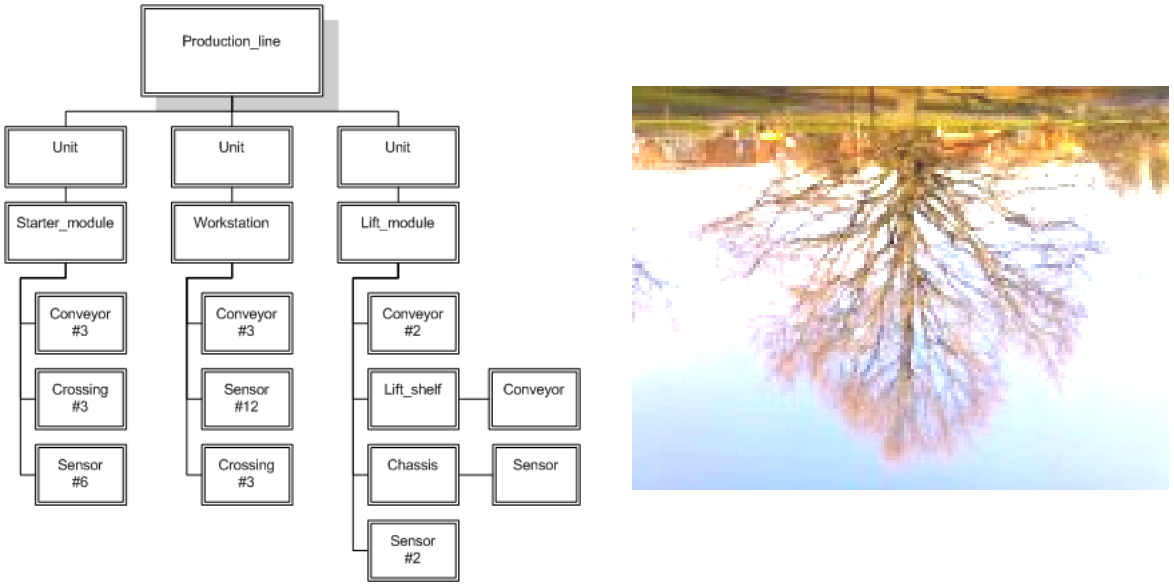
XML-merkintäkieli_teoria.docx
ELEC-C8203 Automaatiojärjestelmät 2
XML Tietointegraatio

XML merkintäkieli - Teoria



Kuva 1. A) Tuotantolinjan fyysisen rakenteen esitys kokonaisuus-osa hierarkiana. B) Väreen edessä oleva tammi

1. XML:n puurakenne

XML on tiedon tallentamista ja rakenteen kuvaamista koskeva määrittely. XML-merkintäkielillä voidaan kuvata hierarkkisia puumuotoisia tietorakenteita (tree). Kaikkien XML tekniikoiden ymmärtämisen kannalta on erittäin tärkeää tiedostaa tämä XML dokumentin puurakenne. Merkinnöillä nimetään puun solmut (node) . Puun lehtisolmut (leaf nodes) sisältävät varsinainen tekstitiedon.

Puurakenteen yleisiä ominaisuuksia

Puulla on vain yksi juurisolmu (root node). Solmuilla voi olla monta lapsi-solmua (child node) mutta vain yksi isä-solmu (parent node). Kaaret (branch) puun solmujen välillä (parent-child relation) voivat vastata mitä tahansa todellista relaatiota esitettävän kohteen osien/objektien välillä

- osa-kokonaisuus (tai kokonaisuus-osa);
- luokka-instanssi;
- ominaisuus;
- jokin assosiaatio;

Erilaiset hierarkiat voidaan esittää puurakenteena, minkä vuoksi XML on soveltuu hyvin luokkahierarkioiden ja järjestelmien rakennehierarkioiden esittämiseen.

2. XML:n rakenneosat

XML-dokumentti voi sisältää seuraavia rakenteita:

1. elementit – muodostavat dokumentin rungon ja rakenteen
2. attribuutit – antavat lisämääreitä elementeille
3. tekstisisältö/leipäteksti – elementtien sisällä olevaa tekstiä
4. kommentit – kommentteja ihmisille tai koneilta väliaikaisesti piilotettuja osioita
5. käsittelyohjeet – ohjeita dokumentin koneelliseen käsittelyyn
6. entiteetit – erikoismerkkien esittämiseen ja dokumenttien liittämiseen toisiinsa

