JouTo

Toimittajan työväline

Suunnitteludokumentti

Versio 0.3

26.2.2001
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos
Ohjelmistotuotantoprojekti, kevät 2001, ryhmä 3

Projektiryhmä:	Petri Hellemaa (PH)
	Antti Koskinen (AK)
	Lari Laine (LL)
	Jarno Saarto (JS)
	Eija Teitto (ET)
	Harri Tuuloskoski (HT)
Ohjaaja:	Kim Viljanen
Vastuuhenkilö:	Raine Kauppinen
Asiakkaat:	Helena Ahonen-Myka
	Greger Lindén

Dokumentin versiohistoria

0.1	19.02.2001	Ensimmäinen hahmotelma (AK)
0.2	23.02.2001	Lisätty arkkitehtuurikuvaus sekä tyyliohje (AK)
0.3	26.02.2001	Korjattu yleiskuvaus ja arkkitehtuurikuvausta

Tästä dokumentista vastaa Antti Koskinen (AK).

Sisältö

1 Johdanto (AK)

Tässä suunnitteludokumentissa esitellaan JouTo-ohjelmistotuotantoprojektin tuotteena syntyvä uutistoimittajan työväline. JouTo on lyhenne englannin kielen sanoista *"A Journalist's Tool"*.

1.1 Määritelmät, termit, lyhenteet

CSS Cascading Style Sheets, W3C:n kehittämä, suosittu HTML- ja XML-dokumenttien ulkoasun määrittelyyn tarkoitettu kieli.
Lisätietoja: `http://www.w3.org/Style/CSS/`

DoReMi Document Management, Information Retrieval, Text and Data Mining. TKTL:n tutkimusryhmä, joka vetää TYTTI-projektia.
Lisätietoja: `http://www.cs.helsinki.fi/research/doremi/`

DTD Document Type Definition, W3C:n kehittämä rakenteisten dokumenttien rakenteen määrittelykieli.

EPS Encapsulated PostScript, kehystetty versio PostScript-kielisestä ohjelmasta. Ohjelmatiedostoon lisätty kehys määrittää rajat tiedoston sisällölle ja mahdollistaa sen sijoittamisen haluttuun kohtaan sivua.

HTML Hypertext Markup Language, WWW-sivujen sisällön kuvauksessa käytetty koodaus.
Lisätietoja: `http://www.w3.org/pub/WWW/MarkUp/`

IPTC International Press Telecommunications Council, kansainvälinen yhteistyöelin, joka organisoi uutisten määrittelyä ja ryhmittelyä.
Lisätietoja: `http://www.iptc.org/`

Java Sunin kehittämä ohjelmointikieli ja järjestelmäriippumaton ajoympäristö.
Lisätietoja: `http://www.sun.com/java/`

Javadoc Sunin kehittämä java-koodin dokumentointi kieli
Lisätietoja: `http://www.sun.com/java/`

Java SDK Sunin kehittämä Java-ohjelmointiympäristö.
Lisätietoja: `http://www.javasoft.com/products/`

JDBC Java Database Connectivity, Java-kielen rajapinta tietokantojen käyttöön.

NITF News Industry Text Format, toimittajien kehittämä ja standardoima XML-pohjainen uutisartikkelien määrittelykieli. Lisätietoja: `http://www.nitf.org`

PostScript Adoben kehittämä suosittu sivunkuvauskieli. Tulostettava sivu voidaan lähettää PostScript-kirjoittimelle tiiviinä PostScript-ohjelmana tilaavievän raakadatan sijaan.

TKTL Helsingin yliopiston tietojenkäsittelytieteen laitos.
Lisätietoja: `http://www.cs.helsinki.fi`

TYTTI Tietotyöläisen työasema. Vuosina 2000-2002 käynnissä oleva tutkimusprojekti Helsingin yliopiston tietojenkäsittelytieteen laitoksella, rahoittajina TEKES sekä viisi suomalaisyritystä: Nokia, Alma Media, Sanoma-WSOY, Vaisala ja Lingsoft. TYTTI-projektin vastuuhenkilö Helena Ahonen-Myka ja projektipäällikkö Greger Lindén toimivat JouTo-projektin asiakkaina.
Lisätietoja: `http://www.cs.helsinki.fi/research/doremi/TYTTI/index.html`

W3C World Wide Web Consortium, kansainvälinen yhteistyöelin, joka koordinoi WWW-tekniikan ja -standardien kehitystä.
Lisätietoja: `http://www.w3.org/pub/WWW/Consortium/`

WWW World-Wide Web, Internetissä toimiva palvelu, joka mahdollistaa tekstin ja kuvien jakamisen helposti ja edullisesti verkon lukemattomille käyttäjille. WWW:ssä aineisto jaetaan sivuihin, jotka sijoitetaan WWW-palvelimelle tavallisina tiedostoina ja hakemistoina.

XML Extensible Markup Language, W3C:n kehittämä rakenteisten dokumenttien määrittelykieli.
Lisätietoja: `http://www.w3.org/XML/`

XSL Extensible Stylesheet Language, W3C:n kehittämä XML-dokumenttien ulkoasun määritykseen tarkoitettu kieli.
Lisätietoja: `http://www.w3.org/Style/XSL/`

1.2 Viitteet

NITFDTD News Industry Text Format DTD version 2.5 8/2000, `http://www.nitf.org`

1.3 Dokumentin rakenne

Ensimmäinen luku, Johdanto, esittelee tämän dokumentin rakenteen.

2 Yleiskuvaus (AK ja ET)

2.1 Sovellusalueen kuvaus

JouTo-ohjelmistotuotantoprojekti on keväällä 2001 Helsingin yliopiston tietojenkäsittelytieteen laitoksella järjestettävän Ohjelmistotuotantoprojekti-kurssin näytetyö.

Työ liittyy osittain myös laitoksella käynnissä olevaan DoReMi-tutkimusryhmän "TYTTI- Tietotyöläisen työasema"-projektiin, jossa on mukana rahoittajina joukko suomalaisia teknologiayrityksiä ja Teknologian kehittämiskeskus. JouTo-projektin tarkoituksena ei kuitenkaan ole valmistaa valmista tuotetta kaupalliseen tarkoitukseen. Tuotettava ohjelmisto toimii prototyyppinä, jota voidaan kehittää myöhemmin lisää.

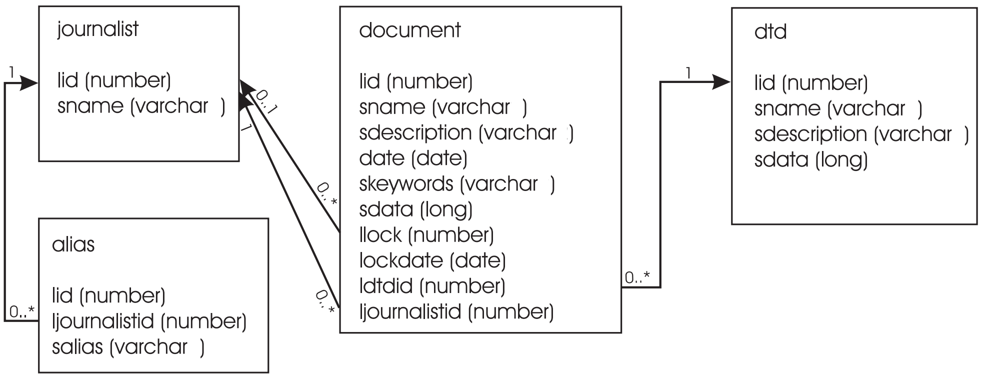
2.2 Laitteistoympäristö (ET)

Ohjelmiston arkkitehtuuri on asiakas-palvelin pohjainen. Asiakaskoneissa pyöritettävä sovellus vaatii Pentium-teholuokan koneen, jossa on jokin toimiva tietoliikeneyhteys. Palvelimen puolella on ratkaista toimiiko palvelinohjelmisto samalla koneella vai eri koneella kuin tietokantapalvelin. Palvelimen kokomääritykset on esitetty määrittelydokumentissa kappaleessa 7.2. Palvelimen suorituskyky. Toimintaynpäristöksi oletetaan toimituksen sisäinen verkko.

