Magento-moduulin kehittäminen

Promootiotuotteet-moduuli

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Ammattikorkeakoulun opinnäytetyö

Tietojenkäsittelyn koulutusohjelma.

Visamäki, syksy 2013

Jukka Kartano

HAMK
HÄMEEN AMMATTIKORKEAKOULU

TIIVISTELMÄ

Visamäki
Tietojenkäsittelyn koulutusohjelma
Systeemityö

Tekijä Jukka Kartano **Vuosi** 2013

Työn nimi Magento-moduulin kehittäminen

TIIVISTELMÄ

Magento on suosittu avoimen lähdekoodin verkkokauppa-alusta, joka on luonteeltaan muokattava ja modulaarinen. Tämä muokattavuus ja modulaarisuus on mahdollistanut verkkokaupan myymisen laajalle asiakaskunnalle.

Opinnäytetyön tekijä tekee työkseen Magento-verkkokauppoja. Työnkuvaan kuuluu paljon Magenton ohjelmointia, eli asiakkaiden haluamien ominaisuuksien toteuttamista verkkokauppakohtaisesti. Suurin osa asioista, joita ohjelmoidaan Magento-verkkokauppoihin, on asiakaskohtaisia uusia ominaisuuksia. Täten ne eivät ole suoraan siirrettävissä toiseen verkkokauppaan, vaikka sama toiminnallisuus haluttaisiin toiseen verkkokauppaan.

Tässä työssä selitetään Magenton rakenne ja se, miten moduulit siinä toimivat. Kun Magenton rakenne on esitelty ja selvitetty. Toteutetaan verkkokauppakohtainen ominaisuus nimeltä "promootiotuotteet" omana modulaarisena toteutuksenaan.

Opinnäytetyölle asetetut tavoitteet saavutettiin, ja vaikka moduulit olivat tekijälle ennestään tuttuja, niin alusta loppuun implementoimalla monesta asiasta tuli kehittäjälle konkreettisempi. Moduulin rakenteen perusteellinen läpikäynti lisäsi tekijän ammattitaitoa ja tietämystä Magentosta.

Välittömiä jatkokehitysideoita opinnäytetyölle ei ole työn kirjoittamisen aikana kertynyt, koska aiheen rajaus on ollut tarkka. Magento Connect -moduulimarkkinapaikka on yksi jatkokehityksen mahdollinen suunta. Yksi mahdollisuus on, käsitellä sitä, miten moduulit toimivat marketplace-ympäristössä, ja mitkä laatuvaatimukset moduulin pitää täyttää päästäkseen Magenton viralliselle markkinapaikalle.

Avainsanat verkkokauppa, open source, avoin lähdekoodi, magento

Sivut 42 s.

HAMK
UNIVERSITY OF APPLIED SCIENCES

ABSTRACT

Visamäki
Degree Programme in Business Information Technology
System Engineering

Author Jukka Kartano **Year** 2013

Subject of Bachelor's thesis Development of a Magento module

ABSTRACT

Magento is a very popular open source e-commerce platform. By design Magento is highly modular and this modularity has allowed Magento to meet the needs of a wide variety of clients.

The writer of this thesis develops Magento web stores and this work includes programming new features for clients. Most of the new features which are implemented are implemented for one web store only and as they are they can't be quickly transferred to other webstores because modifications are required.

The goal of this thesis was to introduce Magento's modular structure and how modules function in it. After Magento's structure has been presented and the basic structure of a module within Magento is explained the feature called promotion products will be implemented as a module of its own.

All the goals set for this thesis were met and even though the modules were already familiar to the writer in many ways a lot of knowledge was gained by doing everything from scratch. The thorough studying of the modular structure of Magento increased the writer's skills and knowledge of Magento.

No further development ideas emerged as the topic was accurately defined. One possible further development would include Magento Connect which is the official marketplace for Magento modules.

Keywords Magento, open source, e-commerce, programming, module

Pages 42 p.

SISÄLLYS

1 JOHDANTO

Olen osakkaana ja ohjelmoijana Insolo Oy yrityksessä. Olen ohjelmoinut Magentolle useita asiakkaalle näkyviä lisäominaisuuksia, joista useimmat vaikuttavat kaupan ulkoasuun. Yrityksessä on tavoitteena kehittää jatkuvasti modulaarisempia ja paketoidumpia kokonaisuuksia, joita asentaa Magento-verkkokauppoihin. Kokemuksen kautta meille on kehittynyt näkemys siitä, minkälaisia ominaisuuksia kauppiaat yleensä verkkokaupaltaan haluavat.

Tähän mennessä olen tehnyt kaikki muutokset ja uudet ominaisuudet verkkokauppoihin asiakaskohtaisesti. Kehitystyön painopiste on kuitenkin siirtymässä modulaarisempaan suuntaan. Näin ollen myös yrityksessämme on syytä muuttaa toteutuksia modulaariseen muotoon kehitysajan nopeuttamiseksi. Modulaarisuuden johdosta ominaisuuksiin on helppo lisätä erilaisia kontrolleja loppukäyttäjälle. Asiakas saa lisää mahdollisuuksia vaikuttaa uusien ominaisuuksien toimintaan Magenton sisäänrakennettujen kontrollien avulla, eikä enää jokaista uutta muutosta tarvitse erikseen ohjelmoida, vaan uudet ominaisuudet voidaan lisätä itse moduulin asetuksiin.

Tässä opinnäytetyössä on valittu suhteellisen yksinkertainen mutta kauppiaalle tehokas ominaisuus toteutettavaksi. Kyseessä on yksittäisten tuotteiden nostaminen erilliseen listaukseen kategorianäkymässä. Tästä lähtien puhutaan moduulista nimellä promootiotuotteet. Asiakkaan selatessa verkkokaupan kategoriapuuta pidemmälle nostettavat ostettavat tuotteet rajautuvat samalla. Jokainen asiakkaan valitsema kategoria rajaa tällöin promootiotuotteiden määrää. Tällä hetkellä toteutus on ohjelmoitu asiakkaan verkkokauppaan käsin. Kun ominaisuus siirretään modulaariseen muotoon, on tarkoituksena antaa kauppiaalle samalla enemmän valintoja, joilla vaikuttaa promootiotuotteiden toimintaan omasta hallintapaneelistaan.

Opinnäytetyö käsittelee moduulin rakentamisen ohjelmistosuunnittelun näkökulmasta ja itse toteutuksen. Työssä selvitetään myös kaikki oleellinen, mitä Magentosta pitää moduulien toiminnan kannalta tietää.

Tämä opinnäytetyö on tilaisuus ylläpitää ammattitaitoani Magento-kehityksessä ja lisätä omaa tuottavuuttani. Kun olen tehnyt yhden moduulin, voin yhä helpommin siirtää muita ominaisuuksia modulaariseen muotoon. Magenton moduulit eivät ole olleet ennen tätäkään opinnäytetyötä vieraita, mutta tässä työssä teen ensimmäistä kertaa perusteellisesti alusta alkaen moduulin käyttämättä valmiita työkaluja tai kopioimatta olemassa olevia moduuleja.

2 AIHEEN ESITTELY JA PROJEKTIN TAUSTAT

Tässä luvussa esitellään projekti ja sen taustatiedot. Kerron taustastani yrityksessä ja siitä miten tämä moduuli lähti liikkeelle. Selvitän työssä yksityiskohtaisesti käyttämäni työkalut ja kehitysympäristöä, jossa ohjelmoin moduulia.

2.1 Toimeksiantaja

Toimin osaakkaana ja ohjelmoijana Insolo Oy yrityksessä, joka on erikoistunut Magento-verkkokauppojen kehittämiseen. Yrityksen brändi perustuu pitkälle räätälöityihin verkkokauppoihin, jotka ovat integroitavissa erilaisiin ohjelmistoihin, kuten taloushallintaohjelmiin. Pitkälle viety räätälöityvyys ja taloushallinto- integraatiot erottavat Insolo Oy:n Suomen muista Magento-tekijöistä.

2.2 Työn taustat ja historiaa

Jouluna 2012 kehitin asiakkaalle räätälöityä verkkokauppaa, johon tuli erikoisominaisuutena mahdollisuus nostaa yksittäisiä tuotteita erikseen kategorianäkymässä esille. Nämä tuotteet listattiin ennen varsinaisen tuotelistauksen alkua omassa osiossaan. Tuotteille pystyi myös antamaan mainostekstin. Olin jo hetken aikaa miettinyt sopivaa väliä opetella moduulien perusteellista rakentamista, ja tämä oli laadultaan sellainen moduuli, joka voisi kiinnostaa muitakin asiakkaita. Se oli myös juuri sopivan kokoinen toteutus opinnäytetyöksi.

Nykyisessä tilassa promootiotuotteiden toteutus ei ole kovin monimutkainen, mutta tarkoituksena on lisätä modulaarisen toteutuksen myötä useita ominaisuuksia ja hallintamahdollisuuksia kauppiasta varten. Lopputuloksena on monipuolinen ja tehokas moduuli kauppiaalle myynninedistämiseen.

