

Huutamo

Toteutusdokumentti

Eric Andrews

eric.andrews@helsinki.fi

Helsingin yliopisto, Tietojenkäsittelytieteen laitos

Tietokantasovellus

1.11.2009

• Sisällysluettelo

1 Johdanto.....	3
1.1 Järjestelmän tarkoitus.....	3
1.2 Toimintaympäristö.....	3
1.3 Rajaukset.....	3
2 Rungon rakenne.....	4
2.1 Yleiskuva ja periaatteet.....	4
2.2 Komponentit.....	5
2.2.1 Uudelleenohjaus.....	5
2.2.2 Konfigurointitiedosto.....	5
2.2.3 Controller-luokat eli ohjaimet.....	5
2.2.4 DAO-luokat.....	5
2.2.5 Model-luokat.....	5
2.2.6 HTMLTemplate.....	5
2.2.7 View eli näkymät.....	6
2.2.8 Helper-luokat eli apuluokat.....	6
3 Sovelluksen rakenne.....	6
3.1 Yleiskuva	6
3.2 Komponentit.....	8
3.2.1 Näkymät.....	8
3.2.2 Modelit.....	10
3.2.3 Ohjaimet.....	10
3.2.4 DAO-luokat.....	11
3.2.5 Apuluokat.....	11
3.2.6 Muut.....	12
4 Asennusohjeet.....	13
5 Käynnistysohje.....	13
6 Liitteet.....	13

1 Johdanto

1.1 Järjestelmän tarkoitus

Huutamo on webhuutokauppajärjestelmä, joka perustuu käyttäjien välisiin myynti-ilmoituksiin ja tarjouspyyntöihin. Järjestelmän tarkoituksena on tarjota käyttäjilleen järjestelmää, jossa mielekkäästi harjoittaa huutokaupan tai kirpputorin tyyppistä kauppaa: vapaasti, mutta myös laajasti. Käyttäjien välisiä kauppvoja ei erikseen ohjata vaan vastuu jätetään suurin osin kauppakumppaneiden harteille.

Internet-pohjaisuutensa ansioista edut perinteisiin kanaviin ovat huikeat: potentiaalisesti suurempi ja hajautetumpi kävijäkunta, joka tuo myynti-ilmoituksille enemmän näkyvyyttä.

1.2 Toimintaympäristö

Sivun käyttäminen vaatii nykyaikaisen www-selaimen ja perustehokaan tietokoneen, jolla selailu toimii sulavasti. Selaimen tulee tukea CSS-tyylimäärittelyitä, jotta sivut näkyisivät oikein. Javascript-tuki ei ole pakollista, vaikkakin suotavaa käytettävyyden kannalta.

Koska palvelinohjelmisto on toteutettu käyttäen PHP5 ja PostgreSQL php -kirjastoa, vaaditaan niiden tukea Huutamoa ajavalta www-palvelimelta. Tietokantapalvelimeksi vaaditaan ”PostgreSQL”-tietokanta.

1.3 Rajaukset

Tehtävänannon laajuuden järjeistämiseksi, tiettyjä ominaisuuksia on tietoisesti karsittu:

- Järjestelmässä ei ole sisäänrakennettua käyttäjien välistä kommunikaatiokanavaa, vaan käyttäjien tulee omatoimiset hoitaa asiat sähköpostein.
- Suurta kaupankäynnin oikeellisuutta valvovaa koneistoa ei ole tarkoitus rakentaa, siispa kaupankäynnin mahdolliset ongelmat jätetään käyttäjien harteille.
- Sivutusta ei toteuteta vaan oletetaan kohteiden lukumäärän olevan tarpeeksi pieni näytettäväksi yhdellä sivulla.