2.3 Ohjelmistoympäristö (ET)

Ohjelmistoa käytetään WWW-selaimen kautta. Ohjelmiston käyttöliittymä, editori, toteutetaan WWW-selaimen graatisena Java Appelttina. Editori -osa toimii rajapintana käyttäjän ja varsinaisen ohjelmiston toiminnan välillä. Varsinainen ohjelmisto, joka huolehtii artikkelin laadinnassa tai muokkauksessa tarvittavista toiminnoista, kuten siirtäminen, kopionti, liittäminen, lataus, tallennus, DTD:n muodostamisen ja DTD:n mukaisen puunrakenteen luomisen, toteutaan Javalla, XML Parsel -ja Wutkan DTD -komponentilla.

Tietokoneella laaditut ja muokatut artikkelit tallentaan palvelimella olevaan tietokantaan XML-muodossa. Tietokannasta voidaan ladata artikkeleita joko käsittelyä varten tai uuden artikkelin pohjaksi. Ohjelmiston käytössä olevat DTD:t sijaisevat myös tietokannassa. Palvelimen ohjelmisto toteutetaan Servlettinä. Tallennus ja lautaus -osan ja palvelimen ohjelmiston välissä on rajapinta.



Kuva 1: Taulujen kuvaus

2.4 Standardit ja Sopimukset

2.4.1 Ohjelmointityyli

Ohjelmointityylistä on kerrottu omassa dokumentissaan.

3 Ohjelmiston arkkitehtuurikuvaus (PH, ET ja HT)

Tässä kappaleessa selvitetään ohjelmiston arkkitehtuuria. Kappale 3.1 sisältää tietokanakuvauksen. Kappale 3.2 sisältää kuvauksen yleisellä tasolla. Kappale 3.3 kuvaa arkkitehtuuria tarkemmalla tasolla tietovuokaavioiden avulla. Kappaleissa 3.4 ja 3.5 kuvataan arkkitehtuuria yksityiskohtaisella tasolla

3.1 Tietokannan kuvaus (JS)

Järjestelmään kuluva tietokanta rakentuu neljästä taulusta (kuva 1): *alias, journalist, document ja dtd.* Jokaisessa taulussa on sarake *lid*, joka yksilöi rivin. Tämä sarake on myös jokaisen taulun pääavain. Tietokantaan kirjoittamisen ja sieltä lukemisen hoitaa yksinomaan palvelinohjelmisto.

3.1.1 Taulu alias

Alias-tauluun tallennetaan toimittajien mahdolliset nimimerkit eli aliakset. Jokaisella toimittajalla voi olla rajattomasti aliaksia. Ohessa jokainen taulun sarake kuvauksineen.

Sarake:	Tehtävä:
lid	Pääavain (*number*), joka yksilöi aliaksen.
ljournalistid	Vierasavain (*number*), joka viittaa journalist-taulun pääavaimeen (*lid*). Sarakkeen avulla selviää kenen toimittajan alias on kyseessä.
salias	Alias tekstinä (*varchar*).

3.1.2 Taulu document

Tauluun *document* tallennetaan editorilla tehdyt dokumentit. Pelkän asiakirjan lisäksi tauluun tallennetaan monia tietoja muun muassa hakujen helpottamiseksi. Ohessa jokainen taulun sarake kuvauksineen.

Sarake:	Tehtävä:
lid	Pääavain (*number*), joka yksilöi aliaksen.
sname	Dokumentin nimi muodossa *varchar*.
sdescription	Pidempi selitys dokumentin sisällöstä (*varchar*).
date	Dokumentin viimeisin tallennusaika (muodossa *date*).
skeywords	Kuvaavia hakusanoja, jotka helpottavat hakua (*varchar*).
sdata	Sarake, johon itse dokumentti on tallennettu (*long*).
llock	Kirjoituslukko, joka toimii samalla vierasavaimena journalist -tauluun. Jos lukko on määritelty, näkee samalla kuka sen on varannut.
lockdate	Lukon asetusaika (muodossa *date*). Tätä tietoa käytetään varmistamaan, että lukko ei ole jäänyt päälle esimerkiksi tietoliikenneyhteyden katkeamisen vuoksi.
ldtdid	Vierasavain, joka osoittaa dtd-taulun pääavaimeen. Sarakkeesta käy ilmi minkä dtd:n mukaan asiakirja on luotu.
ljournalistid	Vierasavain (*number*), joka viittaa journalist-taulun pääavaimeen (*lid*). Tämän sarakkeen avulla selviää dokumentin edellinen tallentaja.
salias	Alias tekstinä (*varchar*).

3.1.3 Taulu dtd

Tauluun *dtd* tallennetaan dokumenttirakenteen määräävät dtd-tiedostot. Pääavaimen *lid* lisäksi tauluun tallennetaan dtd:n nimi (*sname*), kuvaus (*description*) ja itse dtd, joka tulee sarakkeeseen (*sdata*). Ohessa jokainen taulun sarake kuvauksineen.

Sarake:	**Tehtävä:**
lid	Pääavain (*number*), joka yksilöi DTD:n.
sname	DTD:n nimi *(varchar)*.
sdescription	Pidempi kuvaus DTD:stä *(varchar)*.
sdata	Sarake, johon itse DTD on tallennettu (*long*).

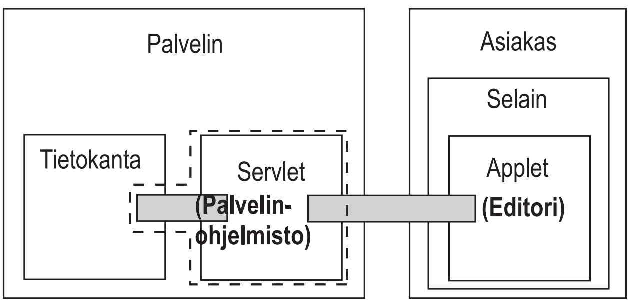
3.1.4 Taulu journalist

Tauluun *journalist* tallennetaan toimittajien tiedot. Taulussa on pääavaimen *lid* lisäksi *sname*, johon tallennetaan toimittajan nimi. Ohessa jokainen taulun sarake kuvauksineen.

Sarake:	**Tehtävä:**
lid	Pääavain (*number*), joka yksilöi DTD:n.
sname	Toimittajan nimi *(varchar)*.

3.2 Yleinen arkkitehtuurikuvaus

Ohjelmiston arkkitehtuuri on asiakas-palvelin pohjainen. Asiakkaan puolen ohjelmisto on WWW-selaimessa toimiva Java-sovelma (Applet), ja palvelimen puolen ohjelmisto (Servlet) huolehtii tietojen tallennuksesta sekä latauksesta tietokannasta.