2.3 Työvälineet ja kehitysympäristö

Projektin toteuttamisessa käytetään WinSCP-ohjelmaa, jolla pidetään yhteyttä auki palvelimelle. Kyseessä on SFTP-yhteys, joka sallii tiedostojen lähettämisen SSH:n yli. Tämä mahdollistaa sulavat ja nopeat muutokset palvelimen kaikkiin tiedostoihin. WinSCP:sta voidaan avata myös erillinen SHELL yhdellä näppäinyhdistelmällä. Ohjelmoimiseen käytettiin Notepad++ tekstieditoria, jossa on ohjelmointiin riittävät perusominaisuudet. Näistä perusominaisuuksista voi mainita esimerkkeinä. syntaksin korostuksen ja koodipalojen tallennuksen.

Kehitysympäristönä ei toimi yleisesti näissä yhteyksissä käytetty Wamp -virtualisointi, vaan yrityksen oma kehitysympäristö. Tämä kehitysympäristö on yksi Debian 6.0 virtuaalipalvelin. Kehitysympäristöön on asennettu 1.7.0.2 Magento oletusasetuksilla.

Työvälineistä vielä maininnan arvoinen on selaimen lisäosa Firebug, joka nopeuttaa web-kehitystyötä huomattavasti ja auttaa testaamaan tyylimuutoksia reaaliajassa. Lisäksi tällä voidaan debugata jossain määrin JavaScriptiä ja tarkkailla, mitä verkkoliikennettä sivut synnyttävät.

2.4 Ohjelmointi- ja merkkauskielet

Esittelen lyhyesti projektissa käytettävät ohjelmointi ja merkkauskielet. Tässä opinnäytetyössä ei mainita itsestäänselvyyksiä, kuten HTML:aa.

2.4.1 PHP

PHP on palvelinpuolella toimiva ohjelmointikieli, jota käytetään dynaamisten web-sivujen luontiin. PHP on varmaan yleisin käytössä oleva ohjelmointikieli webissä. PHP ei ole loppukäyttäjälle mitenkään näkyvissä vaan kaikki koodi suoritetaan palvelimen puolella. Koska PHP suoritetaan kokonaisuudessaan suoritusvaiheessa, PHP:ta voidaan yhdistää esimerkiksi JavaScriptiin.

2.4.2 JavaScript

JavaScript on loppukäyttäjän koneella suoritettua koodia, jolla voidaan esimerkiksi piilottaa elementtejä sivuilta päivittämättä itse sivua. Koska JavaScript toimii käyttäjän koneella, JavaScriptien käyttö mahdollistaa reaaliaikaisten toimintojen toteuttamisen käyttäjän toimintojen perusteella. Yksinkertaisena esimerkkinä tuote voitaisiin lisätä ostoskoriin päivittämättä sivua.

Magenton mukana tulee muutamia valmiita JavaScript-kirjastoja ja ohjelmistokehyksiä, joten ne toimivat lähtökohtana, kun lähdetään rakentamaan moduulia.

2.4.3 XML

XML eli Extensible Markup Language on merkintäkieli, jolla kuvataan tietoa niin, että sitä voidaan välittää järjestelmien välillä. Magento käyttää XML-tiedostoja kokoonpanotiedostoina sitoakseen moduulien osat yhteen, minkä jälkeen XML-tiedostoissa muodostetaan teeman rakenne. Tähän rakenteeseen kiinnitetään moduulit. Tämä on hyvin tehokas ja monipuolinen järjestelmä. Merkkaustapa, jota Magento suosii, on suhteellisen helppolukuista, mutta muutamia kikkoja pitää osata.

3 MAGENTO

Tässä luvussa esitellään taustatiedot Magentosta, Magenton historia ja minkälaisia versioita Magentosta on saatavilla. Magentosta on olemassa muutamia erilaisia versioita, joilla voidaan toteuttaa niin perusverkkokauppoja kuin Nokian ja Niken kokoisten yritysten vaatimukset täyttäviä isompia kokonaisuuksia.

3.1 Magenton historia

Magento on tämän hetken suosituin avoimen lähdekoodin verkkokauppa-alusta. Magenton suosio perustuu sen vahvaan yhteisöön, muokattavuuteen, lukuisiin lisäosiin ja valmiiksi kattaviin ominaisuuksiin, jotka tulevat Magenton mukana.

Magenton on kehittänyt Varien, joka aloitti 2001 web-ohjelmointiyrityksenä, joka keskittyi avoimen lähdekoodin järjestelmiin niiden muokattavuuden ja monipuolisuuden vuoksi. Kun Varien päätti keskittyä verkkokauppojen kehittämiseen, valinta oli avoimen lähdekoodin verkkokappa-alusta OS Commerce.

Verkkokauppojen kehittämisen myötä Varien alkoi kasvaa ja painopiste siirtyi kokonaan verkkokaupankäynnin kehittämiseen. Pian huomattiin tarve tehokkaammalle ja monipuolisemmalle avoimen lähdekoodin verkkokauppa-alustalle. Kun vaatimuksia täyttävää verkkokauppa-ohjelmistoa ei löytynyt valmiina, niin Varien päätti kehittää uuden verkkokauppa-ohjelmiston ja tästä syntyi Magento.

3.2 Magenton eri versiot

Magentosta on olemassa kolme erilaista versiota. Tässä kappaleessa käsitellään pääpiirteittäin jokainen näistä, ja selvitetään miksi tässä opinnäytetyössä on päädytty käyttämään yhteisöversiota. Kappaleessa selvitetään myös, paljonko kehittäjälle on väliä sillä, mitä versiota käyttää.

3.2.1 Magento Community

Tämä on ilmainen versio, joka on tarkoitettu yhteisön ylläpidettäväksi. Magenton virallisen määritelmän mukaan tämä versio ei sovellu tärkeisiin ja isoihin verkkokauppoihin, mutta käytännössä yhteisön avulla saatu tuki riittää ratkaisemaan hankalatkin ongelmat ja yhteisöversion päälle on rakennettu isoja kauppoja. Mahdollisten ongelmien ja kehitystyön määrä riippuu paljon käsin rakennettavan kaupan monimutkaisuudesta.

Tässä työssä käytetään Magenton yhteisöversiota 1.7.0.2. Magenton uudesta 2.x yhteisöversiosta on puhuttu pitkään mutta väliin tulee todennäköisesti vielä 1.8 versio.

3.2.2 Magento Go

Magento Go on SaaS-versio Magentosta, jota ylläpidetään Magenton omalla palvelimella. Etuna yhteisöversioon (Community) on suoraan Magentolta tuleva tekninen tuki ja se, että kauppa on valmiiksi palvelimella. Rajoitteena on rajattu pääsy Magenton lähdekoodiin, mutta SaaS-versiosta kiinnostuneet eivät todennäköisesti halua koskea Magentoon itse.

3.2.3 Magento Enterprise

Enterprise on yhteisöversiosta oma kehityshaaransa, jossa on paljon sellaisia ominaisuuksia, joita yhteisöversiossa ei ole, mutta jotka ovat yhteisöversioon lisäosien kautta toteutettavissa. Esimerkkeinä näistä ominaisuuksista bonuspisteet, asiakkaiden segmentointi markkinointia varten ja kokosivun välimuistin tuki. Magento antaa enterpriselle takuun, että se toimii ja vastaa isojen kauppojen tarpeita niin nopeudeltaan, tietoturvaltaan kuin ominaisuuksiltaan.

4 MAGENTON OSAT RAKENNE, TEEMAT JA MODUULIT

Tämä luku käsittelee osia, joista Magento on rakennettu ja selvittää miten Magento on rakennettu sekä teemoittamisen eli layoutin toimintapiirteet pääosittain. Lisäksi selvitetään, mistä Magenton moduulit löytyvät, ja miksi moduulit on jaettu kolmen eri kansion alle. Selvitetään myös hyviä ohjelmointikäytäntöjä ja kerrotaan pääpiirteittäin, miten moduuli ja teema sidotaan yhteen. Teemoittaminen olisi kokonaan oma opinnäytetyönsä, mutta moduuleja ymmärtääkseen pitää käsitellä myös teemoja. Kaikki tämä tieto auttaa ymmärtämään moduulien rakennetta ja sitä, miten Magento kokonaisuutena toimii. Lisäksi selvitän, mitä kaikkia työkaluja Magentossa on valmiina mukana. Magento on asennuksesta alkaen hyvin kehitysvalmis järjestelmä. Kehittäjät voivat säästää aikaa, kun katsovat mitä Magentossa on valmiina mukana, ja noudattavat muutamia yksinkertaisia ohjeita ja käytäntöjä Magentoon muutoksia tehdessään.

4.1 Ohjelmistokehykset (Frameworkit)

Magentossa tulee mukana muutamia ohjelmistokehyksiä jotka käsitellään tässä luvussa. Useat kehittäjät alkavat lisätä omia ohjelmistokehyksiä ja kirjastoja kehitysvaiheessa, vaikka Magenton mukana tulee valmiina hyvin kattava määrä valmiita kirjastoja ja ohjelmistokehyksiä.

4.1.1 Prototype

Prototype on JavaScript-ohjelmistokehys. Prototypeä käytetään osana tunnettua Ruby on Railsia. Prototype on toinen suosituista JavaScript-ohjelmistokehyksistä toisen ollessa jQuery. Periaatteessa jQuery ei ole ohjelmistokehys, mutta näiden kahden yhtäaikainen käyttäminen johtaa todennäköiseen konfliktiin, joka on helposti vältettävissä.

Kun ohjelmoidaan moduuliin asiakkaan puolella (client sidella) olevia interaktiivisia käyttöliittymätoiminnallisuuksia, niin nämä tulee tehdä Prototypella, eikä sisällyttää turhaan jQuerya.

