Convergence of messaging

Toteutusdokumentti

The Converge Group:
Mikko Hiipakka
Anssi Johansson
Joni Karppinen
Olli Pettay
Timo Ranta-Ojala
Tea Silander

Helsinki 20. joulukuuta 2002

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Sisältö

Kuvat

0.0.1		Dokumentti luotu; kopio suunnittelu-dokumentista	Anssi Joha
0.0.2	15.12.2002	Muutoksia Järjestelmän ydin-lukuun	Olli Petta
0.1	16.12.2002	Järjestemän ydin-lukua muokattu vastaamaan toteutusta	Mikko Hiip
0.1.1	18.12.2002	Muokkausta	Ryhmä
1.0	20.12.2002	Lopputarkistus ja jäädytys	Mikko Hiip

1 Johdanto

1.1 Dokumentin tarkoitus

Tämä toteutusdokumentti esittää teknisen mallin ohjelmistotuotantoprojektin ”Convergence of Messaging” tuotteena syntyneestä suunnitteludokumentin perusteella toteutetusta järjestelmän prototyypistä.

1.2 Määritelmät, termit ja lyhenteet

HTML	Hypertext Markup Language, WWW-sivujen kuvauksessa käytetty kieli http://www.w3.org/MarkUp/
HTTP	Hypertext Transfer Protocol http://www.w3.org/Protocols/
Javadoc	Java-kielen kommentointityökalu http://java.sun.com/j2se/javadoc/
XML	Extensible Markup Language http://www.w3.org/TR/REC-xml
DOM	Document Object Model, määritys dokumenttien luonnille, muuttamiselle ja läpikäynnille, käytetään usein XML-dokumenttien yhteydessä http://www.w3.org/DOM/
XPath	XML Path Language, kyselykieli XML:lle http://www.w3.org/TR/xpath
XUpdate	Kieli XML-dokumenttien muuttamiseksi, http://www.xmldb.org/xupdate/xupdate-wd.html
IMAP	Internet Message Access Protocol, protokolla sähköpostien noutoa varten http://www.imap.org/
SSL	Secure Sockets Layer, verkkoliikenteen salausmenetelmä http://wp.netscape.com/eng/ssl3/

Taulukko 1: Dokumentissa käytetyt termit ja lyhenteet

1.3 Dokumentin rakenne

Ensimmäinen luku esittelee dokumentin sisällön ja rakenteen sekä dokumentissa käytetyt termit ja lyhenteet.
Toisessa luvussa on mainittu ne lisäykset määrittelydokumenttiin, jotka ovat tulleet esiin määrittelydokumentin kirjoittamisen jälkeen.
Kolmas luku sisältää järjestelmän tietosuunnitelman.
Neljännessä luvussa kuvataan järjestelmän arkkitehtuuri yleisellä tasolla.
Viidennessä luvussa on kuvattuna järjestelmän ytimen luokkarakenne, luokkien tehtävät ja metodit sekä ytimen rajapinnat.
Kuudennessa luvussa kuvataan tiedonhakumoduulin yhteydet sähköpostipalvelimille sekä rajapinnat järjestelmän muihin osiin.
Seitsemännessä luvussa kuvataan asiakasmoduulin tarjoamat rajapinnat asiakasohjelmalle ja järjestelmän ytimelle.
Kahdeksannessa luvussa kuvataan asiakasohjelman toiminta.
AttributeInterface -rajapinnan toteutusesimerkit on sisällytetty dokumenttiin liitteenä A.
Liitteessä B on UserPropertyInterface -rajapinnan toteutusesimerkkejä.
Liitteessä C on ActionInterface -rajapinnan toteutusesimerkkejä.
Avoimiksi jääneitä kysymyksiä ja muita kommentteja löytyy liitteestä D.
Asennus- ja ylläpito-ohjeet on liitteessä E ja käyttöliittymän käyttöohje liitteessä F.

2 Tietokannan kuvaus

Järjestelmä käyttää tietojen säilyttämiseen Apache-organisaation Xindice XML-tietokantaa. Järjestelmän dynaamisesta luonteesta johtuen ei tietokannan dokumenttien rakenteelle voida asettaa tiukkoja muotovaatimuksia. Esimerkiksi viestien tallennuksessa käytettävän XML:n muoto riippuu muun muassa viestin tyypistä (onko se esimerkiksi sähköposti) ja siitä, millaisia attribuuttimäärityksiä järjestelmään on tehty. Attribuuttiluokkia saatetaan tulevaisuudessa lisätä järjestelmään dynaa-

misesti, joten voi olla mahdollista, että viestien muoto muuttuu järjestelmän toiminnan aikana.

Vaikka varsinaisille XML-dokumenteille ei tässä määritellä muotoa, tulee dokumenttien kantaantallettaminen määritellä. Xindicen avulla dokumenttien joukkoja ryhmitellään kokoelmiksi (collection). Tietokannan juuri on /db ja sen alaisuudessa seuraavat kokoelmat:

- /db/users
  Sisältää kunkin käyttäjän nimisen alikokoelman käyttäjän tiedoille.
- /db/users/*user*
  Käyttäjän omat tiedot alikokoelmissa.
- /db/users/*user*/pref
  Käyttäjän asetukset dokumentissa preference.xml.
- /db/users/*user*/profiles
  Käyttäjän profiilitiedot.
- /db/users/*user*/contextmodels
  Käyttäjän kontekstimallit.
- /db/users/*user*/messages
  Käyttäjän viestit erillisissä dokumenteissa.
- /db/users/*user*/views
  Käyttäjän näkymien hallintaan liittyvää informaatiota.
- /db/users/*user*/datasources
  Käyttäjän tiedonlähteisiin liittyvät asetukset.
- /db/groups/
  Sisältää ryhmien nimien mukaiset kokoelmat.
- /db/groups/*group*
  Ryhmän tiedot alikokoelmissa.

- /db/groups/*group*/pref
  Ryhmän asetukset.

- /db/groups/*group*/profiles
  Ryhmän profiilit.

- /db/groups/*group*/contextmodels
  Ryhmän kontekstimallit.

- /db/groups/*group*/datasources
  Ryhmän tiedonlähteisiin liittämät asetukset.

  Lokitietoja kerätään erillisiin tiedostoihin.

3 Arkkitehtuuri

Järjestelmä voidaan jakaa toiminnallisesti kolmeen eri moduuliin: järjestelmän ytimeen, tiedonhakumoduuliin ja asiakasmoduuliin.

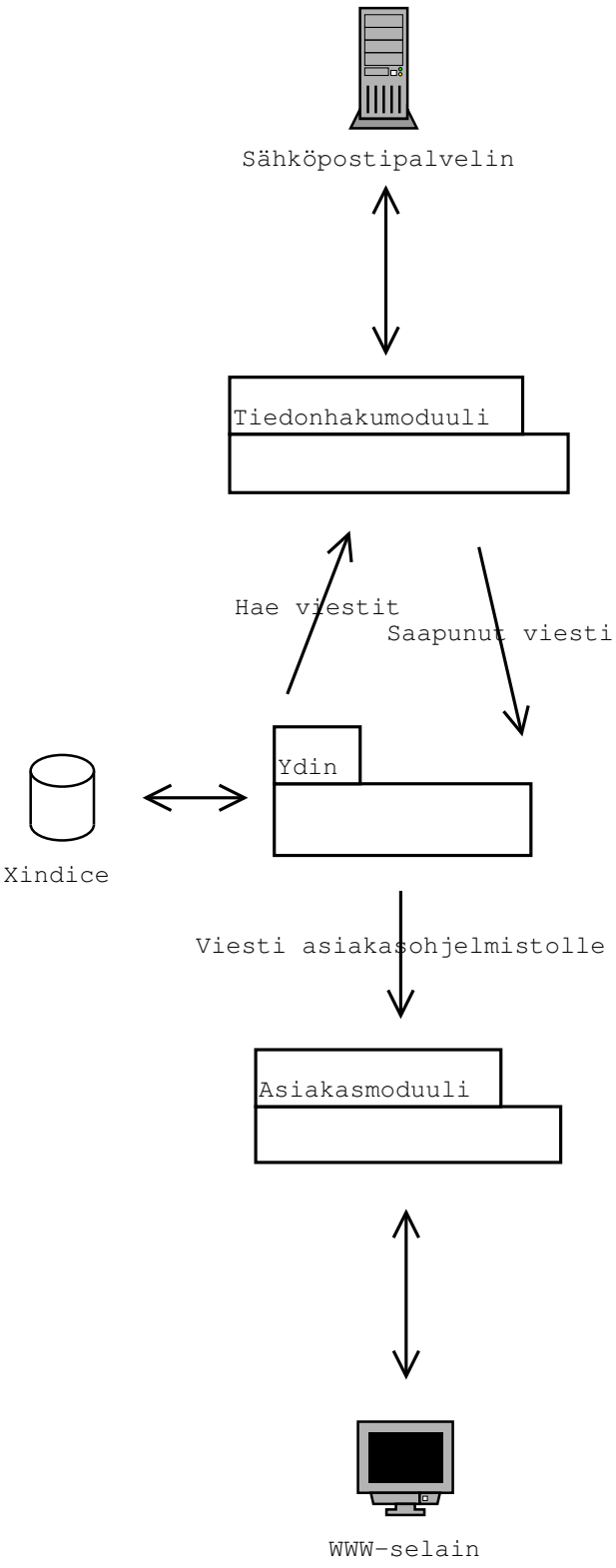
Ytimen rooli järjestelmässä on hallinnoida ryhmiä ja käyttäjiä sekä näihin liitettyjä tietoja kuten kontekstimallit ja profiilit. Toimintoja, joista ydin huolehtii, ovat tietokantaoperaatiot, viestien käsittely kontekstimallivertailujen ja profiilisääntöjen perusteella sekä ajastettujen toimintojen hallinta. Ytimen tehtäviin kuuluu myös suuri osa asiakasmoduulin tarjoaman käyttöliittymän palveluista, muun muassa näkymien hallinta ja tiedonlähdeasetusten ylläpitäminen.

Tiedonhakumoduuli huolehtii yhteyksistä järjestelmän ulkopuolella oleville palvelimille. Näitä palvelimia ovat esimerkiksi sähköpostipalvelimet (sekä saapuvat että lähtevät viestit), uutisryhmäpalvelimet, WWW-palvelimet ym. Tiedonhakumoduuli myös muokkaa saapuneet viestit järjestelmän käyttämään sisäiseen muotoon.

Asiakasmoduuli yhdistää järjestelmän ja järjestelmän käyttäjän. Kaikki tieto järjestelmältä asiakkaalle ja asiakkaalta järjestelmään välitetään asiakasmoduulin kaut-

ta.

Asiakasmoduuli huolehtii järjestelmän ytimestä tulevien viestien muuttamisesta järjestelmän sisäisestä muodosta kunkin asiakasohjelman ymmärtämään muotoon, esimerkiksi muuttaen viestin HTML-muotoon toimitettaessa viestiä WWW-sovellukselle. Asiakasmoduulissa muutetaan myös asiakkaalta tuleva tieto järjestelmän sisäiseen muotoon, esimerkiksi käyttäjän luodessa uusia kontekstimalleja tai profiileja.



Kuva 1: Arkkitehtuuri

4 Järjestelmän ydin

Kokonaisuudessaan toteutetun järjestelmän toiminta-ajatuksena on, että käyttäjä voisi muokata viestintäjärjestelmän toimimaan haluamallaan tavalla. Järjestelmä pystyisi siis itsenäisesti sille annettujen tietojen perusteella päättelemään esimerkiksi mitä viestejä käyttäjä tiettynä ajanhetkenä voisi pitää kiireellisenä. Toteutetun prototyypin ydinluokat huolehtivat nimenomaan tämän toiminnallisuuden toteutuksesta järjestelmässä. Tiedonhakumoduuli sekä asiakasmoduuli huolehtivat puolestaan tietovirrasta järjestelmän ulkopuolella kuten postin hakemisesta sähköpostipalvelimelta tai viestin lähettämisestä käyttäjän päätelaitteeseen.

Jokaiselle järjestelmään rekisteröityneelle käyttäjälle luodaan oma (User, luku 4.1.20, sivulla 40) käyttäjä-olio, jonka tarkoituksena on tarjota mahdollisimman keskitetysti tiedot järjestelmän käyttäjästä (palvelimien osoitteet, profiilit jne.). Käyttäjäolioon liitetään järjestelmän sisäisessä käytössä ominaisuuksia (UserPropertyInterface, luku 4.3.1, sivulla 64), joilla voidaan kuvata järjestelmän käsitystä käyttäjän reaalimaailman ominaisuuksista. Esimerkiksi ominaisuus nimeltä ”paikka” pitäisi sisällään käyttäjän olinpaikan mukaan muuttuvan tiedon, joka kuvaisi hänen sijaintinsa järjestelmän ymmärtämässä muodossa esim.”Helsinki, Kallio, Toinen Linja 10”.

Kontekstimalli (ContextModel, luku 4.1.1, sivulla 12) on edellä mainittua käyttäjän ominaisuutta (UserPropertyInterface)laajempi tapa kuvata käyttäjän reaalimaailman ominaisuuksia. Kontakstimallien tavoitteena on kuvata käyttäjällä tietyllä ajanhetkellä mahdollisesti vallitsevaa tilannetta. Tämä kuvaus toteutetaan järjestelmässä määrittelemällä kontekstimalli, johon liitetään attribuutteja (ContextModelAttributeInterface, luku 4.3.4, sivulla 67), joihin määritetyt arvovälit kuvaavat suhdetta sillä hetkellä päteviin faktoihin (MessageAttributeInterface, luku 4.3.3, sivulla 66). Esimerkiksi kontekstimalliin voitaisiin määritellä attribuutti, jonka looginen nimi olisi ”paikka” ja arvoväli ”Helsinki”. Tällöin kontekstimallivertailussa voidaan sanoa toteuttiko olemassa oleva fakta vaaditun arvovälin. Tällä tavalla vertailemalla attribuutteja olemassa oleviin faktoihin voidaan päätellä

mallin toteutumisprosentti eli kuinka hyvin faktat toteuttivat malliin määritellyt arvovälit.

ContextModelHandler-luokka vastaanottaa viestejä (Message, luku 5.1.6, sivulla 82) ytimen käsiteltäväksi ja luo heti uuden säikeen Process-luokasta, jonka tehtävänä on huolehtia, että kaikki saapuneen viestin vastaanottajakäyttäjän kontekstimallit suorittavat edellä mainitun vertailun. Vertailun tuloksen (ContextModelValueAttribute, luku A.5, sivulla 106) kontekstimallit liittävät viesti-olion attribuutiksi.

Seuraavaksi saapuneen viestin käsittely siirtyy ProfileHandler-luokalle, joka tutkii käyttäjä-olioon liitettyjä profiileja (Profile, luku 4.1.24, sivulla 47). Profiileihin käyttäjä on voinut määritellä sääntöjä (Rule, luku 4.1.25, sivulla 50), joilla järjestelmän toimintaa tietyissä tilanteissa voidaan ennalta määritellä. Jokainen käyttäjän profiili on joko aktiivinen tai passiivinen, mistä vain aktiiviset tullaan tutkimaan. Jokaiseen profiiliin on aina liitetty jokin määritelty kontekstimalli, ja jotta profiilin säännöt suoritettaisiin, on jonkin profiiliin liitetyn mallin kynnysarvo ylityttävä. Jos tämä kynnysarvo ylittyy, tarkoittaa se sitä, että saapuneeseen viestiin liitetyt faktat ovat toteuttaneet profiiliin liitetyn kontekstimallin riittävän hyvin, jotta järjestelmä voi jatkossa toimia säännöissä määritetyllä tavalla. Esimerkiksi jokin sääntö voisi määrittää, että jos ”kellonaika” niminen fakta on suurempi kuin ”16.00” ja ”päivä” niminen fakta on perjantai lähetä kaikki saapuneet viestit gsm-puhelimeen tekstiviestinä.

Kun kynnysarvo on ylitetty, siirtyy suoritus RuleHandler-luokalle, joka huolehtii sääntöjen tutkimisesta sekä niiden määrittelemien toimintojen alustamisesta, muokkaamalla tapahtumaoliota (ActionEvent, luku 4.1.7, sivulla 20). Jokaisella järjestelmän käyttäjä-oliolla on vähintään oletusprofiili, joka määrittää säännön viestin liittämiselle *Inbox*-näkymään. Kun sääntöjen käsittely on saatu päätökseen, viedään toimintatapahtuma EventMonitor-luokan (luku 4.1.8, sivulla 22) säilytettäväksi, josta EventManager (luku 4.1.9, sivulla 23) sen hakee käsittelyyn. Tämän käsittelyn aikana pyydetään ActionLoader-luokalta (luku 4.1.11, sivulla 25) toimintotapahtuma-

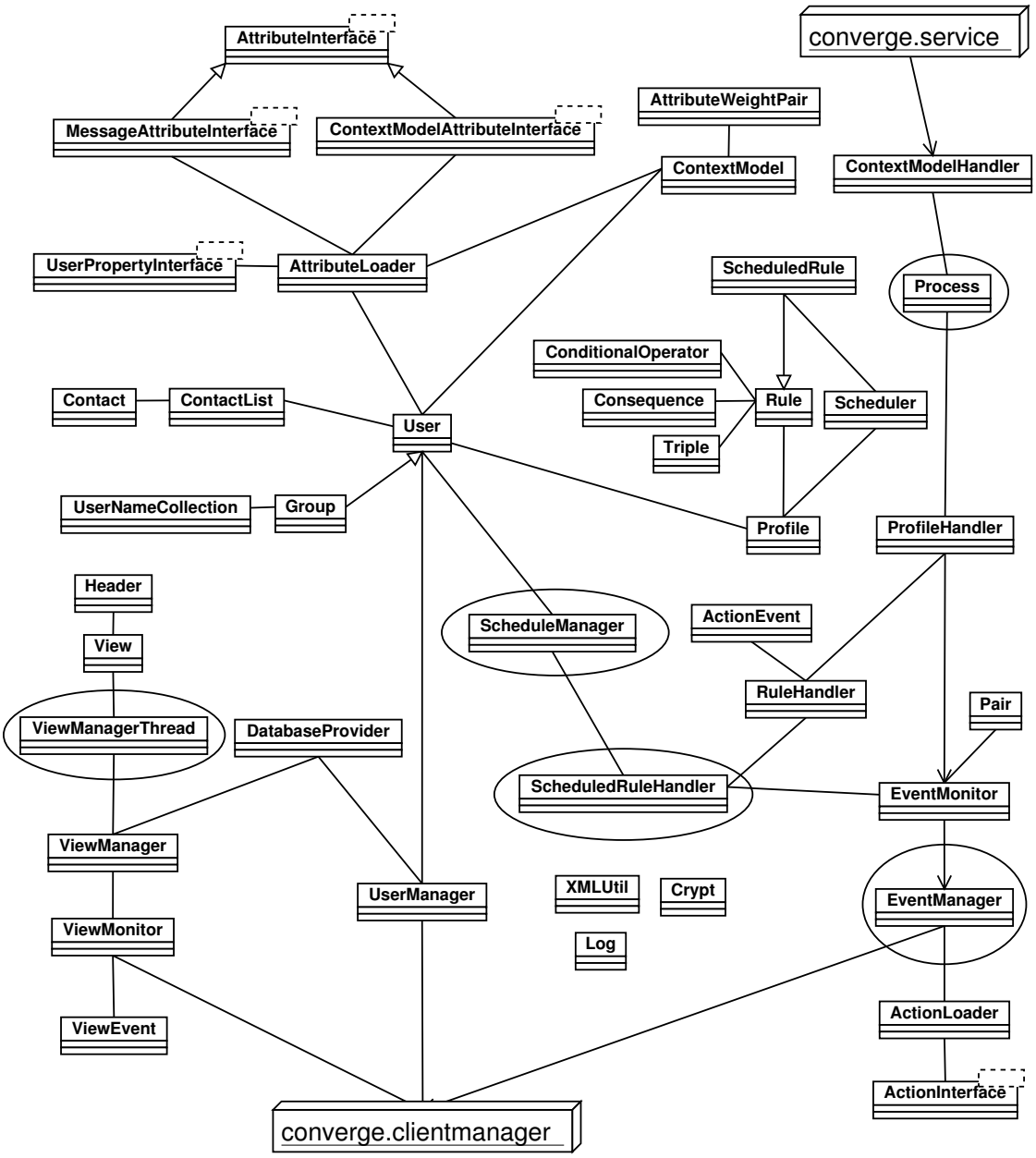
olioon talletetuille toiminnoille toteutuksen tarjoavan luokan ilmentymä (ActionInterface, luku 4.2.1, sivulla 63). Talletetut toiminnot ovat toimintojen loogisia nimiä, joiden mukaan ActionLoader luo toteuttavan luokan.