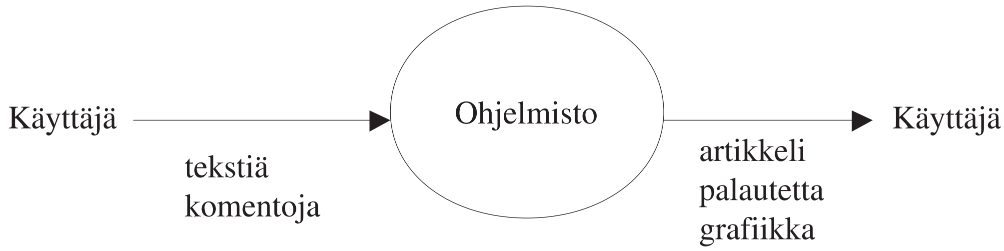


Kuva 2: Ohjelmiston arkkitehtuuri

Kuvassa 2 nähdään ohjelmiston arkkitehtuuri yleisellä tasolla.

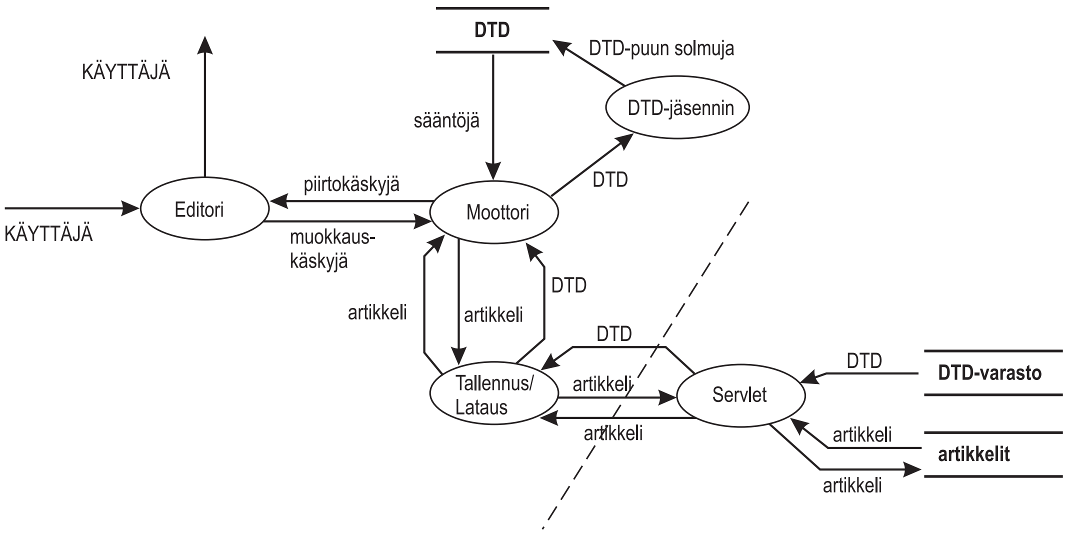
3.3 Tarkempi arkkitehtuurikuvaus

Tässä kappaleessa kuvataan ohjelmiston arkkitehtuuria hieman tarkemmin kuin kappaleessa 4.1.



Kuva 3: Ohjelmiston tietovuokaavio - 1. taso

Kuvassa 3 nähdään ohjelmiston toiminta yleisellä tasolla (tietovuokaavion 1. taso).



Kuva 4: Ohjelmiston tietovuokaavio - 2. taso

Kuvassa 4 nähdään ohjelmiston komponenttien ja erilaisten tietovarastojen sijoittelu ja niiden keskinäinen toimintalogiikka.
Käyttäjä käyttää editoria artikkelien käsittelyyn. Editori on toteutettu Java-sovelmalla (Applet), joka toimii käyttäjän WWW-selaimessa. Käyttäjän ohjelmisto on jakautunut neljään osa-alueeseen, moottoriin, editoriin, DTD-jäsentimeen

sekä tallennus/lataus-osaan.
Editori-osa huolehtii käyttöliittymän luomisesta käyttäjälle, sekä käyttäjän syötteiden välittämisestä eteenpäin moottorille. Editori-osa muokkaa käyttöliittymän valitun artikkelin mukaisesti ja huolehtii että käyttäjä voi luoda vain valitun artikkelin mukaisia rakenteita. Editori-osa toimii rajapintana käyttäjän ja varsinaisen ohjelmiston toiminnan välillä.
DTD-jäsennin huolehtii valitun DTD:n tulkitsemisesta ja sääntöverkon luomisesta. Tämän verkon avulla valitun artikkelin oikeellisuutta pystytään tarkkailemaan ja käyttöliittymän avulla ohjaamaan.
Tallennus/lataus-osa huolehtii kommunikoinnista palvelimen puolen ohjelmiston (Servlet) kanssa. Tallennus/lataus-osan rajapinnan kautta suoritetaan kaikki lataus- ja tallennusoperaatiot.
Mootori huolehtii ohjelman toimintalogiikasta ja muiden osien ohjaamisesta. Moottori huolehtii muiden osien keskinäisestä kommunikoinnista sekä suorittaa tarvittavat operaatiot ohjelman ajamiseksi.

Servlet huolehtii artikkelien ja DTD:ien latauksesta, tallennuksesta ja poistamisesta tietokannasta. Servlet on toteuttu Java-sovelluksena, joka pyörii palvelimen Java-moottorin (Java-engine) avulla palvelimen prosessina. Servlet ottaa vastaan editorista tulleita syötteitä, tekee tarvittavat toimenpiteet palvelimen puolella ja palauttaa tulosteet takaisin editorille. Servlet on ainoa komponentti, joka pääsee käsittelemään tietokannan sisältöä. Servlet tarjoaa appletille rajapinnan tietokanta-operaatiohin. Näin tietokannan rakenne pysyy eheänä ja toteutus piilossa ulkopuolisilta elementeiltä.

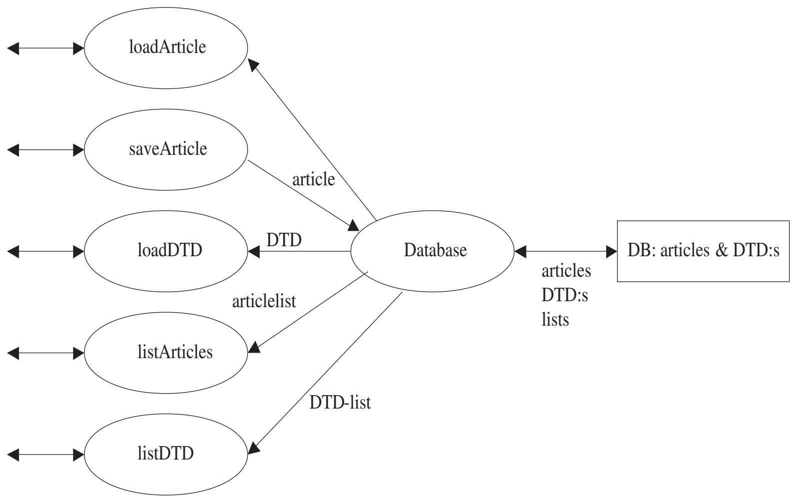
Tietokantaan on tallennettuna eri DTD:t, DTD:den mukaiset artikkelit, eri käyttäjien tiedot sekä muut JouTo-ohjelmiston tarvitsema informaatio. Tietokanta on Oracle 8-pohjainen ja sijaitsee Helsingin Yliopiston atk-keskuksen *Kontti*-tietokantapalvelimella. Tietokantaan on luotu tarvittavat taulut ja oikeudet, jotta JouTo-ohjelmistoa pystyy ajamaan.
Tarkempaa tietoa tietokannasta ja sen rakenteesta löytyy kappaleesta **??**

3.4 Palvelimen yksityiskohtainen arkkitehtuurikuvaus

Kuvassa 5 nähdään palvelimen ohjelmiston komponenttien ja erilaisten tietovarastojen sijoittelu ja niiden keskinäinen toimintalogiikka.