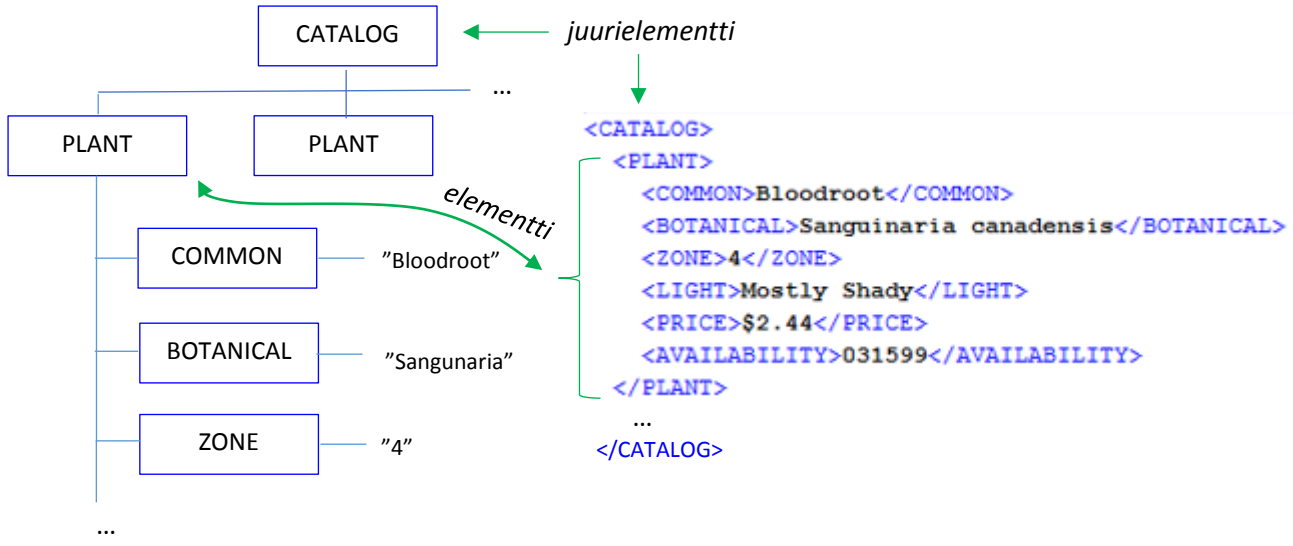
Rakenneosat: Elementit

XML-dokumentti koostuu elementeistä (element). Jokaisella elementillä on nimi, joka kirjoitetaan kulmasulkujen (< ja >) sisään. Elementillä on yleensä alkuosa *<element>* ja loppuosa *</element>,* joiden väliin kirjoitetaan elementin tekstisisältö/leipäteksti.

Elementit voivat sisältää toisia elementtejä ja/tai leipätekstiä. Dokumentin ulointa elementtiä kutsutaan juurielementiksi (root element). Sisempiä elementtejä kutsutaan lapsielementeiksi (child element) ja ulompaa isäelementiksi (parent element). Sisäkkäisyydestä muodostuu dokumentin rakenne

Esimerkki: Tuoteluettelo

Tuoteluettelot esitetään usein yksinkertaisena järjestämättömänä tai järjestettynä listana. Tuotteet listataan elementteinä juurielementin alle sen lapsi-elementeiksi. Tuotteen ominaisuudet listataan myös elementteinä tuote-elementin lapsiksi. Tuotteen ominaisuuden arvo esitetään elementin tekstisisältönä.



Kuva 2. Esimerkki tuoteluettelon osasta esitettynä a) puu-graafina ja b) XML muodossa

Rakenneosat: Attribuutit

Elementteihin voidaan lisätä tietoa (tekstisisältöä) myös attribuuteilla. Attribuutti on *avain-arvopari*. Attribuutin nimeä vastaa sille annettu arvo, joka esitetään lainaus- tai heittomerkeissä. Arvon sisältö voi olla mitä tahansa tekstiä ja entiteettejä.

```
<kurssi koodi=” ELEC-C8203" jaksot=”III”> Automaatiojärjestelmät 2 </kurssi>
```

3. Dokumentin rakenne: Attribuutti vs. lapsielementti

Attribuutti voidaan aina korvata lapsielementillä siten, että attribuutin nimestä saadaan elementin nimi ja attribuutin arvosta saadaan elementin tekstisisältö. Edellisen esimerkin tietosisältö voidaan siis esittää pelkästään elementtejä käyttäen seuraavasti:

```
<kurssi>
   <koodi> ELEC-C8203 </koodi>
   <jaksot> III </jaksot/>
   <nimi> Automaatiojärjestelmät 2 </nimi>
</kurssi>
```