Jotta käyttäjälle yleensäkin saapuu viestejä passiiviselta tiedontarjoajalta (esim. sähköpostipalvelin) on hänen määriteltävä itselleen ajastuksia (Scheduler, luku 4.1.31, sivulla 58), jotka laukeavat niihin määritetyn ajanjakson välein. Näihin ajastuksiin liitetään ajastettuja sääntöjä (ScheduledRule, luku 4.1.28, sivulla 55), jotka ajastuksen laukeamisen yhteydessä lähetetään RuleHandler-luokan käsittelyyn, minkä toimesta luodaan uusia toimintatapahtumia samalla tavalla kuin viestin käsittelyn yhteydessä. Toiminnot, joita ajastetuilla säännöillä voidaan toteuttaa ovat luonteeltaan erilaisia. Esimerkiksi viestin saapuessa ”Set-To-View” asettaa viestin tiettyyn näkymään, mutta ajastuksena se ei ole kovin mielekästä.

Käyttäjä-olioihin on liitettynä myös kontaktilista (ContactList, luku 4.1.27, sivulla 54), johon voi kerätä henkilöiden yhteystietoja (Contact, luku 4.1.26, sivulla 52).

Kaikille tietokantaa käsitteleville luokille on ytimeen luotu DatabaseProvider-luokka (DatabaseProvider, luku 4.1.18, sivulla 36), joka tarjoaa peruskantakäsittelyt.

Järjestelmään saapuvat viestit tallennetaan kantaan ja viite niihin lisätään käyttäjän haluamiin näkymiin. Näkymien ja viestien käsittelyä hallinnoivat ViewMonitor (luku 4.1.14, sivulla 28) ja ViewManager (luku 4.1.16, sivulla 32) yhdessä ViewManagerThread-säikeen kanssa (luku 4.1.17, sivulla 34).



Kuva 2: Ytimen luokkakaavio, jossa soikiot osoittavat säikeitä käynnistävät luokat

4.1 Luokat : *converge.kernel*

4.1.1 ContextModel

ContextModel-luokka tarjoaa attribuuttien hallintaa sekä vertailun viestiolion attribuutteihin. Kontekstimalli-ilmentymälle voidaan määritellä joukko attribuutteja, joihin liitetään arvo sekä vertailussa käytettävä laskusääntö. Näiden avulla luokka laskee itselleen arvon siitä kuinka hyvin viestiolion attribuuttien arvot (eli kyseisellä ajanhetkellä olevat faktat) toteuttavat laskusäännön määrittelemällä tavalla malliin liitettyjen attribuuttien arvovälejä. Saatu mallin toteutumisarvo liitetään viestiolioon ContextModelValueAttribute-tyyppisenä (kts. luku A.5, sivulla 106) attribuuttina.

Tulevaisuudessa tähän luokkaa liitetty toiminta voisi perustua kerättyyn historiadataan, joka korvaisi tarpeen, että käyttäjä itse määrittelee attribuuttien arvovälitietoja. Tämän datan avulla voitaisiin päätellä samankaltaisesti kuin nyt se mikä on käyttäjän konteksti ja täyttääkö viesti eteenpäin lähettämisen vaatimukset.

Määrittely :

public class ContextModel implements XMLContentInterface

Riippuvuudet :

- converge.service.Message
- converge.kernel.XMLContentInterface
- converge.attribute.*
- converge.attribute.attributeImplementations.ContextModelValueAttribute
- java.util.ArrayList
- org.w3c.dom.*

Konstruktorit :

- **ContextModel**(String name)
  Alustaa uuden name-nimisen kontekstimalli ilmentymän valmiiksi, attribuuttien lisäämiselle. Nimi ei saa olla sama kuin mikään järjestelmään määritellyn attribuutin looginen nimi on.
- **ContextModel**(org.w3c.dom.Element contextModelElement)
  Konstruktori, jolla kontekstimalli-ilmentymä muodostetaan antamalla ilmentymän tietosisältö XML-elementtinä.

Kentät :

- private ArrayList **forcedAttributes**
  Kontekstimalliin liitettyjen pakotettujen attribuutti-painoarvo parien (AttributeWeightPair) säilytyslista.
- private ArrayList **attributes**
  Kontekstimalliin liitettyjen attribuutti-painoarvo parien (AttributeWeightPair) säilytyslista.
- private String **name**
  Kontekstimallille annettu nimi

Metodit :

- public void **addAttribute**(String name, String expressionType, Object value, boolean forced, int weight)
  Lisää kontekstimalliin uuden *name*-nimisen attribuutin, jonka arvoväli määritellään *value*:ksi. Laskusääntö siitä miten tätä arvoväliä tullaan vertaamaan viestiolion samannimisen attribuutin arvoon annetaan *expressionType*:lla (laskenta suoritetaan "Viestiolion.Attribuutin.Arvo" expressionType value). *Forced*-parametrilla asetetaan, onko määritelty attribuutti kontekstimallin arvon laskennalle ehdoton vaatimus (pakotettu attribuutti) eli kontekstimalli ei saa arvoa, jos kyseisen attribuutin vertailu arvoväliin ei ole tosi. *weight*-parametri kertoo lisätyn attribuutin painoarvon kontekstimallissa.

- protected void **startProcessing**(Message msg)
  Laukaisee kontekstimalliolion suorituksen, jonka aikana malli laskee itselleen arvon. Tuo arvo liitetään MessageAttributeInterface:n toteuttavana attribuuttina *msg*-olioon, minkä nimeksi määräytyy ilmentymälle määritetty nimi.
- public void **createXML**(Node parent)
  Katso 4.2.2 sivulla 64.
- public boolean **loadXML**(Node myContent)
  Katso 4.2.2 sivulla 64
- public String **getModelName**()
  Palauttaa kontekstimalli-ilmentymälle annetun nimen.
- public ContextModelAttributeInterface[] **getAllAttributes**()
  Palauttaa kaikki ilmentymään liitetyt attribuutit taulukkona.
- public boolean **isForcedAttribute**(ContextModelAttributeInterface attr)
  Palauttaa totuusarvon siitä löytyykö parametrina annettu *attr* attribuutti-ilmentymä kontekstimallin pakotetuista attribuuteista. Jos ilmentymää ei löydy palauttetaan false.
- public int **getAttributeWeight**(ContextModelAttributeInterface attr)
  Palauttaa kokonaislukuarvon, joka kuvaa parametrina annetun *attr* attribuutti-ilmentymään liitetyn painoarvon. Jos ilmentymää ei löydy palautetaan 0
- private boolean **loadAttributes**(NodeList itemList, ArrayList list)
  Luo *itemList*:sta löytyvien elementtien name-attribuutteihin määriteltyjen nimien perusteella uusia ContextModelAttributeInterfacen toteuttavia attribuutteja sekä lisää ne parametrina annettuun *list*:aan.
- private void **createXMLHelp**(ArrayList list, Element parent)
  Lisää *list*:lta löytyvät attribuutti-ilmentymien xml rakenteen *parent*-elementin alle.

4.1.2 AttributeWeightPair

AttributeWeightPair-luokka on tarkoitettu attribuuttien ja niiden painoarvojen toisiinsa liitettynä säilyttämiseen. Luokka on määritelty ContextModel-luokan sisäluokaksi.

Määrittely :

```
public class ContextModel {
    class AttributeWeightPair
}
```

Konstruktorit :

- private **AttributeWeightPair**(ContextModelAttributeInterface ob, long value)
  Alustetaan luokan tietosisältö attribuutti-ilmentymällä *ob* ja siihen liitetyllä painoarvolla *value*. Konstruktori on määrilty *private*, joten luokka on käytettävissä vain ContextModel-luokasta.

Kentät :

- private ContextModelAttributeInterface **data**
  Luokan attribuutti tietosisällön säilytys kenttä.
- private long **weight**
  Luokan painoarvo tietosisällön säilytys kenttä.

Metodit :

- public long **getWeight**()
  Palauttaa ilmentymään määritellyn painoarvotietosisällön.
- public ContextModelAttributeInterface **getAttribute**()
  Palauttaa ilmentymään määritellyn attribuuttitietosisällön.

4.1.3 ContextModelHandler

ContextModelHandler-luokan toiminta alkaa kun tiedonhakumoduuli lähettää järjestelmään saapuneen viestin luokan käsiteltäväksi. Toteutus tukee viestien jatkuvaa saapumista siten, että viestien vastaanotto on säikeistettyä toimintaa. Kun uusi säie on käynnistetty viestin käsittelyyn, suoritetaan kontekstimallimuuttujien arvovälien vertailu viestiolion tarjoamiin konteksti- ja viestiattribuutteihin. Tämä vertailu suoritetaan kaikille vastaanottajana (useamman vastaanottajan viestit käsitellään jokaiselle erikseen) käsiteltävään käyttäjään liitetyille kontekstimalleille (tämän tuloksena kontekstimallit liittävät itselleen saamat arvot viestiolioon nimetyiksi attribuuteiksi). Lopuksi suorittava säie antaa käsittelemänsä viestin edelleen käsiteltäväksi ProfileHandler-luokalle, jonka toiminta suoritetaan siis samassa säikeessä kuin ContextModelHandler:kin toiminta.

Määrittely :

public class ContextModelHandler

Riippuvuudet :

- converge.service.Message

Metodit :

- public static synchronized void **incomingMessage**(Message msg)
  Vastaanottaa uuden viestiolion ja luo uuden (Process) säikeen sen käsittelemiseksi.

4.1.4 Process

Säie-luokka, jonka run-metodissa ContextModelHandler:in kuvauksen toiminnot toteutetaan. Run-metodi pitää saapuneeseen viestiin liitettyä käyttäjää lukulukossa niin kauan kun viestin käsittely järjestelmässä on käynnissä. Mitään käyttäjään liitettyä tietoa ei saa käsittelyn aikana muuttaa, mikä on ehdoton vaatimus sille, että käsittely on luotettavasti mahdollista.

Määrittely :

class Process extends Thread

Konstruktorit :

- public **Process**(Message msg)
  Alustaa luokan asettamalla parametrin *msg* ilmentymän tietosisällöksi.

Kentät :

- private Message **msg**
  Luokan viestiolion tietosisällön säilytys kenttä.
- private User **user**
  Viestiolioon liitetyn käyttäjä olion säilytys kenttä, helpottamaan viittaamista käyttäjään.

Metodit :

- public void **run**()
  Säikeen suoritusmetodi, joka ottaa käyttäjäolion lukulukkoon, jotta kontekstimallien vertailut voidaan suorittaa. Vertailujen jälkeen viestiolio annetaan ProfileHandlerin käsiteltäväksi. Käyttäjälle otettu lukulukko vapautetaan vasta, kun kaikki tarvittava viestin käsittely on suoritettu.

4.1.5 ProfileHandler

ProfileHandlerissa aluksi käyttäjän oletusprofiilin säännöt, viesti ja tyhjä toimintatapahtuma annetaan RuleHandlerille, joka muokkaa toimintatapahtumaa oletusprofiilissa määriteltyjen sääntöjen mukaan. Tämän jälkeen tutkitaan viestiolion eri kontekstimalliarvoja jokaisen profiilin hallinnoimien kontekstimallien kynnysarvoihin. Jos arvon ylittyy, viesti ja käsittelyssä olevan profiilin säännöt, sekä oletusprofiilin muokkaama toimintatapahtuma annetaan RuleHandlerille, joka jatkaa toimintatapahtuman muokkausta annettujen sääntöjen mukaan. Kun kaikki viestiolioon liitetyn käyttäjän profiilit on käsitelty annetaan toimintotapahtuma viestin

kanssa EventMonitorille. Jos kuitenkin vastaanottaja oli ryhmä tallennetaan muokattu toimintotapahtuma ja käynnistetään jokaiselle ryhmänjäsenelle henkilökohtainen viestinkäsittely. Kun tämä suoritus tulee ProfileHandlerille asti otetaan tyhjän toimintotapahtuman sijasta ryhmätasolla muokatun tapahtumakopion käyttäjän profiilien käsittelyn pohjaksi.

Määrittely :

public class ProfileHandler

Riippuvuudet :

- java.util.HashMap
- org.w3c.dom.Document
- converge.service.Message
- converge.attribute.MessageAttributeInterface

Konstruktorit :

- **ProfileHandler**(Message msg)
  Luodaan uusi ProfileHandler-olio viestiä *msg* varten.

Kentät :

- private Message **msg**
  Luokan toiminnassa käytettävän viestiolio-viitteen säiltyskenttä.
- private static HashMap **baseEvents**
  Kokoelma, jossa ryhmälle tulleiden viestien käsittelyssä syntyneitä toimintotapahtumaolioita säilytetään, kunnes kaikkien ryhmänjäsenien henkilökohtaiset käsittelyt on suoritettu.
- private static HashMap **baseCounter**
  Laskuri, joka pitää kirjaa kuinka monta ryhmänjäsentä on jo suorittanut henkilökohtaisen käsittelyn kyseisellä säilytettävällä toimintotapahtumalla.

Metodit :

- void **startProcessing**()
  Metodi laukaisee luokan toiminnan alustetuilla tiedoilla.
- private boolean **limitCheck**(Profile profile)
  Suorittaa parametrina annetun *profile* profiiliin liitettyjen kontekstimallien kynnysarvojen tarkistuksen. Kontekstimallin nimellä pyydetään luokan tietosisältö viestioliolta mallin vertailussa saama arvo, jota verrataan profiiliin määriteltyyn kynnysarvoon. Jos mallin saama arvo ylittää profiiliin määritellyn arvon palautetaan true.
- private static synchronized ActionEvent **getBaseEvent**(String key)
  Palauttaa aina vähintään tyhjän toimintatapahtuman, mutta jos annetulla *key*-parametrilla löytyy talletettu tapahtuma palautetaan siitä kopio.
- private static synchronized void **addBaseEvent**(String key, ActionEvent ae, int count)
  Tallettaa *ae*-toimintatapahtuman annetulla *key*-avaimella sekä alustaa laskurin samalla avaimella.

4.1.6 RuleHandler

RuleHandler huolehtii sääntöjen käsittelystä. Se liittää Drools-sääntökäsittelijän järjestelmään.

Määrittely :

public class RuleHandler

Riippuvuudet :

- java.util.*
- org.w3c.dom.*
- org.drools.*

- org.drools.rule.*;
- org.drools.io.RuleSetLoader

Konstruktorit :

- public **RuleHandler**()
  Uuden sääntökäsittelijän luonti

Kentät :

- private RuleBase **ruleBase**
  Sääntökäsittelyn perusta, tähän olion liitetään käsiteltävät säännöt.
- private TransactionalWorkingMemory **workingMemory**
  Sääntökäsittelyn muistinhallintaa

Metodit :

- public void **processRules**(Object ob, Rule[] rules, ActionEvent ae)
  Tällä metodilla oliota *ob* käsitellään sääntökäsittelijässä sääntöjen *rules* mukaisesti ja tulokset kerätään ActionEvent-tyyppiseen *ae*-olioon.
- private void **setRules**(Rule[] rules)
  Lisätään käytettävät säännöt *rules*.
- private void **assertObject**(Object o)
  Lisätään säännöissä käsiteltävä olio.
- private void **commit**()
  Suoritetaan sääntökäsittely.

4.1.7 ActionEvent

ActionEvent-olioita käytetään sääntöjenkäsittelyn yhteydessä.
Jokaiseen tutkittavaan sääntöön on liitetty toiminta, joka tulisi suorittaa, jos sääntö toteutuu. Näitä suoritettavia toimintoja kerätään ActionEvent-olioon siten, että

aina säännön toteutuessa lisätään suoritettavan toiminnon looginen nimi (esim. Get-Messages) johon liitetään toiminnon toteuttavaa luokkaa ohjaavia arvoja.

Määrittely :

class ActionEvent

Riippuvuudet :

- java.util.*

Konstruktorit :

- public **ActionEvent**()
  Tyhjä oletuskonstruktoritoteutus.
- public **ActionEvent**(ActionEvent event)
  Kopiokonstruktori, joka luo aidon kopio-olion annetusta *event*-oliosta.

Kentät :

- private HashMap **hash** ActionEvent-olion talletettujen tapahtumien sekä niihin liitettyjen tapahtuman toimintaa ohjaavien arvojen säilytystaulu.

Metodit :

- public void **addEvent**(String key, String value)
  Lisätään uusi tapahtuma, jonka nimi määritellään *key*-parametrin avulla ja arvo *value* -parametrillä. Tämän metodikutsun toistaminen samannimisellä tapahtumalla lisää tapahtumaan erillisiä ohjausarvoja.
- public void **replaceEvent**(String key, String value)
  key-nimisen tapahtuman ohjausarvon muuttaminen siten, että kaikki vanhat ohjausarvot korvataan annetulla arvolla *value*.

- public void **replaceEvent**(String key, String oldvalue, String newvalue)
  Etsitään *key*-nimisestä tapahtumasta arvo *oldvalue* ja korvataan se arvolla *newvalue*. Mikäli vanhaa arvoa ei löydy, suoritetaan addEvent-metodin kutsu.
- public void **deleteEvent**(String key)
  Poistetaan kaikki *key*-nimisen tapahtumaan liitetyt ohjausarvot.
- public void **deleteEvent**(String key, String oldvalue)
  Poistetaan *key*-nimisen tapahtuman liitetty ohjausarvo *oldvalue*.
- public HashMap **getEvents**()
  Tämän metodin avulla EventManager pääsee käsiksi olion sisältämiin tapahtumiin. Palautettava hajautustaulu pitää sisällään lisätyt tapahtumat, jotka ovat taulun avaimia sekä tapahtumaan liitetyt toimintaa ohjaavat arvot, jotka on talletettu String-tyyppisinä Set-tyyppiseen tietorakenteeseen.

4.1.8 EventMonitor

EventMonitorissa hallitaan ytimen sisään tulleiden viestien käsittelyssä ja ajastetuissa säännöissä syntyneiden tapahtumaolioiden (ActionEvent) tallentamista siihen saakka, kunnes omassa säikeessä toimiva EventManager hoitaa tapahtumaolioiden ja viestin jatkokäsittelyn.

Määrittely :

class EventMonitor

Konstruktorit :

- private **EventMonitor**()
  Tyhjä konstruktoritoteutus

Kentät :

- private List **eventlist**
  Saapuneiden ActionEventolioiden säilytys lista.
- private static EventMonitor **em**
  Järjestelmän ainoa EventMonitor-ilmentymä

Metodit :

- public synchronized **incomingEvent**(Object o, ActionEvent evt)
  Uuden tapahtumaolion lisääminen käsittelyä odottavien tapahtumien listalle parametrien *o* ja *evt* avulla.
- public synchronized Pair **removeEvent**()
  Vanhimman käsittelemättömän Pairolion poisto EventMonitorin listalta.
- public static EventMonitor **getInstance**()
  Palauttaa arvonaan järjestelmän ainoan EventMonitor-olion.

4.1.9 EventManager

EventManager-luokka suorittaa tapahtumaolioiden määrittelemiä toimintoja. ActionInterface-tyyppisten olioiden avulla se suorittaa muun muassa viestien kantaanlisäyksen, näkymien hallinnan ja viestien lähettämiseen liittyvät toiminnot.

Määrittely :

public class EventManager extends Thread

Riippuvuus :

- java.util.*

Konstruktorit :

- private **EventManager**()
  Tyhjä konstruktori toteutus.