Database hoitaa rajapinnan ja liikennöinnin tietokantaan.

loadArticle huolehtii halutun artikkelin lataamisesta tietokannasta. Tämän servletin kautta hoidetaan artikkelien siirtäminen asiakassovellukselle palvelimelta. Jos tietokantaoperaatio epäonnistuu palautetaan virheilmoitus



Kuva 5: Palvelimen tietovuokaavio - 3. taso

loadDTD huolehtii halutun DTD:n lataamisesta tietokannasta. Tämän servletin kautta hoidetaan DTD:iden siirtäminen asiakassovellukselle palvelimelta. Jos tietokantaoperaatio epäonnistuu palautetaan virheilmoitus

saveArticle huolehtii asiakassovellukselta tulleen artikkelin tallennuksesta tietokantaan. Servletti pitää myös yllä muita artikkeliin liittyviä kenttiä tietokannassa. Jos tietokantaoperaatio epäonnistuu palautetaan virheilmoitus

listArticles listaa tietokannassa olevista artikkeleista otsikkotietoja, jotka se lähettää asiakassovellukselle. Jos tietokantaoperaatio epäonnistuu palautetaan virheilmoitus

listDTD listaa tietokannassa olevista DTD:istä otsikkotietoja, jotka se lähettää asiakassovellukselle. Jos tietokantaoperaatio epäonnistuu palautetaan virheilmoitus

3.5 Asiakassovelman yksityiskohtainen arkkitehtuurikuvaus

Asiakassovelmasta ei tässä vaiheessa voida antaa tarkempaa arkkitehtuurikuvausta. Tarkempi kuvaus voidaan tehdä vasta kun ensimmäinen prototyyppi on valmis.

4 Luokkakuvaukset PH ja LL

Luokat eivät ole vielä missään järjestyksessä. Luokkien järjestystä tullaan muuttamaan tulevaisuudessa.

4.1 ContentManager (PH)

ContentManager-olio huolehtii yhden NITF-dokumentin sisällön hallinnasta. Dokumentin muokkaaminen tapahtuu luokan aksessorien kautta. Olio pitää huolen siitä, että jokaisen muutoksen jälkeen dokumentti noudattaa rakennemäärittelyä. ContentManager perii luokalta *org.xml.sax.helpers.DefaultHandler* rajapinnan SAX-jäsentimeen tapahtumankäsittelymetodeineen. Artikkelia säilytetään XML-elementeistä koostuvassa puurakenteessa. Elementit ovat Elem-olioita ja elementtien attribuutit Attrib-olioita.

Kun sovelma on vastaanottanut DTD:n, luodaan ContentManager-olio kutsumalla luokan konstruktoria, joka ottaa parametreikseen mahdollisesti vastaanotetun artikkelin ja DTD:n InputStream-olioina. Ensin käsitellään DTD *Mark Wutkan* DTD-jäsentimellä[1], joka luo DTD-olion. Se puolestaan sisältää rakennemäärittelyssä esitellyt elementit hajautustaulussa. Elementtien ja attribuuttien määrittelyt jäsentyvät eri luokkien olioiksi. Jäsentimen Javadoc-dokumentaatio on osoitteessa `http://www.wutka.com/dtdparserapi/`

SUUNNITTELUPERIAATTEISIIN: Koska nykyiset validoivat XML-jäsentimet vaativat syötteekseen kokonaisen artikkelin DTD-viitteineen merkkijonomuodossa ja tarkastavat paitsi artikkelin jokaisen osan rakenteen myös hyvämuotoisuuden, niiden käyttö olisi aivan liian hidasta jatkuvan validiuden pakottavassa ohjelmassa.

Kun DTD on jäsennetty, ContentManager käy läpi elementtien lapsirajoitteet ja tekee säännöllisistä lausekkeista epädeterministisiä äärellisiä automaatteja muistuttavia sääntöverkkoja aina, kun kyseessä ei ole tyhjä kieli EMPTY tai universaalikieli ANY. Sääntöverkot rakentuvat ANode-olioista. Sääntöverkot sekä DTDAny- ja DTDEmpty-oliot sijoitetaan hajautustauluun *childConstraints* elementtien nimillä, joilla validointiin käytettävä *acceptChildren*-metodi pääsee niihin käsiksi.

Sääntöverkkojen avulla tarkistetaan, johtaako elementin lapsisarjaan kohdistuva muokkaustoiminto[2] DTD:n sallimaan rakenteeseen. Elementin e lapsisarjaan kohdistuva muutos sallitaan vain, jos e:n säännöllinen lauseke tai siitä luotu sääntöverkko tuottaa muutoksen tuloksena syntyvän lapsisarjan.

Attribuutteihin liittyvät rajoitteet selviävät DTD-jäsentimen elementtikohtaisesti luomista DTDAttribute-olioista.

[1] http://www.wutka.com/dtdparser.html

[2] Lapsisarjaan kohdistuvia muutoksia ovat lasten lisäys, poisto, järjestyksen muuttaminen ja korvaaminen.

Kun tietokannasta ladataan artikkeli, käytetään elementtipuun rakentamiseen *Apache Xerces Java* -kirjaston[3] SAX-jäsennintä. XML-kirjaston dokumentaatio on osoitteessa
`http://xml.apache.org/xerces-j/api.html`
Ennen dokumentin syöttämistä jäsentimelle siitä poistetaan viite rakennemäärittelyyn, koska DTD:t ovat tietokannassa eikä niillä ole URL-osoitetta. SAX-jäsennin tarkistaa siis vain vastaanotettavan dokumentin hyvämuotoisuuden. Jäsennintä käytetään vain rakennettaessa ladatusta dokumentista elementtipuuta.

Dokumentin jäsennys SAX-rajapinnalla on tapahtumaohjattua. Jäsennin lukee dokumenttia suoraviivaisesti alusta loppua kohti ja jokaisen elementin alkaessa ja loppuessa lähettää SAX-tapahtuman. XML-elementti validoidaan vasta jäsentimen kohdatessa elementin lopun, jolloin vastava SAX-tapahtuma siirtää suorituksen endElement-metodiin. Vasta tällöin kaikki lapsielementit ovat tiedossa. Lapsisarja validoidaan sääntöverkoilla ja attribuutit DTD-jäsentimen luomien DTDAttribute-olioiden tiedoilla.

ContentManagerin *toString*-metodi palauttaa artikkelin XML-muodossa.

Kun luodaan uusi artikkeli, kutsutaan ContentManagerin konstruktoria ainoastaan dtd-parametrilla. ContentManager rakentaa hyväksyttävän artikkelipohjan läpikäymällä sääntöverkkojen solmuja. Elementille lisätään aina verkon ensimmäisen polun määrittelemä lapsisarja. Verkot rakennetaan niin, että ensimmäinen polku on lyhin mahdollinen. Jos elementti ei vaadi lapsielementtejä, niitä ei luoda.

Peruutustoiminto?

Leikepöytä?

4.1.1 Attribuutit

private DTD dtd
DTD-jäsentimen palauttama DTD-olio.

private Elem nitf
Elementtipuun juurisolmu, joka esittää NITF-elementtiä.

private Hashtable childConstraints
Hajautustaulu, joka sisältää jokaista DTD:ssä määriteltyä elementtiä kohti sen lapsirajoitteet esittävän säännöllisen lausekkeen pohjalta rakennetun sääntöverkon tai DTDAny-olion tai DTDEmpty-olion. Hajautustaulun avaimina käytetään elementtien nimiä.

