JouTo

Toimittajan työväline

Toteutusdokumentti

Versio 1.0

24.4.2001
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	6.4.2001	Ensimmäinen hahmotelma (AK)
0.2	12.4.2001	Liitetty yleiskuvaus ja muutokset suunnitteluun (AK)
0.3	17.4.2001	Liitetty muutamat luokkakuvaukset (AK)
0.9	20.4.2001	Liitetty loputkin luokkakuvaukset (AK)
1.0	24.4.2001	Viimeistelty versio

Sisältö

1 Johdanto (AK)

Tässä kappaleessa käydään läpi tämän dokumentin tarkoitus ja dokumentissa käytettävien määritelmien ja lyhenteiden kuvausta. Johdanto-luvusta löytyy myös viitteet sekä dokumentin rakenteen selvitys.

1.1 Dokumentin tarkoitus

Tässä toteutusdokumentissa esitellaan JouTo-ohjelmistotuotantoprojektin tuotteena syntyvän uutistoimittajan työvälineen toteutus. JouTo on lyhenne englannin kielen sanoista "A Journalist's Tool".

1.2 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, rakenteisten dokumenttien rakenteen määrittely.

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 dokumentointikieli
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/`

1.3 Viitteet

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

HTTP-protokolla Hypertext Transfer Protocol – HTTP/1.1,
`ftp://ftp.isi.edu/in-notes/rfc2616.txt`

2 Järjestelmän yleiskuvaus

Luvussa kuvataan järjestelmän yleinen toiminta ja sen käyttäjät sekä esitellään millaisissa ympäristössä järjestelmä on toimii.

2.1 Yleinen toiminta

Ohjelmiston arkkitehtuuri on asiakas-palvelin pohjainen. Asiakaskoneissa pyöritettävä sovellus vaatii Pentium-teholuokan koneen, jossa on toimiva tietoliikenneyhteys. Palvelimen puolen tehovaatimukset on esitetty määrittelydokumentissa kappaleessa 7.2. *Palvelimen suorituskyky*. Toimintaynpäristöksi oletetaan toimituksen sisäinen verkko.

Asiakkaan puolen ohjelmisto on WWW-selaimessa toimiva Java-sovelma (applet), ja palvelimen puolen ohjelmisto (servlet) huolehtii tietojen tallennuksesta sekä latauksesta tietokannasta.

2.2 Käyttäjät

Jouto-projektin editorin käyttäjiä ovat eri uutisalueiden toimittajat. Uutistoimittajien tehtäviin nykyisin kuuluu varsinaisen toimittajan työn lisäksi myös artikkelien kirjoitus ja muokkaus sähköiseen muotoon. Editori-osa muokkaa käyttöliittymän valitun artikkelin mukaisesti ja huolehtii että käyttäjä voi luoda vain valitun artikkelin mukaisia rakenteita. Tätä työtä varten voidaan käyttää JouTo-projektin tuotetta, jonka käyttäminen ei edellytetä suurta tietoteknistä osaamista toimittajilta. Editori-osa toimii rajapintana käyttäjän ja varsinaisen ohjelmiston toiminnan välillä.

Ohjelmalla tuotettujen artikkeleiden käyttäjäpiiriin kuuluu myös ylläpitohenkilöitä. He voivat hakea toimittajien kirjoittamia artikkeleita tietokannasta ja taittaa ne lehteen sopivaksi sekä suorittaa muita jatkotoimenpiteitä. Ylläpitöhenkilöt voivat myös vastata artikkelien DTD:eistä. Näitä kaikkia toimintoja ei toteuta tuotteemme editorilla vaan niitä voidaan käyttää esimerkiksi tietokannasta käsin.

JouTo-järjestelmän ylläpito olisi lähinnä tietokannan ylläpitoa. Järjestelmän ylläpitäjän käyttötapauksia tukevia ominaisuuksia ei kuitenkaan toteuteta.

Niitä olisivat ainakin:

- käyttöprofiilien lisäys, muokkaus ja poisto
- vanhentuneiden artikkelien tuhoaminen
- dokumenttityyppimäärittelyjen lisäys, muokkaus ja poisto

2.3 Toteutusympäristö

Demoversio ohjelmistosta toimi Helsingin yliopiston tietokonejärjestelmässä sekä Linux- että Windows-ympäristöissä, jossa ohjelmiston kehittäminen on toteutettu. Demoversio käyttää tietokantaa, joka on Oracle 8-pohjainen ja sijaitsee Helsingin Yliopiston atk-keskuksen *kontti*-tietokantapalvelimella.

