TIETOKANTA MERIKOTKIEN SEURANTAAN
Ylläpitodokumentti

Versiohistoria:

Versio	Päivämäärä	Kuvaus	Tekijä
1.0	12.12.2007	Ensimmäinen luonnos	Janne Piippo
2.0	13.12.2007	Virallinen versio	Janne Piippo

Projektiryhmä:
Janne Piippo
Juha Hiekkamäki
Pekka Maksimainen
Petri Setälä
Teemu Pulkkinen
Tuire Huhtamäki

HELSINGIN YLIOPISTO

Tietojenkäsittelytieteen laitos
Ohjelmistotuotantoprojekti Merikotka

Ohjaaja:
Sanna Keskioja

Sisällysluettelo

1. Johdanto

Ylläpitodokumentti käsittelee sellaisia asioita, joita mahdolliset jatkoryhmät tai asiakkaan edustajat tarvitsevat muokatakseen Haliaeetus-järjestelmää. Dokumenttiin on listattu toteutusvaiheessa ilmenneitä asioita, jotka ovat joko muuttuneet suunnitteluvaiheesta tai muutoin vaikuttavat ohjelman ylläpitoon. Sisältö on rajattu tekniseen toteutukseen, käyttöohje on oma dokumenttinsa. Ylläpitodokumenttia ei ole tarkoitettu luettavaksi yksinään, vaan yhdessä suunnitteludokumentin ja ohjelmakoodin kanssa.

2. Sanasto

Sanasto avaa termien merkityksiä sekä selventää sanojen määrittelyitä. Usein on tarpeen rajata yksinkertaisenkin sanan määritystä siten, että termi on kaikille osapuolille yksikäsitteisesti ymmärrettävissä. Toisaalta sanalyhenteitä sekä harvinaisempaa termistöä on kerrottu selkokielellä auttamaan dokumentin ymmärtämisessä.

2.1 Merikotkiin liittyvä sanasto

Merikotka Haliaeetus albicilla on Suomen suurin petolintu. Sen siipien väli on 190-240 cm ja pituus 76-94 cm. Merikotkalla on tasaruskea höyhenpuku ja suorakaiteen muotoiset siivet, jotka harittavat kärjistään. Pyrstö on lyhyt ja kiilamainen ja vanhoilla linnuilla valkea. Merikotka saavuttaa sukukypsyyden 3-6-vuotiaana.

Merikotkatyöryhmä Torsten Stjernbergin johtama merikotkien suojelua ja tutkimusta edistävä työryhmä Suomen WWF:ssä.

Pesä Merikotkan pesä on Suomessa yleensä hyvin kookas risupesä suuren puun latvassa tai poikkeuksellisesti nykyään jopa maassa. Merikotka pesii myös tekopesissä.

Reviiri Yhdellä reviirillä elää yksi merikotkapariskunta, jolla yleensä on reviirinsä alueella useampi pesä, joista yleensä yksi kerrallaan on käytössä.

Tekopesä Ihmisen tekemä pesä merikotkalle. Näin pyritään siihen, että merikotkat pesisivät häiriöttömillä alueilla ja pesät olisivat tarpeeksi tukevia. Vuonna 1998 23% tunnetuista asutuista pesistä oli tekopesiä.

Pesätarkastus Pesille tehdään tarkastuskäyntejä pesimisaikaan yleensä vain kerran vuodessa toukokuun lopulta alkaen. Tarkastuskäynneillä poikaset mitataan ja rengastetaan sekä kerätään näytteitä ja tietoja pesimäpaikasta myöhempää analyysia varten. Etenkin Pohjois-Suomessa tarkastuksiin käytetään myös lentokonetta, pesinnän tai sen puuttumisen toteamiseen pesintäkauden alkuvaiheessa.

WWF World Wide Fund for Nature, eli Maailman Luonnonsäätiö on maailmanlaajuinen luonnonsuojelujärjestö, joka työskentelee luonnon monimuotoisuuden suojelemiseksi ja ekologisten toimintojen ylläpitämiseksi. Suomessa toimii Maailman luonnonsäätiön (WWF) Suomen rahasto (Suomen WWF).