4.1.2 Metodit

public ContentManager(InputStream dtd)

[3] http://xml.apache.org/xerces-j/

Luo ContentManager-olion ja pohjan uudelle artikkelille. Artikkelipohja on minimaalinen hyväksyttävä elementtipuu. Parametrin *dtd* avulla välitetään rakennemäärittely jäsennettäväksi.

public ContentManager(InputStream dtd, InputStream doc)
Luo ContentManager-olion ja rakentaa elementtipuun ladatusta artikkelista *doc*. Parametrin *dtd* avulla välitetään rakennemäärittely ja parametrin *doc* avulla dokumentti jäsennettäviksi.

private boolean addLastChild()
Tällä metodilla lisätään elementin lapsisarjan jatkoksi uusi elementti. Metodia käytetään vain puuta rakennettaessa.

Artikkelin sisällön muokkaus tapahtuu seuraavanlaisilla metodeilla:

public boolean addAttribute(Attrib a, Elem e)
public boolean delAttribute(Attrib a, Elem e)
public boolean setAttribute(Attrib a, Elem e, String value)
public boolean addChar(PCData e, int index)
public boolean delChar(PCData e, int index)
public boolean addChildAt(Elem child, Elem parent, int index)

Yrittää lisätä elementin *child* *parent*-elementin lapseksi lapsisarjan kohtaan *index*. Palauttaa tiedon onnistumisesta totuusarvona.

Muokkausrajapintaa kehitetään käyttöliittymäkomponenttien tarpeiden mukaisesti.

public Elem getRoot()
Palauttaa viitteen puun juurielementtiin. Metodia käyttää tietoliikenneluokka tallennuksen yhteydessä.

public Elem getHead()
Palauttaa viitteen puun head-elementtiin. Metodia käyttää artikkelin metatietojen muokkauskomponentti.

public Elem getBody()
Palauttaa viitteen puun body-elementtiin. Metodia käyttää artikkelin rungon muokkauskomponentti.

public boolean acceptElement(Elem e)
Onko elementti *e* hyväksyttävä.

public boolean acceptChildren(Elem e)
Onko elementin *e* lapsisarja hyväksyttävä.

public boolean acceptAttributes(Elem e)
Ovatko elementin *e* kaikki attribuutit hyväksyttäviä.

public boolean acceptAttribute(Attrib a, String elem)
Onko *elem*-nimisen elementin attribuutti *a* hyväksyttävä.

public String[] getAcceptableSuccessors(String parent, String[] predecessors)
Palauttaa välittömästi *parent*-elementin lapsien *predecessors* jälkeen sisaruksiksi lisättävissä olevien elementtien nimet.

public String toString()
Palauttaa dokumentin XML-muodossa.

4.2 BodyPanel (PH)

Tämän Panelin aliluokan olion avulla selataan dokumentin runkoa. Se sisältää Scrollbarin, kaksi painiketta ja varsinaisen BodyCanvas-muokkauskomponentin, joka kuuntelee painikkeita ja Scrollbarien säätötapahtumia. Scrollbar on dokumentin selaamiseen riveittäin ja nuolen kuvin varustetut painikkeet sivuttaissuuntaiseen, pitkien esimuotoiltujen tekstirivien selaamiseen. Painikkeilla liikutaan sivusuunnassa puolen sivun verran kerrallaan.

4.2.1 Attribuutit

private Scrollbar lineBar
Scrollbar, jolla dokumenttia selataan riveittäin.

private Button left
Painike vasemmalle selaamiseen.

private Button right
Painike oikealle selaamiseen.

private BodyCanvas bodyCanvas
Varsinainen artikkelin rungon muokkauskomponentti.

4.2.2 Metodit

public BodyPanel()
Luo BodyPanel-olion.

4.3 BodyCanvas (PH)

BodyCanvas-komponentin kautta käyttäjä muokkaa artikkelin runkoa.

4.3.1 Attribuutit

public static final Font TEXT_FONT
Käytetty kirjasintyyppi.

public static final int CHAR_WIDTH
Merkin leveys pikseleinä.

public static final Color BG_COLOR
Taustaväri.

public static final Color TEXT_COLOR
Tekstin väri.

public static final int LINE_HEIGHT
Rivikorkeus.

public static final int TOP_MARGIN
Ylämarginaali.

public static final int BOTTOM_MARGIN
Alamarginaali.

public static final int LEFT_MARGIN
Vasen marginaali.

public static final int RIGHT_MARGIN
Oikea marginaali.

public static final int MAX_COLS
Kuinka monta saraketta voidaan näyttää kerralla.

public static final int MIN_COLS
Kuinka monta saraketta rivin tulisi olla vähintään pitkä, ennen kuin se voidaan katkaista.

public static final int MAX_LINES
Kuinka monta riviä voidaan näyttää kerralla.

private int curLine, curCol
Kohdistimen rivi ja sarake.

private int topLine
Yläreunassa näkyvä rivi.

private int leftCol
Vasemmassa reunassa näkyvä sarake.

private BodyPanel bodyPanel
Viite BodyPaneliin, joka sisältää tämän komponentin.

private Scrollbar lineBar, colBar
Viitteet BodyPanelin Scrollbareihin, joiden säätötapahtumia kuunnellaan.

private Block body
Viite elementtipuun body-elementtiin.

private Image hiddenImage
Kuva piirtopuskurista.

private Graphics hiddenGraphics
Piirtopuskuri, johon teksti piirretään ennen näyttämistä.

private ContentManager man
Viite ContentManager-olioon.

4.3.2 Metodit

public BodyCanvas(Scrollbar lineBar, Scrollbar colBar)
Luo BodyCanvas-olion. Parametreina välitetään viitteet BodyPanelin Scrollbareihin, joiden säätötapahtumia kuunnellaan.

public void paint(Graphics vg)
Piirtää ja näyttää näkymän artikkelin runkoon. *vg* on viite näkyvään grafiikkaan.

public void keyPressed(KeyEvent e)
Suoritetaan, kun jokin erikoisnäppäin painetaan alas näppäintapahtumana *e*.

public void keyReleased(KeyEvent e)
Suoritetaan, kun jokin erikoisnäppäin vapautetaan näppäintapahtumana *e*.

public void keyTyped(KeyEvent e)
Tapahtumankäsittelymetodi, joka suoritetaan, kun käyttäjä kirjoittaa jonkin merkin tai painaa backspace tai delete. *e* on vastaava näppäintapahtuma.

4.4 TreeNode (PH)

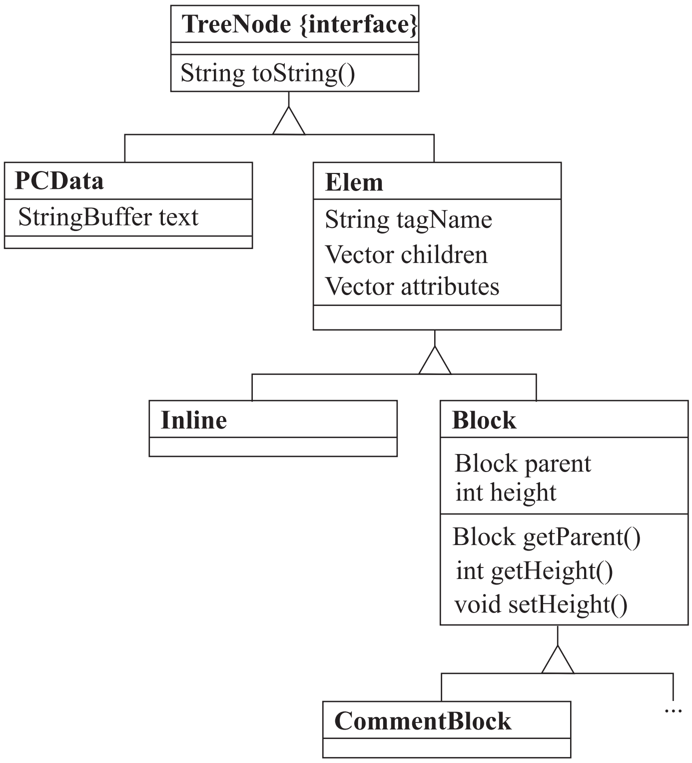
Elementtipuu rakennetaan TreeNode-rajapintaa soveltavista olioista. TreeNode-rajapinnalta peritään metodi *toString*, joka tulostaa solmun sisällön.