Kentät :

- private static EventManager **em**()
  Järjestelmän ainoa EventManager-ilmentymä

Metodit :

- public static EventManager **getInstance**()
  Palauttaa arvonaan järjestelmän ainoan EventManager-olion.
- public void **run**()
  EventManager-säikeen käynnistyessä alkaa run-metodin suoritus, jossa EventMonitorilta haetaan viesti-tapahtumaolio tai ajastettu sääntö-tapahtumaolio -pareja ja toimitaan tapahtumaolion määritysten mukaisesti.

4.1.10 Pair

Hyvin tiivis luokka jolla ei ole toimintaa, vaan tarkoituksena on tarjota täysin julkinen ActionEvent-Object parien säilytys olio.

Määrittely : class Pair

Konstruktorit :

- public **Pair**(Object o, ActionEvent evt)
  Luo uuden Pair-luokan ilmentymän, alustamalla tietosisällön annettujen parametrien mukaan.

Kentät :

- public Object **object**
  Luokan ActionEvent tietosisältöön sidottu data kenttä
- public ActionEvent **event**
  Luokan ActionEvent tietosisällön säilytys kenttä

4.1.11 ActionLoader

ActionLoader-luokalla tuodaan ActionInterface-rajapinnan toimintoja toteuttavia luokkia järjestelmän käyttöön. Kaikki toiminta on luokka tasoista, joten Action-Loaderista ei ole järjestelmässä yhtään ilmentymää.

Määrittely :

public class ActionLoader

Riippuvuudet :

- converge.kernel.Log
- java.util.*
- java.io.*
- org.w3c.dom.*

Kentät :

- private static String **packageName**
  Latauksessa tarvittava pakkausnimi, johon toteutusluokat kuuluvat.
- private static final **String actionDir**
  Javan ajoympäristöön annettu ominaisuuden arvoon viittaava nimi (converge.action.dir), jonka avulla toteutusluokkien latauksessa käytettävä polku (esim. käynnistettäessä "java -Dconverge.action.dir=/home/a/aa/b/actions/"
- private static String **defaultActionDir**
  Oletuksena käytettävä polku kovakoodattuna suhteellisesti käynnistys hakemistosta converge/kernel/actionImplementations/.

Metodit :

- public static ActionInterface **newActionInstance**(String name)
  Palveluna luodaan ActionInterface ilmentymä, joka tarjoaa toteutuksen *name*-nimiselle toiminnolle. Jos luonti epäonnistuu palautetaan null.

- public static synchronized void **loadClasses**()
  Ladataan Action-toteutusluokat määritetystä hakemistosta, ja tarkastetaan niiden tarjoamien toteutusten loogiset nimet, jotka jokainen toteutusluokka tarjoaa supportedActions-metodilla.
- public static String[] **getKnownActions**()
  Palautettaan ladattujen luokkien tarjoamien toimintojen loogiset nimet String-tyyppisessä taulukossa.
- private static void **setActions**(Class aClass)
  Privaatti apumetodi luokkien latauksessa, jossa annetun *aClass* luokan tukemat toiminnot liitetään paikalliseen tietorakenteeseen.

4.1.12 View

View-tyyppisillä olioilla kuvataan tietokannan näkymien sisältöä.

Määrittely :

public class View implements XMLContentInterface

Konstruktorit :

- protected **View**(String name, String[] ids, Header[] headers)
  Luodaan uusi View-olio nimen *name*, viestien tunnusten *ids* ja otsikkotietojen *headers* avulla.

Kentät :

- private String **name**
  Näkymän nimi
- private String[] **ids**
  Näkymään liitettyjen viestien tunnisteet
- private Header[] **headers**
  Näkymään liitettyjen viestien otsikkotiedot

Metodit :

- public String **getName**()
  Palautetaan näkymän nimi.
- public String[] **getMessageIDs**()
  Palautetaan kaikkien näkymään liitettyjen viestien tunnukset.
- public Header[] **getMessageHeaders**()
  Palautetaan olioon liitetyt viestien otsikkotiedot.

4.1.13 Header

Header-olioon on koottu viestin niitä otsikkotietoja, joita tarvitaan näkymän sisällön listauksessa käyttöliittymässä.

Määrittely :

public class

Konstruktorit :

- protected **Header**(String id, String sender, String subject, String date)
  Luodaan uusi Header-olio parametrien *id*, *sender*, *subject* ja *date* avulla.

Kentät :

- private String **subject**
  Viestin otsikko
- private String **date**
  Viestin lähetyspäivämäärä
- private String **id**
  Viestien tunniste
- private String **sender**
  Viestin lähettäjä

Metodit :

- public String **getId**()
  Palautetaan viestin tunniste.
- public String **getSender**()
  Palautetaan viestin lähettäjä.
- public String **getSubject**()
  Palautetaan viestin otsikko.
- public String **getDate**()
  Palautetaan viestin lähetyspäivämäärä.

4.1.14 ViewMonitor

Järjestelmässä ViewMonitor-luokka huolehtii viestien ja näkymien muutosten synkronoinnista tarjamalla

Määrittely :

public class ViewMonitor

Riippuvuudet :

- converge.service.Message
- org.w3c.dom.Document

Konstruktorit :

- protected **ViewMonitor**()

Kentät :

- private ViewEvent **event**
  ViewManagerille annettava ViewEvent
- private View **view**
  Pyynnön seurauksena saatu näkymä-olio

- private Document **dom**
  Viesti XML-muodossa
- private String[] **viewNames**
  Näkymien nimet.
- private boolean **notFound**
  Mikäli kysely ei tuottanut tulosta, *notFound* on true.

Metodit :

- protected synchronized void **waitEvent**()
  ViewManagerThread kutsuu tätä metodia, joka pysäyttää kutsujan toiminnan siihen asti, kunnes ViewMonitor-olioon on lisätty uusi ViewEvent-tapahtumaolio. (Sekä ViewMonitor-olion luonti että tapahtumaolion lisääminen tapahtuu kutsumalla luokan staattisia metodeja.)
- protected synchronized ViewEvent **getEvent**()
  Palautetaan olioon liitetty ViewEvent-tapahtumaolio.
- protected synchronized void **setView**(View view)
  Asetetaan näkymä ViewMonitoriin.
- protected synchronized void **setDom**(Document dom)
  Asetetaan dokumentti ViewMonitoriin.
- protected synchronized void **setViewNames**(String[] names)
  Asetetaan näkymien nimet ViewMonitoriin.
- private static ViewMonitor **newViewMonitor**()
  Luodaan uusi ViewMonitor-olio kutsumalla ViewManager-luokan getNewViewMonitor-metodia.
- private synchronized void **setMessageToView_**(Message msg, String view)
- public static void **setMessageToView**(Message msg, String view)
  Luodaan pyyntö viestin *msg* lisäämisestä näkymään *view*.
- private synchronized void **setMessageToView_**(User user, String id, String view)

- static void **setMessageToView**(User user, String id, String view)
  Luodaan pyyntö viestin *id* liittämisestä käyttäjän *user* näkymään *view*.
- private synchronized void **deleteMessageFromView**_(User user, String id, String view)
- public static void **deleteMessageFromView**(User user, String id, String view)
  Viestin *id* poistaminen käyttäjän *user* näkymästä *view*.
- private synchronized void **deleteMessage**_(User user, String id)
- public static void **deleteMessage**(User user, String id)
  Käyttäjän *user* viestin *id* poistaminen kaikista näkymistä ja kannasta.
- private synchronized View **getView**_(User user,String view)
- public static View **getView**(User user, String view)
  Palautetaan käyttäjän *user* näkymä *view*.
- private synchronized Document **getMessage**_(User user, String id)
- static Document **getMessage**(User user, String id)
  Palautetaan käyttäjän *user* viesti *id*.
- private synchronized String[] **getViewNames**_(User user)
- public static String[] **getViewNames**(User user)
  Palautetaan käyttäjän *user* kaikkien näkymien nimet.

4.1.15 ViewEvent

ViewEvent on apuluokka ViewMonitorin ja ViewManagerin välillä. ViewEvent-olioiden avulla ViewManager suorittaa viestiin ja näkymiin liittyviä pyydettyjä toimintoja.

Määrittely :

class ViewEvent

Riippuvuudet :

- converge.service.Message

Konstruktorit :

- public **ViewEvent**(int type, Message msg, String view)
  Luodaan uusi ViewEvent tyypin *type*, viestin *msg* ja näkymän *view* avulla.
- public **ViewEvent**(User user, int type, String id, String view)
  Luodaan uusi ViewEvent tyypin *type*, viestin tunnisteen *id* ja näkymän *view* avulla ja liitetään tieto käyttäjästä *user*.
- public **ViewEvent**(int type, Message msg)
  Luodaan uusi ViewEvent tyypin *type* ja viestin *msg* avulla.
- public **ViewEvent**(User user, int type, String id)
  Luodaan uusi ViewEvent tyypin *type* ja viestin tunnisteen *id* avulla ja liitetään tieto käyttäjästä *user*.
- public **ViewEvent**(User user, int type, String view)
  Luodaan uusi ViewEvent käyttäjän *user*, tyypin *type* ja näkymän *view* avulla.

Kentät :

- protected static final int **DELETE_MESSAGE_TOTAL**
  Tyyppi: tuhoa viesti järjestelmästä.
- protected static final int **DELETE_MESSAGE_FROM_VIEW**
  Tyyppi: tuhoa viesti näkymästä.
- protected static final int **SET_MESSAGE**
  Tyyppi: Liitä viesti näkymään.
- protected static final int **SET_BY_MESSAGE_ID**
  Tyyppi: Liitä viesti näkymään viestin tunnisteen avulla.

- protected static final int **GET_VIEW**
  Tyyppi: Hae näkymä.
- protected static final int **GET_VIEW_NAMES**
  Tyyppi: Hae näkymien nimet.
- protected static final int **GET_MESSAGE**
  Tyyppi: Hae viesti.
- protected static final in **DELETE_VIEW**
  Tyyppi: Poistetaan näkymä
- public Message **message**
  Tapahtumaan liitetty viesti
- public String **id**
  Tapahtumaan liitetyn viestin tunniste
- public String **view**
  Näkymän nimi
- public User **user**
  Käyttäjäolio, jonka tietoihin tapahtuman toiminnot tehdään.
- public int **type**
  Olion tyyppi, arvona jokin luokan staattisista tyyppimäärittelyistä

4.1.16 ViewManager

ViewManager huolehtii ViewMonitorille annettujen näkymien ja viestien muutospyyntöjen käsittelemisestä ViewManagerThreadin avulla.

Määrittely :

public class ViewManager

Riippuvuudet :

- converge.service.Message

- java.util.*
- org.w3c.dom.*
- converge.attribute.*

Konstruktorit :

- private **ViewManager**()
  Luodaan uusi ViewManager.

Kentät :

- private static ViewManager **vm**
  Viite järjestelmän ainoaan ViewManager-olioon
- private ViewManagerThread **vmt**
  Viite varsinaisen toiminnan suorittavaan säikeeseen
- private List **list**
  Lista ViewMonitor-olioista, joita ei ole vielä käsitelty.
- private boolean **stopped**
  Onko ViewManager pysäytetty.
- private boolean **stop**
  Tulisiko ViewManagerin pysähtyä suoritettuaan ensin kaikki listalla olevat toiminnot.

Metodit :

- public static ViewManager **getInstance**()
  Palautetaan järjestelmän ainoa ViewManager-olio.
- public synchronized void **shutdown**()
  Pysäytetään ViewManagerin toiminta. Tämän metodin kutsu jää odottamaan, kunnes *stopped*-kentän arvo on true.
- private void **start**()
  Käynnistetään ViewManageriin liitetty ViewManagerThread-säie.

- public synchronized ViewMonitor **removeViewMonitor**()
  Poistetaan käsittelyä odottavien listalta ensimmäinen ViewMonitor.
- protected synchronized ViewMonitor **getNewViewMonitor**()
  Palautetaan käsittelyä odottavien listalle liitetty uusi ViewMonitor-olio-(ks. JavaDoc)

4.1.17 ViewManagerThread

Järjestelmässä oleva yksi ainoa ViewManagerThread tarkkailee ViewMonitor-olioiden muodossa tulevia toimintapyyntöjä ja suorittaa niitä saapumisjärjestyksessä. Muutokset tietokantaan hoidetaan DatabaseProviderin kautta.

Määrittely :

class ViewManager extends Thread

Riippuvuudet :

- converge.service.Message
- java.util.*
- org.w3c.dom.*

Kentät :

- private ViewManager **vm**
  Viite ViewManageriin
- private DatabaseProvider **db**
  Viite järjestelmän tietokantayhteyksiä hoitavaan DatabaseProvider-olioon

Metodit :

- protected void **setViewManager**(ViewManager vm)
  Asetetaan säikeen käyttämä ViewManager *vm*.

- public void **run**()
  Säikeen suoritus
- private void **setMessage**(User user, Message msg)
  Liitetään viesti *msg* käyttäjän *user* tietokantakokoelmaan.
- private void **setMessageToView**(User user, String view, String id)
  Lisätään viesti *id* käyttäjän *user* näkymään *view*.
- private void **deleteMessageFromView**(User user, String view, String id)
  Poistetaan viesti *id* käyttäjän *user* näkymästä *view*.
- private void **deleteMessage**(User user, String id)
  Poistetaan käyttäjän *user* viesti *id* kaikista näkymistä ja tietokannasta.
- private View **getView**(User user, String viewname)
  Haetaan kannasta käyttäjän *user* näkymän *viewname* tiedot ja muodostetaan niistä uusi View-olio.
- private Header **getHeader**(User user, String id)
  Muodostetaan käyttäjän *user* viestin *id* tiedoista palautettava otsikkotieto-olio. `Tämä metodi on tehty lähinnä prototyypin käyttöliittymää varten.`
- private Document **getMessage**(User user, String id)
  Haetaan käyttäjän *user* viesti *id* tietokannasta.
- private String[] **getViewNames**(User user)
  Haetaan käyttäjän *user* näkymien nimet tietokannasta.
- private void **deleteView**(User user, String view)
  Poistetaan käyttäjän *user* näkymä *view*.
- private Element **getMessageInView**(Document dom, String id)
  Palautetaan näkymässä *dom* olevaan viestiin *id* liitetty XML-Element-tyyppinen olio (tai null, mikäli näkymässä ei haluttua viestiä ole).

4.1.18 DatabaseProvider

DatabaseProvider tarjoaa tietokantapalveluita muille järjestelmän olioille. Tietokantana se käyttää Apache organisaation Xindice XML-tietokantaa, ja tästä syystä suoritettavat kyselyt ovat lähinnä DOM-puiden läpikäyntiä. Tulevaisuudessa myös XPath-kyselyt ja XUpdate-päivitykset voivat olla mahdollisia.

Määrittely :

public class DatabaseProvider

Riippuvuudet :

- org.w3c.dom.*
- java.io.*
- org.apache.xindice.server.*
- org.apache.xindice.core.*
- org.apache.xindice.core.data.*
- org.apache.xindice.server.services.*
- org.apache.xindice.util.*
- org.apache.xindice.xml.dom.DOMParser
- org.apache.xindice.xml.TextWriter
- converge.service.Message

Konstruktorit :

- private **DatabaseProvider**()
  Luodaan uusi DatabaseProvider.

Kentät :

- private static DatabaseProvider **db**
  Viite järjestelmän ainoaan DatabaseProvider-olioon

- private Kernel **kernel**
  Viite Xindicen Kernel-tyyppiseen olioon, joka tarjoaa varsinaisen tietokantatoteutuksen.
- public static final String **SYSTEM_PREF**
  Järjestelmäasetusten kokoelman nimi
- public static final String **USERS**
  Käyttäjäkokoelman nimi
- public static final String **GROUPS**
  Ryhmäkokoelman nimi
- public static final String **PREF**
  Käyttäjien ja ryhmien asetuskokoelman nimi
- public static final String **PROFILES**
  Käyttäjien ja ryhmien profiilikokoelman nimi
- public static final String **CONTEXTMODELS**
  Käyttäjien ja ryhmien kontekstimallikokoelman nimi
- public static final String **VIEWS**
  Käyttäjien näkymäkokoelman nimi
- public static final String **DATASOURCES**
  Käyttäjien ja ryhmien tiedonlähdekokoelman nimi
- public static final String **PREFERENCE**
  Käyttäjien ja ryhmien asetustiedoston nimi
- public static final String **MESSAGES**
  Käyttäjien viestikokoelman nimi
- private static String[] **defCols**
  Niiden kokoelmien nimet, jotka luodaan järjestelmään ensimmäistä kertaa käynnistettäessä.

Metodit :

- public static DatabaseProvider **getInstance**() Palautetaan järjestelmän ainoa DatabaseProvider-olio.
- protected synchronized Database **getDatabase**()
  Palautetaan org.apache.xindice.core.Database -tyyppinen tietokantaliittymän tarjoava olio.
- public synchronized void **shutdown**()
  Tietokannan sulkeminen
- private void **init**()
  Järjestelmän ensimmäisellä käynnistyskerralla luodaan tarvittavia kokoelmia.
- protected Collection **createCollection**(Collection col, String name)
  Luodaan uusi kokoelman *name* kokoelman *col* alle.
- protected synchronized Collection **getUserCollection**(String name)
  Palautetaan käyttäjän *name* kokoelma.
- protected synchronized Collection **newUserCollection**(String name)
  Luodaan uusi kokoelma käyttäjää *name* varten.
- protected synchronized void **removeUserCollection**(String name)
  Poistetaan käyttäjän *name* kokoelma kannasta.
- protected synchronized Collection **getGroupCollection**(String name)
  Palautetaan ryhmän *name* kokoelma.
- protected synchronized Collection **newGroupCollection**(String name)
  Luodaan uusi kokoelma ryhmää *name* varten.
- protected synchronized void **removeGroupCollection**(String name)
  Poistetaan ryhmän *name* kokoelma kannasta.
- protected synchronized void **setDocument**(Collection col, String sub, String id, Document dom)
  Asetetaan dokumentti *dom* tunnisteella *id* kokoelmaan *col*, tai mikäli *sub* ei ole null, kokoelman *col* alikokoelmaan *sub*.

- protected synchronized Document **getDocument**(Collection col, String sub, String id)
  Palautetaan dokumentti *id* kokoelmasta *col*, tai mikäli *sub* ei ole null, kokoelman *col* alikokoelmasta *sub*.
- protected synchronized void **deleteDocument**(Collection col, String sub, String id)
  Poistetaan dokumentti *id* kokoelmasta *col*, tai mikäli *sub* ei ole null, kokoelman *col* alikokoelmasta *sub*.
- protected synchronized boolean **setCollectionEmpty**(Collection parent, String collection)
  Poistetaan kokoelma *collection* kokoelmasta *parent* ja luodaan se uudestaan. Palautetaan tieto siitä, onnistuiko operaatio.

4.1.19 Crypt

Crypt-luokka tarjoaa palveluita datan salaamiseksi salasanan avulla. Salattu tieto muutetaan lisäksi Base64-muotoon, jotta sen tallentaminen tietokantaan onnistuisi.

Määrittely :

public class Crypt

Riippuvuudet :

- javax.crypto.spec.*
- javax.crypto.*

Kentät :

- private static byte[] **salt**
  Salauksessa hyödynnettävä taulukko
- private static int **count**
  Iteraatioiden määrä

Metodit :

- public static String **encode**(String pw, byte[] data)
  Salataan *data*-parametrin sisältämä tieto salasanalla *pw* ja palautetaan salattu tietosisältö base64-muotoisena String-oliona.
- public static String **decode**(String pw, String crypted)
  Salasanan *pw* avulla puretaan *crypted*-parametrin base64-muodossa antama salattu tieto ja palautetaan salaamaton tieto.

4.1.20 User

User-luokan ilmentymillä ylläpidetään järjestelmässä eri käyttäjien tietoja. Ilmentymät sisältävät profiileja, kontekstimalleja, viittauksia tiedonlähteisiin, ominaisuusolioita ja kontaktilistan.