Seuraavia saatetaan osin toteuttaa jos vain aika sallii:

- Olemaosaslevien myyntikohteiden muokkaaminen
- Käyttäjätietojen päivitys ja profiileiden poistaminen.
- **Mahdollisuutta lisätä kuvia kohteeseen. (Toteutettu)**

2 Rungon rakenne

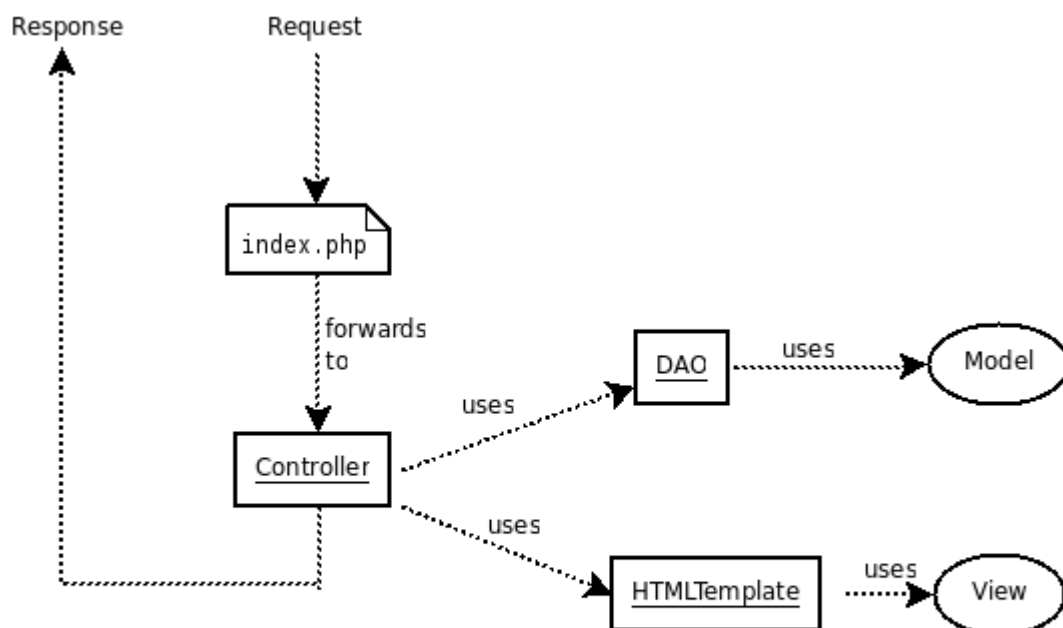
Ohjelman koostuu kahdesta kokonaisuudesta: rungosta (tai ”frameworkista”) ja itse ohjelmaa määrittävästä koodista eli sovelluksesta.

2.1 Yleiskuva ja periaatteet

Kirjoitin geneerisen rungon ennen kun aloitin varsinaisen ohjelman koodaamisen. Tällä tavoin saisin jaettua ohjelman loogisiin kokonaisuuksiin. Varsinkin kuin toteutuskielenä toimii PHP niin luokkapohjainen ratkaisu vaati hieman uudelleenohjauskoodia toimiakseen järkevästi. Luokkapohjaisuuden rinnalle otin myös ”model-view-controller”-ohjelmistoarkkitehtuurityylin, jonka avulla ohjelmakoodit voidaan jakaa loogisiin osiin sen mukaan mitä kunkin koodin tehtävänä on.

Tietokantapuolen toteutin niin, että sovelluskoodissa on mahdollista käyttää ORM-tekniikkaa (Object-relational mapping), eli sovellus voi ikään kuin suoraan hakea ja tallettaa objekteja tietokannasta. Näkymäpuoli toimii ”push”-pohjaisena eli tulostettava data työnnetään prosessoinnin jälkeen yksinkertaiselle HTML-PHP sekakoodille aseteltavaksi ja käsiteltäväksi, ja lopuksi tulos esitellään (vertaa JSP).

Rungossa on myös osittain ”Convention over configuration”-filosofia, joka tarkoittaa sitä, että suurten konfigurointitiedostojen sijasta, oletetaan ohjelmoijan käyttävän tiettyjä nimeämiskäytäntöjä. Tämä näkyy käytännössä esim. uudelleenohjauksessa ja ”HTML-lomake → *model*”-muunnoksissa.



Pyynnön käsittelyn 'flow'

Kaaviossa näkyy, miten rungolle toteutettu sovellus suurpiirteisesti toimii, kun se saa pyynnön selaimelta. Aluksi *index.php* ottaa pyynnön käsittelyyn, ja ohjaa sen tietylle *controllerille* eli ohjaimelle, joka toimii kaiken koordinoijana. *DAO*-luokkien avulla ohjain tallettaa tai hakee *modeleita* tietokannasta. Lopuksi luodaan yksi tai useampi *HTMLTemplate*-objekti, joka rakentaa selaimelle palautettavan näkymän tietyn *view*-koodin sekä ohjaimen välittämän datan perusteella. Lopuksi lähetetään näkymä vastausviestinä asiakaspäätteen (selaimen) näytettäväksi.