Ohjelmistoa käytetään WWW-selaimen kautta. Ohjelmiston asiakkaan puolen käyttöliittymä, editori, toteutettiin WWW-selaimessa toimivana graafisena Java applettina. Appletissa käytetään Javan version 1.2 ominaisuuksia ja valmiita komponentteja, kuten Swing-kirjastoa. Lisäksi käytetään JouTo-projektin ulkopuolella toteutettuja Java-kirjastoja, Xerces XML-parseria, Mark Wutkan tekemää DTD-parseria ja aelfred-parseria. Näiden lisäksi ohjelmistossa luodaan omia tietorakenteita sekä XML-puun käsittelyä että varsinaisen artikkelin muokkausta varten.

Tietokanta -operaatioista ja tietokannan käsittelystä huolehtivat JouTo-projektissa luodut Java Servlet-komponentit. Nämä komponentit käyttävät JDBC-rajapintaa tietokannan käsittelyssä. JDBC-tietokantarajapinta kuuluu osana Java 1.2 -standardiin.

2.4 Toimintaympäristö

JouTo-projektin editori toimii siinä ympäsitössä, jossa tuotteen kehittäminen on toteutettu.

Jos demoversiosta kehitetään varsinainen tuote, tällöin toimintaympäristöksi on oletettu toimituksen sisäinen verkko. Uutistoimittajat kirjoittavat artikkeleitaan tietokoneen äärellä usein toimittajan työpaikalla, toimituksessa. Toimittajilla on käytössään tietokoneita, jotka ovat yhteydessä verkon sisällä olevaan palvelimeen. Artikkelit kirjoitetaan ja muokataan tietokoneella ja tallennetetaan sekä ladataan palvelimella olevaan tietokantaan. Koska kyseessä on tomituksen sisäinen verkko, voimme olettaa yhteyksien olevan suojattuja.

3 Suunnitteluun toteutuksen aikana tehdyt muutokset ja rajoitukset

Järjestelmän suunnittelu on kuvattu suunnitteludokumentissa. Tässä luvussa esitellään toteutuksen aikana suunnitteluun tehdyt muutokset pääpiirteissään.

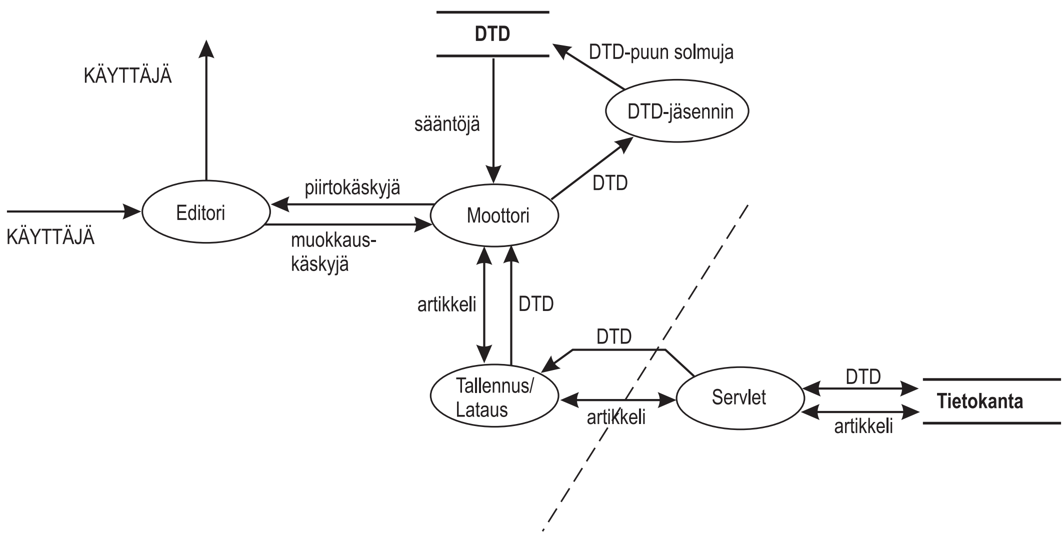
3.1 Toimintaympäristö

Järjestelmä suunniteltiin alunperin toimimaan selaimessa toimivana applettina. Toteutuksen aikana huomattiin, että Tietojenkäsittelytieteen laitoksen ATK-ympäristön Java Runtime Environment oli väärässä versiossa, eikä tarvittavaa versiota saatu asentumaan. Näin ollen ohjelmaa ei voitu ajaa laitoksen koneiden selaimissa. Ohjelma toteutettiin toimimaan siten myös omana sovelluksenaan.