2.2 Tekninen sanasto

CSV Comma-Separated Values on tiedostorakenne, jolla voidaan esittää taulukkomuotoista dataa erottamalla sarakkeiden arvot jollakin välimerkillä.

CVS Concurrent Versions System. Versionhallintaohjelmisto, joka on luotu helpottamaan ohjelmistojen versionhallintaa.

Datamalli Puutietorakenne, jonka avulla siirretään dataa järjestelmän sisällä. Tarkemmin selostettu arkkitehtuurisuunnitelmassa.

Digikuva Digitaalisessa muodossa oleva kuvatiedosto. Kuvatiedostolla tarkoitetaan tässä dokumentissa JPG-tiedostoa.

Hakemisto Sisäkkäiset hakemistot muodostavat hakemistorakenteen eli hakemistopuun. Hakemisto voi sisältää tiedostoja ja hakemistoja.

Hali Haliaeetus-järjestelmän keväällä 2003 ohjelmistotuotantoprojektina toteuttaneen ryhmän nimi.

Hali2 Haliaeetus-järjestelmän jatkokehityksestä keväällä 2004 vastaavan ohjelmistotuotantoprojektiryhmän nimi.

Haliaeetus Tässä dokumentissa määritellyn tietokantajärjestelmän nimi.

HTML HyperText Markup Language. World Wide Webin eli WWW:n julkaisukieli.

HTTP Hypertext Transfer Protocol. Yhteyskäytäntö eli protokolla, jonka varaan WWW rakentuu. Hypertekstidokumenttien siirtoa verkossa tukeva komentokieli.

HTTPS HTTP over Secure Sockets Layer. HTTP:n salakirjoitettu versio.

Java Ohjelmointikieli, jota käytetään projektin toteutuksessa.

Järjestelmä Tässä dokumentissa järjestelmällä tarkoitetaan pääasiallisesti toteutettavaa ohjelmistoa tai jo valmista ohjelmistoa. Järjestelmä sisältää käyttöliittymän, tietokannan ja näiden välillä olevat toiminnallisuudet.

JDBC Java Database Connectivity. Ohjelmointirajapinta, jota käytetään kommunikoitaessa järjestelmän tietokannan kanssa.

Käyttöliittymä Ne välineet ja toiminnot, joilla käyttäjä on yhteydessä tietojärjestelmään eli käytännössä ohjelmistoon.

MS Excel -yhteensopiva Yhteensopivuudella tarkoitetaan, että data on helposti saatavissa Excelin tai jonkin muun tilasto-ohjelman käyttöön. Esimerkiksi CSV-muotoinen teksti täyttää vaatimuksen.

PDF Portable Document Format on standardiksi muodostunut esitystapa dokumenteille. PDF-tiedoston katseluun tarvitsee PDF-lukijan.

Servlet Java-ohjelmointikielellä kirjoitettu palvelinsovelma, servletti.

Tietokanta Jotain käyttötarkoitusta varten laadittu kokoelma toisiinsa liittyviä säilytettäviä tietoja. Tietokannan teknisiä ominaisuuksia ovat mm. tiedon riippumattomuus sitä käsittelevistä ohjelmista, tietojen samanaikainen käyttö, monipuoliset tiedonhakumahdollisuudet, tietojen suojaus, mutkikkaat riippuvuudet tietojen välillä ja automaattinen varmistus ja elpyminen häiriöistä.

Tietokantajärjestelmä Tässä dokumentissa tietokantajärjestelmällä tarkoitetaan pääasiallisesti toteutettavaa ohjelmistoa tai jo valmista ohjelmistoa. Tietokantajärjestelmä muodostuu tietokannasta, sitä hallinnoivasta tietokannanhallintajärjestelmästä sekä tietokantaa käyttävistä sovelluksista.

Tulostus Tässä dokumentissa tulostus-termiä käytetään paperille tulostamisen lisäksi myös raportin ja/tai digikuvan tiedostoon tallentamisesta yleisesti käytössä olevassa tiedostomuodossa.

WWW World Wide Web. Maailmanlaajuinen verkko, "verkko", Internet-verkko hypertekstimuodossa.