Määrittely :

public class User

Riippuvuudet :

- java.util.*
- converge.service.DataSource
- converge.attribute.*

Konstruktorit :

- protected **User**(String name, String password)
  Uuden User-olion luominen on sallittua vain ytimessä (pakkaus converge.kernel). Parametreinä nimi, *name* ja salasana, *password*

Kentät :

- private HashMap **contextModels**
  Käyttäjän kontekstimallit hajautustaulussa
- private HashMap **profiles**
  Käyttäjän profiilit hajautustaulussa
- private ContactList **contacts**
  Kontaktilista
- private HashMap **datasources**
  Käyttäjän tiedonlähteet hajautustaulussa
- private HashMap **userProperties**
  Käyttäjän ominaisuusoliot hajautustaulussa
- private String **name**
  Käyttäjän nimi
- private String **password**
  Käyttäjän salasana
- private Set **readLocks** User-olion lukulukot
- private Object **theWriteLock** User-olion kirjoituslukko
- private boolean **waitForWriteLock**
  Yrittääkö joku saada User-olioon kirjoituslukkoa.
- private boolean **deleted**
  Onko käyttäjän tiedot poistettu tietokannasta.

Metodit :

- synchronized void **readLock**(Object locker)
  Olio voidaan lukulukita, jolloin kukaan ei voi muuttaa olion tietoja. Lukulukkoja User-oliota kohti voi olla useita. Parametrinä *locker* saadaan se olio, joka lukituksen haluaa tehdä.
- synchronized void **writeLock**(Object locker)
  User-olio voidaan kirjoituslukita, jolloin lukuoperaatiot on estetty. Kir-

joituslukon tulisi olla päällä mahdollisimman vähän aikaa, sillä lukituksen ollessa päällä vain yksi olio tai luokka pystyy käsittelemään User-ilmentymää. Parametrinä *locker* saadaan se olio, joka lukituksen haluaa tehdä.

- synchronized void **releaseLock**(Object locker)
  Lukon vapautus, toimii sekä luku- että kirjoituslukoille. Parametrinä *locker* saadaan se olio, jonka saama lukko vapautetaan.
- synchronized boolean **hasLock**(Object locker)
  Voidaan testata onko oliolla *locker* User-olion lukko.
- ArrayList **getContextModels**()
  Palautetaan käyttäjän kaikki konteksimallit.
- void **addContextModel**(ContextModel model)
  Lisätään kontekstimalli *model*.
- ArrayList **getProfiles**()
  Palautetaan kaikki käyttäjän profiilit.
- void **addProfile**(Profile profile)
  Lisätään profiili *profile*.
- void **addDatasource**(Datasource src)
  Lisätään uusi tiedonlähde *src*.
- void **deleteDatasource**(Datasource src)
  Poistetaan tiedonlähde *src*.
- Datasource **getDatasource**(String name)
  Palautetaan tiedonlähde nimen *name* perusteella.
- String[] **getDatasourceNames**()
  Palautetaan kaikkien User-olioon liitettyjen Datasource-olioiden nimet.
- ContactList **getContactList**()
  Palautetaan käyttäjän kontaktilista.

- UserPropertyInterface **getProperty**(String name)
  Pyydetään käyttäjäoliolta ominaisuusoliota nimen *name* perusteella.
- void **setProperty**(String name, Object value)
  Asetetaan ominaisuusolion *name* arvoksi *value*.
- void **save**()
  Kun käyttäjän tiedot halutaan tallentaa tietokantaan, kutsutaan tätä metodia.

4.1.21 UserManager

UserManager hallinnoi käyttäjä- ja ryhmäolioita ja huolehtii niiden kantaantallennuksesta.

Määrittely :

class UserManager

Riippuvuudet :

- java.util.*
- org.w3c.dom.*
- org.apache.xindice.core.*
- converge.service.DataSource

Konstruktorit :

- private **UserManager**()
  Uuden UserManager-olion luonti

Kentät :

- private static UserManager **um**
  Viite järjestelmän ainoaan UserManager-olioon

- private Map **users**
  Järjestelmän käyttäjät
- private Map **groups**
  Järjestelmän ryhmät
- private DatabaseProvider **db** Viite järjestelmän tietokantapalveluita tarjoavaan DatabaseProvider-olioon

Metodit :

- static UserManager **getInstance**()
  Palautetaan järjestelmän ainoa UserManager-olio.
- public synchronized void **shutdown**()
  UserManager-ilmentymän sulkeminen aiheuttaa kaikkien käyttäjien tietojen tallentamisen tietokantaan.
- public synchronized Group **newGroup**(String name, String password)
  Luodaan uusi ryhmä nimellä *name* ja salasanalla *password*, mikäli sellaista ei järjestelmässä jo ole.
- public synchronized User **newUser**(String name, String password)
  Luodaan uusi käyttäjä nimellä *name* ja salasanalla *password*, mikäli sellaista ei järjestelmässä jo ole.
- public synchronized Group **getGroup**(String name)
  Palautetaan Group-olio nimen perusteella.
- public synchronized User **getUser**(String name)
  Palautetaan User-olio nimen perusteella.
- public synchronized Group **getGroup**(String name, String password)
  Palautetaan Group-olio nimen ja salasanan perusteella.
- public synchronized User **getUser**(String name, String password)
  Palautetaan User-olio nimen ja salasanan perusteella.
- public synchronized boolean **hasGroup**(String name)
  Onko järjestelmässä ryhmää nimeltä *name*.

- public synchronized boolean **hasUser**(String name)
  Onko järjestelmässä käyttäjää nimeltä *name*.
- public synchronized void **deleteGroup**(String name)
  Poistetaan ryhmä *name*.
- public synchronized void **deleteUser**(String name)
  Poistetaan käyttäjä *name*
- public synchronized void **saveUser**(User user)
  Tallennetaan käyttäjän *user* tiedot kantaan. Parametrinä voidaan antaa myös Group-tyyppinen olio.
- private void **saveUserNow**(User user)
  Tallennetaan käyttäjä
  ryhmä *user* kantaan.
- protected synchronized void **saveAllUsers**(int sleep)
  Tallennetaan järjestelmän kaikki käyttäjät. (Järjestelmän toimintaa tutkittaessa voidaan haluttaessa tallennusten välissä keskeyttää toiminto parametrin *sleep* määrittelemäksi ajaksi millisekunneissa.)
- protected synchronized void **loadUsers**()
  Ladataan järjestelmän käyttäjien tiedot tietokannasta.
- protected synchronized void **loadGroups**()
  Ladataan järjestelmän ryhmien tiedot tietokannasta.
- private User **load**(String name, boolean group)
  Ladataan käyttäjän tai ryhmän *name* tiedot tietokannasta. Parametri *group* kertoo, onko kyseessä ryhmä.
- public synchronized String[] **getUserNames**()
  Palautetaan järjestelmän käyttäjien nimet.
- public synchronized String[] **getGroupNames**()
  Palautetaan järjestelmän ryhmien nimet.

4.1.22 Group

Järjestelmän ylläpitäjä voi muodostaa käyttäjäryhmiä, joita kuvaa Group-luokka. Käyttäjäryhmien toiminta on useilta osin sama kuin varsinaisilla käyttäjillä, mutta ryhmillä ei ole näkymiä. Ryhmälle tullut viesti käsitellään samoin kuin tavallisetkin viestit, mutta ProfileHandler ei annakaan viestiä EventMonitorille, vaan siitä luodaan kopio jokaista ryhmän jäsentä varten, mitkä käsitellään uudestaan jokaisen ryhmän jäsenen kontekstimallien ja sääntöjen avulla.

Määrittely :

public class Group extends User

Konstruktorit :

- protected **Group**(String name, String password, String[] userNames)
  Uuden ryhmän *name* luominen salasanalla *password* ja käyttäjillä *userNames*.

Kentät :

- private UserNameCollection **group**
  Tieto ryhmään liitetyistä käyttäjistä

Metodit :

- public UserNameCollection **getUserNames**()
  Palautetaan ryhmään kuuluvien käyttäjien nimet UserNameCollection-tyyppisenä oliona.

4.1.23 UserNameCollection

Määrittely :

public class UserNameCollection implements XMLContentInterface

Riippuvuudet :

- java.util.*
- org.w3c.dom.*

Kentät :

- private HashSet **users**
  Kokoelma käyttäjänimiä

Metodit :

- public String[] **getUserNames**()
  Palautetaan käyttäjien nimiä sisältävä merkkijonotaulukko.
- public void **addUser**(User user)
  Lisätään käyttäjän *user* nimi kokoelmaan.
- public void **addUser**(String user)
  Lisätään käyttäjän *user* nimi kokoelmaan.
- public boolean **hasUser**(String name)
  Löytyykö kokoelmasta käyttäjä nimellä *name*.
- public void **removeUser**(String name)
  Poistetaan käyttäjä *name* kokoelmasta.
- public void **createXML**(Node parent)
  Katso 4.2.2 sivulla 64.
- public boolean **loadXML**(Node myContent)
  Katso 4.2.2 sivulla 64

4.1.24 Profile

Profile-oliolla kuvataan järjestelmässä käyttäjään liitettyä profiilia. Se sisältää tiedon siihen liitettyjen kontekstimallien kynnysarvoista, ajastetuista säännöistä ja varsinaisista viestin hallintaan liittyvistä säännöistä.

Määrittely :

public class Profile implements XMLContentInterface

Riippuvuudet :

- java.util.*
- org.w3c.dom.*

Konstruktorit :

- protected **Profile**(User user)
  Luodaan käyttäjälle *user* uusi profiili.
- public **Profile**(User user, String name)
  Luodaan käyttäjälle *user* uusi profiili nimeltä *name*.

Kentät :

- public static final String **DEFAULT**
  Oletusprofiilin nimi.
- public User **user**
  Viite profiilin käyttäjään
- private String **name**
  Profiilin nimi
- private boolean **active**
  Onko profiili aktiivinen
- private LinkedHashMap **rules**
  Profiiliin liitetyt säännöt
- private HashMap **contextModelValues**
  Profiiliin liitetyt konteksimallien kynnysarvot
- private HashMap **scheduledRules**
  Profiiliin liitetyt ajastukset (, jotka pitävät sisällään sääntöjä)

Metodit :

- public boolean **isDefaultProfile**()
  Onko profiili käyttäjän oletusprofiili.
- public String **getName**()
  Palautetaan profiilin nimi.
- public boolean **isActive**()
  Onko profiili aktiivinen.
- public void **setActive**(boolean active)
  Asetetaan profiilin aktiivisuus parametrillä *active*.
- public Rule **getRule**(String name)
  Palautetaan sääntö nimen *name* perusteella.
- public void **setRule**(Rule rule)
  Lisätään profiiliin sääntö *rule*.
- public void **deleteRule**(String name)
  Poistetaan sääntö nimeltä *name*.
- public String[] **getRuleNames**()
  Palautetaan kaikkien profiilin sääntöjen nimet.
- public Rule[] **getRules**()
  Palautetaan säännöt taulukossa.
- public void **addScheduledRule**(Scheduler rule)
  Lisätään ajastettu sääntö *rule*.
- public Scheduler **getScheduledRule**(String name)
  Palautetaan ajastettu sääntö *name*.
- public void **deleteScheduledRule**(String name)
  Poistetaan ajastettu sääntö *name*.
- public Scheduler[] **getScheduledRules**()
  Palautetaan kaikki ajastetut säännöt taulokossa.

- public void **setContextModelValue**(String name, int value)
  Lisätään profiiliin uusi kontekstimallin kynnysarvo.
- public int **getContextModelValue**(String name)
  Palautetaan kontekstimalliin liitetty arvo.
- public void **removeContextModel**(String name)
  Poistetaan profiilista viittaus kontekstimalliin.
- public String[] **getContextModels**()
  Palautetaan profiilin viittaamat kontekstimallit.
- public void **createXML**(Node parent)
  Katso 4.2.2 sivulla 64.
- public boolean **loadXML**(Node myContent)
  Katso 4.2.2 sivulla 64.

4.1.25 Rule

Rule-luokalla kuvataan järjestelmässä käytettäviä sääntöjä, jotka ovat muodoltaan ehdot-toiminnot -pareja.

Määrittely :

class Rule implements XMLContentInterface

Riippuvuudet :

- java.util.*
- org.w3c.dom.*
- org.drools.rule.*

Konstruktorit :

- protected **Rule**()
  Luodaan uusi sääntö.

- **Rule**(String name)
  Luodaan uusi sääntö, jonka nimi on *name*.

Kentät :

- public static final int **EVENT_ADD**
  Uuden toiminnan lisäys.
- public static final int **EVENT_DELETE**
  Toiminnan poisto.
- private String **name**
  Säännön nimi
- private LinkedList **conditions**
  Ehdot
- private LinkedList **actions**
  Toiminnot
- private boolean **opAdded**
  Onko ehdon lisäämisen jälkeen lisätty myös ehdollinen operaattori (conditional operator) (joko AND tai OR).

Metodit :

- public String **getName**()
  Palautetaan säännön nimi.
- public void **addCondition**(String attributeName, String value, String expressionType)
  Lisätään säännön ehtoon vertailu *attributeName*-attribuutin ja arvon *value* välillä *expressionType*:n mukaisesti esimerkiksi *Subject* in "converge". *AttributeName* viittaa suoraan attribuuttien nimiin .
- public void **nextCondition**(String conditionalOperator)
  Jatketaan ilmentymän sisältämää ehtoa *conditionalOperator*:n mukaisesti. Parametrin tyyppinä tämä prototyyppi tukee vain AND ja Or -operaattoreita.

- public void **consequence**(int eventType, String eventname, String eventvalue)
  Lisätään toiminto sääntöön. Toiminnon tyyppi on *eventType* (joko EVENT_ADD tai EVENT_DELETE) ja toiminta määritellään avain *eventname* - arvo *eventvalue* pareina.
- public List **getConsequences**()
  Palauttaa listan toiminnoista.
- public org.drools.rule.Declaration[] **getParametersForDrools**()
  Palautetaan Declaration-taulukko, joka sisältää viitteen sääntöjen käsittelyssä käytettävien olioiden nimiin ja tyyppeihin.
- public org.drools.semantics.java.ExprCondition **getConditionForDrools**()
  Palautetaan Droolsin ymmärtämässä muodossa säännön ehto.
- public org.drools.semantics.java.BlockConsequence **getConsequenceForDrools**()
  Palautetaan Droolsin ymmärtämässä muodossa säännön toiminto.
- public void **createXML**(Node parent)
  Katso 4.2.2 sivulla 64.
- public boolean **loadXML**(Node myContent)
  Katso 4.2.2 sivulla 64.

4.1.26 Contact

Contact-luokka kuvaa yhtä kontaktilistassa olevaa tietoalkiota. Rakenteeltaan Contact on nimetty hajautustaulu, eli sillä on nimi ja muut tiedot esitetään avain-arvo -pareina.

Määrittely :

public class Contact implements XMLContentInterface

Riippuvuudet :

- java.util.*

- org.w3c.dom.*

Konstruktorit :

- public **Contact**()
  Luodaan uusi Contact-olio.
- public **Contact**(String name)
  Luodaan uusi *name*-niminen Contact-olio.

Kentät :

- private HashMap **props**
  Kontaktitiedon ominaisuudet hajautustaulussa
- private String **name**
  Kontaktitiedon nimi

Metodit :

- public String **getName**()
  Palautetaan Contact-olion nimi.
- public String **get**(String key)
  Palautetaan avaimeen *key* liitetty arvo.
- public void **set**(String key, String value)
  Lisätään uusi tieto avaimella *key* ja arvolla *value*. Korvataan mahdollinen vanha arvo.
- public String[] **getKeys**()
  Palautetaan Contact-olion kaikkien avain-arvo -parien avaimet.
- public void **createXML**(Node parent)
  Katso 4.2.2 sivulla 64.
- public boolean **loadXML**(Node myContent)
  Katso 4.2.2 sivulla 64

4.1.27 ContactList

Jokaisella järjestelmän käyttäjällä on yksi kontaktilista (ContactList), joka voi sisältää sekä varsinaisia kontaktitietoja (Contact) että toisia kontaktilistoja.

Määrittely :

public class ContactList implements XMLContentInterface

Riippuvuudet :

- java.util.*
- org.w3c.dom.*

Konstruktorit :

- public **ContactList**()
  Luodaan uusi kontaktilista.
- public **ContactList**(String name) Luodaan uusi kontaktilista nimeltään *name*.

Kentät :

- private String **name**
  Kontaktilistan nimi
- private HashMap **contacts**
  Kontaktilistan yksittäiset kontaktitiedot
- private HashMap **contactlists**
  Kontaktilistan alikontaktilistat

Metodit :

- public String **getName**()
  Palautetaan kontaktilistan nimi.

- public Set **getContactNames**()
  Palautetaan kaikkien ContactList-olion sisältämien Contact-olioiden nimet.
- public Contact **getContact**(String name)
  Etsitään nimellä *name* kontakti tietoa.
- public Set **getContactListNames**()
  Palautetaan kontaktilistan alikontaktilistojen nimet.
- public ContactList **getContactList**(String name)
  Etsitään ContactList-olio nimen *name* perusteella.
- public void **addContact**(Contact contact)
  Lisätään uusi kontaktitieto *contact*.
- public void **addNewContactList**(ContactList clist)
  Lisätään uusi alikontaktilista *clist*
- public void **removeContact**(String name)
  Poistetaan kontaktitieto kontaktilistalta.
- public void **removeContactList**(String name)
  Poistetaan alikontaktilista.
- public void **createXML**(Node parent)
  Katso 4.2.2 sivulla 64.
- public boolean **loadXML**(Node myContent)
  Katso 4.2.2 sivulla 64

4.1.28 ScheduledRule

ScheduledRule muuttaa Rule-luokan määrityksiä siten, että sääntöjä voidaan käyttää myös ajastetuissa säännöissä. Tämä tarkoittaa erityisesti sääntökäsittelijän käyttämän Message-olioviittauksen muuttamista User-tyyppiseksi.

Määrittely :

public class ScheduledRule extends Rule

Riippuvuudet :

- org.w3c.dom.*

Konstruktorit :

- public **ScheduledRule**()
  Luodaan uusi ScheduledRule.
- public **ScheduledRule**(String name)
  Luodaan uusi ScheduledRule nimeltään *name*.

Metodit :

- public org.drools.rule.Declaration[] **getParametersForDrools**()
  Katso 4.1.25 sivulla 50..
- public org.drools.semantics.java.ExprCondition **getConditionForDrools**()
  Katso 4.1.25 sivulla 50.

4.1.29 ScheduleManager

ScheduleManager käy säännöllisin väliajoin läpi käyttäjien ja ryhmien aktiivisten profiilien ajastetut säännöt ja mikäli jokin sääntö tulee laukaista, luodaan uusi ScheduledRuleHandler-säie, joka huolehtii säännön käsittelystä.

Määrittely :

public class ScheduleManager extends Thread

Riippuvuudet :

- org.w3c.dom.*
- java.util.*

Kentät :

- public static final int **sleepTime**
  Tämä luku kertoo sekunneissa kuinka usein ScheduleManager tutkii aktiivisia profiileita.
- private static ScheduleManager **sm**
  Viite järjestelmän ainoaan ScheduleManager-olioon
- private boolean **stop**
  Tulisiko ScheduleManagerin pysähtyä
- private boolean **stopped**
  Onko ScheduleManager pysähtynyt

Metodit :

- public static ScheduleManager **getInstance**()
  Palautetaan järjestelmän ainoa ScheduleHandler-olio.
- public void **run**()
  Säikeen toiminnallisuus on run-metodissa.
- public synchronized void **shutdown**()
  ScheduleManagerin toiminnan pysäyttäminen

4.1.30 ScheduledRuleHandler

ScheduledRuleHandler huolehtii yhden ajastetun säännön suorittamisesta.

Määrittely :

class ScheduledRuleHandler extends Thread

Riippuvuudet :

- org.w3c.dom.*

Konstruktorit :

- public **ScheduledRuleHandler**(User user, Document rules)
  Luodaan uusi käsittelijä käyttäjän *user* ajastetulle säännölle *rules*.

