima se nalaze specijalizovane klase za transformaciju po istoimenim parserima. Pored ovih paketa, sam transformacioni paket sadrži nekoliko opštih klasa i interfejsa za transformaciju. Ovaj set za rukovanje XML transformacijom u Javi naziva se TrAX, odnosno, transformation API for XML i
  - **javax.xml.stream** se sastoji od klasa za preuzimanje i čuvanje XML dokumenata putem tokova podataka, a takođe sadrži i poseban potpaket sa interfejsima za rukovaoce XML događajima

SAX

- Simple API for XML (SAX) predstavlja API za takozvano Push parsiranje, kod koga se XML dokument sekvencijalno parsira od početka do kraja. Razvila ga je organizacija XML-DEV, kao alternativu DOM parsiranju. SAX je prvobitno prevashodno postojao za Java programski jezik, ali danas postoje verzije za više programskih jezika (PHP, C++,C#...).

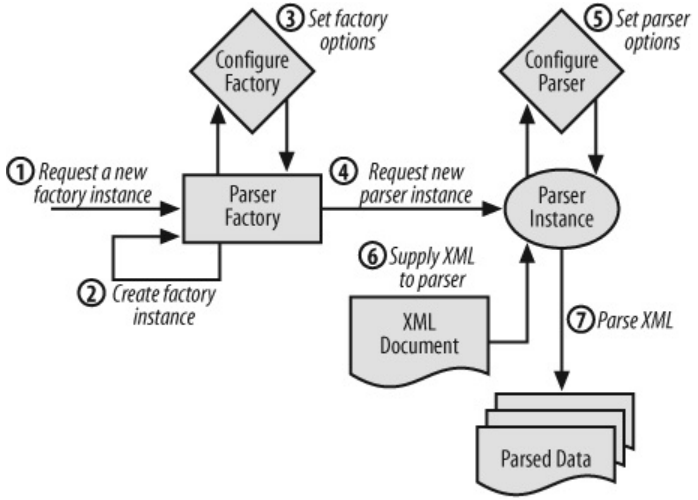
Kako SAX funkcioniše?

- Simple API for XML, SAX, podrazumeva takozvani event based metod rukovanja XML dokumentima. XML dokument se tretira tako što se čita sekvencijalno, tag po tag. Prilikom svakog pristupa i obrade određenog taga dolazi do aktivacije događaja. Na primer, početak čitanja elementa izaziva događaj, kraj čitanja elementa takođe izaziva događaj, čitanje atributa takođe itd.
- Kao rukovaoci događaja definišu se određene callback funkcije u kojima se obrađuje objekat na kome se događaj dogodio (na primer element, ukoliko je u pitanju start ili end element događaj).
- SAX iščitava XML dokument sekvencijalno, pri čemu se, nakon pročitanog taga, nije moguće vratiti nazad na taj tag (Unidirectional Read). Takođe, SAX tehnologija ne podrazumeva mogućnost upisa u XML dokument. Odnosno, ona je Read Only.



Parsiranje dokumenta korišćenjem SAX API-ja (jwsx012014 SimpleSAXParsing)

- Da bi se korišćenjem Java programskog jezika izvršilo parsiranje XML dokumenta korišćenjem SAX API-ja, neophodno je koristiti objekat klase **SAXParser**. Objekat ove klase dobija se korišćenjem objekta**SAXParserFactory** klase. Iz ovoga se može zaključiti da je prvo neophodno dobiti instancu SAXParserFactory. Za to se koristi metoda newInstance ove klase
- Nakon što je kreiran objekat ove factory klase, može se pristupiti kreiranju instance samog parsera. To se postiže metodom newSAXParser




```
SAXParserFactory spf = SAXParserFactory.newInstance()

SAXParser sp = spf.newSAXParser();
```

SAX Parsiranje

- Postojanje parsera je preduslov za parsiranje XML dokumenta. Sada kada je parser instanciran, može se izvršiti parsiranje nekog XML dokumenta. Za parsiranje se koristi metoda parse:

```
sp.p
parse(File f, DefaultHandler dh)                    void
parse(File f, HandlerBase hb)                       void
parse(InputSource is, DefaultHandler dh)            void
parse(InputSource is, HandlerBase hb)               void
parse(InputStream is, DefaultHandler dh)            void
parse(InputStream is, HandlerBase hb)               void
```

- Prvi perametar se odnosi na lokaciju dokumenta koji je potrebno parsirati, dok se **dh** odnosi na objekat u kome će biti definisana ponašanja, odnosno različite reakcije na događaje parsiranja.