Prototype on automaattisesti laajennettu kattamaan script.acu.ous-kirjaston, joka on sama asia kuin jQuery UI. Tässä kirjastossa on efektejä ja toimintoja käyttöliittymän toteutusta varten.

Frontend-kehityksen kannalta tämä on oleellinen tieto. Suurin osa toiminnallisuuksista tullaan kirjoittamaan palvelinpuolen PHP-kielellä. Tieto siitä, että prototype tulee Magenton mukana, on kuitenkin hyvin oleellinen kaiken kehitystyön kannalta. Monet kehittäjät lisäävät turhaan jQueryn, ja muita kirjastoja, joiden vastaava toiminnallisuus on jo olemassa. Tästä syystä voi tarpeettomien konfliktien selvittämiseen kulua aikaa, mikäli Magentoon kuuluvat ominaisuudet eivät ole täysin kehittäjälle selvillä.

4.1.2 Zend

Zend-ohjelmistokehys on avoimen lähdekoodin ohjelmistokehys PHP5-kehitykseen. Kaikki Zendin osat ovat täysin olio-ohjelmoinnin mukaisesti toteutettuja ja Zend tukee useimpia tietokantoja.

Zendin etu muihin PHP-ohjelmistokehyksiin nähden on se, miten tarkkaan ohjelmistokehys noudattaa Olio-ohjelmointimalleja. Tämä tarkoittaa sitä, että Zend on erittäin modulaarinen ohjelmistokehys, jossa on helppo kierrättää vanhaa koodia ja laajentaa jo olemassa olevia osia. Valmiina mukana tuleva kirjasto on laaja ja täysin käyttäjän laajennettavissa. Haittapuolena tässä on, että pienien uusien asioiden ohjelmoimiseksi pitää kirjoittaa paljon koodia.

4.2 MVC (Model, View, Controller) arkkitehtuuri

MVC on ohjelmistoarkkitehtuurityyli jonka tarkoituksena on erottaa käyttöliittymä sen taustalla toimivasta datasta. MVC:ssa ohjelma jaetaan kolmeen osaan: malli (model), käsittelijä (controller) ja näkymä (view).

Magentossa moduulien MVC:ssa on muutamia yleisestä MVC:sta olevia poikkeuksia, joita selvitetään opinnäytetyön luvussa 4.5. MVC ei ole käsitteenä tai käytäntönä kovin jäykkä vaan sen osa-alueilla on paljon tulkintaa.

4.2.1 Konventionaalinen ja kokoonpanotiedostoilla toimiva MVC

Magenton MVC-arkkitehtuuri on sidottu yhteen kokoonpanotiedostoilla. Tämä tarkoittaa sitä, että toisin kuin konventionaalisessa järjestelmässä, Magentolle pitää kertoa jokaisesta uudesta mallista ja käsittelijästä kokoonpanotiedostoissa. Nämä ovat xml-tiedostoja. Jokaisella moduulilla on oma kokoonpanotiedostonsa, joka kertoo Magentolle moduulin käyttämät käsittelijät, mallit ja näkymät. Konventionaalisessa MVC:ssa järjestelmä

hakee uudet luokat automaattisesti lukemalla tiedostojärjestelmästä esiasetettuja polkuja.

Esimerkki Moneybookers-moduulin kokoonpanotiedostosta

Otetaan esimerkki olemassa olevan yhteisömoduulin kokoonpanotiedostosta. En ota koko tiedostoa tähän, ainoastaan muutaman esimerkkikohdan, että idea tulee selväksi. Alla esimerkissä 1 tämä kohta.

Esimerkki 1. Moneybookers-moduulin kokoonpanotiedosto

```
<models>
   <moneybookers>
      <class>Phoenix_Moneybookers_Model</class>
   </moneybookers>
</models>
```

Moneybookers-moduulin config.xml-kokoonpanotiedosto kertoo Magentolle, mistä löytyvät kyseisen moduulin mallit (model) osat. Teksti **`Phoenix_Moneybookers_Model`** kertoo nimiavaruuden, josta Magento hakee mallit. Ensimmäinen osa on yhtiö eli Phoenix, toinen on moduulin nimi ja kolmas on itse moduulin osa, jonne halutaan viitata. Tässä tapauksessa Moneybookers-moduulin mallit sijaitsevat polussa `Phoenix/Moneybookers/Model` eli juuri se mitä on `<class>`-tagien sisällä. Kokoonpanotiedostoa käsitellään tarkemmin luvussa 6.1.

Kokoonpanotiedostoilla toimivan järjestelmän etuna on se, että jos jokin menee vikaan, ongelman selvittäminen on huomattavasti helpompaa, koska moduulin jokainen osa pitää kertoa järjestelmälle käsin kokoonpanotiedostoissa, sen sijaan että konventionaalinen MVC poimii kaikki luokat automatiikalla tiedostopoluista.

4.2.2 Malli (Model)

Mallissa on tiedot siitä, miten ohjelma tallentaa tietoja. Tähän sisältyy tietokantakuvaukset ja kyselyt. Käytetään esimerkkinä yksinkertaista lineaarista laskinta, lineaarisen laskimen nappia painettaessa, kasvaisi laskimen näytöllä oleva luku yhdellä. Tämä lineaarisen laskimen ohjelman malli muistaa siis tiedon, monesko luku on menossa, ja kuinka paljon lukuun lisätään kun nappia painetaan. Lisäksi mallista löytyy tieto, minne lineaarisen laskurin numero päivittämisen yhteydessä tallennetaan. Tässä luvussa esiteltyä esimerkkiä lineaarisesta laskimesta jatketaan MVC:n muita osia käsiteltäessä.

4.2.3 Käsittelijä (controller)

Käsittelijä tarkkailee käyttäjän syöttämiä arvoja käyttöliittymän näkymään (view) ja välittää ne mallille samalla huolehtien siitä, että päivitetty tieto näkyy käyttäjälle. Palataan mallissa esiteltyyn lineaariseen laskimeen. Tässä tapauksessa eli lineaarisessa laskimessa käsittelijä tarkkailee lineaarisen laskurin nappia. Kun käyttäjä painaa lineaarisen laskurin nappia, käsittelijä välittää tiedot mallille, että malli päivittää oikeat tiedot.

4.2.4 Näkymä (view)

Näkymä esittää mallin käsittämät tiedot käyttäjälle ja tarjoaa käyttöliittymän näiden muokkaamiseen. Näkymä ei tiedä mitään mallista, vaan käsittelijä hakee tiedot mallilta, jonka jälkeen ne esitetään näkymässä. Käsittelijä kuuntelee muutoksia näkymässä ja välittää ne mallille.

4.3 Magenton tiedostorakenne

Tässä kappaleessa käydään läpi Magenton tiedostorakenne, mitä löytyy mistäkin ja miksi Magenton osat on sijoitettu, kuten ne on sijoitettu. Tarkoituksena ei ole käsitellä koko Magenton tiedostorakennetta vaan ainoastaan moduulin ja Magenton ohjelmoinnin kannalta oleellisimmat rakenteet ja hakemistot.

4.3.1 App-kansio

Magento-asennuksen juurta selatessa löytyy kansio nimeltä app. App-kansion alla on kaikki Magenton toiminnallisuudet ja teemat. Mitä app-kansiossa ei ole, on teemojen eri tyylitiedostot (css-tiedostot) ja javascript-kirjastot. Magentossa kehitystyötä tehdessä suurin osa ajasta toimitaan app-kansion alla. Magenton muut kansiot eivät ole kovin oleellisia tämän opinnäytetyön kannalta, mutta kaikki mitä löytyy app-kansion alta on.

4.3.2 Ydin ja paikallisten moduulien ero - hakemistot app/code/core ja local

Core-kansioon on sijoitettu kaikki Magenton ohjelmallinen toiminnallisuus moduulien mukaan. Moduuleja voi sijoittaa kolmeen eri kansioon, joilla on keskinäiset prioriteetit. Ensimmäisenä on core, jonka alla sijaitsevat Magenton ydintiedostot ja toiminnallisuudet. Kehittäjän on tärkeää ymmärtää, että Magenton ytimeen eli app/code/coren alle ei saa missään tilanteessa tehdä muutoksia.

Jos Magenton ydintä mennään muokkaamaan, Magento ei ole enää automaattisesti päivitettävissä, ja tämä tietää yhteensopivuusongelmia. Monet suhteellisen kokeneetkin Magento-kehittäjät tekevät muutoksia suoraan Magenton ytimeen. Työpaikallani on yllättävän usein korjattu muiden kehittäjien jättämiä jälkiä ydintiedostoihin.

Kauppakohtaisia muutoksia ohjelmoidessa muutokset pitää tehdä local-kansion alle. Kaikki, mikä sijaitsee local-kansiossa, ylittää core-kansion sisällön. Kun muutoksia tehdään Magentoon tätä kautta, yhteensopivuus säilyy ja kehitystyössä tulevia ongelmia on huomattavasti helpompi ratkaista.

Esimerkki
Magento 1.7.0.2 haukutoiminnossa on sql-kysely sellaisessa muodossa, jonka johdosta haku ei aina käyttäydy johdonmukaisesti. Jotta hausta voidaan tehdä tarkempi, pitää muokata Magenton ydintiedostoa, jonka polku on:

```
app/code/core/Mage/CatalogSearch/Model/Resource/Fulltext.ph
p
```

Tästä polusta huomataan, että tiedosto sijaitsee coren alla, jolloin se on Magenton ydintiedosto. Polku myös kertoo, että toiminnallisuus liittyy katalogin hakutoimintoihin ja kyseessä on MVC:n Malli-osa.