3. Toteutetut ominaisuudet

Suunnitellut muutokset

Suunnitteludokumentin järjestelmävaatimuksista toteutettiin kaikki prioriteetilla *välttämätön* olleet vaatimukset. Alempien prioriteettien vaatimuksista toteutettiin tietokannassa olevien koordinaattien tarkistukset ja korjaaminen (vaatimusdokumentin koodi J 7), koordinaattimuunnosten korjaaminen (J5.2) sekä osittain pudotusvalikkojen kirjainkoodien auki kirjoittaminen (J 5.1). Muita prioriteettien *hyödyllinen* ja *mahdollinen* osia ei ehditty ajan puutteen vuoksi tehdä. Järjestelmävaatimukset on tarkemmin listattu suunnitteludokumentin kappaleessa 5.

Suunnittelemattomat muutokset

Kuvatiedostojen vastaanottoa varten täytyi järjestelmään lisätä tuki selaimen lähettämän hajautetun sisällön parsimiselle. Toteutus löytyy luokan *Hali* metodeista *parseMultiPartContent(), parseMultipartHeaderSeparator()* sekä *parseValue()*. Tomcat -ympäristö osaa suoraan käsitellä vain normaaleja lomakkeita eikä tätä huomattu suunnitteluvaiheessa tarkistaa.

Pesän näkymien uudet kentät vaativat enemmän koodimuokkauksia kuin suunnittelussa otettiin huomioon (jossa mainittiin vain mallipohjien ja validoinnin muutoksia). Validoinnin lisäksi täytyi muokata myös tietojen tallennus-, haku- sekä vakioluokkia ottamaan huomioon uudet kentät. Sama koski kielitiedostoja. Toteutuksen alkuvaiheessa päädyttiin tekemään yhtenäisiä mallipohjia edellisten *subTemp*-osien sijaan. Täten suunnitelmassa näihin viitatut muutokset kohdistuivat vain yksittäisiin tiedostoihin.

Tietokantaan lisättiin uusia kenttiä ja joitain suunniteltuja kenttiä muokattiin, kun kenttien tarkoitusta selvennettiin toteutusvaiheessa.

4. Koodin ylläpitoon liittyvät seikat

Lähdekoodin käsittely

Järjestelmän CVS -versionhallintahakemisto löytyy db-palvelimen hakemistosta *"/home/tkt_hal3/cvs_rep_2*". Ajamalla juurihakemistossa komennon "make" haetaan kaikki tiedostot lokaaliin kopioon hakemistoon *"/home/tkt_hal3/cvs_local_2*". Tämän jälkeen kyseinen komento kääntää tarvittavat tiedostot ja kopioi ne edellee oikeille paikoilleen Tomcat -hakemistopuun alle. Mikäli automatisoitua prosessia täytyy muokata, tulee muutokset tehdä juurihakemiston tiedostoon "*Makefile*".

Tarkempi ohjeistus versionhallinnasta ja sen käytöstä löytyy liitteenä olevasta sisäisestä toteutusohjeistusdokumentista.

Sisäänkirjautuminen

Järjestelmän käyttäjätunnukset on edelleen määritelty ohjelmakoodissa luokassa *ValidateLogin*.

Puutteellinen virheiden käsittely

Monissa virhetilanteissa järjestelmä tulostaa vain tyhjän valkoisen sivun. Mallipohjien käsittelyyn on lisätty toiminnallisuutta, jonka pitäisi kirjoittaa näihin liittyvät virheet lokitiedostoon. Sen sijaan järjestelmä ei usein anna mitään ulospäin näkyvää informaatiota ajonaikaisista virheistä. Poikkeus- ja virhetilanteiden käsittelyä sekä näihin liittyvän informaation tallennusta lokiin tulisi edelleen parantaa.