2.2 Komponentit

2.2.1 Uudelleenohjaus

Index.php ohjaa pyyntöjä ohjaimille. Uudelleenohjaus perustuu konfigurointitiedoston asetuksiin ja pyynnössä esiintyvän ”*controller*”-parametriin. Jokainen ohjain perii luokan *BaseController.class.php*, joka taas on vastuussa metodin valinnasta (metodiin ohjaus). Metodien valinta perustuu ”*action*”-parametriin tai sen olemattomuuteen.

2.2.2 Konfigurointitiedosto

Löytyy *config/config.php* tiedostosta. Mahdollisuus määritellä mm. tietokantayhteys, sovelluksen internetissä näkyvä juurikansio ja eri komponenttityyppien hakemistot.

2.2.3 Controller-luokat eli ohjaimet

Vastaavat pyynnönkäsittelystä ja toimivat ”orkesterinjohtajana” käyttäen suuresti apuna *model* ja *view* komponentteja. Voidaan myös nähdä ikään kuin *modelin* ja *viewin* välisenä tulkkina, joka välittää viestin toiselta toiselle.

2.2.4 DAO-luokat

DAO-luokat (”Data Access Object”) vastaavat tietokantaoperaatioista, sekä objekti- ja relaatiopohjaisen datan muuntamisesta muodosta toiseen, niin ettei sovelluskoodin täydy ajatella dataa muunna kuin objekteina (ORM). Siispä *DAO*-luokat palauttavat ja ottavat vastaan *model*-luokkia. Kaikki *DAO*-luokat perii *BaseDAO*n, josta saa valmiiksi tietokantayhteysresurssin, ja joka tarjoaa yksinkertaiset CRUD-operaatiot.

2.2.5 Model-luokat

Edustavat tilanteesta riippuen mm. ohjelman säilyttämää dataa, käyttäjän syöttämää kenttädataa tai tietokantataulun yhtä riviä. *Model*-luokissa määritellään validointimetodi (esim. *validate()*), jolla validoidaan objektissa oleva data erilaisin perustein. Myös erilaisia taulukkomuunnosmetodeja (associative array) on tarjolla. Niiden avulla objektin data voidaan muuttaa taulukoksi tai taulukosta voidaan suoraan luoda objekti. Jälkimmäinen tapaus on varsin kätevä tapa esim. luoda *model* suoraan käyttäjän täyttämästä lomakkeesta.

2.2.6 HTMLTemplate

HTMLTemplate luokalla voi ladata *view*-komponentteja ja populoida ne datalla, sekä lopuksi tulostaa tai ottaa talteen populoitu näkymä.

2.2.7 View eli näkymät

”View” tai näkymät ovat HTML-tiedostoja, joiden sekaan on lisätty yksinkertaisia php-lausekkeita datan esitystä varten. Näkökomponenteille ”työnnetään” dataa esim. objekteina, joista sitten yksinkertaiset php-lausekkeet lukevat halutut datat haluttuihin kohtiin.

2.2.8 Helper-luokat eli apuluokat

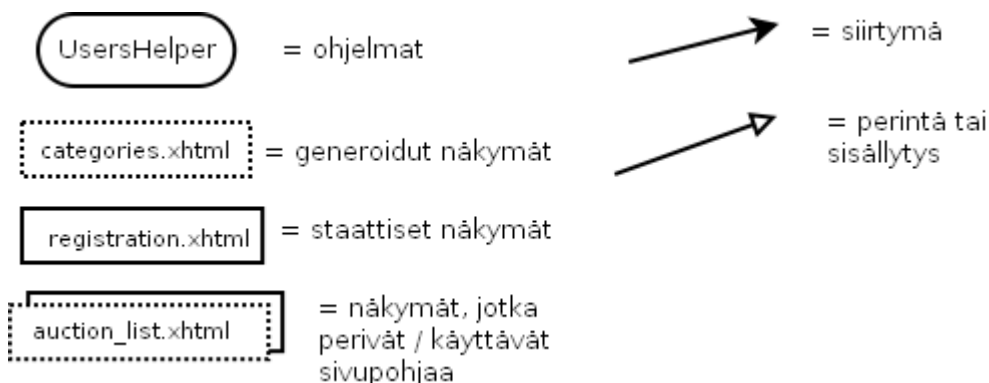
Tarjoavat perinteisesti erilaisia apumetodeja näkymiä varten, jotta niissä tarvitsisi ilmaista mitään monimutkaisia lausekkeita. Tässä rungossa *helper*-luokkia käyttää myös muutkin kuin vain näkymät.