Jotta haku voidaan korjata ja yhteensopivuus päivitysten välissä säilyttää, luodaan local-kansion alle samanlainen polku, jonne muokattu tiedosto Fulltext.php sijoitetaan. Tämä polku näyttää tältä:

```
app/code/local/Mage/CatalogSearch/Model/Resource
```

Kun local-kansioon sijoitetaan muokattu Fulltext.php, niin tämä local-kansiossa sijaitseva moduulin malli-osa ohittaa prioriteettina Magenton ydinmoduulin vastaavan tiedoston, ja Magento siirtyy käyttämään korjattua Fulltext.php:ta, joka sijaitsee local-kansiossa.

Useat Magento-kehittäjät muokkaavat virheellisesti ydintiedostoja kun yllä oleva tapa on oikeaoppinen tapa hoitaa koodimuutokset. Se on myös yleisesti Magenton dokumentaatiossa kerrottu tapa ylittää Magenton ydinmoduuleja.

4.3.3 Yhteisön moduulit - hakemisto app/code/community

Polussa app/code/community on yhteisön luomat lisämoduulit. Kun Magenton sisäänrakennetulla lisäosien hallinnalla Magento connectilla asennetaan lisämoduuleita, ne tulevat community-kansion alle.

Community-kansiossa sijaitsevat toiminnallisuudet ohittavat core-kansiossa olevat, mutta eivät mene paikallisten (local-kansio) ohi.

Tässä opinnäytetyössä tehtävä moduuli voisi sijaita local-kansion alla tai se voi sijaita community-kansion alla. Ainoa paikka, missä moduuli ei voi sijaita on ytimessä eli app/code/core-polussa. Esimerkiksi kaikki lisämoduulit, joita yritys tekee, tulisivat tänne yrityksen nimellä. Silloin promootiotuotteet-moduulin sijainti näyttää tältä:

```
app/code/community/Insolo/InsoloPromotions
```

4.3.4 Moduulirekisteri - hakemisto app/etc/modules

Magento lukee oletuksena kaikki tämän kansion xml-tiedostot. Tämä on niin sanottu moduulirekisteri Magentolle, jos moduulia ei ole olemassa täällä, niin Magento ei voi löytää sitä. Jokainen moduuli on listattuna omina xml-tiedostoinaan. Xml-tiedosto on hyvin yksinkertainen ja siitä löytyy tiedot moduulin tilasta eli tiedot onko moduuli käytössä vai pois päältä. Sekä tiedot myös siitä sijaitseeko moduulin koodi community- vai local-polussa. Moduulin versio ja mahdolliset riippuvuudet myös kerrotaan tässä XML-tiedostossa.

Esimerkki 2. Aschroder_SMTPPro.xml

```
<?xml version="1.0"?>
<config>
    <modules>
        <Aschroder_SMTPPro>
            <active>true</active>
            <codePool>community</codePool>
        </Aschroder_SMTPPro>
    </modules>
</config>
```

Itse XML-tiedoston nimi voi olla mikä vain, mutta sisällössä moduulin nimi pitää olla oikein.

Tämä aihe käsitellään tarkemmin luvussa 6. Tämä on siis nopea paikka tarkistaa kaikki Magentoon asennetut moduulit, ja niiden tila, jos hallintapaneelista ei jostain syystä ole mahdollista sitä tehdä. Lisäksi jos moduulissa menee jokin vikaan tai Magentossa pitää muuten tehdä ongelmanratkaisua, joka ei mahdollista hallintapuolelle kirjautumista, niin tätä kautta moduuleja voi ottaa pois päältä manuaalisesti.

4.3.5 Käännöstiedostot - hakemisto app/locale

Locale-kansion alla sijaitsevat käännöstiedostot. Kun moduulia kirjoitetaan, kannattaa käyttää Magenton sisäänrakennettua kääntäjää tekstien esittämiseen, jolloin moduulin antama teksti kääntyy helposti useammallekin kielelle, ilman että joka kerta muokataan moduulin view/näkymä -tiedostoa.

Moduulille voidaan määrittää oma käännöstiedosto tai sitten voidaan käyttää jo valmiina olevia käännöstiedostoja. Käännöstiedostojen vaikutusta käydään tarkemmin läpi moduulia toteuttaessa.

4.3.6 Teeman tiedostot - hakemisto app/design

Designin alta löytyvät magenton teeman tiedostot. Magentossa on olemassa kaksi eri näkymää, joita voidaan teemata. Nämä näkymät ovat frontend ja adminhtml. Frontend on siis asiakkaalle näkyvä puoli ja adminhtml on backend eli hallintapuoli, joka näkyy kaupan ylläpitäjälle. Kun klikataan frontend auki, saadaan kaksi alikansiota default ja base. Tässä on sama

idea kuin moduuleiden kanssa, joissa on siis kolme eri tasoa; core, community ja local. Teemoilla on kaksi eri tasoa.

Jos Magento ei löydä teemasta jotain tiedostoa, jota tarvitaan, Magento siirtyy käyttämään base-kansion alta löytyviä tiedostoja. Tällöin Magento hakee teeman tiedot polusta frontend/base/default.

Kun Magentolle määritellään oletusteemaa hallintapuolella, kaupalle pitää kertoa, käyttääkö se teemaa, joka sijaitsee defaultin tai basen alla. Useimmat jälkeenpäin asennettavat teemat menevät defaultin alle. Jos jälkeenpäin asennettavasta teemasta ei löydy jotain tiettyä osaa, Magento ei mene rikki vaan siirtyy käyttämään base:n alla olevia tiedostoja.

Moduulia luotaessa, tiedostot voidaan luoda joko base- tai default -kansion alle. Kyseessä on pitkälti kehittäjän oma mieltymys minne tiedostot perustetaan. Kuten Magenton ydinmoduulien kanssa, niin ei ole suositeltavaa muokata base:n alla olevia Magenton tiedostoja, vaan niistä pitää tehdä kopiot defaultin alle. Näin siis, jos muokkaa olemassa olevan teeman tiedostoja sen sijaan että luo uusia.

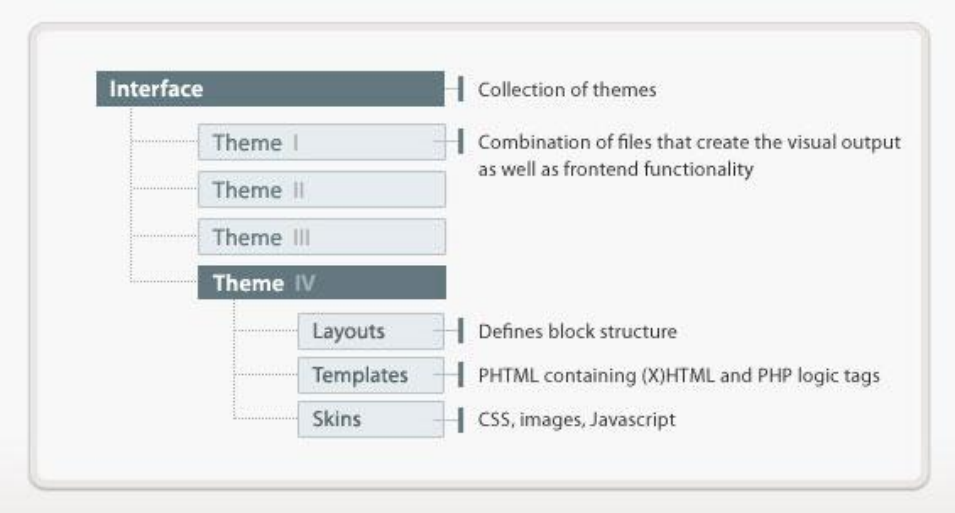
4.4 Magento-layout toimintaperiaatteet ja moduulien asemointi teemaan

Tässä kappaleessa esitellään Magenton layoutin toimintaa ja miten layouteihin voi tehdä muutoksia muokkaamatta itse teeman koodia. Tieto siitä, miten Magento teemoitetaan, on tärkeää kaiken kehitystyön kannalta, koska muuten etenkin aloitteleva kehittäjä tulee liian helposti poistaneeksi toiminnallisuuksia huonoilla menetelmillä, esimerkiksi piilottamalla ne tyylitiedostoista tai kommentoimalla ulos teeman template-tiedostoista. Teemojen toiminnan hahmottaminen ja läpikäyminen auttaa ymmärtämään, miten moduulien ja teemojen suhde Magentossa toimii.

Moduulit muuttavat teemoja omilla xml-tiedostoillaan, joissa on tiedot siitä, missä moduuli haluaa tulla esitetyksi. Magento lukee xml-tiedostot järjestyksessä, jonka mukaan koostetaan teemassa näytettävät moduulit ja myös, missä moduulit näytetään. Järjestelmä on erittäin monipuolinen ja tehokas.

4.4.1 Mistä teema koostuu?

Magentossa teema on yhdistelmä xml-tiedostoja ja .phtml template -tiedostoja jotka tyylitellään skin-kansiossa sijaitsevilla tyylitiedostoilla. Tässä kappalessa käsitellään pääpiirteittäin, mistä Magenton teema on rakennettu. Pelkästä teemasta ja sen luomisesta voisi tehdä jo oman työnsä kokonaan. Tämän työn kannalta käsitellään sen verran verran teemoja kuin se auttaa ymmärtämään moduulien toimintaa Magentossa. Kuvassa 1 näkyy teeman perusrakenne Magentossa.