Kentät :

- private Document *rules*
  Säännöt XML-muodossa.
- private User *user*
  Viite käyttäjään, kenen sääntöjä ollaa käsittelemässä
- private ActionEvent **event**
  Toimintotapahtuma, jonka sisältö muodostetaan RuleHandlerissa.

Metodit :

- public void **run**()
  Säikeen suoritus, jossa ensin RuleHandlerin avulla muodostetaan uusi ActionEvent, joka annetaan edelleen EventMonitorille User-viitteen kanssa.

4.1.31 Scheduler

Scheduler kuvaa aikaväliä tai tarkkaa aikaa, jolloin jonkin toiminnon pitäisi tapahtua.

Määrittely :

public class Scheduler implements XMLContentInterface

Riippuvuudet :

- org.w3c.dom.*
- java.util.*

Konstruktorit :

- public **Scheduler**(String name, int type, int multiply)
  Luodaan uusi ajastin *name* tyypin *type* ja kertoimen *multiply* avulla.

- public **Scheduler**(String name, Date date)
  Luodaan uusi kerran suoritettava ajastin *name* päivämääräolion *date* avulla.
- public **Scheduler**()
  Luodaan uusi ajastus.

Kentät :

- public static final int **MINUTE**
  Tyyppi: minuutti.
- public static final int **HOUR**
  Tyyppi: tunti.
- public static final int **DAY**
  Tyyppi: vuorokausi.
- public static final int **WEEK**
  Tyyppi: viikko.
- public static final int **MONTH**
  Tyyppi: kuukausi.
- private Date **date**
  Mikäli ajastus on vain kerran suoritettava, pitää *date* llä viitettä päivämääräolioon, joka kertoo ajastuksen halutun laukeamisajan.
- private int **type**
  Ajastuksen tyyppi
- private int **multiply**
  Ajastuksen kerroin
- private LinkedHashMap **rules**
  Ajastukseen liitetyt säännöt
- private long **lastTime**
  Viimeisin laukeamisajankohta

- private String **name**
  Ajastuksen nimi

Metodit :

- protected long **getLastTime**()
  Edellinen laukeamisajankohta
- protected void **setLastTime**(long time)
  Asetetaan edellinen laukeamisajankohta.
- public String **getName**()
  Palautetaan ajastimen nimi.
- public boolean **exact**()
  Onko ajastin luotu Date-olion avulla, jolloin tiedetään tarkasti milloin ajastuksen tulisi laueta.
- public Date **getDate**()
  Palautetaan date-olio.
- public int **getType**()
  Palautetaan ajastimen tyyppi.
- public int **getMultiply**()
  Palautetaan ajastimen kerroin.
- public long **getInterval**()
  Palautetaan ajastuksen laukeamisväli sekunneissa.
- public void **setRule**(Rule rule)
  Lisätään ajastimeen uusi sääntö *rule*.
- public ScheduledRule **getRule**(String name)
  Palautetaan sääntö nimen *perusteella*.
- public void **deleteRule**(String name)
  Poistetaan sääntö nimeltä *name*.

- public String[] **getRuleNames**()
  Palautetaan kaikkien sääntöjen nimet.
- public Document **getRuleSet**()
  Katso 4.1.24 sivulla 47.
- public void **createXML**(Node parent)
  Katso 4.2.2 sivulla 64.
- public boolean **loadXML**(Node myContent)
  Katso 4.2.2 sivulla 64

4.1.32 XMLUtil

XMLUtil-luokka tarjoaa XML:n käsittelyä helpottavia staattisia metodeja.

Määrittely :

public class XMLUtil

Riippuvuudet :

- org.w3c.dom.*

Kentät :

- public static final String **NS_DROOLS**
  Droolsin nimiavaruus.
- public static final String **NS_DROOLS_JAVA**
  Droolsin vaatima Javan nimiavaruus.

Metodit :

- public static Document **newXMLDocument**()
  Luodaan uusi w3c.org.dom.Document -tyyppinen olio.
- public static Document **newDroolsDocument**()
  Luodaan uusi w3c.org.dom.Document -tyyppinen olio, joka pitää sisällään Droolsin tarvitsemia XML-elementtejä.

- public static void **outputPrinting**(Document dom, OutputStream output)
  Tulostetaan XML-dokumentti *dom output*-virtaan, tai mikäli *output* on null, Käytetään System.out -virtaa.

4.1.33 Log

Log-luokkaa käytetään järjestelmän lokitietojen tallennukseen.

Määrittely :

public class Log

Riippuvuudet :

- java.io.*
- java.util.*

Kentät :

- public static PrintStream **out**
  Järjestelmän oletustulostusvirta.
- public static PrintStream **err**
  Järjestelmän virhetulostusvirta.

Metodit :

- public static synchronized PrintStream output **output**(String name)
  Palautetaan viittaus *name*-nimiseen tulostusvirtaan, jonka avulla tiedot tallennetaan tulostusvirran nimiseen tiedostoon.
- public synchronized static void **flush**() Kutsutaan java.io.PrintStream-luokan flush-metodia kaikille järjestelmän loki-tulostusvirroille.
- public static synchronized void **stack**(Exception e)
  Tulostetaan poikkeuksen *e* pinokuvaus Log.err -virtaan.

- public static synchronized void **stack**(Exception e, String output)
  Tulostetaan poikkeuksen *e* pinokuvaus parametrin *output* nimiseen tulostusvirtaan.
- public static synchronized void **stack**(Exception e, PrintStream output)
  Tulostetaan poikkeuksen *e* pinokuvaus *output* -tulostusvirtaan.

4.2 Rajapinnat : *converge.kernel*

4.2.1 ActionInterface

ActionInterface:lla määritetään ne toiminnallisuudet, jotka jokaisen järjestelmään toteutettavan toiminnallisuusluokan on tarjottava. Toteutusvaiheessa tällaisia luokkia tullaan tekemään esimerkiksi viestien lähettämistä ja hakua varten.

Määrittely :

interface ActionEvent extends XMLContentInterface

Metodit :

- void **setValue**(User user, Object ob, String[] values)
  Asetetaan toiminnan suorittamiseksi arvoja, missä toiminta liitetään käyttäjään *user*, käyttäjän johonkin olioon *ob* (esimerkiksi Message) sekä arvotaulukon *values* sisältämien arvojen avulla. *Values*-arvotaulukon sisältö riippuu kutsutun toteutus luokan toiminnallisuuden ohjaamiseksi tarvittavista tiedoista, jotka voivat olla esimerkiksi joukko tietokantakyselyjä.
- void **doAction**(String action)
  Metodi laukaisee ilmentymän toiminnan, se mitä luokan tarjoamista toiminnallisuuksista annetuilla arvoilla halutaan laukaista määritetään *action*-parametrilla.
- String[] **supportedActions**()
  String-tyyppinen taulukko toiminnallisuuksista, joiden toteutuksen luok-

ka tarjoaa. Taulukon sisältämät määritykset ovat *doAction*-metodin hyväksymiä syötteitä.

4.2.2 XMLContentInterface

XMLContentInterface-rajapinnan toteuttavien luokkien on osattava luoda omasta tietosisällöstään XML-kuvaus, jonka avulla ilmentymät voidaan myöhemmin palauttaa samaan tilaan kuin missä ne olivat kuvauksen luonnin yhteydessä.

Määrittely :

interface XMLContentInterface

Metodit :

- void **createXML**(Element parent)
  Parametrinä *parent* saadaan se XML-elementti, johon olion pitäisi liittää oma XML-kuvauksensa org.w3c.dom.Node -rajapinnan appendChild() -metodilla (lapsielementiksi).
- boolean **loadXML**(Element myContent)
  Oliota kutsuttaessa loadXML-metodilla, muuttaa se tilansa vastaamaan parametrinä saatua tietosisältöä, jonka kuvaus löytyy *myContent*-parametrista.

4.3 Rajapinnat : *converge.attribute*

4.3.1 UserPropertyInterface

UserPropertyInterface:lla määritetään ne ominaisuudet, jotka jokaisen käyttäjäolioon liitettävien ominaisuuksien toteutusluokkien on tarjottava. Tällaisia ominaisuuksia ovat kaikki käyttäjän ulkomaailman kontekstia kuvaavat tiedot esimerkiksi käyttäjän paikkatieto (”paikannus”=”Helsinki, Kallio, Toinen Linja 10”) tai päätelaitteen saavutettavuus (”GSM-OnLine”=”false”).

Määrittely :

interface UserPropertyInterface

Metodit :

- public boolean **compare**(String expressionType, Object value)
  Metodi jonka avulla voidaan vertailla annettavaa arvoa ilmentymän sisältämään arvotietosisältöön. Laskenta suoritetaan, ”Ilmentymän arvo” *expressionType value* esim. 100 < value. Epäselvän laskusäännön tapauksessa palautetaan*false*.
- public void **setUser**(User user)
  Asettaa ilmentymään liitetyn käyttäjäolion
- public public void **setName**(String name)
  Asetetaan ominaisuuden nimi.
- public String[] **supportedAttributes**()
  String-tyyppinen taulukko niiden kaikkien attribuuttien loogisista nimistä joille toteutettu luokka tarjoaa käsittelyn toteutuksen.
- public String **getDescription**()
  Palautetaan selkokielinen kuvaus ominaisuusluokasta.

4.3.2 AttributeInterface

AttributeInterface:lla määritetään ne ominaisuudet, jotka jokaisen attribuuttien käsittelyn toteuttavien luokkien on tarjottava.

Määrittely :

interface AttributeInterface extends XMLContentInterface

Metodit :

- public void **setName**(String name)
  Asetetaan attribuutin nimi (jokin supportedAttributes:in palauttama nimi)

- public String **getName**()
  Palauttaa ilmentymälle annetun nimen, joka vastaa loogista attribuutin nimeä, jolle ilmentymä tarjoaa toteutuksen.
- public void **setValue**(Object value)
  Ilmentymän arvotietosisältö alustetaan antamalla *value*. Toteutus itse käsittelee Object-tyypiä kuten se sitä tarvitsee.
- public Object **getValue**()
  Palauttaa ilmentymän arvotietosisällön Object-tyyppisenä.
- public String[] **supportedAttributes**()
  Jokaisen attribuuttien käsittelyyn toteutetun luokan on pystyttävä kertomaan niiden kaikkien attribuuttien loogiset nimet, joiden käsittelyt se tarjoaa.
- public String **getDescription**()
  Palauttaa luokkaan kirjoitetun kuvauksen siitä miten ja mihin tarkoitukseen siinä määriteltyjä loogisia attribuutteja on mahdollista käyttää.

4.3.3 MessageAttributeInterface

MessageAttributeInterface:lla määritetään kaikki ne ominaisuudet, jotka viestiolioon liitettävien attribuuttiolioiden toteutusluokkien on tarjottava.

Määrittely :

public interface MessageAttributeInterface extends AttributeInterface

Metodit :

- public boolean **compare**(String ExpressionType, Object value)
  Metodi jonka avulla voidaan vertailla annettavaa arvoa ilmentymän sisältämään arvotietosisältöön. Laskenta suoritetaan, ”Ilmentymän arvo” *expressionType value* esim. 100 < value. Epäselvän laskusäännön tapauksessa palautetaan *false*.

- public void **setMessage**(Message msg)
  Annetaan ilmentymälle viite viestiin.

4.3.4 ContextModelAttributeInterface

ContextModelAttributeInterface:lla määritetään ne ominaisuudet, jotka kontekstimalliolioon liitettävien attribuuttiolioiden toteutusluokkien on tarjottava.

Määrittely :

public interface ContextModelAttributeInterface extends AttributeInterface

Metodit :

- public boolean **compare**(Message msg)
  Suorittaa vertailun ilmentymän oman arvotietosisällön ja *msg*-olioon liitetyn samannimisen attribuutin arvotietosisällön välillä käyttämällä ilmentymään asetettua laskusääntöä. Laskenta suoritetaan ”Ilmentymän arvo” laskusääntö msg:n attribuutin arvo esim. 100 < msg.attr.value.
- public void **setExpressionType**(String type)
  Ilmentymän compare-metodin laskennassa käytettävä laskusäännön asetus. *Type*-parametriksi hyväksyttäviä arvoja ”<”, ”>”, ”>=”, ”<=”, ”=” ja ”in”, joista viimeinen tarkoittaa attribuutista riippuen joko alimerkkijonoa tai alkiota taulukossa.
- public String **getExpressionType**()
  Palauttaa ilmentymän vertailulaskennassa käytettävän laskusäännön.

4.4 Luokat : *converge.attribute*

4.4.1 AttributeLoader

AttributeLoader-luokalla ladataan (tarvittaessa dynaamisesti) AttributeInterface -rajapinnan toteuttavia luokkia järjestelmän käyttöön.

Määrittely :

public class AttributeLoader

Metodit :

- public static UserPropertyInterface **newUserPropertyInstance**(String name)
  Luodaan uusi ilmentymä luokasta, joka toteuttaa käyttäjäominaisuudelta vaadittavat ominaisuudet (UserPropertyInterface-rajapinta) ja tarjoaa toteutuksen *name*-nimiselle attribuutille.
- public static ContextModelAttributeInterface **newContextModelAttributeInstance**(String name)
  Luodaan uusi ilmentymä luokasta, joka toteuttaa kontekstimalliattribuutilta vaadittavat ominaisuudet (ContextModelAttributeInterface-rajapinta) ja tarjoaa toteutuksen *name*-nimiselle attribuutille.
- static MessageAttributeInterface **newMessageAttributeInstance**(String name)
  Luodaan uusi ilmentymä luokasta, joka toteuttaa viestiolioattribuutilta vaadittavat ominaisuudet (MessageAttributeInterface-rajapinta) ja tarjoaa toteutuksen *name*-nimiselle attribuutille.
- static synchronized void **loadClasses**()
  Luokka-metodi, jonka suorituksessa käydään kaikki attribuuttien toteutuksia tarjoavat luokat läpi. Tällöin päivitetään AttributeLoaderin tietosisältö siitä, minkä nimisille attribuuteille luokat tarjoavat toteutuksen.

4.4.2 Attribute

Attribute on abstrakti yliluokka muille Attribute-sarjan luokille. Yksi Attribute-sarjan aliluokan ilmentymä pitää sisällään yhden viestin attribuutin (lähettäjä, otsikko, lähetysaika, tärkeys, viestityyppi). Erityyppisiä attribuutteja varten luodaan

omia aliluokkia Attribuuteista, esim. merkkijonoja varten on luokka Attribute-String.

Määrittely :

public abstract class Attribute
implements converge.attribute.MessageAttributeInterface,
converge.attribute.ContextModelAttributeInterface

Kentät :

- private Object value
  Viestiattribuutin tieto.
- private String name
  Viestiattribuutin nimi.
- private String expressionType
  Vertailutyyppi, katso 4.3.4 sivulla 67

Metodit :

- abstract boolean **compare**(String expressionType, Object value)
  Katso 4.3.3 sivulla 66
- abstract boolean **compare**(Message msg)
  Katso 4.3.4 sivulla 67
- Object **getValue**()
  Katso 4.3.3 sivulla 66
- void **setValue**(Object value)
  Katso 4.3.2 sivulla 65
- void **setExpressionType**(String expressionType)
  Katso 4.3.4 sivulla 67
- String **getExpressionType**()
  Katso 4.3.4 sivulla 67

- String **getName**()
  Katso 4.3.2 sivulla 65
- void **setName**(String name)
  Katso 4.3.2 sivulla 65
- abstract String[] **supportedAttributes**()
  Katso 4.3.2 sivulla 65
- abstract void **createXML**(Element parent)
  Katso 4.2.2 sivulla 64.
- abstract boolean **loadXML**(Element myContent)
  Katso 4.2.2 sivulla 64

4.5 Luokat : *converge*

4.5.1 Converge

Järjestelmän käynnistävä luokka.

Määrittely :

public class Converge

Riippuvuudet :

- converge.kernel.*
- converge.clienmanager.*

Metodit :

- public static void **main**(String[] args)
  Käynnistetään järjestelmä. Mikäli parametrinä *args* saadaan -gui, käynnistetään myös ylläpitäjän käyttöliittymä. Järjestelmä voidaan sulkea käynnistyksen jälkeen esimerkiksi antamalla komento Ctrl-C komentorivikehotteessa.

4.5.2 ConvergeShutdown

Järjestelmän sulkemisesta huolehtiva luokka.

Määrittely :

class ConvergeShutdown extends Thread

Riippuvuudet :

- converge.kernel.*
- converge.clienmanager.*

Metodit :

- public static void **run**()
  Järjestelmän sulkeminen hoidetaan säikeen suorituksen yhteydessä.

4.5.3 PairPanel

Kahden käyttöliittymää rakentavan olion yhdistäminen toisiinsa.

Määrittely :

class PairPanel extends Panel

Riippuvuudet :

- java.awt.*

Konstruktorit :

- public **PairPanel**(Component left, Component right, boolean horiz)
  Luodaan uusi PairPanel yhdistämällä *left* ja *right* parametrin *horiz* mukaisesti joko horisontaalisesti tai vertikaalisesti.

4.5.4 UIUpdater

Ylläpitäjän käyttöliittymää päivittävä säie.

Määrittely :

class UIUpdater extends Thread

Riippuvuudet :

- java.awt.*

Konstruktorit :

- public **UIUpdater**(ConvergeAWT awt)
  Luodaan uusi käyttöliittymää *awt* päivittävä säie.

Metodit :

- public void **run**()
  Päivitetään käyttöliittymää kerran sekunnissa.

4.5.5 ConvergeAWT

Ylläpitäjän käyttöliittymän (yksinkertainen) toteutus.

Määrittely :

class ConvergeAWT extends Frame implements WindowListener, ItemListener, ActionListener

Riippuvuudet :

- java.awt.*
- java.awt.event.*

Konstruktorit :

- public **ConvergeAWT**()
  Luodaan uusi käyttöliittymä

Metodit :

- public void **itemStateChanged(ItemEvent e)**
  Käsitellään group-valikon muutokset.
- public void **actionPerformed(java.awt.event.ActionEvent e)**
  Käsitellään käyttöliittymän tapahtumat.
- public void **windowClosing**(WindowEvent e)

- public void **windowClosed**(WindowEvent e)
  Suljettaessa ikkuna, suljetaan myös koko järjestelmä.