3 Sovelluksen rakenne

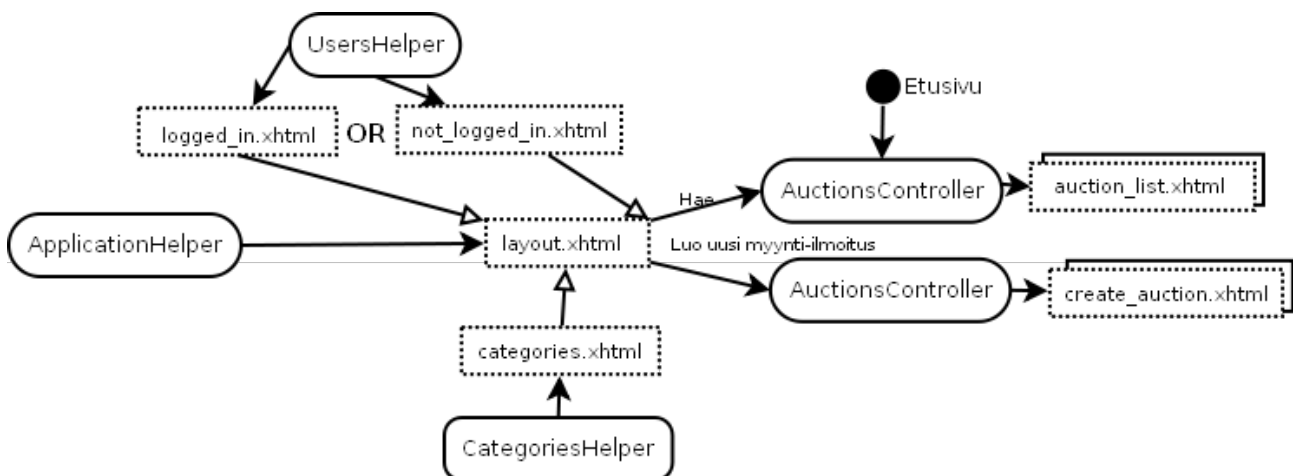
3.1 Yleiskuva

Siirtymät ovat tässä sovelluksessa usein ennalta arvaamattomia, eivätkä ne välttämättä noudata mitään tiettyä järjestystä. Siksi päätin poiketa kuvaustekniikassa hieman ohjeistuksesta. Sen sijaan, että esittelisin yhden ison sekavan kaavion niin jaan sen useampaan, toivottavasti luettavampaan osaan.