Kuva 1. Magenton teeman osat

Interface
Interface on kokoelma teemoja (Kuva 1). Magenton hallintapuolella käytettävä Interface valitaan nimellä package (paketti). Base on oletus, jota Magento käyttää, jos käyttäjän määrittelemästä teemasta ei löydy jotain tarvittua osaa. Tämä siis riippumatta siitä, minkä paketin tai liittymän käyttäjä valitsee.

Layouts
Layoutit (Kuva 1) ovat XML-tiedostoja, joiden mukaan sijoitetaan moduulien template-tiedostot teemassa valmiiksi määriteltyihin paikkoihin. Nämä valmiiksi määritellyt paikat ovat rakennelohkoja, joita käsitellään enemmän luvussa 4.4.2.

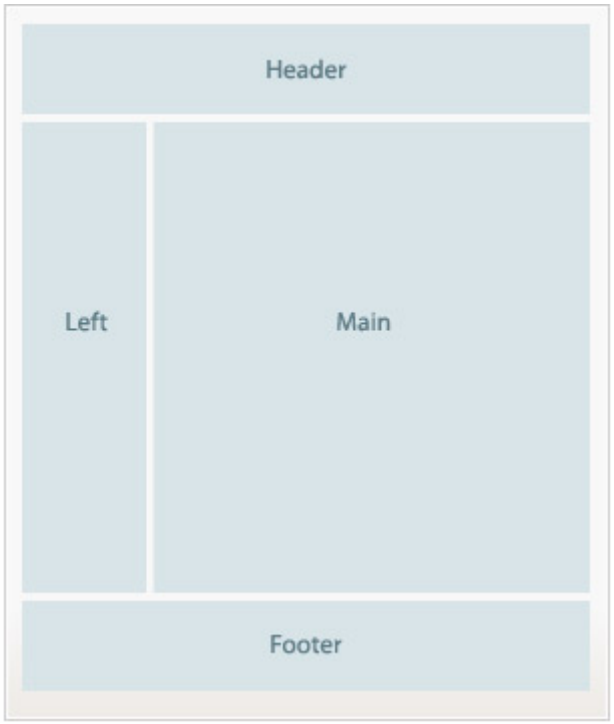
Jokaisen moduulin mukana tulee oma layout-tiedosto, jonka Magento lukee ja sen perusteella muodostetaan muutokset teemaan niin hallintapuolelle kuin asiakaspuolelle.

Templatet
Templatet kattavat html-, php- ja javascript-tiedostot ja ovat asiakkaalle näkyvä osa kaupasta. Tämä on siis käyttöliittymä, joka kootaan loppukäyttäjälle eli verkkokaupan asiakkaalle.

Skins
Skin-kansiossa sijaitsevat tyylitiedostot (CSS), kuvamateriaalin ja javascript-tiedostoja.

4.4.2 Rakennelohkot (structural blocks)

Magentossa on kahdenlaisia lohkoja, joita käytetään, kun moduulit sijoitellaan näytettäväksi. Structural block eli rakennelohko, on lohko johon sisältölohkot (content blocks) viittaavat, ja jonka perusteella sisältölohkon sisältö tulee esiintymään sille osoitetussa rakennelohkossa. Kuvassa 2 näkyy useimpien teemojen mukana tulevat valmiit rakennelohkot, joihin voidaan viitata. Kuvan lohkojen lisäksi 3-kolumnin malleissa on myös oikea kolumni, johon voidaan sisältölohkoja viitata.



Kuva 2. Rakennelohkot, joihin viitataan sisältölohkoilla

Rakennelohkot määritellään teeman pohja-templatessa, joka käsittää koko sivun perustan. Esimerkki 3 on esimerkki hyvin yksinkertaisesta pohja-templatesta. Esimerkissä 3 on siis seuraavat rakennelohkot, joihin sisältölohkoja voi sijoittaa; header, left, content ja footer.

Esimerkki 3. Magenton pohja-template, jossa rakennelohkot paikallaan.

```
<html>
<head></head>
<body>
<div class="header"><?=$this->getChildHtml('header')?></div>
<div class="middle">
    <div class="col-left"><?=$this->getChildHtml('left')?></div>
    <div class="col-main"><?=$this->getChildHtml('content')?></div>
</div>
<div class="footer"><?=$this->getChildHtml('footer')?></div>
</body>
</html>
```

4.4.3 Sisältölohkot (content blocks)

Sisältölohkoja sijoitetaan edellä esitettyihin rakenteellisiin lohkoihin layout-tiedostoissa. Nämä rakenteelliset lohkot on esitetty kuvassa 3 alla. Joka kerta kun halutaan päättää, missä tietty moduuli esiintyy, mennään moduulin layout-tiedostoon, jossa vaihdetaan viittausta, missä sisältölohkossa moduulin halutaan esiintyvän.

Kuva 3. Sisältölohkot sijoitettuna rakennelohkoihin.

4.4.4 Esimerkki newsletter moduulin layout-tiedostosta

Jos halutaan vaihtaa newsletter-moduulin esityspaikka verkkokaupassa tai ottaa newsletter-moduuli pois näkyviltä, pitää muokata newsletter-moduulin layout-tiedostoa polussa:
app/design/frontend/default/modern/layout/newsletter.xml

Tässä ei siis muokata newsletter-moduulin asiakkaalle näkyvää osaa eli templatea, vaan käsittellään moduulin layout-tiedostoa. Käsitellään tämä layout- tiedosto Esimerkissä 4, ja käydään läpi opinnäytetyön kannalta olennaisin sisältö.

Esimerkki 4. Newsletter-moduulin layout-tiedosto

```
<?xml version="1.0"?>
<layout version="0.1.0">
<default>

<!-- Mage_Newsletter -->
<reference name="footer">
<block type="newsletter/subscribe" name="newsletter"
as="newsletter" template="newsletter/subscribe.phtml"/>
</reference>

</default>
</layout>
```

`<default>` On layout-handle, joka tarkoittaa sitä, että tämä moduuli ladataan useimmilla sivuilla. Jos halutaan, että moduuli ladataan eri tavalla jollakin muulla sivulla tämä `<default>` pitää vaihtaa. Esimerkiksi `<cms_index_index>` on handle, joka lataisi moduulin ainoastaan etusivulla.

Sen lisäksi, että layoutin handle vaihdetaan, pitää moduulille määrätä uusi sijainti teemassa. Oletuksena newsletter sijaitsee footerissa, kuten layout-tiedostosta voi pätellä: `<reference name="footer">`. Reference name tarkoittaa sitä rakennelohkoa, johon sisältölohko viittaa. Useimmissa teemoissa on olemassa seuraavat rakennelohkot eli sijainnit; content, left, right, top, header. Yhtenä vaihtoehtona siis esim. `<reference name="left">`

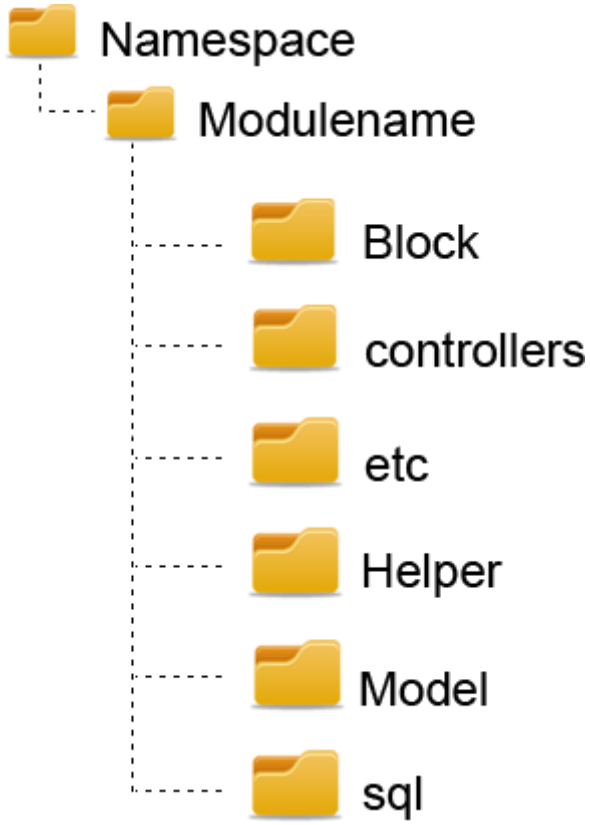
`<block>` tästä alkaa itse sisältölohkon tiedot.
`<block type>` on sisältölohkon tyyppi, kyseinen newsletter/subscribe on Magenton sisäänrakennettu tyyppi. `name` on sisältölohkon nimi. Nämä eivät ole kovin oleellisia tässä yhteydessä. `Template` viittaa itse moduulin template-tiedostoon, joka esitetään käyttäjälle.

Muita asioita, mitä layoutille voi kertoa on missä järjestyksessä tämä sisältölohko esitetään. Esimerkiksi footerissa voi olla vierekkäin useampi sisältölohko, jolloin kyseiseen xml:aan voidaan sisällyttää `before="toinen_lohko"` joka tarkoittaa, että kyseinen lohko esitetään teemassa ennen lohkoa nimeltä ”toinen_lohko”.