4.6 Rajapinta tiedonhakumoduulille

Ytimen tiedonhakumoduulille tarjoamat rajapinnat löytyvät seuraavista luokista:

- 4.1.3 **ContextModelHandler**, s. 16
- 4.1.20 **User**, s. 40
- 4.3.2 **AttributeInterface**, s. 65
- 4.3.3 **MessageAttributeInterface**, s. 66

4.7 Rajapinta asiakasmoduulille

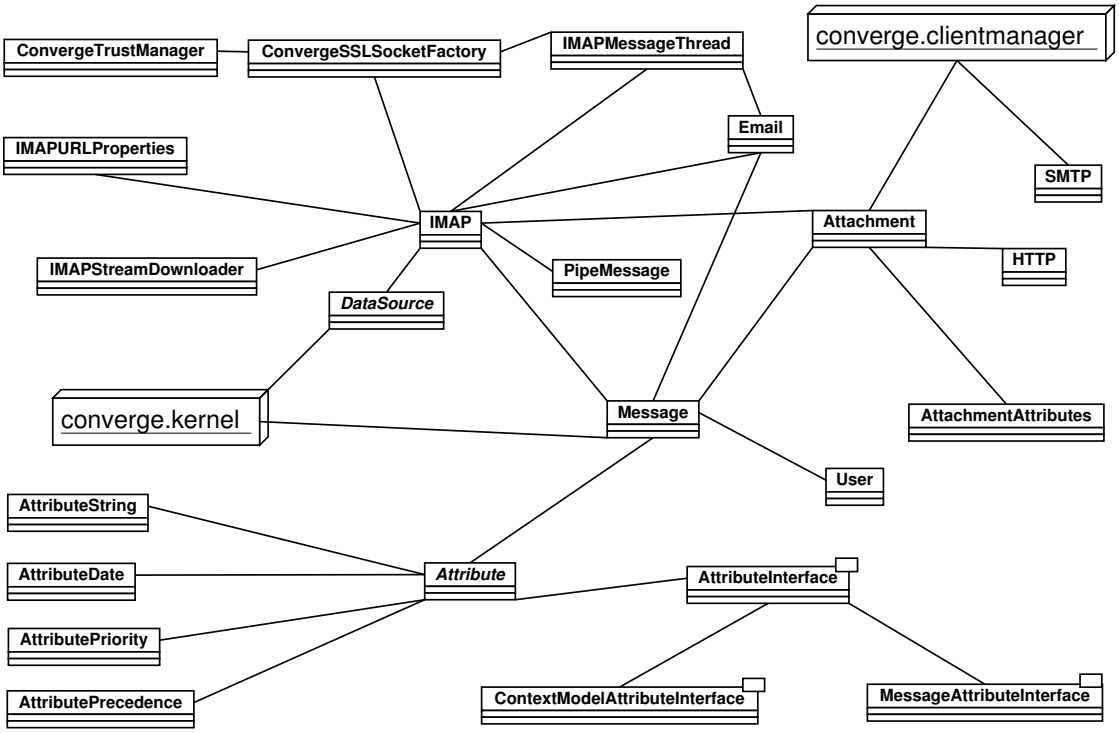
Ytimen asiakasmoduulille tarjoamat rajapinnat löytyvät seuraavista luokista:

- 4.1.21 **UserManager**, s. 43
- 4.1.20 **User**, s. 40

- 4.1.22 **Group**, s. 46
- 4.1.24 **Profile**, s. 47
- 4.1.1 **ContextModel**, s. 12
- 4.1.25 **Rule**, s. 50
- 4.1.31 **Scheduler**, s. 58
- 4.1.14 **ViewMonitor**, s. 28
- 4.1.12 **View**, s. 26
- 4.1.13 **Header**, s. 27
- 4.1.26 **Contact**, s. 52
- 4.1.27 **ContactList**, s. 54
- 4.1.33 **Log**, s. 62

5 Tiedonhakumoduuli

Tiedonhakumoduuli huolehtii yhteyksistä muihin Internetin palvelimiin. Moduuli käyttää hyväkseen JavaMail- ja Java Secure Socket Extension -paketteja (katso http://java.sun.com/products/javamail/ ja http://java.sun.com/products/jsse/). Järjestelmän ydin kutsuu tiedonhakumoduulia kun käyttäjälle saapuneet uudet viestit halutaan noutaa, ja viestit noudettuaan tiedonhakumoduuli muuttaa ne järjestelmän sisäiseen muotoon ja välittää ne järjestelmän ytimelle jatkokäsittelyä varten. Tiedonhakumoduuli huolehtii myös viestien lähettämisestä muille postipalvelimille.



Kuva 3: Tiedonhakumoduulin luokkakaavio

5.1 Luokat : *converge.service*

5.1.1 DataSource

DataSource on abstrakti yläluokka kaikille tiedonlähteille. Luokka tarjoaa aliluokilleen joitakin tiettyjä yleisiä metodeja, mutta suurin osa metodeista pitää tehdä tiedonlähdekohtaisesti. DataSource-luokasta ei tehdä koskaan ilmentymiä, ainoastaan sen aliluokista.

Uusia DataSourcea laajentavia luokkia voidaan kirjoittaa tarpeiden mukaisesti, periaatteena on luoda yksi uusi luokka jokaista verkkoprotokollaa varten.

Määrittely :

public abstract class DataSource implements XMLContentInterface

Kentät :

- protected String **dataSourceName**
  Tiedonlähteen luonnollinen nimi, esim. "Yliopiston IMAP".
- protected String **latestErrorMessage**
  Viimeisin tiedonlähdettä käytettäessä tullut virheilmoitus.
- protected int **consecutiveFailures**
  Virhelaskuri, kasvaa aina kun palvelimeen ei saada yhteyttä ja nollautuu kun yhteys saadaan muodostettua.

Metodit :

- public abstract boolean **canHandle**(String URL)
  Kertoo pystyykö tämä DataSource käsittelemään parametrina annetun osoitteen mukaisia osoitteita.
- public abstract void **getNewData**(User user)
  Tätä kutsutaan ytimestä kun halutaan hakea uudet viestit.
- public String **getName**()
  Palauttaa tiedonlähteen luonnollisen nimen.
- public void **setName**(String newName)
  Asettaa tiedonlähteen luonnollisen nimen.
- public abstract String **getType**()
  Palauttaa tiedonlähteen tyypin, esim. "IMAP"
- public static DataSource **getDataSource**(String name)
  Palauttaa uuden halutuntyyppisen DataSource-olion.
- public boolean **isValid**()
  Palauttaa tiedon siitä, onko tiedonlähde kunnossa eli onko tietoa haettaessa tullut ongelmia.
- public String **getErrorMessage**()
  Palauttaa viimeksi tulleen tiedonlähteen virheilmoituksen, null jos ei virheitä.

- public void **setErrorMessage**(String err)
  Asettaa viimeksi tulleen tiedonlähteen virheilmoituksen, null jos ei virheitä.
- public abstract void **createXML**(Element parent)
  Katso 4.2.2 sivulla 64.
- public abstract boolean **loadXML**(Element myContent)
  Katso 4.2.2 sivulla 64.

5.1.2 IMAP

IMAP on DataSource-luokan aliluokka ja laajentaa DataSourcea toteuttamalla tuen IMAP-postilaatikoille.

Määrittely :

public class IMAP extends DataSource

Riippuvuudet :

- java.util.*
- javax.mail.*
- javax.mail.search.MessageIDTerm
- java.security.Security
- org.w3c.dom.*
- converge.kernel.Log
- webcore.HtmlUtils

Konstruktorit :

- public **IMAP**()
  Oletuskonstruktori, käytetään vain loadXML:n yhteydessä.

- public **IMAP**(String dataSourceName, String serverAddress, String serverLogin, String serverPassword)
  Luo uuden IMAP-tiedonlähteen nimeltään *dataSourceName* jonka osoite on *serverAddress* käyttäjätunnuksella *serverLogin* ja salasanalla *serverPassword*.

Kentät :

- protected String **serverAddress**
  Palvelimen osoite, esim. ml.mappi.helsinki.fi
- protected String **serverLogin**
  Käyttäjänimi.
- protected String **serverPassword**
  Salasana.
- protected Date **lastchecked**
  Milloin postit on viimeksi tarkistettu.
- protected boolean **useSSL**
  Käytetäänkö SSL:ää muodostettaessa yhteys IMAP-palvelimelle

Metodit :

- public void **getNewData**(User user)
  Tätä metodia kutsutaan ytimestä kun halutaan hakea uudet postit.
- public String **getType**()
  Palauttaa tiedonlähteen tyypin "IMAP"
- public PipeMessage **getAttachment**(String URL, boolean testOnly)
  Tätä metodia kutsutaan Attachment-luokasta kun haetaan liitetiedosto IMAP-postilaatikosta. Metodi yrittää hakea liitetiedoston osoitteesta *URL* ja palauttaa PipeMessagen jossa on tieto noutamisen tuloksista. Parametri *testOnly* asetetaan arvoon true kun halutaan vain tarkistaa onko liitetiedosto saatavilla.

- public static DataSource **getDataSource**(String name)
  Palauttaa uuden halutuntyyppisen DataSource-olion.
- public String **getErrormessage**()
  Kertoo tätä DataSourcea viimeksi käytettäessä tulleen virheilmoituksen.
- public String **getName**()
  Palauttaa IMAP-ilmentymän luonnollisen nimen, esim. "Yliopiston IMAP".
- public String **getServerAddress**()
  Palauttaa IMAP-palvelimen osoitteen.
- public String **getServerLogin**()
  Palauttaa IMAP-postilaatikon käyttäjätunnuksen.
- public boolean **isValid**()
  Kertoo onko DataSource käytettävissä (palvelinvirheet).
- public boolean **canHandle**(String URL)
  Kertoo onko parametrina annettu *URL* sellainen, jonka juuri tämä IMAP-olio voisi hakea.
- public void **createXML**(Element parent)
  Katso 4.2.2 sivulla 64.
- public boolean **loadXML**(Element myContent)
  Katso 4.2.2 sivulla 64.

5.1.3 IMAPMessageThread

IMAPMessageThread on säie, joka hakee uudet viestit IMAP-postilaatikosta. Jos viestejä haettaessa tulee virhetilanne, siitä lähetetään käyttäjälle järjestelmän sisäinen viesti.

Määrittely :

public class IMAPMessageThread extends Thread

Riippuvuudet :

- java.util.*
- javax.mail.*
- java.security.Security
- converge.kernel.Log

Konstruktori :

- public **IMAPMessageThread**(IMAP imap, User user)
  Luo uuden säikeen, joka hakee tietoa palvelimelta *imap* käyttäjälle *user*.

Kentät :

- private IMAP **imap**
  IMAP-olio, josta viestit haetaan.
- private User **user**
  Käyttäjä jolle noudetut viestit välitetään.

Metodi :

- public void **run**()
  Käynnistää säikeen ajamisen.

5.1.4 IMAPStreamDownloader

IMAPStreamDownloader on säie, joka lukee tietoa aiemmin avatusta lukijaolioon toiseen aiemmin avattuun kirjoittajaolioon kunnes luettavaa ei enää ole, ja sulkee lopuksi IMAP-postilaatikon.

Määrittely :

public class IMAPStreamDownloader extends Thread

Riippuvuudet :

- java.io.*

- javax.mail.*
- converge.kernel.Log

Konstruktori :

- public **IMAPStreamDownloader**(BufferedReader bufr, BufferedWriter bufw, InputStreamReader isr, OutputStreamWriter osw, InputStream is, OutputStream os, Folder f, Store s)
  Luo uuden säikeen joka lukee tiedot mainituista lukijaolioista ja kirjoittaa tiedot mainittuihin lukijaolioihin, ja sulkee lopuksi IMAP-kansion *f* ja IMAP-palvelinyhteyden *s*.

Metodi :

- public void **run**()
  Käynnistää säikeen ajamisen.

5.1.5 IMAPURLProperties

IMAPURLProperties on apuluokka, jonka avulla voidaan pilkkoa osiin ja tarkistaa IMAP-postilaatikkoihin osoittavat pointterimerkkijonot. Tämän luokan avulla voidaan helposti poimia pointterista esimerkiksi vain palvelimen osoite tai käyttäjätunnus.

Määrittely :

public class IMAPURLProperties

Konstruktori :

- protected **IMAPURLProperties**(String pointer)
  Tarkistaa liitetiedostopointterin *pointer* syntaksin ja lajittelee sen sisältämät tiedot IMAPURLProperties-luokan kentiksi.

Kentät :

- protected String **file**

  Tiedostonimi.

- protected String **msgid**

  Sen viestin Message-ID -kenttä, jossa tämä tiedosto on liitteenä.

- protected int **msgnum**

  Moniosaisen viestin osa, jossa liitetiedosto sijaitsee.

- protected String **protocol**

  Käytettävän protokollan tunnus, normaalisti siis "imap".

- protected String **server**

  IMAP-palvelimen osoite.

- protected String **username**

  IMAP-postilaatikon käyttäjätunnus.

5.1.6 Message

Message sisältää viestin järjestelmän sisäisessä muodossa.

Määrittely :

public class Message implements XMLContentInterface

Riippuvuudet :

- java.util.Date
- org.w3c.dom.*
- converge.attribute.*
- converge.attribute.attributeImplementations.*
- converge.kernel.Log
- converge.kernel.User

Konstruktorit :

- public **Message**(User user, Document doc)
  Muodostaa uuden Message-olion XML-dokumentista *doc* ja asettaa käyttäjäksi *user*.
- public **Message**(User user)
  Luo tyhjän Message-olion ja muodostaa sille uuden yksikäsitteisen viestitunnuksen, ja asettaa lopulta käyttäjäksi *user*.

Kentät :

- private User **user**
  Järjestelmän käyttäjä, jolle tämä viesti on tullut.
- private HashMap **attr**
  Viestiattribuuttien säilytystä varten.
- private HashSet **attrPerm**
  attr:n tiedot tallennetaan tietokantaan vain jos *attrPerm*:stä löytyy merkintä samalla nimellä.
- private Attachment[] **attachments**
  Tiedot viestiin liitetyistä liitetiedostoista.

Metodit :

- public User **getUser**()
  Palauttaa viestin sisältämän käyttäjäolion.
- public MessageAttributeInterface **getMessageAttribute**(String name)
  Palauttaa *name*-nimisen viestiattribuutin.
- public void **setMessageAttribute**(String name, Object value, boolean permanent)
  Asettaa *name*-nimisen viestiattribuutin arvoksi olion *value*, attribuutti tallennetaan myös kantaan jos *permanent* on true.
- public Attachment[] **getAttachmentList**()
  Palauttaa listan viestin mukana tulleista liitetiedostoista

- public void **addAttachment**(Attachment attachment)
  Lisää viestiin uuden liitetiedoston.
- public String **getMessageID**()
  Palauttaa Messagen yksikäsitteisen viestin tunnuksen.
- public User **getUser**()
  Palauttaa Message-olioon liitetyn User-olion.
- public void **addAttribute**(ContextModelValueAttribute cmAttr)
  Lisää viestiolioon ContextModelValueAttribuutin.
- public void **createXML**(Element parent)
  Katso 4.2.2 sivulla 64.
- public boolean **loadXML**(Element myContent)
  Katso 4.2.2 sivulla 64.

5.1.7 Email

Email on luokka, joka osaa muuntaa saamansa Internetin sähköpostiviestin järjestelmän sisäiseen muotoon.

Määrittely :

public class Email

Riippuvuudet :

- javax.mail.*
- java.util.*
- java.io.*
- com.sun.mail.imap.IMAPNestedMessage
- converge.kernel.Log

Metodit :

- public static Message **convertMessage**(User user, MimeMessage mm, String URLprefix)
  Muodostaa uuden Message-olion JavaMail-oliosta *mm* ja asettaa käyttäjäksi *user*. Palauttaa null jos viestin muodostaminen ei onnistu. *URLprefix*iä käytetään muodostettaessa liitetiedostojen osoitteita.
- public static PipeMessage **getAttachment**(Part p, String shouldBeFileName, Folder folder, Store store)
  Noutaa liitetiedoston viestin osasta *p*, jonka tiedostonimi pitäisi olla *shouldBeFileName*. Viesti sijaitsee kansiossa *folder* ja IMAP-palvelimella *store*. Palauttaa *PipeMessage*-olion, josta löytyy tieto liitetiedoston noutamisen onnistumisesta.
- public static void **handleMultiPart**(Multipart multipart, Message outmsg, String URLprefix)
  Käsittelee moniosaisen viestin *multipart* ja tallentaa sen sisältämät tiedot Convergen Message-olioon *outmsg*. *URLprefix*iä käytetään muodostettaessa liitetiedostojen osoitteita.
- public static void **handlePart**(Part part, Message outmsg, int partNum, String URLprefix)
  Käsittelee yhden viestin osan *part*, tai koko viestin jos osia on vain yksi. Tiedot tallennetaan Convergen Message-olioon *outmsg*. *URLprefix*iä ja viestin osanumeroa *partNum* käytetään muodostettaessa liitetiedostojen osoitteita.

5.1.8 HTTP

HTTP hakee liitetiedostojen tiedot WWW-sivuilta.

Määrittely :

public class HTTP

Riippuvuudet :

- java.net.HttpURLConnection
- java.net.URL
- java.util.Date
- javax.net.ssl.HttpsURLConnection
- converge.kernel.Log

Metodi :

- public static AttachmentAttributes **getAttachmentAttributes**(String URL)
  Tätä metodia kutsutaan Attachment-luokasta kun halutaan tietää liitetiedoston *URL* ominaisuudet.

5.1.9 Attachment

Kaikista liitetiedostoista tallennetaan tietokantaan ainoastaan liitetiedoston osoite. Jos viestin mukana tulee perinteinen liitetiedosto, se korvataan pointterilla IMAP-postilaatikkoon, jolloin URLiksi tulee esim. imap://kayttaja@palvelin/msgid/nro-tiedosto.txt.

Jos viestin liitetiedostona tulee määrätynmuotoinen osoitteen sisältämä tekstitiedosto, asetetaan viestin liitetiedoston osoitteeksi saapunut osoite. Tekstitiedosto noudattaa mm. Internet Explorerin käyttämää url-tiedostomuotoa.

Tässä esimerkki eräästä järjestelmän ymmärtämästä WWW-liitetiedostopointterista:

```
[DEFAULT]
BASEURL=http://news.google.com/
[InternetShortcut]
URL=http://news.google.com/
Modified=F0EFFF2169A3C201CE
```

Määrittely :

public class Attachment implements XMLContentInterface

Riippuvuudet :

- org.w3c.dom.*

Konstruktori :

- public **Attachment**(String URL)
  Muodostaa uuden *Attachment*-olion, jonka liitetiedosto on *URL*.

Kentät :

- private AttachmentAttributes **attributes**
  Viestin liitetiedoston ominaisuudet.
- private String **URL**
  Liitetiedoston osoite.

Metodit :

- public PipeMessage **getAttachment**(User user)
  Noutaa liitetiedoston. Liitetiedoston osoite on jo Attachment-olion tiedossa (URL). Palauttaa PipeMessagen joka pitää sisällään mahdollisen virheilmoituksen ja pipen josta liitetiedoston voi lukea. Käynnistää säikeen liitetiedoston hakemiseksi.
- public AttachmentAttributes **getAttachmentAttributes**()
  Palauttaa liitetiedoston ominaisuudet *AttachmentAttributes* -oliona.
- public void **setAttachmentAttributes**(AttachmentAttributes aa)
  Asettaa liitetiedostolle AttachmentAttributes-olion *aa*.
- public boolean **refreshAttachmentAttributes**(User user)
  Tarkistaa ovatko liitetiedoston tiedot muuttuneet ja päivittää attributes-arvot.
- public String **getPointerAddress**()
  Palauttaa liitetiedoston osoitteen.
- public void **createXML**(Element parent)
  Katso 4.2.2 sivulla 64.

- public boolean **loadXML**(Element myContent)
  Katso 4.2.2 sivulla 64

5.1.10 AttachmentAttributes

Tietovarasto liitetiedoston ominaisuuksille.

Määrittely :

public class AttachmentAttributes implements XMLContentInterface

Riippuvuudet :

- java.util.Date
- org.w3c.dom.*

Konstruktorit :

- public **AttachmentAttributes**()
  Oletuskonstruktori.
- public **AttachmentAttributes**(String error)
  Asettaa virheilmoitustekstiksi merkkijonon *error*.

Kentät :

- private boolean **available**
  Onko liitetiedosto saatavilla (jos esim. palvelinongelmia).
- private long **size**
  Liitetiedoston koko, -1 jos tuntematon.
- private Date **date**
  Liitetiedoston viimeisimmän muokkauksen päivämäärä.
- private String **MIMEtype**
  Liitetiedoston MIME-tyyppi.

- private String **description**
  Liitetiedoston sanallinen kuvaus.
- private String **filename**
  Liitetiedoston tiedostonimi.
- private String **errorMessage**
  Liitetiedostoa haettaessa tullut virheilmoitus.

Metodit :

- public boolean **getAvailable**()
  Palauttaa liitetiedoston saatavuustiedon.
- public void **setAvailable**(boolean available)
  Asettaa liitetiedoston saatavuustiedon.
- public long **getSize**()
  Palauttaa liitetiedoston koon.
- public void **setSize**(long size)
  Asettaa liitetiedoston koon.
- public Date **getDate**()
  Palauttaa liitetiedoston viimeisimmän muokkauspäivämäärän.
- public void **setDate**(Date date)
  Asettaa liitetiedoston viimeisimmän muokkauspäivämäärän.
- public String **getMIMEType**()
  Palauttaa liitetiedoston MIME-tyypin.
- public void **setMIMEType**(String MIMEtype)
  Asettaa liitetiedoston MIME-tyypin.
- public String **getDescription**()
  Palauttaa liitetiedoston sanallisen kuvauksen.
- public void **setDescription**(String description)
  Asettaa liitetiedoston sanallisen kuvauksen.