5. Muut ylläpitoon vaikuttavat seikat

Mallipohjat

Mallipohjassa *uusikuva.ftl* osaa pudotusvalikoiden sisällöstä ei haeta tietokannasta. Suomenkielisiä valintoja ei lisätty tietokannan aputauluihin, sillä sekin olisi ollut virheellistä toiminnallisuutta. Tukeakseen kielikohtaisia pudotusvalikoita pitäisi tietokannasta hakea avain vastaavaan kielitiedoston arvoon. Tälle ei järjestelmässä ole tukea. Puutteen vuoksi pudotusvalikot ovat aina suomeksi, käytössä olevasta kielitiedostosta riippumatta.

"Pesä"-lomakkeet

Päivämääräkenttiin sijoitettua tekstimuotoista syötettä ei havaita kunnolla (ei pääse läpi, mutta ei tulosta virhesanomaakaan). Tämä näyttäisi olevan yleinen järjestelmän virhe. Poikasten mittaustietoihin voi lisätä negatiivisia arvoja, sillä ne ovat liukulukutyyppisiä ja näitä tarkistava metodi luultavasti ei käsittele niitä oikein.

Kielitiedostot

Vain suomenkielinen kielitiedosto on ajan tasalla. Muilla järjestelmä ei toimi kunnolla.

Raportti T, Raportti V

Raporttien otsikkorivien toteutuksessa ei ole huomioitu kielitukea. Mikäli saman vuoden aikana on samaan pesään tehty useampi kuin yksi pesätarkastus, perustuu "Vuosittaiset pesimistulokset" -sarake ko. vuoden viimeisimpään pesätarkastusraporttiin.

Raportti T:n toteuttavaan luokaan ReportT.java on sisäänrakennettu valmiiksi asiakkaan toivoma optio tulostaa pesälistaus haluttaessa ilman tuhoutuneita pesiä. Tämä ominaisuus jäi pois nykyversiosta käyttöliittymäoption jäämättä toteutumatta aikataulullisten tekijöiden vuoksi. (Ominaisuus on "testattavissa" kirjoittamalla raportin T hakuvalintoihin pesänimen kohdalle "test22".)

Raportti P, Raportti U

Raporttisivun hakuehdoista käytetään vain yksilöivintä mahdollista. Esimerkiksi jos on valittu sekä suuralue, reviiri että pesänimi haku tehdään ainoastaan pesänimen perusteella. Prioriteettijärjestys on seuraava: pesänimi, reviirinimi, kunta, ympäristökeskus, suuralue. P-raportin rauhoitusaste- tieto otetaan valitun vuoden ensimmäisenä päivänä (01.01.XXXX) voimassaolevasta pesämuuttuva taulusta.

Liitteet

LIITE 1 Toteutusohjeistusdokumentti

This page last changed on 2007-12-12 by ptmaksim.

Toteutusohjeistus

Ohjeita tarvittavien työkalujen käyttöön sekä ohjelmakoodin muotoiluun.

1. CVS

CVS (Concurrent Versions System) on versionhallintajärjestelmä jota käytetään kehityksen tukena. Projektin CVS -repository löytyy palvelimelta *db.cs.helsinki.fi* hakemistosta *cvs_rep*.

Eclipsen ja Puttyn konfiguroiminen (Pekan mailin pohjalta):

```
Eli Eclipse -> CVS -yhteys toimii seuraavasti:
Eclipse ottaa yhteyden db.cs.helsinki.fi-palvelimeen ja käyttää salaista avainta
(tkt_hal3.priv.zip, liitteenä).

1. Eclipseä varten tarvitsee nyt Plinkin - [http://www.chiark.greenend.org.uk/~sgtatham/putty/
download.html].
Hae plink.exe ja tallenna vaikka C:\windows\ -hakemistoon.
2. Käynnistä Eclipse.
3. Window -> Preferences -> Team -> Ext Connection Method
CVS_RSH: C:\windows\plink.exe
Parameters: -l {user} {host} -i C:\tkt_hal3.priv.ppk
(HUOM: laita polku tkt_hal3.priv.ppk:hon oikein!)
4. File -> New -> Project -> CVS -> Projects from CVS -> Create a new repository location.
5. Host: db.cs.helsinki.fi
Repository path: /home/tkt_hal3/cvs_rep
User: tkt_hal3
Password: jätä tyhjäksi
Connection type: ext
6. Use an existing module
Tästä voi valita sitten CVSROOTin.
```