PCData vastaa XML-dokumentin PCDATA-osuutta eli tekstiä. Tekstisolmuilla ei ole lapsisolmuja.

Elem-luokan olio vastaa XML-elementtiä. Elem-olioilla on tunnistenimi *tagName* ja vektorit lapsielementeille ja attribuuteille.

Rivielementit ovat Inline-olioita ja lohkoelementit Block-olioita. Block on kuin Elem, mutta se muistaa korkeutensa riveinä ja tuntee isäelementtinsä. Dokumentin selaaminen riveittäin perustuu lohkojen korkeustietoihin. Kun luodaan uusi tyhjä lohko, sen korkeudeksi asetetaan 1. Samalla lisätään isäelementin korkeutta yhdellä rivillä. Aina kun korkeutta muutetaan, muutetaan elementin isän korkeutta yhtä paljon. Kun lohko tuhotaan, vähennetään isälohkon korkeutta tuhotun lohkon korkeuden verran. Lohkoja käytetään vain dokumentin body-elementin alipuussa. bodyn isäelementiksi asetetaan null-osoite. BodyCanvas on linkitetty body-elementtiin.

DTD:ssä ei rivi- ja lohkoelementtejä ole eritelty, vaan BodyCanvas-luokan hajautustaulussa on kaikkien lohkotyyppisten elementtien tunnistenimet. Jos body-alipuun elementti mainitaan taulussa, siitä tehdään Block, muutoin Inline. body-alipuun ulkopuolella elementit ovat Elem-olioita.



Kuva 6: Elementtipuun solmujen luokat.

4.5 PCData (PH)

PCData-olio vastaa PCDATAa XML-dokumentissa.

4.5.1 Attribuutit

private StringBuffer text

4.6 Elem (PH)

Elem-luokan olio esittää XML-dokumentin yhtä elementtiä.

4.6.1 Attribuutit

private String tagName
Elementin tunnistenimi.

private Vector children
Elementin lapsielementit.

private Vector attributes
Elementin attribuutit Attrib-olioina.

4.7 Inline (PH)

Rivielementtien luokka.

4.8 Block (PH)

Block on lohkotyyppinen XML-elementti ja Elem on sen yliluokka. Dokumentin rungon selaaminen perustuu siihen, että jokainen lohko tuntee korkeutensa riveinä. Lohkoja ovat muun muassa kappaleet, listat ja otsikot. Kun uusi lohko luodaan, se saa korkeuden 1. Aina, kun korkeutta muutetaan, lisätään myös lohkon isäelementin korkeutta muutoksen verran.

4.8.1 Attribuutit

private int height
Lohkon korkeus riveinä.

4.8.2 Metodit

public Block getParent()
Palauttaa isäelementin.

public int getHeight()
Palauttaa lohkon korkeuden riveinä.

public void setHeight(int height)
Asettaa lohkon korkeudeksi *height* riviä.

4.9 Attrib (PH)

Tämän luokan olio esittää XML-elementin yhden attribuutin tietosisältöä.

public String toString()
Palauttaa attribuutin XML-esitysmuodossa.

4.10 ANode (PH)

ANode on sääntöverkon solmu. Solmuilla on arvo ja taulukko seuraajasolmuja. Seuraajasolmujen arvot kertovat siirtymät. Verkon polut päättyvät null-viitteeseen.

Kunkin solmun seuraajataulukkoon asetetaan seuraajat seuraavaan järjestykseen: ensiksi mahdollinen ohitussiirtymä, seuraavaksi kaikki ei-tyhjät siirtymät ja viimeiseksi mahdollinen silmukkasiirtymä. Kuvassa 7 on esitetty verkon rakennussäännöt.

Käytännössä solmut sisältävät elementtien nimiä ja verkkoja käytetään elementin lapsisarjan hyväksyttävyyden tarkistamiseen. ContentManager kutsuu verkon juurisolmun *acceptPath*-metodia parametreilla *path* ja *index*. *path*in pitää sisältää lapsisarjan elementtien nimet String-taulukossa ja *index*-muuttujan arvon pitää olla 0. *acceptPath* on rekursiivinen metodi ja *index* osoittaa kohtaa, johon asti polku on hyväksytty. Silmukoiden välttämiseksi matkan varrella merkitään tyhjän merkkijonon solmut (lambda-solmut) kulloisellakin indeksillä. Jos saavutaan lambda-solmuun samalla indeksillä kuin millä se on merkattu, tiedetään, että jatkamalla jouduttaisiin silmukkaan. Näissä tilanteissa kurkistus hylätään ja siirrytään seuraavaan haaraan.

Kun käyttäjä yrittää lisätä elementin, joka vaatii lapsielementtejä, sovelletaan aina solmun ensimmäisen seuraajan sääntöä. Esimerkiksi lausekkeella (a | b) luodaan lapsi a, lausekkeella (a | b)* ei luoda yhtään lasta, ja lausekkeella (((a | b), c) | d)+ luodaan lapset a ja c. Myöhemmin käyttäjä voi tietysti korvata elementit, joihin ei ole tyytyväinen, joillakin toisilla. Ensimmäisen säännön periaate tulee ottaa huomioon rakennemäärittelyä kirjoitettaessa.

4.10.1 Attribuutit

private String value
Solmun (säännön) tuottama merkkijono.

private int index
Apukenttä, jolla voidaan merkitä kuljettu polku silmukoiden välttämiseksi.

private ANode[] successors
Solmun seuraajat.

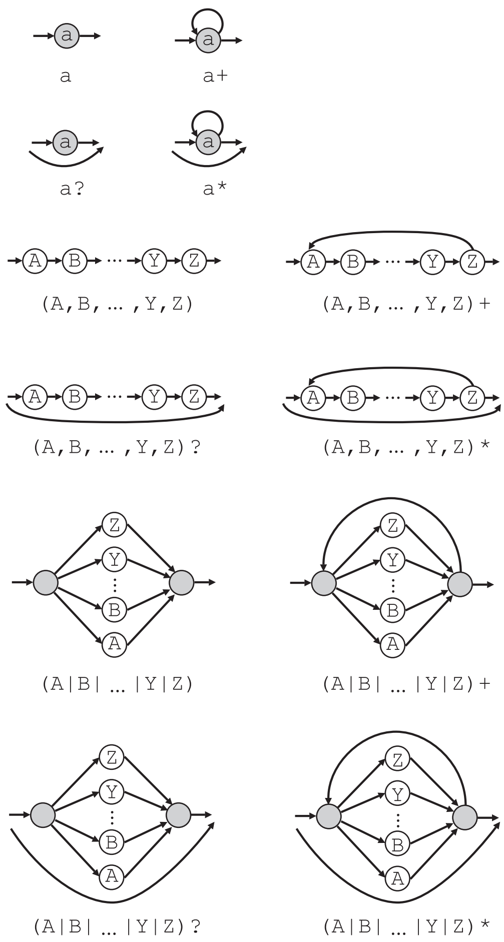
public static final String LAMBDA
Tyhjä merkkijono "".