Kaavioiden symbolit



Sivupohja



Kuva 1: sivupohjan rakenne

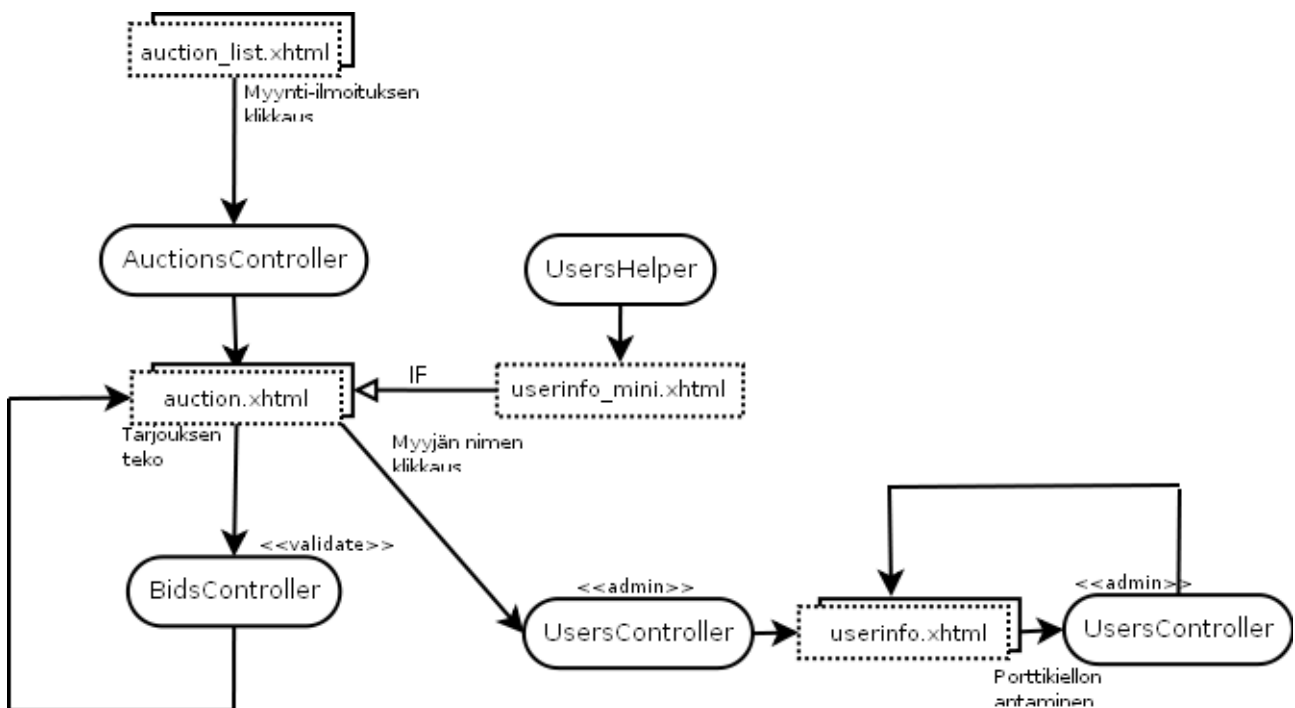
Näkymien yhtenäisen tyylin, rakenteen ja navigoinnin määrittelee sivupohja (eng. ”template”) *layout.xhtml*. Kaaviosta nähdään, että *ApplicationHelper*:in avulla muut näkymät voivat ottaa käyttöön sivupohjan. Sivupohjaan lisätään myös *UsersHelper*:illä käyttäjäboksi, joka vaihtelee sen mukaan onko käyttäjä sisäänkirjautunut vai ei. Pohjaan lisätään myös *CategoriesHelper*:illä kategoriat hakutoimintoja varten.

Oikealla puolella näkyy navigointipalkin siirtymät: myynti-ilmoitusten listaus ja myynti-ilmoituksen luontilomake. Diagrammissa on esitelty myös sovelluksen etusivu. Etusivuna toimii *AuctionsController*:in kautta generoitu *auctions_list.xhtml*, joka oletuksena näyttää suosituimmat voimassaolevat myyntikohteet.

Myynti-ilmoitusten listaus

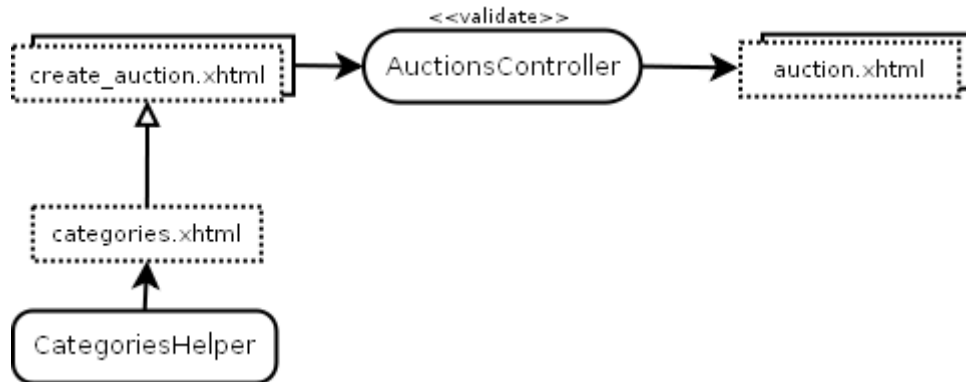
Tarkastellaan lähemmin *auction_list.xhtml*-tiedostoa ja sen siirtymiä. Kuten alla olevasta diagrammista näkyy, myynti-ilmoitusta klikkaamalla päästään tarkastelemaan tiettyä myynti-ilmoitusta. *Auction.xhtml*:n oikealla puolella oleva *IF* toteutuu, kun myynti-ilmoitus on sulkeutunut ja käyttäjällä on oikeus nähdä kauppakumppanin tiedot. Ostotarjoukset validoidaan *BidsController*:issa, jonka jälkeen palataan takaisin juuri tarjottuun ilmoitukseen.