Versionhallinta Eclipsellä:
Tarvittavat komennot löytyvät oikealla napilla aukeavan valikon Team -osiosta. Komentoja voi ajaa sekä yksittäisille tiedostoille tai
kokonaisille hakemistoille. Yksinkertaisuuden vuoksi emme tule käyttämään brancheja.
Tiedostojen päivittäminen:
Team -valikon "Update". Hakee tiedoston/hakemiston uusimman version. Versionhallinnasta tulevat muutokset yhdistetään versioosi
automaattisesti - myös silloin, jos olet tehnyt samaan tiedostoon omia muutoksia.
Muutosten tarkistaminen
On suositeltavaa, että ennen tiedostojen lähettämistä tarkistat tekemäsi muutokset. Tämä onnistuu ajamalla Team -valikosta
"Show History." Valitse haluamallesi versionumerolle "Compare Current with x.y". Tämän jälkeen näet kyseistä versiota vasten tekemäsi
muutokset.Mikäli haluat poistaa muutoksesi ja palata versionhallinnasta löytyvään versioon, valitse "Get Contents".
Tiedostojen lähettäminen:
Team -valikon "Commit". Päivittää muutoksesi versionhallintaan. Muista testata muutoksesi käyttämällä järjestelmää erillisen

päivityskomennon jälkeen.

Hakemistorakenne:

/dev
Kehitystiedostoja, joita ei päivitetä ajettavaan järjestelmään. Tätä hakemistoa voi käyttää ryhmäläisten välisten tiedostojen synkronointiin.
Kaikki kyseiseen kategoriaan kuuluvat hakemistot luodaan tämän moduulin alle (esimerkiksi web-templates, etc)

/**lib**
Järjestelmän käyttämät Java-kirjastot (.jar).

/pdf
Järjestelmän tarvitsemat PDF -syötepohjat.

/properties
Järjestelmäasetukset (debug-tila, käytetyt hakemistot, etc) sekä kielitiedostot.

/public_html
Järjestelmän staattiset kuvat (jpg) sekä css -tyylitiedostot.

/source
Järjestelmän Javalla kirjoitettu lähdekoodi.

/templates
Mallipohjat dynaamisille html -sivuille.

2. Järjestelmän päivittäminen

Järjestelmän päivitys versionhallintajärjestelmästä on automatisoitu. Tarkoitusta varten on luotu ns. makefile, joka määrittää ja lopulta suorittaa tarvittavat operaatiot. Kyseinen tiedosto löytyy käyttäjän "tkt_hal3" kotihakemistosta, joten myös päivityskomennot tulee ajaa samasta paikasta.

Kaikki CVS -repositoryyn päivitetyt tiedostot kopioidaan ajettavaan järjestelmään, joten muutosten toimivuus tulee aina tarkistaa ajamalla komento "make". Poikkeuksena tästä on moduulin "dev" alta löytyvät tiedostot.

Tarvittavat komennot:

make
Hakee uusimmat tiedostot versionhallintajärjestelmästä. Kääntää Java -lähdekoodin mikäli se on muuttunut. Kopioi päivitetyt tiedostot
oikeille paikoilleen järjestelmään.

make clean
Poistaa päivityksessä käytetyt väliaikaistiedostot. Komento kannattaa ajaa mikäli järjestelmän päivitys pysähtyy muuhun kuin
Java -lähdekoodin käännösvirheeseen. Lisäksi näin voidaan varmistaa, että automaatio on täydellinen - ts. järjestelmä ei vaadi
käsin tehtäviä lisäasetuksia toimiakseen.

Päivitystiedostoihin tarvittavat muutokset kannattaa kertoa/mailata Juhalle. Muutoin monimutkaisen tiedoston ymmärtämiseen menee liikaa aikaa.

2.1 Make-päivitysten synkronointi usean käyttäjän kesken

Samanaikainen make-päivitys on omiaan aiheuttamaan ongelmia kehityksessä. Ei ole toivottavaa, että joku muu päivittää järjestelmää juuri kun olet itse testaamassa järjestelmän toimintoja. Tätä tarkoitusta varten make-päivitykseen voi varata lukon. Lukko varataan *cvs*-scriptillä (liitteenä).