Tietokanta saatiin toimimaan suunniteludokumentissa selvitetyllä tavalla Tietojenkäsittelytieteen tietokantapalvelmilla (db.cs.helsinki.fi).

3.2 Yleisarkkitehtuuri

Yleisarkkitehtuurissa ei ole tehty muutoksia. Arkkitehtuuri kuvataan kuvassa 1. Yleinen arkkitehtuurikuvaus on esitelty ja selitetty suunnitteludokumentissa.



Kuva 1: Ohjelmiston arkkitehtuurikuvaus

3.3 Luokkarakenne

Järjestelmän luokkarakenteen muutokset on kuvattu kappaleessa 4 *Järjestelmän luokkarakenne*.

3.4 DTD:hen tehdyt muutokset

Suunnitteluvaiheessa päätettiin toteuttaa JouTo-projektia varten oma DTD, joka on rajoitettu versio NITF v.2.5 DTD:stä. Sovelluksessa käytettävä DTD luotiin JouTo-järjestelmän toimintoja vastaaviksi.

Muutoksena suunniteltuun DTD:hen tuli seuraava korjaus:
Kaikki editorilla muutettavissa olevat otsikkotiedot ovat pakollisia.

Toteutettu DTD on kuvattu liitteessä (Liite A).

Järjestelmän ideana oli luoda mahdollisuus käyttää myös muita DTD:eitä. Toteuksen aikana luotiin siten toteutusta DTD:stä vielä rajoitetumpi DTD kolumnien (esim. pääkirjoituksia) varten. Myös tämä DTD löytyy liitteistä (Liite B).

3.5 TreeNode-puu

Puusta tehtiin kaksisuuntaisesti linkitetty. Toisin sanoen paitsi Block-oliot myös PCData- ja Element-oliot tuntevat isäelementtinsä. CommentBlock-luokkaa ei toteutettu.

3.6 XML-jäsennin

Apachen monipuolinen mutta raskas *Xerces*-jäsennin vaihdettiin Microstarin *Ælfred*-jäsentimeen. Ælfred toimii SAX 1 -rajapinnalla, mikä johti vähäisiin muutoksiin tapahtumankäsittelymetodeissa. *characters*-metodi jouduttiin nimeämään uudelleen metodiksi *charData* ja jouduttiin lisäämään uusi metodi *attributes*. Katso SAX 1:n määrittelydokumentteja.

4 Järjestelmän luokat

Tässä luvussa esitellään toteutetun järjestelmän luokat. Tämän luvun tarkoituksena on käydä läpi vain muutokset jotka tehtiin toteutusvaiheessa. Suunnitellut luokat ja luokkarakenne esitellään suunnitteludokumentissa.

4.1 IPTC

IPTC-luokka on suunnitteluvaiheen jälkeen lisätty luokka. IPTC-luokan tehtävä on tallettaa IPTC-numeroiden ja niiden kuvausten suhde, sekä eri IPTC-tasojen väliset suhteet. Jokainen IPTC-numero talletetaan luokan instanssiin, jossa on tieto IPTC-numerosta, sen kuvauksesta sekä mahdollisen ylemmän tason isäelementin numerosta.

Luokka saa luonnin yhteydessä siihen liittyvät tiedot, jotka eivät muutu sen eliniän aikana. Näitä tietoja on tarkoitus lukea suoraan luokan muuttujista UIManager-luokasta.

4.1.1 Rajapinnat

Luokan rajapintana toimivat sen julkiset muuttujat, joita käsitellään suoraan. Mitään erillisiä rajapintametodeja luokassa ei ole.

4.1.2 Attribuutit

public String ownIPTC Kyseisen ilmentymän IPTC-numero.

public String parentIPTC Mahdollisen yläluokan IPTC-numero.

public String iptcString Tämän ilmentymän IPTC-numeron kuvaus luonnollisella kielellä.

4.2 Editor

Editor-luokka on suunnitteluvaiheen jälkeen lisätty luokka. Editor-luokan tarkoituksena on toimia ohjelman käynnistäjänä luomalla UIManagerista uusi ilmentymä ja asettamalla sen ominaisuuden halutuiksi. Editor-luokan avulla pystyy ohjelmistoa ajamaan niin Applettina kuin sovelluksena vain vähäisin muutoksin ohjelmakoodiin.

Editor-luokalle ei anneta parametrejä. Luokka tekee varsinaiset toimensa omassa konstruktorissaan tai init()-metodissaan riippuen ajetaanko ohjelmaa Applettina vai sovelluksena. Editor-luokka on JApplet-luokan aliluokka.