Jos käyttäjällä on ylläpito-oikeudet, voi hän klikata myyjän nimeä ja päästä tutkimaan käyttäjän tietoja, sekä mahdollisesti antamaan tälle porttikiellon.



Kuva 2: myynti-ilmoitusten listaus, myynti-ilmoituksen tarkastelu, tarjouspyynnön teko, käyttäjätietojen tarkastelu ja porttikieltojen jako

Myynti-ilmoituksen luonti



Kuva 3: myynti-ilmoituksen luonti

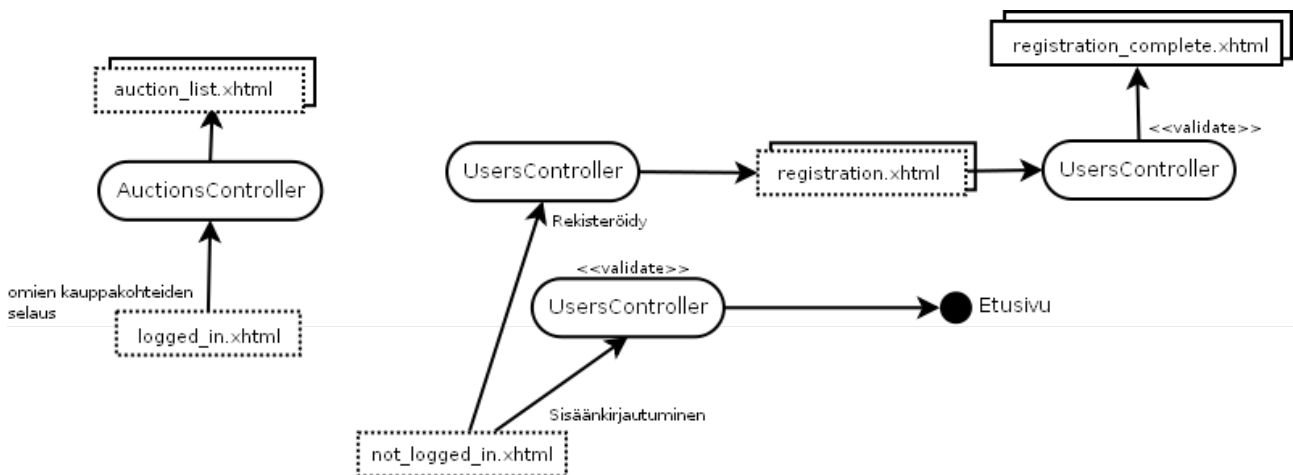
Kaikki

lähtee liikkeelle *create_auction.xhtml*-lomakkeesta, johon lisätään kategoriat. Lomakkeen täytettyä ja lähetettyä se yksinkertaisesti validoidaan *AuctionsController*issa ja validoinnin onnistuttua käyttäjä siirretään uunituoreen ilmoituksen sivulle.

Käyttäjäkapsin käyttäminen

Käyttäjän ollessa sisäänkirjautunut, voi hän käyttää *logged_in.xhtml*-käyttäjäkapsia omien kauppakohteidensa selaukseen.

Kun käyttäjä ei ole sisäänkirjautunut voi hän sisäänkirjautua *not_logged_in.xhtml*-käyttäjäkapsista. Vaihtoehtoisesti käyttäjä voi rekisteröityä, joka onnistuneen validoinnin jälkeen siirtää käyttäjän ”onnittelusivulle”.