Alla esimerkki päivityslukon varaamisesta. Lukon varauksen yhteydessä kysytään varaajan nimi sekä arvio varausajan pituudesta. Jos lukkoa yrittää varata sen ollessa jo varattuna, näkee käyttäjä lukon varanneen nimen sekä aika-arvion. Näin järjestelmän kehitys on helpompaa hoitaa usean käyttäjän kesken ilman häiritseviä katkoksia.

tkt_hal3@alkokrunni:~$ ./cvs
Olet varannut CVS-lukon.
Tue Dec 11 15:22:01 EET 2007
Kerro kuka olet ja kuinka kauan varaat lukkoa: Pekka, 1h

Jos toinen käyttäjä yrittää nyt varata lukkoa, niin hän näkee vain varausilmoituksen.

tkt_hal3@alkokrunni:~$ ./cvs
CVS varattu
Lukko varattu (Tue Dec 11 15:32:25 EET 2007): Pekka, 1h

Vaikka lukko on varattu, niin järjestelmä on silti mahdollista päivittää ajamalla make. Kehitysyhteistyössä luotetaan siis siihen, että jos lukko on varattu, niin kukaan ei päivitä järjestelmää maken kautta.

Cvs-scripti pitää järjestelmän päivityshistoriatietoja tiedostossa *history*. Jos cvs-scripti jostain syystä ei koskaan anna varata lukkoa, niin lukon voi vapauttaa poistamalla tiedoston *lock-file*.

3. Ohjelmakoodin (Java) muotoilu

Usean henkilön ohjelmistoprojekteissa on tärkeää säilyttää ohjelmakoodilla yhtenäinen ulkoasu. Muutoin järjestelmän luettavuus ja tätä kautta ylläpidettävyys kärsii. Käytännöt tämän saavuttamiseen vaihtelevat. Joissain yrityksissä (usein nämä ovat pieniä) luotetaan sanattomaan "kuten olemassaoleva toteutus" -periaatteeseen, toisissa muotoiluoppaat ovat kymmeniä sivuja pitkiä. Yksityiskohtaisista oppaista ei kuitenkaan ole mitään hyötyä, jos ohjeita ei noudateta. Tästä hyvänä esimerkkinä toimii hali2 -ryhmän toteuttama järjestelmä. Dokumentaatiossa mainitaan, että toteutus tehdään Java Code Conventions -ohjeistuksen mukaiseksi (http://java.sun.com/docs/codeconv/CodeConventions.pdf, 24 sivua), mutta käytännössä linja ei ole kunnolla pitänyt. Vastaavan välttämiseksi tähän kappaleeseen on koottu joitain yleisiä periaatteita, joilla saadaan suurin osa hyödystä ilman raskaslukuisia dokumentteja joista kukaan ei jaksa asioita tarkistaa.

Rakenne:

Aloittava hakasulku ei tule omalle rivilleen.
Lopettava hakasulku tulee omalle rivilleen.
Sisennykseen käytetään aitoja tabeja.
Ryhmäkommentointia (Javadoc -tyylillä) käytetään luokkien ja metodian kommentointiin.
Rivikommentointia toteukseen.
Jokainen muuttuja ja metodi esitellään omalla rivillään.
Jokainen rivi voi sisältää vain yhden lauseen.
Muuttujat tulee aina alustaa esittelyn yhteydessä.

Esimerkkejä tästä lötyy kommentointikohdan alta.

Nimeäminen:

Ohjelmakoodi kirjoitetaan englanniksi. Kommentit ovat suomea.

Paketit (hali, servlet):
Kirjoitetaan pienillä kirjaimilla.

Luokat (View, ValidateSaaliit):
Jokaisen sanan ensimmäinen kirjain kirjoitetaan isolla, muut pienellä.

Metodit (execute, getEmptyErrors):
Muiden paitsi ensimmäisen sanan ensimmäinen kirjain isolla, muut pienellä.