Tällaisilla muutoksilla voidaan helposti ja nopeasti vaikuttaa siihen, miten ja missä moduuleja esitetään Magentossa.

4.5 Moduulin rakenne

Tähän mennessä on käsitelty sitä, miten moduulit asettuvat teemaan. Kun nyt on kuvattu teemojen ja moduulejen suhdetta selväksi, ja mitä kaikkia työkaluja on saatavilla moduulin ohjelmoimiseksi. Seuraavaksi käsitellään itse moduulin rakennetta tarkemmin. Kuvassa 4 on esitetty moduulin hakemistorakenne.



Kuva 4. Moduulin rakenne

Moduulin rakenne on MVC-mallin mukainen eli siitä löytyy Malli (Model), Näkymä (Block) ja Käsittelijä (Controller). Moduulissa namespace viittaa nimiavaruuteen, joka moduulille annetaan. Tässä tapauksessa se voi olla yrityksen nimi, jonka alle sijoitetaan moduulit. Itse moduulin tiedostopolku, jonne tämä moduuli sijoitetaan, on selvitetty aiemmin tässä opinnäytetyössä. Alla selvitetään jokainen moduulin kansio erikseen, ja mikä rooli niillä on moduulin toiminnassa.

4.5.1 Block-hakemisto

Blockien tarkoitus on yhdistää mallin (Model) tarjoamat tiedot teeman template-tiedostoihin, jotka ovat siis teeman-kansiossa sijaitsevia .phtml-loppuisia tiedostoja. Näillä tiedostoilla esitetään ne tiedot, mitkä näkyvät asiakkaalle.

Block ei ole MVC:n mukainen täydellinen vastike View/Näkymälle vaan Block enemmänkin sitoo moduulin teeman tiedostossa olevaan view-tiedostoon. Block koostuu siis funktioista, joita templaten view-tiedostot

kutsuvat esittääkseen tietoa, jonka seurauksena teeman view-tiedostojen ei pidä tietää mitään moduulin mallista, jota kautta tiedot haetaan.

4.5.2 Controllers-hakemisto

Nämä ovat käsittelijöitä, jotka vastaavat business-logiikan toimituksesta muualle Magentoon ja moduulin sisällä. Tätä kautta ajetaan malliin päivityksiä, jotka tulevat näkymän kautta käyttäjän antamina.

Jokaisessa PHP:lla toteutetussa järjestelmässä pääasiallinen piste, josta tietoa esitetään, on index.php. Itse index.php:hen ei kuitenkaan koskaan kirjoiteta ohjelmalogiikkaa vaan MVC:lla toteutetussa järjestelmässä index.php kutsuu koodia, joka tutkii seuraavat asiat.

1. Tutkii URL:n
2. Ennalta määrättyjen sääntöjen perusteella kääntää osoitteesta Käsittelijä-luokan ja toiminta-metodin. Tätä kutsutaan routingiksi eli reitityksekseksi.
3. Alustaa käsittelijä-luokan ja kutsuu toiminta-metodin. Tätä kutsutaan dispatchingiksi eli lähettämiseksi.

Esimerkiksi seuraava URL:
http://www.magentokauppa.fi/catalog/category/view/id/21

Käsittelijä tutkii domainin jälkeen urlin seuraavalla tavalla.

```
1 Osa. Front controller
http://www.magentokauppa.fi/catalog/category/view/id/21
```

Ensimmäinen osa on front-controller eli catalog. Tämä tarkoittaa, että Magento alkaa hakea moduulin catalog-käsittelijää. Catalog-moduuli sijaitsee polussa app/code/core/Mage/Catalog

```
2 Osa. Controller eli käsittelijä
http://www.magentokauppa.fi/catalog/category/view/id/21
```

Toinen osa kertoo, mitä käsittelijää moduulista pitää käyttää. Tässä tapauksessa kyseessä on category-käsittelijä. Kaikki käsittelijät, joita moduulille luodaan, pitää nimetä muodossa nimiController.php. Tässä tapauksessa moduulin käsittelijä sijaitsee polussa
app/code/core/Mage/Catalog/controllers/CategoryController.php

```
3 Osa. Toimintametodi
http://www.magentokauppa.fi/catalog/category/view/id/21
```

Kolmas osa on itse toiminta-metodin nimi, tässä tapauksessa view eli halutaan esittää kategoria.

```
4 Osa. Parametrit
http://www.magentokauppa.fi/catalog/category/view/id/21
```

Kaikki toiminta-metodia seuraavat tiedot osoitteessa lasketaan olevan arvopareja. Tässä tapauksessa GET-muuttujan nimi on id ja sen arvo on 21.

4.5.3 Etc-hakemisto

Etc-hakemiston alla on moduulin kokoonpanotiedostoja. Kokoonpanotiedostossa config.xml kerrotaan Magentolle, mistä löytyvät moduulin mallit, blockit, helperit jne. Ainoa vaadittu tiedosto, on config.xml, jonka pitää näyttää minimissään alla olevan esimerkki 5:n mukaiselta.

Esimerkki 5. config.xml -tiedosto, minimissään nämä tiedot.

```
<config>
  <modules>
      <Namespace_Modulename>
          <version>0.1.0</version>
      </Namespace_Modulename>
  </modules>
</config>
```

Config.xml on moduulin pääasiallinen kokoonpanotiedosto, jossa kerrotaan, mistä löytyvät moduulin eri osat, riippuvuudet ja käännöstiedostot, jos niitä on. Lisäksi jos moduulilla on omia javascript-kirjastoja, niihin pitää myös viitata tässä. Tyylitiedostot, joita moduuli käyttää, kerrotaan myös config.xml:ssa. Muita kokoonpanotiedostoja, joita moduulilla voi pakollisen config.xml:n lisäksi olla, ovat system.xml ja adminhtml.xml

Siinä missä config.xml, kertoo moduulin kokoonpanotiedot, eli mistä löytyy mikäkin MVC:n osa. System.xml kertoo tarvittaessa moduulin vaatimia asetuksia, kuten tekstikenttiä tai dropdown-asetuksia. Näillä asetuksilla voidaan sitten vaikuttaa moduulin toimintaan kaupan hallintapuolelta.

Esimerkki 6:ssa alla on esimerkki system.xml, joka lisäisi hallintapuolelle tyhjän välilehden nimeltä "Tämä on välilehti". Tätä aihetta käsitellään tarkemmin moduulin toteuttamisvaiheessa luvussa 6.

Esimerkki 6. System.xml, jossa määritellään hallintapuolelle välilehti.

```
<config>
   <tabs>
      <moduuli translate="label" module="module">
         <label>Tämä on välilehti</label>
         <sort_order>100</sort_order>
      </moduuli>
   </tabs>
<config>
```

4.5.4 Helper-hakemisto

Helpereihin voidaan sijoittaa metodeja, joista on hyötyä myös moduulien ulkopuolella. Kaikki metodit, jotka kirjoitetaan helpereihin, ovat jokaisen templatetiedoston, blockin, mallin tai käsittelijän käytettävissä. Moduulin ulkopuolelta pääsee siis aina moduulin helpereihin käsiksi. Yleensä ottaen helperit eivät koskaan käsittele tietoa, kuten mallit, ainoastaan esittävät sitä.

Helppo esimerkki helperistä on Magenton ytimessä oleva tekstin lyhennyksessä käytettävä helperi truncate, joka esitellään esimerkissä 7 alla.

Esimerkki 7. Truncate-helper

```
Mage::helper('core/string')->truncate($mystring,200)
```

Useimmat helperit perivät Magenton ytimen helperit luokasta Mage_Core_Helper_Abstract.

4.5.5 Model-hakemisto

Yleisesti MVC:ssa malli/model yhdistää suoraan tietokantaan ja käsittelee dataa, mutta Magentossa mallit eivät käsittele tietokantaa suoraan, eivätkä suorita edes suoria SQL-kyselyitä. Magentossa käytetään ORM:aa eli Object-Relational Mappingia, jonka johdosta tietokannan ja ohjelmoijan välille syntyy yksi kerros lisää abstraktiota.

ORM:ssa ohjelmoija käsittelee tietokannan rivejä olioina, tämän johdosta ohjelmoijalle ei näy, eikä ohjelmoijan tarvitse suorittaa perinteisiä SQL-kyselyitä. Tämä tekee tiedon validioimisesta helpompaa, ja nopeuttaa tiedon hakua tietokannasta, koska perinteisiä SQL-lauseita ei tarvitse kirjoittaa. Magenton mukana tulee omat ohjelmakirjastot, joilla tietokannat näyttäytyvät olioina järjestelmässä, jota kautta kehittäjä voi manipuloida tietokantaa. ORM itsessään olisi jo yhden opinnäytetyön arvoinen asia, mutta pääasia on se, että Magenton mallit eivät itsessään tee suoria SQL-kyselyitä. Seuraavassa esimerkki siitä miten Magenton model käsittelee tietokantaa ORM:n kautta.

Esimerkki 8. Magenton model ja ORM

```
$model = Mage::getModel('catalog/product')->load(27);
$price = $model->getPrice();
$price += 5;
$model->setPrice($price)->setSku('SK83293432');
$model->save();
```

Tässä esimerkissä ladataan katalogista tuote jonka ID on 27. Haetaan tuotteen hinta, ja lisätään siihen 5 €. Tämän jälkeen päivitetään ladatun olion tiedot, setPrice ja setSku –metodeilla. Lopuksi tallennetaan muutokset.