Editor()

parametrit: -

palauttaa: -

toiminta: Luokan konstruktori, joka ajetaan kun ohjelmaa suoritetaan sovelluksena. Konstruktori käynnistää UIManagerin kutsumalla createUIManager()-metodia, ja asettaa ohjelmaan tiedot että sitä ajetaan nyt sovelluksena.

void createUIManager()

parametrit: -

palauttaa: -

toiminta: Tämä metodi luo uuden ilmentymän UIManager-luokasta, asettaa sen koon ja sijainnin sekä ikkunankuuntelijan. Käytönnässä siis tämä metodi siis käynnistää koko ohjelman.

void destroy()

parametrit: -

palauttaa: -

toiminta: Tämä metodi huolehtii ohjelman sammuttamisestaja UIManagerin luoman ikkunan sulkemisesta. Mikäli ohjelmaa ajetaan sovelluksena, niin metodi kutsuu lopuksi System-luokan exit()-metodia, joka aiheuttaa ohjelman lopullisen keskeytyksen.

void start()

parametrit: -

palauttaa: -

toiminta: Tämä metodia kutsutaan WWW-selaimen toimesta automaattisesti, kun Appletti käynnistetään ja sitä ryhdytään jälleen suorittamaan. Editor-luokan start()-metodi ei sisällä mitään toimintoja.

void stop()

parametrit: -

palauttaa: -

toiminta: Tämä metodia kutsutaan WWW-selaimen toimesta automaattisesti, kun Appletin toiminta pysäytetään. Editor-luokan stop()-metodi ei sisällä mitään toimintoja.

void getParameterInfo()

parametrit: -

palauttaa: -

toiminta: Tämä metodi lukee Appletille annetut parametrit. Tämä metodi on varattu tulevalle ohjelmiston kehitykselle, eikä sitä käytetä tässä vaiheessa mihinkään.

String getAppletInfo()

parametrit: -

palauttaa: Merkkijonon, joka ilmoittaa appletin tiedot

toiminta: Tämä metodi tarjoaa ulkoisille objekteille mahdollisuuden tiedustella Appletin ominaisuuksia. Tämä metodi on varattu tulevaa ohjelman- kehitystä varten, eikä sitä käytetä tässä vaiheessa mihinkään.

void init()

parametrit: -

palauttaa: -

toiminta: Tätä metodia kutsutaan automaattisesti, kun ohjelmaa ajetaan Applettina. Metodi kutsuu createUIManager()-metodia, ja asettaa tiedot, että tälä hetkellä ohjelmaa ajetaan Applettina.

void main(String[] args)

parametrit: Merkkijonotaulukko annetuista parametreistä

palauttaa: -

toiminta: Tätä metodia kutsutaan automaattisesti, kun ohjelma käynnistetään sovelluksenä. Se kutsuu tekee Editor-luokasta (itsestään) uuden ilmentymän, ja näinollen käynnistää oman konstruktorinsa.

4.2.1 Attribuutit

Luokalla ei ole julkisia attribuutteja. Luokalla on yksi sisäinen attribuutti

private isStandAlone Tieto siitä ajetaanko ohjelmaa tällä hetkellä Applettina ja sovelluksena.

4.3 UIManager

4.3.1 Muutetut metodit kuvauksineen

void save()

parametrit: Boolean saveAs, joka kertoo metodille, onko kyseessä save-, vai saveAs toiminto.

palauttaa: boolean arvon operaation onnistumisesta.

toiminta: Sama kuin suunniteltu.

void load() -> boolean loadArticle()

Metodin nimi muutettiin load:sta loadArticleen.

parametrit: -

palauttaa: boolean arvon operaation onnistumisesta.

toiminta: Sama kuin suunniteltu.

Node new() -> Node showCreateNewArticleDialog()

Metodin nimi muutettiin new:stä showCreateNewArticleDialog:iin.

parametrit: Sama kuin suunniteltu.

palauttaa: Sama kuin suunniteltu.

toiminta: Sama kuin suunniteltu.

4.3.2 Muutetut attribuutit kuvauksineen

TextDialog textDialog -> JDialog dialog
Attribuutin nimi on muutettu.

TreeManager treeManager -> Treemanager treemanager
Attribuutin nimi on muutettu.

ClipBoardManager clipBoardManager -> ClipboardManager clipboardmanager
Attribuutin nimi on muutettu.

int documentID -> int articleID
Attribuutin nimi on muutettu.