Muuttujat (currentYear, nestId):
Muiden paitsi ensimmäisen sanan ensimmäinen kirjain isolla, muut pienellä.

Kommentointi:

Luokat ja metodit dokumentoidaan Javadoc -yhteensopivalla syntaksilla. Tällöin lähdekoodista voidaan automaattisesti generoida nämä selittävä dokumentaatio.

Luokkakommentointi:

```
/**
* Luokka View hoitaa Haliaeetus-järjestelmän näyttöjen generoimisen.
*/
public class View
```

Metodikommentointi:

```
/**
* Tarkistaa, onko annettu muuttuja vaaditulla arvovälillä.
*
* Aseta minValue ja maxValue samaksi, jos muuttujan tulee olla täsmälleen jonkin arvoinen.
*
* @param num - Tarkistettava muuttuja.
* @param minValue - Numeron minimiarvo. Vertailu sisältää yhtäsuuruuden.
* @param maxValue - Numeron maksimiarvo. Vertailu sisältää yhtäsuuruuden.
* @return Palauttaa vertailun tuloksen.
*/
protected boolean checkValue(int number, int minValue, int maxValue)
```

Kommentoinnissa on tärkeää, ettei toista informaatiota joka selviää lähdekoodista nopealla vilkaisulla. Ylläpitovaiheessa yksityiskohtaiset kommentit eivät pysy koodin perässä, ja mikäli kommentteihin ei voi luottaa, muuttuvat ne täysin hyödyttömiksi. Luokkien käyttötarkoitus ja rajapinnat - ja erityisesti näiden erikoistilanteet - sen sijaan tulee dokumentoida huolellisesti. Toteutustasolla on järkevintä, että kommentoinnin sijasta järjestää ohjelmakoodin niin, että metodit pysyvät lyhyinä ja tarkoitus selviää metodin nimestä.

```
// Laskee neliöjuuren. <-- Huono kommentti, ei kerro mitään mikä ei selviäisi jo metodin nimestä.
float calculateSquareRoot(float value)

// Laskee neliöjuuren. Hajoittaa tietokoneesi negatiivisilla arvoilla. <-- Dokumentoi oleellisen,
// eli mitä tapahtuu erikoistilanteissa
float calculateSquareRoot(float value)

//
**********************************************************************************************************

// Jos karkausvuosi
if(year % 4 == 0) {
if(year % 100 != 0) {
if(year % 400 != 0)
prepareOlympics();
}
else
prepareOlympics();
}

// Ylläolevan sijasta kannattaa kirjoittaa erillinen metodi. Tällöin kommentteja ei tarvitse,
// ja samalla oleellinen toiminallisuus hajoitetaan helpommin uusiokäytettäviin osiin. DRY!

boolean isLeapYear(int year) {
if(year % 4 == 0) {
if(year % 100 != 0) {
if(year % 400 != 0)
return true;
}
else
return true;
}

return false;
}

// Selkeää ilman turhaa kommentointia
if(isLeapYear(year))
prepareOlympics();
```

4. Debug-tiedostojen käsittely Linuxin työkaluilla

Linux sisältää perustoiminnallisuutta tarjoavat GNU utils -apuohjelmat. Tuttuja GNU utils -ohjelmia ovat ls, cp, rm ja jne. Hieman vähemmän tuttuja lienee grep ja tail. Useimmille täysin tuntemattomia ovat find, cut ja xargs. Näiden perustyökalujen avulla satamegaistenkin tiedostojen käsittely käy leikiten.

Järjestelmän debug-tulostusta luettaessa usein on tarpeen tietää vain mitä viimeisimmät 20 - 100 riviä ovat. Käsitellään esimerkiksi tiedostoa debug_log_2007-12-9.txt (158MB).

```
tail -n 3 debug_log_2007-12-9.txt

2007-12-09 23:58:56     fi.hy.hali.view.ViewTemplateExceptionHandler|- Expression
data.pesatarkastus.nayte_p is undefined on line 1931, column 81 in pesatarkastus.ftl.
2007-12-09 23:58:56     fi.hy.hali.view.ViewTemplateExceptionHandler|- Expression
data.pesatarkastus.nayte_a is undefined on line 1933, column 81 in pesatarkastus.ftl.
2007-12-09 23:58:56     fi.hy.hali.view.ViewTemplateExceptionHandler|- Expression
data.pesatarkastus.nayte_r is undefined on line 1935, column 81 in pesatarkastus.ftl.
```