4.5.6 Sql-hakemisto

Sql-hakemiston alla on sql-asennuskriptat. Jos moduuli vaatii omia tauluja tietokantaan, luontiskriptat sijoitetaan tänne, ja ne osoitetaan moduulin kokoonpanotiedostossa.

4.5.7 Etc/modules-hakemisto

Itse moduulin kokoonpanotiedoston lisäksi Magentolle pitää ilmoittaa moduulin olemassaolosta etc/modules -hakemistossa xml-tiedostona. Tässä tiedostossa kerrotaan Magentolle moduulin nimi, ja mistä kokonaisuudesta, communitysta vai localista moduulin kokoonpanotiedostot haetaan. Itse xml-tiedoston sisältö on käsitelty aiemmin tässä opinnäytetyössä.

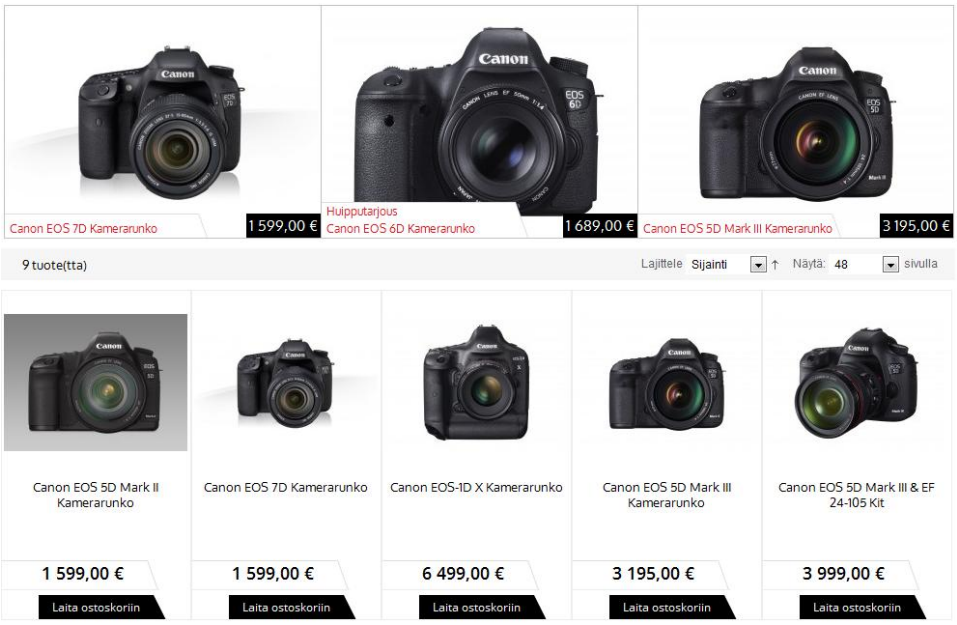
5 PROMOOTIOTUOTTEET-MODUULIN SUUNNITTELU

Tähän mennessä on selvitetty, mistä osista Magenton moduulit on tehty, ja miten ne elävät Magenton teemassa. Tässä luvussa aloitetaan itse toteutettavan moduulin suunnittelutyö. Käydään läpi, miten promootiotuotteet toimivat nykytilassa, ja miten tämä vaikuttaa moduulin suunnitteluun. Mitä sellaisia asioita moduulin pitää tehdä, mitä nyt on tehty käsin, ja mitä lisäominaisuuksia modulaarinen muoto mahdollistaa.

5.1 Promootiotuotteiden nykyinen toteutus

Ennen kuin aletaan suunnittelemaan modulaarista muotoa, käydään läpi miten promootiotuotteet on tällä hetkellä ei-modulaarisessa muodossa toteutettu. Tästä tulee selväksi, mitä haetaan, ja mitä rajoitteita nykyinen toteutustapa asettaa. Kun puhutaan promootiotuotteiden nykytilasta, pitää painottaa sitä, että tämä on tehty ainoastaan yhden asiakkaan tarpeita vastaamaan. Nykyisestä toteutuksesta puuttuu useita asioita, joita voitaisiin pitää itsestään selvyyksinä. Promootiotuotteiden toteutusta ei voi siirtää 1:1 sellaisenaan moduuliksi, vaan tarkoitus on laajentaa promootiotiotuotteiden-toimintaa huomattavasti.

5.1.1 Asiakkaan näkymä (frontend)



Kuva 5. Näin promootiotuotteet näkyvät kaupassa

Kuvassa 5 näkyy kolme promootiotuotetta, jotka nostetaan erikseen esille normaalista tuotelistauksesta. Normaali tuotelistaus alkaa kategorian lajittelutyökalun jälkeen.

Promootiotuotteet ovat kategoriakohtaisia, tuote tulee näkymään promootiotuotteena siinä kategoriassa, mihin tuote on asetettu. Magentossa kategorian voi asettaa näyttämään alakategorian tuotteet automaattisesti, vaikka tuote ei kuuluisikaan kyseiseen kategoriaan. Tämän johdosta promootiotuotteet tarkentuvat asiakkaalle sitä mukaan, mitä ylemmäksi asiakas kategoriapuussa liikkuu.

Kuva 6. Promootiotuote ilman mainostekstiä ja mainostekstin kanssa

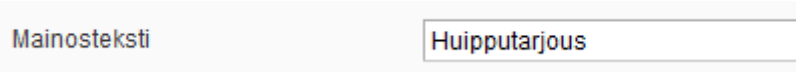
Kuvassa 6 esimerkki siitä, miten kauppias voi määrittää promootiotuotteelle mainostekstin hallintapuolella. Jos mainostekstiä ei ole asetettu, niin tuotteen tyylitelty näkymä kategorialistauksessa muuttuu sitä mukaa.

5.1.2 Kauppiaan näkymä (backend)

Kauppiaalle on perustettu kauppaan käsin kaksi uutta tuoteattribuuttia (moduulissa tämä pitää tehdä ohjelmallisesti), joita promootiotuotteet-moduuli käyttää. Kuvassa 7 kauppias on kirjautunut hallintaan, ja siirtyy tarkastelemaan tuotetta, jonka hän haluaa asettaa promootiotiotuotteeksi. Itse Magenton tuoteattribuutteihin ja attribuuttien kokonaisuuksiin ei tässä opinnäytetyössä ole tarpeellista tarkasti puuttua.

Kuva 7. Tuotehallinnan osa, josta tuotteen voi asettaa promootiotuotteeksi

Kuvassa 8 näytettävä tuotteen mainosteksti on mitoitettu asiakkaan tarpeiden mukaan, ja tässä tapauksessa ohjeena oli 1-2 sanaa. Jos tässä ei ole mitään tekstiä, promootiotuotteet-moduuli katsoo, että tälle promootiotuotteelle ei ole asetettu mainostekstiä. Eli tälle attribuutille ei ole erikseen alasvetovalikkoa, jossa kerrotaan, onko mainosteksti käytössä vai ei. Tarkistus suoritetaan ohjelmallisesti, ja jos attribuutti on täytetty, se on käytössä.



Kuva 8. Tuotteen mainosteksti

5.1.3 Promootiotuotteiden nykyinen asettelu Magenton teemaan

Promootiotuotteet tällaisenaan on suhteellisen nopeasti siirrettävissä toiseen kauppaan, mutta nykyinen järjestely on kaukana parhaasta mahdollisesta. Tuoteattribuutit pitää perustaa käsin ja liittää erikseen tuotekokonaisuuksiin. Teemojen template-tiedostoja pitää myös käsin muokata sen sijaan, että promootiotuotteet lisättäisiin teemaan layout-päivitysten kautta. Kaupan tuotelistauksen view-templatessa eli tiedostossa: app/design/frontend/default/teeman_nimi/template/catalog/product/list.phtml

List.phtml-tiedostossa kutsutaan toista view-tiedostoa (.phtml) joka hakee Magentosta kaikki kategorian ja sen alla olevat tuotteet joille on promootiotuote-attribuutti asetettuna. Tämän jälkeen se muotoilee haetun datan. Tapa, jolla haetaan view-templateen toisen view-templaten sisältö, on alla esimerkissä 9.

Esimerkki 9. Toisen view-tiedoston sisällyttäminen view-tiedostoon

```
<?
echo $this->getLayout()->createBlock('core/template')-
>setTemplate('nostot/tuotenostot.phtml')->toHtml();
?>
```

Tätä kutsutaan tuotelistauksessa ennen tuotelistauksen järjestelytyökalua eli toolbaria.

5.1.4 Promootiotuotteiden ohjelma

Käydään jonkin verran läpi tuotenostojen ydintä eli koodia, jolla nostot Magentosta haetaan ja esitetään. Nykyisellään koodia on 70 riviä. Arvioin, että moduulin jälkeen moduulin ohjelmakoodi tulee olemaan 100-150 riviä sen mukaan, mitä ominaisuuksia halutaan ja ennätetään implementoimaan. En käy läpi koko koodia tässä opinnäytetyössä vaan otan esille pääkohdat. Esimerkissä 10 ladataan kaikki tuotteet, jotka sijaitsevat nykyisessä kategoriassa ja sen alla. Kuten koodista huomataan, niin kaikki ehdot täyttävät tuotteet haetaan Magenton mallin kautta muuttujaan `$products`.