4.3.3 Lisätyt metodit ja attribuutit kuvauksineen

Image getImage(String name)

parametrit: Tagin nimi, joka ilmaisee haettavan kuvan.

palauttaa: Kahvan haluttuun kuvaan.

toiminta: Metodi etsii halutun kuvan Pair-listasta, ja palauttaa siihen viitteen.

boolean copy(Areaselection selection)

parametrit: Valittu alue bodysta.

palauttaa: Boolean arvon operaation onnistumisesta.

toiminta: Metodi käyttää ClipboardManagerin copy-toimintoa.

boolean cut(Areaselection selection)

parametrit: Valittu alue bodysta.

palauttaa: Boolean arvon operaation onnistumisesta.

toiminta: Metodi käyttää ClipboardManagerin cut-toimintoa.

boolean showLoadArticleDialog()

parametrit: -

palauttaa: Boolean arvon metodin onnistumisesta.

toiminta: Metodi näyttää käyttäjälle dialogin, jonka avulla käyttäjä voi ladata haluamansa artikkelin. Metodi käyttää loadArticle-metodia dokumentin lataamiseen.

boolean showSaveArticleDialog()

parametrit: -

palauttaa: Boolean arvon metodin onnistumisesta.

toiminta: Metodi näyttää käyttäjälle dialogin, jonka avulla käyttäjä voi tallentaa artikkelistaa uuden kopion tietokantaan. Metodi käyttää saveArticle-metodia dokumentin tallentamiseen.

void showTextDialog(String text, String header, int dialogType)

parametrit: Näytettävä teksti, ikkunan otsikko ja dialogi tyyppi. Dialogityyppi vaihtelee välillä 1-3.

palauttaa: -

toiminta: Metodi näyttää halutun tyyppisen dialogin. Dialogit näyttävät halutun tekstin, ja niissä on yksi nappi: OK.

boolean setButtonsActive(Areaselection selection)

parametrit: Valittu alue bodysta.

palauttaa: boolean arvon operaation onnistumisesta.

toiminta: Metodi tutkii selection-muuttujan, clipboardmanager.canCut:in, clipboardmanager.canCopyn ja clipboardmanager.canPasten avulla mitkä napit ovat aktiivisia. Se myös aktivoi tarvittavat napit käyttöliittymästä.

boolean loadDTD(int dtdIdentification)

parametrit: Ladattavan DTD:n tunniste.

palauttaa: boolean arvon operaation onnistumisesta.

toiminta: Metodi lataa DTD:n tietokannasta.

4.3.4 Poistetut metodit ja attribuutit

UIManagerista on poistettu seuraavat suunnitellut metodit:

1. paste (String Text)
2. setUndoer (Object Reference)
3. undo()
4. new()

UIManagerista on poistettu seuraavat suunnitellut attribuutit:

1. SaveAsDialog saveAsDialog
2. OpenDocumentDialog openDocumentDialog
3. Node bodyRoot
4. Node headersRoot
5. Object undoer

4.4 BodyPanel

BodyPanel on toteutettu suunnitteludokumentissa määritellyllä tavalla.

4.5 Body

Body-komponentti on toteutettu lähes samanlaisena kuin suunnitteludokumentissa on suunniteltu. Lähinnä muutokset koskeva piirto-osaa. Piirtäminen on toteutettu kolmeen metodiin joista yhdessä käydään puuta läpi ja kahdessa muussa piirretään. Toinen suunnitteluun tehty muutos on myös se että attributeDialog ja popupmenu on toteutettu Body-luokan sisälle, eikä niille ole omia luokkia.

private Element curParent -> currentNode

void paint(Graphics vg)

parametrit: Saa parametrinä käsiteltävän solmun ja piirtoalustan.

palauttaa: —

toiminta:

void drawNode(TreeNode node, Graphics g)

parametrit: Saa parametrinä käsiteltävän solmun ja piirtoalustan.

palauttaa: —

toiminta: Käy rekursiivisesti läpi koko XML-puun. Ensin kutsuu drawImage:a joka piirtää alku tagin. Sen jälkeen, jos solmulla on lapsia käydään kaikki lapset läpi kutsumalla tätä samaa funktiota. Lopuksi kun lapsia ei enää löydy, jos ollaan PCDATA elementissä, tulostetaan PCDATA:n sisältö kutsumalla splitRow-funktiota. Tämän jälkeen rekursiosta takaisin palattaessa kutsutaan vielä drawImage:a, joka piirtää loppu tagin.

void drawImage(TreeNode node, boolean start, Graphics g)

parametrit: Saa parametrinä käsiteltävän solmun, tiedon onko piirrettävä kuva alku- vai loppukuva ja piirtoalustan.

palauttaa: —

toiminta: Piirtää sekä tagi-kuvat että li-elementtien numeroinnit ja pallot. Tarkistaa myös osuuko kursori kyseisen kuvan päälle, jos osuu muuttaa currentNode-muuttujan sisältöä.

void splitRow(PCData node, Graphics g)

parametrit: Saa parametrinä käsiteltävän solmun ja piirtoalustan.

palauttaa: —

toiminta: Rivittää parametrinä saadun PCData-elementin sisältämän tekstin. Tarkistaa myös kursorinpaikan ja korjaa sitä tarvittaessa. Muuttaa myös kursorinpaikan mukaan currentNode-muuttujan sisältöä.