- public String **getFilename**()
  Palauttaa liitetiedoston tiedostonimen.
- public void **setFilename**(String filename)
  Asettaa liitetiedoston tiedostonimen.
- public String **getErrorMessage**()
  Palauttaa liitetiedostoa haettaessa tulleen virheilmoituksen.
- public void **setErrorMessage**()
  Asettaa liitetiedostoa haettaessa tulleen virheilmoituksen.
- public

5.1.11 PipeMessage

Tietovarasto liitetiedoston noudossa tulleille virheilmoituksille. Sisältää joko tiedostokahvan liitetiedoston lukemiseksi tai virheilmoituksen.

Määrittely :

public class AttachmentAttributes

Riippuvuudet :

- java.io.PipedOutputStream
- java.io.PipedInputStream

Konstruktorit :

- public **PipeMessage**(PipedOutputStream pipe, String reply)
  Luo uuden PipedInputStreamin ja yhdistää sen parametrina saatuun *pipe*en. Asettaa ilmoitustekstiksi tekstin *reply*. public **PipeMessage**(String reply)
  Virheilmoituskonstruktori. Asettaa ilmoitustekstiksi tekstin *reply*.

Kentät :

- private PipedInputStream **pipe**
  Putken toinen pää.
- private String **reply**
  Virhe- tai muu ilmoitus

Metodit :

- public String **getReply**()
  Palauttaa mahdollisen virheilmoituksen.
- public PipedInputStream **getPipe**()
  Palauttaa siirtoputken toisen pään.

5.1.12 StreamDownloader

StreamDownloader on säie, joka lukee tietoa aiemmin avatusta lukijaolioon toiseen aiemmin avattuun kirjoittajaolioon kunnes luettavaa ei enää ole. Tämä on yleinen luokka, jota voidaan käyttää erilaisten datavirtojen lukemiseen erilaisista tiedonlähteistä.

Määrittely :

public class StreamDownloader extends Thread

Riippuvuudet :

- java.io.*
- converge.kernel.Log

Konstruktori :

- public **StreamDownloader**(BufferedReader bufr, BufferedWriter bufw, InputStreamReader isr, OutputStreamWriter osw, InputStream is, OutputStream os)
  Luo uuden säikeen joka lukee tiedot mainituista lukijaolioista ja kirjoittaa tiedot mainittuihin lukijaolioihin.

Metodi :

- public void **run**()
  Käynnistää säikeen ajamisen.

5.1.13 SMTP

SMTP hoitaa sähköpostiviestien lähettämisen.

Määrittely :

public class SMTP

Riippuvuudet :

- javax.mail.*
- javax.mail.internet.*
- java.util.*
- converge.kernel.Log

Metodit :

- public static boolean **sendMail**(String from, String to, String subject, String message, User user)
  Lähettää sähköpostiviestin annetuilla parametreilla. To-kenttä voi olla lista vastaanottajista pilkuilla erotettuna. User-oliota tarvitaan jos joudutaan lähettämään virheilmoituksia käyttäjälle.
- public static void **sendErrorMessage**(User user, String body)
  Apumetodi virheilmoitusten lähettämiseksi.

5.1.14 ConvergeSSLSocketFactory

ConvergeSSLSocketFactory luo SSL-suojattuja verkkosocketeja. Näitä tarvitaan haettaessa viestejä SSL-suojatulta IMAP-palvelimelta tai SSL-suojatulta WWW-palvelimelta.

Määrittely :

public class ConvergeSSLSocketFactory

Riippuvuudet :

- javax.net.ssl.*
- java.io.IOException
- java.net.*
- javax.net.*

Konstruktorit :

- public **ConvergeSSLSocketFactory**()
  Oletuskonstruktori. Asettaa SSLContextin käyttämään ConvergeTrustManageria sertifikaattien hallitsemiseen.

Metodit :

- public static SocketFactory **getDefault**()
  Palauttaa uuden SSL Socket Factoryn.
- public Socket **createSocket**(Socket socket, String server, int port, boolean autoClose)
  Luo uuden socketin. *socket* on jo olemassaoleva socket, *server* on palvelimen osoite, *port* on palvelimen portti, *autoClose* kertoo suljetaanko allaoleva socket kun tämä socket suljetaan.
- public Socket **createSocket**(InetAddress inaddr1, int port1, InetAddress inaddr2, int port2)
  Luo uuden socketin. *inaddr1* on palvelimen osoite, *port1* on palvelimen portti, *inaddr2* on oman koneen osoite, *port2* on oman koneen portti.
- public Socket **createSocket**(InetAddress inaddr, int port)
  Luo uuden socketin. *inaddr* on palvelimen osoite, *port* on palvelimen portti.

- public Socket **createSocket**(String server, int port1, InetAddress inaddr, int port2)
  Luo uuden socketin. *server* on palvelimen osoite, *port1* on palvelimen portti, *inaddr* on oman koneen osoite, *port2* on oman koneen portti.
- public Socket **createSocket**(String server, int port)
  Luo uuden socketin. *server* on palvelimen osoite, *port* on palvelimen portti.
- public String[] **getDefaultCipherSuites**()
  Palauttaa listan oletuksena käytetyistä salakirjoitusmenetelmistä.
- public String[] **getSupportedCipherSuites**()
  Palauttaa listan tuetuista salakirjoitusmenetelmistä.

5.1.15 ConvergeTrustManager

Yksinkertainen luokka, joka hyväksyy kaikki annetut turvasertifikaatit. Tätä voisi joskus tulevaisuudessa virittää turvallisemmaksikin eli toteuttaa puuttuvat metodit. Tämä taso riittää kuitenkin prototyyppiin.

Määrittely :

public class ConvergeTrustManager implements X509TrustManager

Riippuvuudet :

- javax.net.ssl.X509TrustManager
- java.security.cert.X509Certificate

Metodit :

- public boolean **isClientTrusted**(X509Certificate[] cert)
  Tarkistaa onko palvelimen sertifikaatti *cert* oikea. Tätä metodia ei tarvittane enää, checkClientTrusted hoitaa asian J2SE 1.4:stä lähtien.

- public boolean **isServerTrusted**(X509Certificate[] cert)
  Tarkistaa onko palvelimen sertifikaatti *cert* oikea. Tätä metodia ei tarvittane enää, checkClientTrusted hoitaa asian J2SE 1.4:stä lähtien.
- public X509Certificate[] **getAcceptedIssuers**()
  Palauttaa tyhjän taulukon tiedetyistä sertifikaattien julkaisijoista.
- public void **checkClientTrusted**(X509Certificate[] cert, String auth)
  Tarkistaa onko asiakasohjelman sertifikaatti oikea.
- public void **checkServerTrusted**(X509Certificate[] cert, String auth)
  Tarkistaa onko palvelimen sertifikaatti oikea.

5.2 Rajapinta järjestelmän ytimelle

Tiedonhakumoduulin järjestelmän ytimelle tarjoamat rajapinnat löytyvät seuraavista luokista:

- 5.1.6 **Message**, s. 82
- 5.1.9 **Attachment**, s. 86
- 5.1.1 **DataSource**, s. 75

5.3 Rajapinta asiakasmoduulille

Tiedonhakumoduulin asiakasmoduulille tarjoamat rajapinnat löytyvät seuraavista luokista:

- 5.1.6 **Message**, s. 82
- 5.1.9 **Attachment**, s. 86
- 5.1.13 **SMTP**, s. 92
- 5.1.1 **DataSource**, s. 75

- 5.1.11 **PipeMessage**, s. 90

6 Asiakasmoduuli

Asiakasmoduuli toimii välittäjänä järjestelmän ytimen ja erilaisten asiakasohjelmien tai päätelaitteiden välillä. Jokaista tuettua asiakasohjelmatyyppiä kohden on asiakasmoduulissa oma komponentti, joka osaa kaikki tarvittavat protokollat joiden avulla kyseisen ohjelman tai päätelaitteen kanssa voidaan kommunikoida.

Ohjelmointirajapinta jota vasten uudentyyppisiä asiakasohjelmaluokkia voidaan rakentaa, koostuu ClientHandler -yläluokasta ja ClientManager-luokan tarjoamista metodeista.

Huomautus: asiakasmoduulin ne osat, jotka toteuttavat tämän prototyypin ainoan asiakasohjelmatyypin, WWW-asiakasohjelman, on kuvattu seuraavassa luvussa.

6.1 Luokat

Pakkaus *converge.clientmanager*

6.1.1 ClientManager

Järjestelmässä on yksi ClientManager-ilmentymä, jonka tehtävä (tässä prototyypissä) on mm. pitää kirjaa aktiivisista käyttäjistä, hoitaa sessionhallintaa ja välittää User-olioita asiakasohjelmaluokille.

Ytimelle tarjottavat palvelut

- public void **start**()
  Aiheuttaa erilaisten asiakasrajapintojen käynnistymisen (toteutettavassa prototyypissä koskee HTTP-palvelinta)

- public void **shutdown**()
  Aiheuttaa erilaisten asiakasrajapintojen sulkemisen (toteutettavassa prototyypissä koskee HTTP-palvelinta)

- public void **sendMessage**(Message message)
  Parametrina saatu *message* lähetetään sille päätelaitteelle jota vastaanottaja sillä hetkellä käyttää. Päätelaitteiden / asiakasohjelmien erilaisista luonteista johtuen tämän metodin kutsuminen ei takaa että viesti menee perille. (Vastaanottaja tunnetaan, sillä Message-olio sisältää viitteen User-olioon.)

Erityyppisille asiakasohjelmaluokille tarjottavat palvelut

- public synchronized User **getUserBySessionId**(long sessionId)
  Palauttaa annettua *sessionId*:tä vastaavan User-olion. Jos *sessionId* ei ole validi (esimerkiksi istunto on vanhentunut), metodi palauttaa null-arvon.

- public synchronized long **login**(String username, String password, boolean createNew, boolean group)
  Tätä metodia kutsutaan sisäänkirjautumisen yhteydessä. Jos annetut *username* ja *password* ovat validit, palautetaan long-tyyppiä oleva session-id. Uuden käyttäjän luonti ja ryhmäkirjautuminen (eli kirjaudutaan sisään muuttamaan käyttäjäryhmän asetuksia) tapahtuvat antamalla sopivat *createNew* ja *group* -parametrit.

- public synchronized void **logout**(long sessionId)
  Asiakasohjelmaluokat kutsuvat tätä metodia, jos käyttäjä haluaa eksplisiittisesti kirjautua ulos palvelusta. Parametrina annettu *sessionId* ja sitä vastaava User-olio poistetaan ClientManagerin sisäisestä tietorakenteesta. (Eli jos samalla *sessionId*:llä kutsuttaisiin tämän jälkeen getUserBySessionId-metodia, palauttaisi se arvon null.)

Muut metodit

- public static synchronized ClientManager **getInstance**()
  Palauttaa järjestelmän ainoan ClientManager-olion.

6.1.2 ClientHandler

ClientHandler on abstrakti yläluokka kaikentyyppisten asiakasohjelmien käsittelijöille. Siinä on määritelty joitakin metodeja, jotka ovat yhteisiä kaikille aliluokille, mutta jotka jokainen aliluokka toteuttaa omalla tavallaan.

Metodit

- public abstract void **start**()
  Käynnistää asiakasohjelmatyyppiin mahdollisesti liittyvän palvelimen, kuuntelijan tms.
- public abstract void **shutdown**()
  ClientHandlerin alasajo
- public abstract String **getType**()
  Palauttaa ClientHandler-olion tyypin
- public abstract void **sendMessage**(Message msg)
  Viestin lähetys kyseiseen asiakasohjelmaan / päätelaitteeseen

7 WWW-asiakasohjelma

Järjestelmän prototyypissä on WWW:ssä toimiva asiakasohjelma. Tämä ohjelma sisältää varsin rajoitetun määrän toiminnallisuutta, eikä esimerkiksi käyttöliittymän suunnittelussa ole nähty siinä määrin vaivaa kuin ”oikean” asiakasohjelman kohdalla pitäisi tehdä. Sen avulla on kuitenkin mahdollista käyttää järjestelmän yleisimpiä toimintoja.

Asiakasmoduuliin kuuluu Javalla tehty palvelin, joka kuuntelee jotakin TCP/IP -porttia ja ymmärtää HTTP-protokollaa. Asiakasohjelmana toimii siis oikeastaan WWW-selain, jolla käyttäjä voi suoraan ottaa yhteyden järjestelmään. Tässä luvussa ei käsitellä varsinaista asiakasohjelmaa (WWW-selainta), vaan esitellään ne järjestelmän (asiakasmoduulin) osat, jotka mahdollistavat WWW-selaimen käyttämisen asiakasohjelmana.

HTTP-palvelimen toiminnallisuutta ei toteuteta itse, vaan prototyypissä käytetään valmista **webcore**[1]-nimistä open source Java-palvelinta.

7.1 Luokat

7.1.1 WebClientHandler

WebClientHandler on ClientHandler-luokan aliluokka. Tämä on prototyypin ainoa luokka joka mahdollistaa järjestelmän käytön jollain tietyllä päätelaiteella / asiakasohjelmalla, tässä tapauksessa WWW-selaimella.

WebClientHandlerin ainoa tehtävä tässä prototyypissä on webcore-palvelimen käynnistäminen ja sulkeminen; se tarjoaa ytimelle metodit näihin tarkoituksiin. Luokan sendMessage-metodi ei tee tässä tapauksessa mitään. (WWW-selaimelle ei voida push-tyyppisesti lähettää viestejä, vaan kaikki toiminta tapahtuu selaimen aloitteesta.)

Metodit

- public void **start**()
  Käynnistää webcore-palvelimen
- public void **shutdown**()
  Sulkee webcore-palvelimen

[1]Lisätietoja osoitteesta http://www.hunter-lovell.org/webcore/doc/overview.html

- public static synchronized WebClientHandler **getInstance**()
  Palauttaa järjestelmän ainoan WebClientHandler-olion.

7.1.2 HTML:ää tuottavat webcore-luokat (pakkaus *webcore*)

Nämä luokat huolehtivat tietynlaisten HTML-sivujen luomisesta ja niiden palauttamisesta WWW-selaimelle vastauksena HTTP-pyyntöön. Suurin osa asiakasmoduulin toiminnallisuudesta löytyy webcore-pakkauksen luokista.

Periaatteessa kaiken toiminnallisuuden ja HTML:n generoinnin voisi tehdä esimerkiksi vain yhdessä luokassa; toiminnallisuus on jaettu useampaan luokkaan lähinnä selkeyden vuoksi. Kukin luokka vastaa periaatteessa käyttöliittymän yhtä ”näkymää”.

Lista webcore-luokista

Login - sisään- ja uloskirjautumiseen liittyvät asiat, etusivu
Main - sisäänkirjautumisen jälkeinen pääsivu, lista kaikista näkymistä
View - yhden näkymän sisältö
Message - yhden viestin näyttäminen
Attachment - liitetiedoston välittäminen käyttäjälle
Settings - asetusten muokkaaminen (sisältää tässä prototyypissä vain IMAP-palvelinten asetukset, käyttöliittymässä kohta "DataSources")
ContextModel - kontekstimallien luominen, muokkaaminen ja poistaminen
Profile - profiilien ja niihin liittyvien sääntöjen luominen, muokkaaminen ja poistaminen
ContactList - kontaktilistan muokkaaminen
Write - viestin kirjoittaminen ja lähetys

Kaikki edellämainitut luokat toteuttavat webcore-järjestelmän WebRequestable -rajapinnan. Luokassa on rajapinnan vaatima metodi

public void **processRequest**(HTTPRequest httpRequest)

joka saa parametrina HTTP-pyynnön, ja jonka tehtävänä on tuottaa selaimelle vastauksena lähetettävä HTML-sivu. Lisäksi luokissa on mahdollisesti omia apumetodeja; esimerkiksi Login-luokassa sisäänkirjautumislomakkeen tulostamista varten on pieni apumetodi.

webcore-luokkien toiminta Sessionhallinta hoidetaan HTTP-”keksien” (cookie) ja yksilöllisten session-id -tunnusten avulla. Sisäänkirjautumisen jälkeen jokainen selaimelta tuleva HTTP-pyyntö sisältää session-id:n, ja pyynnön käsittelevä webcore-luokka kutsuu ClientManagerin getUserBySessionId -metodia, joka palauttaa session-id:tä vastaavan User-olion. Tämän jälkeen kyseinen webcore-luokka voi suoraan käyttää User-olion metodeja esimerkiksi listatakseen käyttäjän tietyn näkymän viestit, tai tehdäkseen käyttäjän pyytämiä muutoksia johonkin profiiliin tai kontekstimalliin.

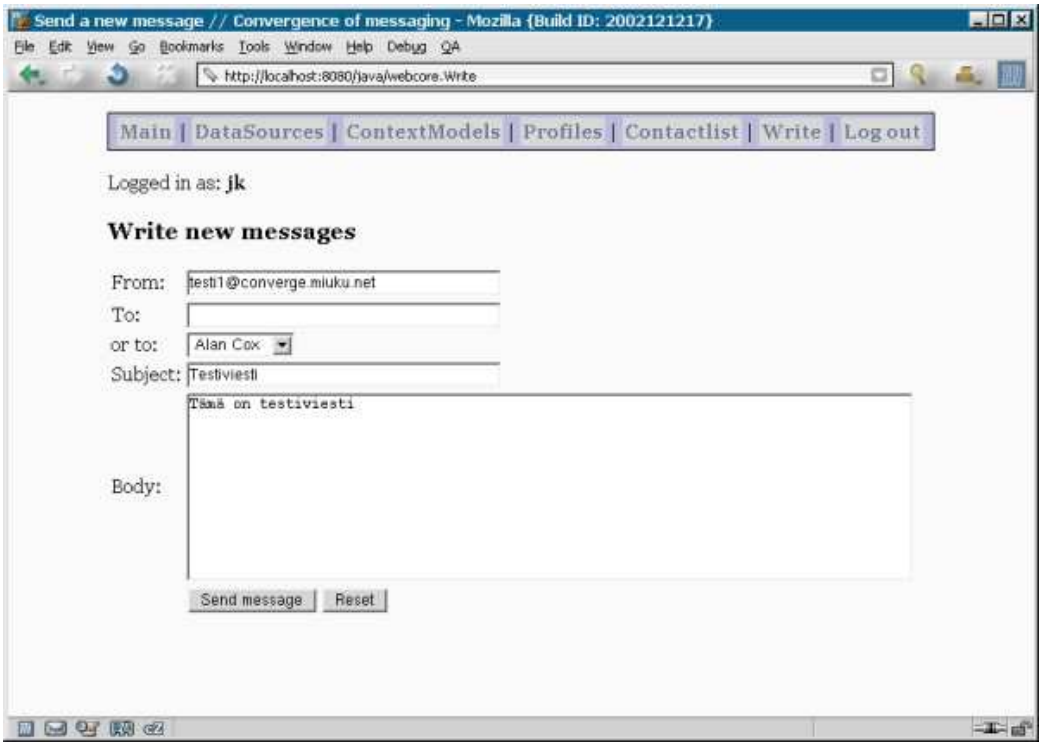
Login -luokka toimii muista poiketen siten, että se saa käyttäjältä käyttäjänimen ja salasanan, jotka välitetään ClientManagerin login-metodille, ja näin saadaan viite User-olioon. Onnistuneen sisäänkirjautumisen yhteydessä Login asettaa selaimelle session-id -”keksin”, jonka avulla tunnistus ja sessionhallinta jatkossa tapahtuu.

Useimmissa tapauksissa sivulla oleva HTML-lomake lähetetään samalle sivulle, toisin sanoen sama luokka tuottaa HTML:n joka sisältää lomakkeen, sekä koodin joka käsittelee ko. lomakkeen kun käyttäjä on syöttänyt siihen selaimella tietoja.

Webcore-luokkien toiminta selviää parhaiten itse lähdekoodia tutkimalla. Kannattaa kuitenkin muistaa, että ne toteuttavat vain yhden (alkeellisen) asiakasohjelman (joka tässä tapauksessa toimii HTTP- ja HTML -tekniikoiden avulla), eikä niiden merkitys kontekstisensitiivisen viestintäjärjestelmän kannalta ole kovin suuri.