4.10.2 Metodit

public ANode(String value, ANode[] successors)
Luodaan uusi solmu, jolle annetaan arvoksi *value* ja seuraajiksi *successors*.

public boolean acceptPath(String[] path, int index)
Kertoo, tuottaako jokin solmusta alkava polku merkkijonot *path[index]...path[N]*, missä *N* on taulukon suurin indeksi.

public void addSuccessor(ANode successor)
Lisää solmulle uuden seuraajasolmun *successor*.



Kuva 7: Sääntöverkon rakentaminen säännöllisestä lausekkeesta.

4.11 Communications.java

Tämä olio tarjoaa rajapinnan palvelin puolelle.

4.11.1 String loadArticle(integer articleID)

parametrit: Artikkelin id-numero

palauttaa: Tietokannasta ladatun artikkelin yhtenä string oliona.

toiminta: Kutsuu loadArticle nimistä servlettiä, joka lataa tietokannasta halutun artikkelin.

4.11.2 String loadDTD(integer DTDID)

parametrit: DTD:n id-numero

palauttaa: Tietokannasa ladatun DTD:n yhtenä string oliona.

toiminta: Kutsuu loadDTD nimistä servlettiä, joka lataa tietokannasta halutun DTD:n.

4.11.3 String saveArticle(String[][] artikkeli)

parametrit: Haluttu artikkeli contentManager-oliona, tieto siitä halutaanko tallentaa samalla nimellä vai uudella

palauttaa: mahdollinen tietokannan virheilmoitus

toiminta: Ensin muutetaan artikkeli xml-dokumentin muotoon ja sen jälkeen lähetetään saveArticle-servletille artikkeli. Uusi-parametrillä määritellään tehdäänkö tallennus vanhan päälle vai uutena rivinä kantaan.

4.11.4 String[][] listArticles()

parametrit: —

palauttaa: String taulukon joka sisältää kannassa olevien artikkeleiden listauksen.

toiminta: Kutsuu listArticles-servlettiä, joka hakee kannasta artikkelien otsikot.

4.11.5 String[][] listDTD()

parametrit: —

palauttaa: String taulukon joka sisältää kannassa olevien DTD:iden otsikot

toiminta: Kutsuu listDTD-servlettiä, joka hakee kannasta DTD:iden otsikot.

4.12 Servletit

Kaikki servletit rakentuvat yhdestä *service*-functiosta. Tämän lisäksi ne kutsuvat Database-olion functioita tietokannan käyttämiseksi.

4.12.1 loadArticle.java

parametrit: id = artikkelin id-numero

palauttaa: Halutun artikkelin

toiminta: Hakee tietokannasta artikkelin id-numeron perusteella oikean artikkelin ja lähettää sen appletille.

4.12.2 loadDTD.java

parametrit: id = DTD:n id-numero

palauttaa: Halutun DTD:n

toiminta: Hakee tietokannasta DTD:n id-numeron perusteella oikean DTD:n ja lähettää sen appletille.

4.12.3 saveArticle.java

parametrit: ??? Kaikki taulun kentät ja artikkeli kokonaisena

palauttaa: ok tai mahdollisen tietokantavirheen tuloste

toiminta: Tallentaa artikkelin ja muut artikkeliin liittyvät kentät tietokantaan.

4.12.4 listArticles.java

parametrit: —

palauttaa: Artikkelin otsikko / id / description / DTD-id

toiminta: Hakee tiedon artikkelit taulun sisällöstä ja lähettää sen appletille.

4.12.5 listDTD.java

parametrit: —

palauttaa: DTD:n otsikko / id / kuvaus

toiminta: Hakee tiedon DTD taulun sisällöstä ja lähettää sen appletille

5 Käyttöliittymä(HT ja ET)

Ohjelmiston käyttöliittymä toteutetaan WWW-selaimeen graafisena Java-Applettina. Käyttöliittymässä käytetään valmiita Java Swing-komponentteja mahdollisimman laajasti. Varsinainen käyttöliittymä totetutetaan käyttäen Bordland JBuilder 4 -kehitystyökalua ja sen ominaisuuksia. Käyttöliittymän kieli on englanti.

5.1 Käytettävät tekniikat(HT)

Ohjelmiston kehityksessä käytetään Bordland JBuider 4 -kehitystyökalua ja sen tarjoamia ominaisuuksia. JBuilder 4 käyttää standardia Java-kirjastoa sekä Swing-komponentteja joita tarvitaan käyttöliittymän kehittämisessä.

Ohjelmistossa käytetään Javan versiota 1.2.2 ja vastaavia Swing-komponentteja.

(Tähän pitäisi kirjoittaa lista käytettävistä UI-komponenteista.!)

5.2 Käyttöliittymän elementit(HT)

Tässä luvussa kuvataan ohjelmiston asiakkaan puolen käyttöliittymä käyttöliittymä ja sen toiminnat.

5.2.1 Valikot

Ohjelmiston valikoista käyttäjä pääsee ohjaaman monia ohjelman toimintoja. Ohjelmassa on neljä valikkoa: File, Edit, Options ja About.

File-valikosta löytyvät seuraavat toiminnot

- *New*-toiminnolla käyttäjä voi avata uuden artikkelin.
- *Open*-toiminnolla käyttäjä voi avata aikaisemmin tallennetun artikkelin.
- *Save*-toiminnolla käyttäjä tallettaa editorissa olevan artikkelin.
- *Save As New*-toiminnolla käyttäjä voi tallettaa artikkelin uudella nimellä tallentamatta sitä edellisen artikkelin päälle.
- *Exit*-toiminnolla käyttäjä pääsee pois ohjelmasta.

Edit-valikosta löytyvät seuraavat toiminnot

- *Undo*-toiminnolla käyttäjä voi perua edellisen tekstinmuokkaustoimintonsa.
- *Redo*-toiminnolla käyttäjä voi perua edellisen Undo-operaation.
- *Cut*-toiminnolla käyttäjä voi leikata valitun tekstialueen.
- *Copy*-toiminnolla käyttäjä voi kopioida valitun tekstialueen.
- *Paste*-toiminnolla valittu tekstialue voidaan joko siirtää tai kopioida.

Options-valikosta löytyvät seuraavat toiminnot

- *Article Options*-toiminnolla käyttäjä pääsee muokkaamaan ja tarkastelemaan artikkeliin liittyviä tietoja.

About-valikosta löytyvät seuraavat toiminnot

- *About JouTo*-toiminnolla käyttäjä pystyy tarkastelemaan ohjelman tietoja.

Kuvassa ?? nähdään ohjelman valikot Java-elementeillä toteutettuina.

5.2.2 Metatieto-osio

Metatiedot tarkoittavat varsinaisesta artikkelista suoraan riippumatonta tietoa. Artikkeliin liittyvää metatietoa voi syöttää tähän osioon. Syötettäviä asioita ovat kirjoittaja, kirjoitusajankohta sekä uutiset tyyppi.

Kuvassa ?? nähdään metatieto-osion ulkoasu Java Swing-komponenteilla toteutettuna.

5.2.3 Artikkelin tiedot-osio

Artikkelin tiedot tarkoittavat valittuun artikkelityyppiin liittyviä tiedoja, jotka artikkelin kirjoittajan on ehdottomasti täytettävä ennenkuin artikkeli voidana hyväksyä.