Kuva 4: käyttäjäkapsin käyttäminen

3.2 Komponentit

3.2.1 Näkymät

Näkymätiedostot löytyvät *views*-kansion alta, jaettuna kansioihin tehtävänsä ja sisältönsä mukaan.

auction/auction.xhtml

Myynti-ilmoitussivu, joka sisältää kattavat tiedot myyntikohteesta sekä lomakkeen, jolla voi tehdä tarjouspyynnön.

auction/auction_list.xhtml

Käytetään myynti-ilmoituslistan rakentamiseen. Jokaisesta kohteesta näytetään: kohteen nimi, nykyinen hinta ja tarjousten määrä.

auction/create_auction.xhtml

Lomake, jolla voidaan laatia uusia myynti-ilmoituksia.

layouts/layout.xhtml

Sivupohja, joka määrittelee yhtenäisen rakenteen, tyylin ja navigoinnin monille sivuston näkymille.

partials/categories.xhtml

Palauttaa HTML:n ”select”-valintaboksin, jonka vaihtoehtoina löytyy eri kategoriat.

partials/errors_flash.xhtml

Virheilmoituslaatikko, joka näytetään sivupohjalla esim. kun jokin lomake ei validoitunut tai käyttäjä yritti käyttää toimintoa, johon hänellä ei ollut oikeuksia.

user/registration.xhtml

Rekisteröitymislomake.

user/registration_complete.xhtml

Sivu, jolla ilmoitetaan rekisterilomakkeen lähettäneelle käyttäjälle, että rekisteröinti onnistui.

user/userinfo.xhtml

Käyttäjätietosivu, jolle pääse vain ylläpito-oikeudet omaava, sisäänkirjautunut käyttäjä.

user/userinfo_mini.xhtml

”Miniversio” yllä mainitusta, joka näytetään myynti-ilmoitussivulla, kun huutoaika on päättynyt ja käyttäjä on kohteen myyjä tai korkein tarjoaja. Käyttäjätiedoissa näkyy kauppakumppanin tiedot.

userbox/logged_in.xhtml

Käyttäjäksi kun käyttäjä ei ole sisäänkirjautunut. Sisältää sisäänkirjautumislomakkeen ja linkin rekisteröitymislomakkeeseen.

userbox/logged_in.shtml

Käyttäjäksi kun käyttäjä on sisäänkirjautunut. Sisältää ”henkilökohtaisen navigoinnin”, jolla voi hakea sellaisia myynti-ilmoituksia, joissa on itse osallisena.

3.2.2 Modelit

Sijaitsevat *models*-kansion alla.

Auction.class.php

Määrittelee luokan *Auction*, joka vastaa instantioituna yhtä myynti-ilmoitusta. Tiedostossa määritellään myös *AuctionException*-poikkeus joka voidaan heittää esim. ohjaimelle validoinnin epäonnistuessa. *AuctionException* sisältää kattavat tiedot siitä, mitkä kentät eivät läpäissyt niille asetettuja ehtoja.

Bid.class.php

Sisältää luokan *Bid*, joka vastaa yhtä tarjousta. Tiedostossa määritellään myös *BidException*-poikkeus, joka voidaan heittää validoinnin epäonnistuessa.

Category.class.php

Sisältää *Category*-luokan määrittelyn, joka vastaa yhtä kategoriaa.

User.class.php

Sisältää luokan *User*, joka vastaa yhtä käyttäjää. Tiedostossa määritellään myös *UserException*-poikkeus, joka voidaan heittää validoinnin epäonnistuessa.

3.2.3 Ohjaimet

Sijaitsevat *controllers*-kansion alla.

AuctionsController.class.php

Määrittelee *AuctionsController*-ohjaimen, joka vastaa:

- Etusivun ”suosituimmat kohteet”-listasta
- Hausta ja hakutulosten listaamisesta
- Myynti-ilmoitus -kohtaisista sivuista
- Myynti-ilmoituksen luontilomakkeesta ja sen käsittelystä
- Myynti-ilmoituksen poistamisesta

BidsController.class.php

Määrittelee *BidsController*-ohjaimen, joka lähinnä vastaa tarjouspyyntöjen käsittelystä.

UsersController.class.php

Määrittelee *UserController*-ohjaimen, joka vastaa:

- Rekisteröitymislomakkeesta ja sen käsittelystä
- Sisään- ja uloskirjautumisesta
- Käyttäjätietojen näyttämisestä
- Käyttäjäkohtaisten porttikieltojen antamisesta ja poistamisesta

BaseController.class.php – ks. 2.2.1

3.2.4 DAO-luokat

Sijaitsevat *daos*-kansion alla.