Lisäksi webcore-pakkauksessa on luokka **HtmlUtils**, johon on koottu useissa webcore-luokissa tarvittavia apumetodeja. Näihin kuuluu mm. metodeja jotka tekevät erilaisia usein tarvittavia HTML-tulostuksia, luovat HTTP-protokollan otsikkotietoja (mm. ”keksien” asettamista varten) ja käsittelevät merkkijonoja eri tavoilla (esim. metodi joka muuttaa välilyönnit ja rivinvaihdot alaviivoiksi). Tarkat kuvaukset

apumetodeista on Javadoc-dokumentaatiossa.



Kuva 4: WWW-käyttöliittymä

A AttributeInterface -rajapinnan toteutusesimerkkejä

A.1 AttributeString

Attribute-luokan aliluokka, joka osaa käsitellä merkkijonoja.

Määrittely :

public class AttributeString
extends Attribute
implements converge.attribute.MessageAttributeInterface,
converge.attribute.ContextModelAttributeInterface

Konstruktorit :

- public **AttributeString**(String name, String value)

Metodit :

- boolean **compare**(String ExpressionType, Object value)
  Katso 4.3.3 sivulla 66
- boolean **compare**(Message msg)
  Katso 4.3.4 sivulla 67
- String[] **supportedAttributes**()
  Katso 4.3.2 sivulla 65
- void **createXML**(Element parent)
  Katso 4.2.2 sivulla 64.
- boolean **loadXML**(Element myContent)
  Katso 4.2.2 sivulla 64

A.2 AttributeDate

Attribute-luokan aliluokka, joka osaa käsitellä päivämääriä (esim. viestin lähetysaika).

Määrittely :

public class AttributeDate
extends Attribute
implements converge.attribute.MessageAttributeInterface,
converge.attribute.ContextModelAttributeInterface

Konstruktorit :

- public **AttributeDate**(String name, Date value)

Metodit :

- boolean **compare**(String ExpressionType, Object value)
  Katso 4.3.3 sivulla 66
- boolean **compare**(Message msg)
  Katso 4.3.4 sivulla 67
- String[] **supportedAttributes**()
  Katso 4.3.2 sivulla 65
- void **createXML**(Element parent)
  Katso 4.2.2 sivulla 64.
- boolean **loadXML**(Element myContent)
  Katso 4.2.2 sivulla 64

A.3 AttributePriority

Attribute-luokan aliluokka, joka osaa käsitellä viestin mukana tullutta Priority-arvoa (normal, high, low).

Määrittely :

public class AttributePriority
extends Attribute

implements converge.attribute.MessageAttributeInterface,
converge.attribute.ContextModelAttributeInterface

Konstruktorit :

- public **AttributePriority**(String name, String value)

Metodit :

- boolean **compare**(String ExpressionType, Object value)
  Katso 4.3.3 sivulla 66
- boolean **compare**(Message msg)
  Katso 4.3.4 sivulla 67
- String[] **supportedAttributes**()
  Katso 4.3.2 sivulla 65
- void **createXML**(Element parent)
  Katso 4.2.2 sivulla 64.
- boolean **loadXML**(Element myContent)
  Katso 4.2.2 sivulla 64

A.4 AttributePrecedence

Attribute-luokan aliluokka, joka osaa käsitellä viestin mukana tullutta Precedence-arvoa (bulk, normal, priority ym.)

Määrittely :

public class AttributePrecedence
extends Attribute
implements converge.attribute.MessageAttributeInterface,
converge.attribute.ContextModelAttributeInterface

Konstruktorit :

- public **AttributePrecedence**(String name, String value)

Metodit :

- boolean **compare**(String ExpressionType, Object value)
  Katso 4.3.3 sivulla 66
- boolean **compare**(Message msg)
  Katso 4.3.4 sivulla 67
- String[] **supportedAttributes**()
  Katso 4.3.2 sivulla 65
- void **createXML**(Element parent)
  Katso 4.2.2 sivulla 64.
- boolean **loadXML**(Element myContent)
  Katso 4.2.2 sivulla 64

A.5 ContextModelValueAttribute

ContextModelValueAttribute on attribuutin toteutusluokka, joka on tarkoitettu viestiolioon liitettäväksi (implements MessageAttributeInterface) ja sen toiminnallisuus on tarkoitettu käsittelemään kontekstimalliolion itselleen määrittelemää arvoa, joka kuvastaa prosentuaalisesti (kokonaislukuna 0-100) vertailun onnistumista.

Määrittely :

class ContextModelValueAttribute implements MessageAttributeInterface

Konstruktorit :

- **ContextModelValueAttribute**(int value)

Metodit :

- void **setName**(String name)
  Katso 4.3.2, sivulla 65.
- String **getName**()
  Katso 4.3.2, sivulla 65.
- void **setValue**(Object value)
  Katso 4.3.2, sivulla 65. Luokka hyväksyy syötteekseen (Integer) kokonaislukuarvoja väliltä 0 - 100, millä ilmaistaan kontekstimallin itselleen saamaa arvoa prosentteina.
- String[] **supportedAttributes**()
  Katso 4.3.2, sivulla 65. Luokan toteutus on käyttökelpoinen vain kontekstimallien luodessa itsestään arvoilmauksen, ja koska käyttäjän kontekstimalleille antamia nimiä ei voida tietää etukäteen, palauttaa tämä metodi tyhjän String-taulukon.
- boolean **compare**(Object value, String expressionType)
  Katso 4.3.3, sivulla 66. *Value*-parametrina hyväksytään vain kokonaislukuarvoja (Integer) väliltä 0-100.
- Object **getValue**()
  Katso 4.3.3, sivulla 66. Palauttaa ilmentymän arvotietosisällön Integer-tyyppisenä kokonaislukuarvona.
- void **createXML**(Element parent)
  Katso 4.2.2, sivulla 64.
- boolean **loadXML**(Element myContent)
  Katso 4.2.2, sivulla 64.

A.6 AttributeStringArray

AttributeStringArray-luokka on toteutus luokka viestiolioon liitettävillä (implements MessageAttributeInterface) attribuuteille, joiden arvotietosisältö voi olla moniosainen tekstiarvo.

Määrittely :

class AttributeStringArray implements MessageAttributeInterface

Konstruktorit :

- **StringArrayImplementation**(String[] value)

Metodit :

- void **setName**(String name)
  Katso 4.3.2, sivulla 65.
- String **getName**()
  Katso 4.3.2, sivulla 65.
- void **setValue**(Object value)
  Katso 4.3.2, sivulla 65. Hyväksyttävä syöte on String-tyyppinen lista.
- String[] **supportedAttributes**()
  Katso 4.3.2, sivulla 65.
- boolean **compare**(Object value, String expressionType)
  Katso 4.3.3, sivulla 66. *Value*-parametrina hyväksytään String-tyyppinen taulukko tai yksi String-tyyppinen olio.
- Object **getValue**()
  Katso 4.3.3, sivulla 66. Palauttaa ilmentymän tietosisällön String-tyyppisenä taulukkona.
- void **createXML**(Element parent)
  Katso 4.2.2, sivulla 64.
- boolean **loadXML**(Element myContent)
  Katso 4.2.2, sivulla 64.

B UserPropertyInterface -rajapinnan toteutusesimerkkejä

B.1 TimeProperty

TimeProperty-luokassa toteutetaan vertailu vuorokauden tunteihin (0-23).

Määrittely :

public class TimeProperty implements UserPropertyInterface

Metodit :

- public boolean**compare**(String expressionType, Object value)
  Metodi jonka avulla voidaan vertailla annettavaa arvoa ilmentymän sisältämään arvotietosisältöön. Tämä toteutus tukee vertailua kulloiseenkin ajanhetkeen (tuntiin) *expressionType*-parametrin ollessa jokin seuraavista: =, !=, < tai >. *value*-parametrissä annetaan vertailtava tunti numerona String-olioksi muutettuna.
- public void **setUser**(User user)
  Asettaa ilmentymään liitetyn käyttäjäolion
- public void **setName**(String name)
  Asetetaan ilmentymän nimi.
- public String[] **supportedAttributes**()
  String-tyyppinen taulukko niiden kaikkien attribuuttien loogisista nimistä joille toteutettu luokka tarjoaa käsittelyn toteutuksen. Tämä toteutus palauttaa taulukossa vain arvo ”Hour”
- public String **getDescription**()
  Palautetaan luokan selkokielinen kuvaus.

C ActionInterface -rajapinnan toteutusesimerkkejä

C.1 ActionActivateProfile

ActionActivateProfile-luokka toteuttaa profiilien aktivoimiseen liittyviä toimintoja.

Määrittely :

public class ActionActivateProfile implements ActionInterface

Riippuvuudet :

- converge.kernel.ActionInterface
- converge.kernel.User

Kentät :

- private String[] **supports**
  Taulukko tämä toteutuksen tukemista toiminnoista: Activate-Profile, DeActivate-Profile ja Toggle-Profile-Activation.
- private String **description**
  Luokan selkokielinen kuvaus
- private User **user**
  Viite käyttäjäolioon
- private String[] **conf**
  Toiminnon parametrit

Metodit :

- public void **setValue**(Object ob, String[] values)
  Asetetaan toiminnan suorittamiseksi arvoja, missä toiminta liitetään käyttäjään (User) arvotaulukon *values* sisältämien arvojen avulla. Tässä toteutusluokassa taulukko pitää sisällään profiilien nimiä.

- public void **doAction**(String action)
  Metodi laukaisee ilmentymän toiminnan. Se mitä luokan tarjoamista toiminnallisuuksista annetuilla arvoilla halutaan laukaista määritetään *action*-parametrilla.
- public String[] **supportedActions**()
  String-tyyppinen taulukko toiminnallisuuksista, joiden toteutuksen luokka tarjoaa. Taulukon sisältämät määritykset ovat *doAction*-metodin hyväksymiä syötteitä.
- public String **getDescription**()
  Palautetaan luokan selkokielinen kuvaus.
- public boolean **supportsScheduledRule**(String name)
  Tukeeko luokan toteuttama toiminto *name* ajastettujen sääntöjen toiminnan käsittelyä.
- public boolean supportsRule(String name) Tukeeko luokan toteuttama toiminto *name* tavallisten sääntöjen toiminnan käsittelyä. Tässä toteutusluokassa palautetaan aina false.

C.2 ActionGetMessages

ActionGetMessages-luokka toteuttaa viestien hakemiseen liittyvää toiminnallisuutta.

Määrittely :

public class ActionGetMessages implements ActionInterface

Riippuvuudet :

- converge.kernel.ActionInterface
- converge.kernel.User

- import converge.service.*

Kentät :

- private String[] **supports**
  Taulukko tämä toteutuksen tukemista toiminnoista: Get-Message.
- private String **description**
  Luokan selkokielinen kuvaus
- private User **user**
  Viite käyttäjäolioon
- private String[] **conf**
  Toiminnon parametrit

Metodit :

- public void **setValue**(Object ob, String[] values)
  Asetetaan toiminnan suorittamiseksi arvoja, missä toiminta liitetään käyttäjään (User) arvotaulukon *values* sisältämien arvojen avulla. Tässä toteutusluokassa taulukko pitää sisällään tiedonlähteiden nimiä.
- public void **doAction**(String action)
  Metodi laukaisee ilmentymän toiminnan. Se mitä luokan tarjoamista toiminnallisuuksista annetuilla arvoilla halutaan laukaista määritetään *action*-parametrilla.
- public String[] **supportedActions**()
  String-tyyppinen taulukko toiminnallisuuksista, joiden toteutuksen luokka tarjoaa. Taulukon sisältämät määritykset ovat *doAction*-metodin hyväksymiä syötteitä.
- public String **getDescription**()
  Palautetaan luokan selkokielinen kuvaus.
- public boolean **supportsScheduledRule**(String name)

Tukeeko luokan toteuttama toiminto *name* ajastettujen sääntöjen toiminnan käsittelyä.

- public boolean supportsRule(String name) Tukeeko luokan toteuttama toiminto *name* tavallisten sääntöjen toiminnan käsittelyä. Tässä toteutusluokassa palautetaan aina false.

C.3 ActionIncomingMessage

Tämä ActionInterface-rajapinnan toteutus liittää järjestelmän sisään tulleen viestin johonkin näkymään.

Määrittely :

public class ActionIncomingMessage implements ActionInterface

Riippuvuudet :

- converge.kernel.ActionInterface
- converge.kernel.*
- import converge.service.*

Kentät :

- private String[] **supports**
  Taulukko tämä toteutuksen tukemista toiminnoista: Set-To-View.
- private String **description**
  Luokan selkokielinen kuvaus
- private Message **msg**
  Viite viestiolioon
- private String[] **conf**
  Toiminnon parametrit

Metodit :

- public void **setValue**(Object ob, String[] values)
  Asetetaan toiminnan suorittamiseksi arvoja, missä toiminta liitetään viestiin (Message) arvotaulukon *values* sisältämien arvojen avulla. Tässä toteutusluokassa taulukko pitää sisällään näkymien nimiä.
- public void **doAction**(String action)
  Metodi laukaisee ilmentymän toiminnan. Se mitä luokan tarjoamista toiminnalisuuksista annetuilla arvoilla halutaan laukaista määritetään *action*-parametrilla.
- public String[] **supportedActions**()
  String-tyyppinen taulukko toiminnallisuuksista, joiden toteutuksen luokka tarjoaa. Taulukon sisältämät määritykset ovat *doAction*-metodin hyväksymiä syötteitä.
- public String **getDescription**()
  Palautetaan luokan selkokielinen kuvaus.
- public boolean **supportsScheduledRule**(String name)
  Tukeeko luokan toteuttama toiminto *name* ajastettujen sääntöjen toiminnan käsittelyä. Tämä toteutusluokka palauttaa aina false.
- public boolean supportsRule(String name) Tukeeko luokan toteuttama toiminto *name* tavallisten sääntöjen toiminnan käsittelyä.

D Avoimeksi jääneitä kysymyksiä ja muita lyhyitä kommentteja

- ActionInterface-rajapinnan voisi jakaa kahtia, jolloin ajastettujen sääntöjen toiminnot olisivat eroteltu muiden sääntöjen toiminnoista.
- User-luokkaan voisi getProperty-metodin rinnalle lisätä myös setProperty-metodin.

E Asennus ja ylläpito

E.1 Toimintaympäristö

Prototyypissä käytetään seuraavia ohjelmointikieliä, ohjelmistoja, ym. tekniikoita:

Linux 2.4.18
Java SDK, versio 1.4.1
Xindice, versio 1.0 (pienin omin korjauksin)
Drools, versio 2.0 beta7
WWW-käyttöliittymä toimii millä tahansa HTML- ja CSS-standardeja kohtuullisen hyvin tukevalla selaimella. (Esimerkiksi Mozilla 1.0 (tai uudempi) ja MS Internet Explorer 6 kelpaavat hyvin. Sen sijaan esim. Netscapen versiolla 4.x käyttöliittymä ei välttämättä toimi oikein.) Selaimen asetuksissa on oltava ”keksit” eli cookiet päällä ja proxy-palvelinten käyttö pois päältä.

E.2 Järjestelmän asennus ja käyttöönotto

Ryhmän kotisivulta voi ladata järjestelmän lähdekoodit paketissa converge.tar.gz. Paketti puretaan komennolla `tar xvfz converge.tar.gz`, jolloin syntyy uusi hakemisto sources, jonka alta löytyvät Java SDK:ta lukuunottamatta kaikki tarpeelliset ohjelmakirjastot ja lähdekoodit.

E.2.1 Kääntäminen

Mikäli lähdekoodit halutaan kääntää ja käyttöympäristön `javac` -komento on Java SDK:n version 1.4.1:n mukainen, voidaan kääntäminen suorittaa sources-hakemistossa komennolla `./compile`

Tietojenkäsittelytieteen laitoksen koneissa käytetään vielä oletuksena Javan vanhempaa versiota, mutta compile-skripti osaa käyttää /opt/j2sdk1.4.1_01/bin/ -hakemistosta löytyvää kääntäjää.

E.2.2 Käynnistäminen

Järjestelmä voidaan käynnistää komennolla `./start` , mikäli käyttöympäristön Java-versio on 1.4.1.

Tietojenkäsittelytieteen laitoksen koneissa käytetään vielä oletuksena Javan vanhempaa versiota, mutta start-skripti osaa käyttää /opt/j2sdk1.4.1_01/jre/bin -hakemistosta löytyvää Java-tulkkia.

Jos halutaan käynnistää myös ylläpitäjän käyttöliittymä (esimerkiksi ryhmien luontia varten), voidaan käynnistyksen yhteydessä antaa parametri `-gui`.

Käynnistyessään järjestelmä avaa `sources/config/clientmanager.cfg`-tiedostossa määritellyt portit (oletuksena 8080 suojaamattomalle ja 8081 SSL-salatulle yhteydelle), joihin voi viitata WWW-selaimella esimerkiksi osoitteilla `http://localhost:8080/` tai `https://localhost:8081/`.

E.2.3 Järjestelmän sulkeminen

Järjestelmä voidaan sulkea komentorivikehoitteessa komennolla `Ctrl-c`, jolloin käynnistyy järjestelmän hallittu alasajo. (Myös ylläpitäjän käyttöliittymäikkunan sulkeminen aiheuttaa järjestelmän alasajon.)

E.2.4 Javadoc -dokumentaatio

Järjestelmän Javadoc-dokumentaatioon on linkki ryhmän kotisivuilla osoitteessa `http://www.cs.helsinki.fi/group/converge`, mutta tarvittaessa voidaan dokumentaatio luoda paikallisestikin antamalla sources-hakemistossa komento `./makejavado` Tällöin luodaan uusi hakemisto sources/javadoc, joka sisältää kaiken lähdekoodeista tuotetun dokumentaation.

Javadoc-dokumenttien luonnissa esiintyy varoitusilmoituksia, mutta ne ovat hyväksyttäviä.

E.3 Ylläpito

E.3.1 Lokitiedostot

Tässä prototyypissä kerätään runsaasti erilaista ohjelmiston toiminnasta kertovaa lokitietoa `sources/logs` -hakemistoon. Lokitiedostot luodaan aina uudestaan järjestelmää käynnistettäessä, viimeistään siinä vaiheessa, kun kuhunkin lokitiedostoon kirjoitetaan ensimmäistä kertaa.

E.3.2 Tietokannan varmuuskopio

Tietokannasta voidaan ottaa varmuuskopio kopioimalla hakemisto `sources/db` sisältöineen haluttuun paikkaan. Kopion palautus toimii vastaavasti kopioimalla varmuuskopio sources/db-hakemiston päälle (tätä ennen kannattanee sources/db joko tyhjentää tai poistaa kokonaan).

E.3.3 Asetustiedostot

WWW-käyttöliittymän ulkoasu on suurimmaksi osaksi helposti muutettavissa hakemistosta `sources/webcore/public_html/` löytyvää `converge.css` -tyylitiedostoa muokkaamalla. Saman hakemiston `index.html` on webcore-palvelimen juureen tuleva dokumentti (eli tavallaan järjestelmän etusivu osoitteessa `http://localhost:8080/`). (Muutokset `public_html` -hakemiston tiedostoihin tulevat voimaan ilman järjestelmän uudelleenkääntämistä.)

`sources/config/clientmanager.cfg`
Asiakasmoduuliin ja webcoreen liittyvät asetukset:
- Session vanhentumisaika (sekunteina)
- Webcoren käyttämät porttinumerot (SSL ja ei-SSL)
- Tulostetaanko WWW-käyttöliittymään testausinformaatiota (HTTP-parametrit ym.)

`sources/config/converge.key`

- SSL-palvelimen salausavain

`sources/config/profile.xml`

- Käyttäjän oletusprofiili XML-muodossa

`sources/config/groupprofile.xml`

- Ryhmän oletusprofiili XML-muodossa

Muut `sources/config`-hakemiston asetustiedostot ovat Xindicen asetuksia.

F WWW-käyttöliittymän käyttöohje