SAX Parsiranje

- Prilikom SAX parsiranja neophodno je obezbediti hendler koji će obraditi događaje parsiranja
- Ovaj hendler se može napisati ručno, ili se može kreirati nasleđivanjem ugrađene klase **DefaultHandler**

```
sp.parse(
        JavaApplication311.class.getResourceAsStream("newXMLDocument.xml"),
        new DefaultHandler()
);
```

- Program iznad ne daje rezultat, je se koristi neprepisan DefaultHandler
- Ipak, program će funkcionisati bez greške (iako ništa ne radi)

Obrada SAX događaja (Prepisivanje DefaultHanlder klase)

```
sp.parse(
        JavaApplication311.class.getResourceAsStream("newXMLDocument.xml"),
        new DefaultHandler(){
            @Override
            public void startDocument() throws SAXException {
                System.out.println("Document reading has been started");
            }
            @Override
            public void endDocument() throws SAXException {
                System.out.println("Document reading has ended");
            }
        }
);
```

Parsiranje elemenata

- Prethodnim kodom definisani su handleri samo za početak i kraj čitanja kompletnog dokumenta. Moguće je pregaziti i metode koje će se aktivirati pri parsiranju pojedinačnih elemenata.

```
@Override
public void startElement (String uri, String localName, String qName, Attributes attributes)
            throws SAXException
{
      System.out.println("Element started: " + qName);
}
```

```
@Override
public void endElement(String uri, String localName, String qName)
throws SAXException {
        System.out.println("Element finished: " + qName);
}
```

Parsiranje atributa

- Korišćenjem metoda Attributes tipa moguće je izvršiti parsiranje atributa. Da bismo pristupili određenom atributu, koristimo metodu getLocalName za naziv atributa i getValue za vrednost atributa. Oba metoda prihvataju indeks (poziciju) atributa u listi kao parametar. Ova lista takođe poseduje metod getLength koji prikazuje broj atributa u listi, na osnovu čega je moguće proći kroz listu i putem petlje.

```
for (int i = 0; i < attributes.getLength(); i++) {
        System.out.println("Attribute: "
                + attributes.getLocalName(i) + ": "
                + attributes.getValue(i));
    }
```

Parsiranje sadržaja

- Da bismo pročitali sadržaj samog elementa, potrebno je pregaziti metod characters. Ovaj metod prihvata tri parametra: niz karaktera, početnu poziciju (Offset) i dužinu teksta u karakterima. Ovako prosleđen tekst možemo pročitati putem petlje ili kroz konstruktor String objekta, kao u sledećem primeru:

```
@Override
public void characters (char ch[], int start, int length)
 throws SAXException
{
        String nodeValue = new String(ch,start,length);
         if(!nodeValue.trim().equals(""))
            System.out.println("Node value: " + nodeValue);
}
```

Selektivno parsiranje

- U prethodnim primerima čitaju se sve vrednosti svih elemenata XML dokumenta. Šta bi bilo da hoćemo da pročitamo samo neke od elemenata XML dokumenta?
- U tom slučaju morali bismo da nekako damo do znanja metodu characters koji je naziv elementa koji trenutno čita

Vežba 1
(jwsx RSSRead)

- Na osnovu RSS fida (http://rss.news.yahoo.com/rss/stocks) potrebno je napraviti odgovarajući čitač uz pomoć SAX tehnologije. Ovaj čitač prikazaće sadržaje fidova u sledećem formatu:

Title: News title
Link: News link
Publishing date: News publishing date
Description: News description