4.6 Communications

Communications on toteutettu suunnitteludokumentissa määritellyllä tavalla.

4.7 Servletit

Kaikki servletit on toteutettu suunnitteludokumentissa määritellyllä tavalla.

4.8 XMLAssistant

Ainoana erona suunniteltuun lisättiin metodi *removeExtraSpaces*.

public static removeExtraSpaces(String s)

parametrit: String s, muokattava merkkijono.

palauttaa: -

toiminta: Korvaa merkkijonon *s* rivinvaihto- ja tabulointimerkit välilyönneillä ja tämän jälkeen muuttaa peräkkäisten välilyöntien jonot yksittäisiksi välilyönneiksi.

4.9 InvalidityException

Kun TreeManagerin jäsentäessä XML-dokumenttia ilmenee ongelmia, luodaan InvalidityException.

public InvalidityException()

parametrit: -

palauttaa: -

toiminta: Luo InvalidityException-olion. Tämän jälkeen toString() palauttaa merkkijonon "The document could not be loaded due to its invalidity."

4.10 TreeSelection

Toteutettiin kuten oli suunniteltu.

4.11 TreeNode

Toteutettiin kuten oli suunniteltu.

4.12 PCData

Toteutettiin kuten oli suunniteltu.

4.13 Element

Toteutettiin kuten oli suunniteltu.

4.14 Block

Toteutettiin kuten oli suunniteltu.

4.15 ANode

Toteutettiin kuten oli suunniteltu.

4.16 AreaSelection

Toteutettiin kuten oli suunniteltu.

4.17 ClipboardManager

Toteutettiin kuten oli suunniteltu.

4.18 ElementPopup

Erillistä luokkaa ei tarvittu. Korvaava toteutus on Body-luokassa.

4.19 AttributesDialog

Erillistä luokkaa ei tarvittu. Korvaava toteutus on Body-luokassa.

4.20 TreeManager

Xerces-jäsentimen sijaan käytettiin Ælfred-nimistä SAX 1 -jäsennintä. Tämän vuoksi TreeManagerista tuli *com.microstar.xml.HandlerBase*-luokan aliluokka ja tapahtumankäsittelymetodi *characters* korvattiin metodilla *charData*. SAX 1 -jäsentimet antavat elementin attribuutit yksitellen ennen *element*-metodin kutsua, *attribute*-metodissa. Attribuutteja kasataan *attVector*-vektoriin. TreeManageriin tuli myös useita pieniä apumetodeja, jotka on kommentoitu lähdekoodissa. Edellisten lisäksi ainoa huomattava muutos suunniteltuun oli metodin *divide* käyttöönotto. *divide* jakaa elementin kahteen osaan. *canDivide* kertoo, voidaanko *divide* suorittaa.

Muutokset suunniteltuun:

private Vector attVector
Vektori, johon attribuutteja kasataan ennen liittämistä seuraavaan Element-olioon.

public void attribute(String name, String value, boolean isSpecified)

toiminta: Suoritetaan, kun jäsennin kohtaa attribuutin. *name* on attribuutin nimi ja *value* sen arvo. Muuttujaa *isSpecified* ei käytetä.

public void charData(char[] ch, int start, int length)

toiminta: Suoritetaan, kun jäsennin kohtaa merkkijonon. *length* merkkiä pitkä merkkijono poimitaan taulukosta *ch* indeksistä *start*.

public boolean canDivide(Element parent, int childI, int charI)

toiminta: Voidaanko elementti *parent* jakaa kahtia kohdasta (*childI*, *charI*). *childI* on indeksi ensimmäiseen *parent*in lapseen, joka ainakin osittain kuuluu jaon jälkimmäiseen puoliskoon. *charI* on indeksi ensimmäiseen merkkiin *childI*:ssä, joka kuuluu jaon jälkimmäiseen puoliskoon.

public boolean divide(Element parent, int childI, int charI)

toiminta: Yrittää jakaa elementin *parent* kahtia kohdasta (*childI*, *charI*). *childI* on indeksi ensimmäiseen *parent*in lapseen, joka ainakin osittain kuuluu jaon jälkimmäiseen puoliskoon. *charI* on indeksi ensimmäiseen merkkiin *childI*:ssä, joka kuuluu jaon jälkimmäiseen puoliskoon.