Kuvassa ?? nähdään Artikkelin tiedot-osion toteutus Java Swing-komponenteilla. Ulkoasu kuitenkin riippuu valitusta artikkelityypistä.

5.2.4 Varsinainen artikkeli

Varsinainen artikkeli-osio sisältää varsinaisen artikkelin ja siihen liittyvät muotoilutiedot.

Kuvassa ?? nähdään esimerkki osion toteutuksesta.

5.3 Käyttöliittymän dialogit (ET)

Ohjelmistossa dialogeja käytetään ottamaan vastaan käyttäjän syötteitä sekä näyttämään käyttäjälle ohjelman vaihtoehtoja.

Valikot

Käyttöliittymän dialogeja ovat New, Open, Save ja Save As New -toiminnot. Käyttöliittymässä näytetään myös työn eteneminen käyttäjälle. Tällöin käyttäjälle annetaan palautetta siitä, että järjestelmä suorittaa pyydettyä toimenpidettä.

5.3.1 New-dialogi

Käyttöliittymään tuodaan Servletiltä listaus DTD:istä, joista listan ensimmäinen on se tyypillisin DTD. Tyypillisin DTD on myös valittu, jolloin käyttäjän ei tarvitse muuta kuin jatkaa suoraan eteenpäin mikäli käyttäjä ei halua valita jotakin muuta DTD:tä. Valitulta DTD:ltä saadaan sen id, jonka perusteella haetaan Servletiltä DTD:n mukainen artikkelipohja. Käyttöliittymässä on OK -ja Cancel -painikkeet.

Kuvassa **??** nähdään Java Swing-komponenteilla toteutettu dialogi.

5.4 Open-dialogi

Käyttöliittymään tuodaan Serletiltä listaus artikkeleista laskevassa järjestyksessä kirjoituspäivämäärän mukaan, joista viimeisin artikkeli on valittuna. Liittymässä näytetään vain kymmenen ensimmäistä artikkelia. lisää artikkeleita saa vierittämällä palkista. Artikkelista liittymässä näytettävät tiedot ovat: pääotsikko, kuvaus, kirjoittaja ja kirjoituspäivämäärä. Pääotsikon kentän pituus on 25 merkkiä ja kuvauksen 50 merkkiä. Mikäli merkkijonot ovat pitempiä kuin mitä liittymässä on mahdollista näyttää, näytetään käyttäjälle merkkijonon jatkuminen kolmella pisteellä. Valitulta artikkelista saadaan sen id, jonka perusteella saadaan Servletiltä artikkelin DTD:n id. Käyttöliittymään haetaan ensin DTD:n mukainen artikkelipohja ja sen jälkeen täydennetään pohjaan artikkelin tiedot. Käyttöliittymässä on OK -ja Cancel -painikkeet.

Kuvassa **??** nähdään Java Swing-komponenteilla toteutettu dialogi.

5.4.1 Save-dialogi

Tallennetaan käyttäjän kirjoittama artikkeli sellaisenaan vanhan artikkelin päälle. Tallenuksen eteneminen näytetään käyttöliittymässä joko palkkirivistönä tai jollakin liikkuvalla kuvalla. Käyttäjälle annetaan palautetta siitä, että järjestelmä suorittaa pyydettyä toimenpidettä. Tätä palautetta käyttöliittymässä näytetään myös new ja open -toimintojen yhteydessä.

Kuvassa **??** nähdään Java Swing-komponenteilla toteutettu dialogi.

5.4.2 Save As New-dialogi

Käyttöliittymässä näytetään artikkelin nimi ja kuvaus. Artikkelin nemä käytetään pääotsikkoa. Mikäli käyttäjä on kirjoittanut artikkelille kuvauksen, tuodaan se liittymään esille, käyttäjä voi tarvittaessa muuttaa kuvausta. Artikkelin kuvaus ei ole pakollinen. Tallennus-toiminnon yhteydessä artikkelille muodostetaan sen id. Käyttöliittymässä on OK -ja Cancel -painikkeet.

Kuvassa ?? nähdään Java Swing-komponenteilla toteutettu dialogi

5.5 Käyttöliittymän virheilmoitukset (ET)

Järjestelmä voi antaa käyttäjälle kolmea eri tyyppiä olevia virheilmoituksia joko antaa ei toimenpiteitä vaativan ilmoituksen, kertoa tilarivillä tilanteen tai kertoa toimenpiteen epäonnistumisesta ja vaatia käyttäjältä hyväksymisen. Tässä kappaleessa esitellään virheilmoituksen yleinen rakenne, metodien kuvauksissa kerrotaan kyseisen metodin virheilmoitukset ja niiden tyyppi.

5.5.1 Ei toimenpiteitä-virheilmoitus

Jos pyydettyä toimenpidettä, esimerkiksi tekstin siirtäminen, kopioiminen tai liittäminen, ei voida suorittää, ilmoitetaan virheestä käyttäjälle esiin pomppaavassa ikkunassa. Tällöin pyydetty toimenpide jätetään kokonaan suorittaamatta, toisin sanoen liittymässä ei tapahdu mitään muutoksia. Esiin pompannut ikkuna pysyy näytössä kolme sekunttia. Mikäli käyttäjä haluaa sulkea virheilmoituksen aikaisemmin, käyttäjä voi sulkea sen oikeassa ylälaidassa olevasta ruksista.

Kuvassa ?? nähdään Java Swing-komponenteilla toteutettu dialogi.

5.5.2 Tilarivi

Käyttöliittymän alareunassa tilarivillä kerrotaan suorittamatta jätetystä toimenpiteestä. Tilarivi on vaihtoehtoinen tapa esiin pomppaavalle ikkunalle. Virheen vakavuus ratkaisee sen kumpaa tapaa käytetään, kun kerrotaan käyttäjälle suorittamatta jätetystä toimenpiteestä.

Kuvassa ?? nähdään Java Swing-komponenteilla toteutettu komponentti.

5.5.3 Toimenpiteitä vaativa virheilmoitus

Jos pyydetyn toimenpiteen seurauksena joko tietoja tuhoutuisi tai toimenpidettä ei voida suorittaa, käyttäjälle kerrotaan virheestä esiin pomppaavassa ikkunassa ja käyttäjältä vaaditaan hyväksyntä suorittamatta jätetystä toimenpiteestä. Tällainen toimenpide voisi esimerkiksi olla tallennuksen epäonnistuminen. Käyttöliittymässä ja järjestelmässä palataan pyydettyä toimenpidettä edeltävään tilaan. Käyttöliittymässä on vain OK -painike.

Kuvassa ?? nähdään Java Swing-komponenteilla toteutettu esimerkki.

5.6 Rajapinnat

Tässä luvussa kuvataan käyttöliittymän ohjelmisto-rajapinnat ulospäin sekä luokan tarjoamat rajapinnat muille luokille. Kappaleen tarkoituksena on antaa lukijalle kuva käyttöliittymän tarvitsemista resursseista ja sen tarjoamista operaatioista.

5.6.1 Sisäiset rajapinnat

Koska ohjelma jakaantuu kahteen erilliseen selkäesti erotettavaan kokonaisuuteen palvelimeen ja asiakassovellukseen, joutuvat nämä kaksi kokonaisuutta keskustelemaan verkon ylitse. Keskusteluun käytetään HTTP-POST -standardia, jonka avulla tietoa saadaan siirrettyä verkossa

5.6.2 Rajapinnat ulospäin

JouTo tarjoaa muille ohjelmille rajapintanaan oman tietokantansa, jonne kaikki artikkelit on tallennettu.