AuctionsDAO.class.php

Määrittelee myynti-ilmoituksiin liittyvät tietokantaoperaatiot sekä *model*-luokan *Auction* ja tietokantataulun *auctions* välisen ORM-suhteen.

BaseDAO.class.php – ks. 2.2.4

BidDAO.class.php

Määrittelee tarjouspyyntöihin liittyvät tietokantaoperaatiot sekä *model*-luokan *Bid* ja tietokantataulun *bids* välisen ORM-suhteen.

CategoryDAO.class.php

Määrittelee kategorioihin liittyvät tietokantaoperaatiot sekä *model*-luokan *Category* ja tietokantataulun *categories* välisen ORM-suhteen.

UserDAO.class.php

Määrittelee käyttäjiin liittyvät tietokantaoperaatiot sekä *model*-luokan *User* ja tietokantataulun *users* välisen ORM-suhteen.

3.2.5 Apuluokat

Löytyvät *helpers*-kansion alta.

ApplicationHelper.class.php

Tarjoaa ratkaisuja yleisiin, sovelluksen eri osissa ilmeneviin ongelmiin esim: sivupohjalle julkaiseminen, pilkkudesimaalien muuntaminen pistedesimaalimuotoon ja toisin päin.

AuctionsHelper.class.php

Tarjoaa apumetodeja myyntikohteiden esittämistä varten.

CategoriesHelper.class.php

Tarjoaa erilaisia apumetodiratkaisuja kategorioihin liittyvien ongelmien helpottamiseksi.

UsersHelper.class.php

Tarjoaa erilaisia apumetodeja käyttäjätietojen esityksen ja käsittelyn helpottamiseksi.

HTMLTemplate.class.php – ks. 2.2.6

3.2.6 Muut

views/css

Sisältää CSS-tyylimäärittelyt näkymille.

views/images

Sisältää sivuston tyyliin kuvia.

views/js

Sisältää sivuston JavaScriptit.

config/config.php – ks. 2.2.2

index.php – ks. 2.2.1

.htaccess

Sisältää ”URL-rewrite”-säännöt, jolla sovelluksen www-osoitteista saadaan luettavammat. Vertaa:
users/show/5

vs

index.php?controller=users&action=show&id=5.

Tiedoston sisältö:

```
# Enable rewriting
```

```
RewriteEngine on
```

```
RewriteBase /~andrews/tsoha
```

```
# Rewrite controllers nicely
```

```
#
# /users -> index.php?controller=users
# /auctions -> index.php?controller=auctions
#
RewriteRule ^([\w]+)/([A-Za-z]+)/?([0-9]+)/?$ index.php?
controller=$1&action=$2&id=$3
RewriteRule ^([\w]+)/([0-9]+)/?$ index.php?controller=$1&id=$2
RewriteRule ^([\w]+)/?$ index.php?controller=$1
RewriteRule /views/(.*) views/$1
```

4 Asennusohjeet

Sovelluksen asentaminen onnistuu purkamalla liitteenä oleva arkisto siihen polkuun, josta se tullaan ajamaan. Tämän jälkeen tulee vielä konfiguroida seuraavat tiedostot:

1. *config/config.php*:sta tulisi muuttaa ainakin ***PGSQL_CONNECTION_STRING***, niin että se viittaa oikeaan tietokantaan, ja ***__DOCUMENT_ROOT_URL***, niin että se viittaa internetiin näkyvään sovelluksen juureen.
2. *.htaccess*-tiedostosta tulisi muuttaa ***RewriteBase*** */~andrews/tsoha*, niin että jälkimmäinen osa (*'~andrews/tsoha'*) viittaa ulospäin näkyvään sovelluksen juurikansioon eli samaan kuin äsken mainittu ***__DOCUMENT_ROOT_URL***. Jos *.htaccess*-konfiguraatiota ei aio käyttää niin joutuu käsin muokkaaman näkymissä olevia linkkejä ”rumempiin”-versioihin.

5 Käynnistysohje

Sovellusta pääsee käyttämään osoitteesta: <http://db.cs.helsinki.fi/u/andrews/tsoha>

6 Liitteet

Löytyy kansioista *liitteet*.