5 Jatkokehitysajatuksia

Seuraavassa luettelossa esitetään muutamia jatkokehitys- sekä parannusehdotuksia JouTo-projektille:

- SystemClipboardin käyttöönotolla saavutettaisiin toimivuutta ja käytettäisiin yleisä käytössä olevia rajapintoja.
- Nykyisen tuotteen vailinaisten ominaisuuksien toteuttaminen ja toteuttamatta jätettyjen ominaisuuksien toteuttaminen, tällöin saataisiin käyttöön tavanomaiset editoriominaisuudet.
- Satatus Barin toteutus editoriin.
- Kaikki IPTC-luokat käyttöön.
- LoadDialog -dialogin listan toteutus niin, että kaikki listaelementit näkyvät.
- Päivämäärän oikeellisuuden tarkistus, nitfissä pvm:t ovat vain pcdataa.
- Html- ja ps-konvertointi.
- Käyttäjäprofiilien käyttöotto.
- Ylläpidon työkalut: käyttäjäprofiilien -, DTD:iden -ja tietokannan ylläpito.
- Käyttäjien käyttöoikeudet ja oikeustasot.
- Käyttäjällä olisi mahdollisuus poistaa omia artikkeleitaan.
- Ohjeet helpissä.

A Liite: Toteutettu DTD

```
<!ENTITY % enriched-text '
        #PCDATA
        | location
        | object.title
        | org
        | person
        | a
        | br
        | em
        | q
        '>
<!ENTITY % block.content "p | hl2 | media | ol | ul | pre ">

<!ELEMENT nitf (head, body)>
<!ATTLIST nitf
        version      CDATA    #FIXED    "-//IPTC-NAA//DTD NITF-XML 2.1//EN"
        change.date  CDATA    #FIXED    "4 July 2000"
        change.time  CDATA    #FIXED    "1900"
        >
<!ELEMENT head (title, tobject, docdata)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT tobject (tobject.subject)>
<!ATTLIST tobject
        tobject.type            CDATA   "news"
        >
        <!ELEMENT tobject.subject  EMPTY>
        <!ATTLIST tobject.subject
                tobject.subject.ipr             CDATA           "IPTC"
                tobject.subject.refnum          NMTOKEN         #REQUIRED
                >
<!ELEMENT docdata (
        evloc,
        doc-id,
        date.issue,
        doc.copyright?,
        key-list
        )>
        <!ELEMENT evloc EMPTY>
        <!ATTLIST evloc
                city            CDATA   #IMPLIED
                >
        <!ELEMENT doc-id EMPTY>
        <!ATTLIST doc-id
                id-string       CDATA   #IMPLIED
```

```
                >
        <!ELEMENT date.issue EMPTY>
        <!ATTLIST date.issue
                norm            CDATA   #IMPLIED
                >
        <!ELEMENT doc.copyright EMPTY>
        <!ATTLIST doc.copyright
                year            NMTOKEN         #IMPLIED
                holder          CDATA           #IMPLIED
                >
        <!ELEMENT key-list (keyword)*>
                <!ELEMENT keyword EMPTY>
                <!ATTLIST keyword
                        key             CDATA   #REQUIRED
                        >
<!ELEMENT body (body.head, body.content)>
<!ELEMENT body.head (
        hedline,
        byline,
        distributor?,
        abstract
        )>
<!ELEMENT hedline (hl1, hl2*)>
        <!ELEMENT hl1 (%enriched-text;)*>
        <!ELEMENT hl2 (%enriched-text;)*>
<!ELEMENT byline (#PCDATA)>
<!ELEMENT distributor (#PCDATA)>
<!ELEMENT abstract (p)>
<!ELEMENT body.content (%block.content;)*>
<!ELEMENT p (%enriched-text;)*>
<!ELEMENT media (media-reference, media-caption?)>
<!ATTLIST media
        media-type      CDATA #FIXED "image"
        >
        <!ELEMENT media-reference (#PCDATA)>
        <!ATTLIST media-reference
                data-location           CDATA           #REQUIRED
 mime-type  CDATA   #REQUIRED
                >
        <!ELEMENT media-caption (%enriched-text;)*>
<!ELEMENT ol (li)+>
<!ELEMENT ul (li)+>
        <!ELEMENT li (%enriched-text;)*>
<!ELEMENT pre (#PCDATA)>
<!ELEMENT location (#PCDATA)>
<!ELEMENT object.title (#PCDATA)>
```

```
<!ATTLIST object.title
idsrc  CDATA  #REQUIRED
value  CDATA  #REQUIRED
>
<!ELEMENT org (#PCDATA)>
<!ATTLIST org
idsrc  CDATA  #REQUIRED
value  CDATA  #REQUIRED
>
<!ELEMENT person (#PCDATA)>
<!ATTLIST person
idsrc  CDATA  #REQUIRED
value  CDATA  #REQUIRED
>
<!ELEMENT a (%enriched-text;)*>
<!ATTLIST a
        href            CDATA           #IMPLIED
        name            CDATA           #IMPLIED
        title           CDATA           #IMPLIED
        >
<!ELEMENT br EMPTY>
<!ELEMENT em (%enriched-text;)*>
<!ELEMENT q (%enriched-text;)*>
<!ATTLIST q
        quote-source    CDATA   #IMPLIED
        >
```