Nyt debugin lukeminen on jo huomattavasti helpompaa, kun ei tarvitse rullata montaa tuhatta riviä nähdäkseen uusimmat debug-viestit. Yllä olevakaan tuloste ei kovin siististi näy kapeassa konsoli-ikkunassa. Debugin alkuosa toistuu aina samana, joten leikataan se pois. Se onnistuu cut-ohjelmalla:

```
tail -n 3 debug_log_2007-12-9.txt | cut -c 37-

ViewTemplateExceptionHandler|- Expression data.pesatarkastus.nayte_p is undefined on line 1931,
column 81 in pesatarkastus.ftl.
ViewTemplateExceptionHandler|- Expression data.pesatarkastus.nayte_a is undefined on line 1933,
column 81 in pesatarkastus.ftl.
ViewTemplateExceptionHandler|- Expression data.pesatarkastus.nayte_r is undefined on line 1935,
column 81 in pesatarkastus.ftl.
```

Vastaavasti voitaisiin leikata neljännestä pisteestä eteenpäin.

```
tail -n 3 debug_log_2007-12-9.txt | cut -d "." -f 5-
```

Tulos on sama kuin yllä.

Jos debug-tiedostosta taas pitää hakea jotakin, niin se onnistuu yleensä grepin avulla. Yritetään etsiä esimerkiksi "blob"-sanan sisältäviä debug-viestejä. Joskus debug-viestejä tulee niin paljon ja hajanaisesti, että koodissa miltei peräkkäin olevat debug-viestit voivatkin olla debug-tiedostossa kymmeniä rivejä erillään.

```
tail -n 200 debug_log_2007-11-26.txt | grep -i blob | grep -vi execute
```

Komento siis hakee debug-tiedoston 200 viimeisintä riviä. Hakee sitten sanaa "blob" ja poistaa rivit, joilla esiintyy sana execute. Näin saadaan tulostettua haluttu rivi:

```
2007-11-26 23:44:47     fi.hy.hali.operation.KuvaOperation|- Trying to get image blob:
oracle.jdbc.driver.OracleResultSetImpl@fe315d
```

Logien selaus käy taas helposti less-komennon avulla. Näin haetaan 200 viimeisintä riviä less-ohjelman näytettäväksi:

```
tail -n 200 debug_log_2007-12-9.txt | less
```

Nyt less-ohjelmassa on helppo kelata sivua ylös ja alas ilman, että pitää käynnistää "raskasta" tekstieditoria. Less:stä pääsee pois painamalla q. Hakutoiminnallisuuden saa /-merkillä.

Kunhan nämä komennot tulevat tutuiksi, niin järjestelmästä pystyy helposti ja nopeasti etsimään haluttua tietoa. Tässä esimerkki komentojen tehokkuudesta. Etsitään järjestelmän java-tiedostoista koodirivejä, jotka käsittelevät oraclen BLOB-olioita. Esimerkin vuoksi on lisätty vielä sed-editori, joka poistaa riveiltä turhat whitespacet.

```
find . -iname "*.java" | xargs grep -i "blob" | grep -i oracle | sed -e 's/\t*\|  //g' | cut -d "/"
-f 9-
```

Tämänkin komentolitanian voi helposti jakaa osiin ja yrittää ymmärtää jokaisen osasen toiminnan erikseen. Find-komento etsii kaikki java-tiedostot alihakemistoja myöten. Xargs-komento antaa jokaisen löydetyn java-tiedoston parametrina grepille. Grep etsii tiedostojen sisältä blob-sanaa. Löydetyistä riveistä etsitään sitten oracle-sanaa. Sen jälkeen sed poistaa whitespacen. Tämän jälkeen pitkä rivi vielä lyhennetään sopivasti. Tähän lopputulokseen pääsee yrityksen ja erehdyksen kautta.