4. Hyvämuotoinen XML dokumentti

XML dokumentin ensimmäisellä rivillä on oltava seuraava alkumäärittely:

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

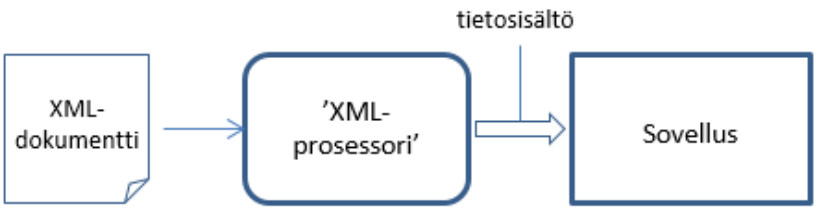
XML-dokumentti on hyvämuotoista (well-formed), kun

1. dokumentti alkaa XML-määrittelyllä
2. dokumentissa on vain yksi juurielementti, jonka lapsia kaikki muut elementit ovat
3. Isä-elementeillä on alku- ja loppuosa
4. tyhjät elementit on merkitty loppukauttaviivalla
5. elementit ovat sisäkkäin, eivät ristikkäin
6. dokumentissa ei käytetä merkkejä < ja & kuin elementtien ja entiteettien alussa

Jos dokumentti ei täyty näitä vaatimuksia, se ei ole virallisesti XML:ää.

5. XML:n peruskäyttö sovelluksissa

Sovelluksen tulee voida lukea ja kirjoittaa XML-dokumentteja ohjelmallisesti. XML:n prosessointiin on kehitetty valmiita ohjelmakirjastoja, joita sovellus voi kutsua niiden rajapintojen (API) kautta (kuvassa 3: ’XML prosessori’). XML-dokumenttien käyttö on yleensä kaksivaiheista: ’XML-prosessori’ lukee XML-dokumentin jostakin lähteestä ja sovellus saa ’XML-prosessorilta’ XML-dokumentin sisällön käyttöönsä (Ks. Kuva 3)



Kuva 3.

Näin on pyritty helpottamaan sovellusten laatijoiden työtaakkaa ja samalla pitämään huoli, että XML-dokumentteja käsitellään samalla tavalla sovelluksesta toiseen. Melkein kaikkiin ohjelmointikieliin löytyy yksi tai useampia rajapintoja XML-dokumenttien käsittelyyn näiden rajapintojen takana on yleensä XML-parseri (tai -jäsennin), joka lukee ja muokkaa XML-dokumentin johonkin ohjelmointikielelle sopivaan muotoon. XML ei ota kantaa, mitä sovellus dokumentilla tekee.

6. Lyhyt XML ja JSON esitystapojen vertailu

Sekä XML että JSON ovat sekä ihmisen että koneen luettavaksi tarkoitetun tiedon esitysmuotoja (data format). Molemmat ovat ohjelmointikielestä riippumattomia esitysmuotoja, mutta JSON on erityisesti JavaScript käsittelyyn optimoitu esitysmuoto. Seuraavassa esimerkissä henkilöiden yhteystietoja on esitetty sekä XML:llä että JSON:lla

```xml
<contacts>
   <contact type="personal">
      <prefix type="common">Mr.</prefix>
      <firstname>Michael</firstname>
      <lastname>Szul</lastname>
   </contact>
</contacts>
```

```json
{ "contacts" : [
   {
      "contacttype" : "personal",
      "prefixtype" : "common",
      "prefix" : "Mr.",
      "firstname" : "Michael",
      "lastname" : "Szul"
   }
] }
```

Kuva 4. Yhden henkilön yhteystiedot kontaktilistassa esitettynä a) XML-kielellä ja b) JSON- kielellä

XML:ssä on kaksi tiedon kuvaustasoa: elementit ja attribuutit. Elementit merkataan alkumerkillä (<contact>) ja loppumerkillä (</contact>) ja ne voivat olla sisäkkäisiä (esim: firstname-elementti on contact-elementin sisällä). Elementteihin voidaan lisätä metatietoa (toinen taso) attribuuttien avulla (esim. type=”personal”). Attribuutti on avain-arvo pari.
JSON:ssa kaikki tieto esitetään avain-arvo pareina (esim: ”firstname” : ”Michael”), minkä vuoksi kuvan esimerkissä metatieto (tyyppitiedot) joudutaan esittämään samoin kuin varsinainen henkilötieto. Lisäksi kuvan esimerkissä joudutaan luomaan kaksi type-avainta, koska kahta saman nimistä avainta ei voi käyttää samassa objektissa ({ objekti }).

[JSON] https://codepunk.io/xml-vs-json-why-json-sucks/