B Liite: Kolumni DTD

```
<!ENTITY % enriched-text '
        #PCDATA
        | location
        | object.title
        | org
        | person
        | a
        | br
        | em
        | q
        '>
<!ENTITY % block.content "p | hl2 ">

<!ELEMENT nitf (head, body)>
<!ATTLIST nitf
        version      CDATA    #FIXED    "-//IPTC-NAA//DTD NITF-XML 2.1//EN"
        change.date  CDATA    #FIXED    "4 July 2000"
        change.time  CDATA    #FIXED    "1900"
        >
<!ELEMENT head (title, tobject, docdata)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT tobject (tobject.subject)>
<!ATTLIST tobject
        tobject.type            CDATA   "news"
        >
        <!ELEMENT tobject.subject  EMPTY>
        <!ATTLIST tobject.subject
                tobject.subject.ipr             CDATA           "IPTC"
                tobject.subject.refnum          NMTOKEN         #REQUIRED
                >
<!ELEMENT docdata (
        evloc,
        doc-id,
        date.issue,
        doc.copyright?,
        key-list
        )>
        <!ELEMENT evloc EMPTY>
        <!ATTLIST evloc
                city            CDATA   #IMPLIED
                >
        <!ELEMENT doc-id EMPTY>
        <!ATTLIST doc-id
```

```
                id-string       CDATA   #IMPLIED
                >
        <!ELEMENT date.issue EMPTY>
        <!ATTLIST date.issue
                norm            CDATA   #IMPLIED
                >
        <!ELEMENT doc.copyright EMPTY>
        <!ATTLIST doc.copyright
                year            NMTOKEN         #IMPLIED
                holder          CDATA           #IMPLIED
                >
        <!ELEMENT key-list (keyword)*>
                <!ELEMENT keyword EMPTY>
                <!ATTLIST keyword
                        key             CDATA   #REQUIRED
                        >
<!ELEMENT body (body.head, body.content)>
<!ELEMENT body.head (
        hedline,
        byline,
        distributor?,
        abstract
        )>
<!ELEMENT hedline (hl1, hl2*)>
        <!ELEMENT hl1 (%enriched-text;)*>
        <!ELEMENT hl2 (%enriched-text;)*>
<!ELEMENT byline (#PCDATA)>
<!ELEMENT distributor (#PCDATA)>
<!ELEMENT abstract (p)>
<!ELEMENT body.content (%block.content;)*>
<!ELEMENT p (%enriched-text;)*>
<!ELEMENT location (#PCDATA)>
<!ELEMENT object.title (#PCDATA)>
<!ATTLIST object.title
idsrc  CDATA  #REQUIRED
value  CDATA  #REQUIRED
>
<!ELEMENT org (#PCDATA)>
<!ATTLIST org
idsrc  CDATA  #REQUIRED
value  CDATA  #REQUIRED
>
<!ELEMENT person (#PCDATA)>
<!ATTLIST person
idsrc  CDATA  #REQUIRED
value  CDATA  #REQUIRED
```

```
>
<!ELEMENT a (%enriched-text;)*>
<!ATTLIST a
        href            CDATA           #IMPLIED
        name            CDATA           #IMPLIED
        title           CDATA           #IMPLIED
        >
<!ELEMENT br EMPTY>
<!ELEMENT em (%enriched-text;)*>
<!ELEMENT q (%enriched-text;)*>
<!ATTLIST q
        quote-source    CDATA   #IMPLIED
        >
```

C JavaDoc

Seuraavassa erillisessä liitteessä esitellään ohjelman lähdekoodin *Javadoc*.