Esimerkki 10. Promootiotuotteiden ohjelman rivit, jossa haetaan tuotteet.

```
<?php

    $categoryId = $currentCategoryPromotion;
    $category = Mage::getModel('catalog/category')-
>load($categoryId);

    $products = Mage::getModel('catalog/product')
        ->getCollection()
        ->addAttributeToSelect('*')
        ->addCategoryFilter($category)
        ->addAttributeToFilter('promotion', 1);

    $products->getSelect()->limit(3)->order(new
Zend_Db_Expr('RAND()'));
?>
```

Käydään läpi, miten tuotteita suodatetaan.

```
->getCollection()
```

Mallin kautta haetaan kokoelma. Yleensä kokoelmia käytetään tuotelistauksissa, jolloin voidaan hakea tuotenimet nousevassa tai laskevassa järjestyksessä hinnan, nimen tai jonkin muun arvon perusteella.

Kokoelma vastaa tuotelistauksen näkymää eli jos Magenton kategoriassa on ankkuri-ominaisuus käytössä kokoelmaan tulevat myös ne tuotteet, jotka sijaitsevat alakategoriassa, vaikka niitä tuotteita ei olisi merkitty siihen kategoriaan, missä asiakas nyt on. Sitten kun asiakas liikkuu kategoriapuussa ylöspäin, myös promootiotuotteet tarkentuvat sitä myöten.

`->addAttributeToSelect('*')` metodilla valitaan kaikki attribuutit kokoelmaan, että niiden perusteella voidaan suodattaa jälkeenpäin. Tämä nopeuttaa suodattamista, jos tuotteille lisätään jälkeenpäin attribuutteja, niin itse promootiotuotteiden ydintä on paljon nopeampi muokata.

`->addCategoryFilter($category)` metodilla lisätään kokoelman suodatukseen kategoriafiltteri, tässä tapauksessa sen kategorian tunniste, jossa asiakas on. Kokoelma ladataan tästä kategoriasta, ja jos ankkuri-ominaisuus on käytössä, myös ne tuotteet tulevat mukaan, jotka sijaitsevat kategorian alla.

`->addAttributeToFilter('promotion', 1);` metodilla valitaan ne tuotteet jossa attribuutti promotion on asetettu ”Kyllä”- tilaan.

```
$products->getSelect()->limit(3)->order(new
Zend_Db_Expr('RAND()'));
```

Tällä valitaan kolme tuotetta kokoelmasta satunnaisessa järjestyksessä. Jos tuotteita on vähemmän kuin kolme, se ei haittaa.

Tämän jälkeen `$products`-muuttujassa on ladattuna ne tuotteet, jotka halutaan promootiotuotteissa esittää. Tuotteet esitetään yksinkertaisella foreach-ohjausrakenteella, jonka aikana ne muotoillaan ja tuotteen tiedot haetaan esille. Ohjausrakenteen sisällä suoritetaan tuotetiedon muotoilut ja esitetään ne asiakkaalle. Esimerkki 11 alla esittelee, mitä tapahtuu foreachin sisällä.

Esimerkki 11. Ohjausrakenteen sisällä tapahtuvaa muotoilua

```
<div class="holder-promotion-col col<?php echo $i; ?>">
<div class="content<?php echo $last; ?>">
```

PHP:lla voidaan luoda tyylitiedostoille valmiiksi luokkia, joille voidaan sitten kirjoittaa CSS:ssa tyylit. Tässä tapauksessa jokaisen tuotepromootion kolumni on yksilöllisesti numeroitu. Kaikissa listoissa ja navigaatioissa viimeisestä tietueesta löytyy aina maininta viimeisen arvon kohdalta, koska viimeinen osa pitää aina tyylitellä eri tavalla. On hyvä kirjoittaa mahdollisimman monia html-tagien luokkia valmiiksi template-tiedostoon, vaikka kaikkia ei alkuun tyylitiedostossa käyttäisikään. Kun tilanne tulee, niin tyylien vaihtaminen ja ulkoasun säätäminen jokaiselle asiakkaalle sopivaksi onnistuu koskematta itse template-tiedostoon.

Esimerkissä 12 promootiotuotteet tarkistaa, onko tuotteelle asetettu mainosteksti. Metodi `$_product->getMainosteksti()` hakee tuoteattribuutin ”Mainosteksti” arvon ja php:n **`empty`** -funktiolla tarkistetaan, onko se tyhjä. Tässä voitaisiin myös käyttää muuttujan pituuden tarkistamista mutta **`empty`** on täysin riittävä tähän tehtävään. `$eiMainostekstia` on muuttuja, jolla luodaan luokka html-elementille.

Jos mainostekstiä ei ole, niin html-elementille annetaan tyylitiedostoa varten luokka nimeltä ”noattribute”, joka löytyy `$eiMainostekstia` muuttujasta. Tyylitiedostossa tämä tarkoittaa, että kyseisen kolumnin tuotenimikohdassa ei varata tilaa mainostekstin esittämiseen vaan toista tyyliä käytetään tässä. Jos mainosteksti on olemassa, niin CSS:n oletustyyliä käytetään.

Esimerkki 12. Html-tagien muodostamisesta

```
<?php $checkMainosteksti = $ product->getMainosteksti(); ?>
<?php if(empty($checkMainosteksti)){
$eiMainostekstia = " noattribute";
}else{
$eiMainostekstia = "";
}?>
```

Esimerkissä 13 liian pitkät tuotenimet lyhennetään, että muotoilu säilyy asiallisena ja myös kohta, jossa mainosteksti luodaan näytettäväksi asiakkaalle. Tässä on yksi esimerkki Magenton valmiista ominaisuuksista, jossa käytetään Magenton sisäänrakennettua helperiä, joten omia funktioita ei tarvitse tällaista yksinkertaista asiaa varten kirjoittaa.

Esimerkki 13. Liian pitkän tuotenimen lyhentäminen

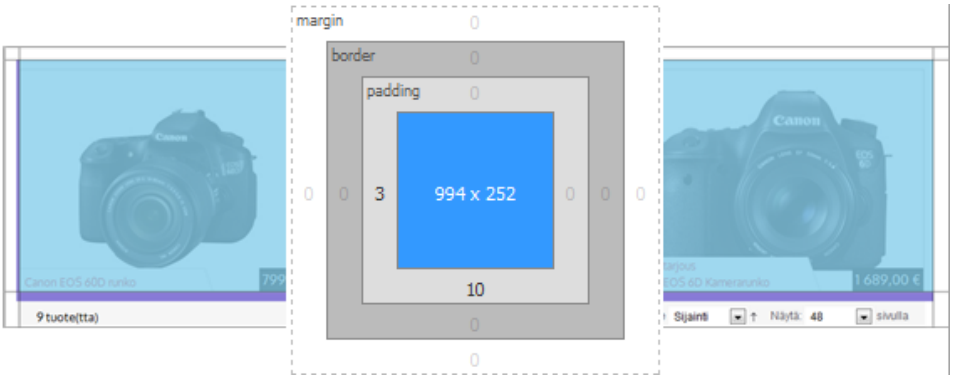
```
<?php echo $_product->getMainosteksti(); ?><br>
<?php $pNamePromotion= $ product->getName();
echo $this->helper('core/string')-
>truncate($pNamePromotion,35);  ?>
```

5.2 Moduulin tulevat ominaisuudet nykyiseen toteutukseen nähden

Nyt on käsitelty promootiotuotteiden nykyinen tila ja pääpiirteittäin, miten ominaisuus on tällä hetkellä toteutettu verkkokaupan yhteyteen. Seuraavaksi teen luettelon asioista, joita pitää parantaa ja mitä uusia ominaisuuksia moduulia toteuttaessa haluan implementoida, että promootiotuotteista tulisi mahdollisimman yleispätevä ja laajalle asiakaskunnalle sopiva moduuli. Lisäksi otan esille ne asiat, joita moduulia suunnitellessa pitää ottaa huomioon. Minulla on hyvä käsitys erilaisista verkkokaupoista, joten tiedän, miten eri elementit sopivat erilaisille kaupoille, ja miten ne asettuvat teemaan. Olen kuitenkin suorittanut tähän mennessä kauppakohtaista räätälöintiä, joten tämä on minulle uusi lähestymistapa.

5.2.1 Asiakaspuolen näkyvyys ja toiminnallisuus

Promootiotuotteet on tehty tällä hetkellä kauppaan, jossa on käytössä kolumniton teema. Tämä tarkoittaa sitä, että kaupan teemassa ei ole vasenta tai oikeaa kolumnia. Tämän ansiosta promootiotuotteet voivat viedä todella paljon tilaa ja käyttää isoja tuotekuvia. Myös tuotteiden määrä eli kolme on kiinteä ja asiakkaan tarpeiden mukaan tehty. Kuvassa 9 alla on esitetty tämä tilanne.



Kuva 9. Firebugilla esitetään html-elementin kokoa, joka on tässä kuvassa promootiotuotteiden viemä tila.

Useimmissa verkkokaupoissa ei kuitenkaan ole käytettävissä 994 px tilaa kategorianäkymässä. Tyypillisesti tilaa on käytössä noin 580 px. Kehitysympäristö, johon olen moduulia rakentamassa, on kauppa, jossa on paljon tilaa esittää tietoa. Minun pitää muistaa testata moduulin toimivuus kapeammilla teemoilla, koska useimmissa kaupoissa on käytössä kapeammat teemat. Yksinkertaisin ratkaisu on tehdä moduulin css-muotoilusäännöistä responsiivisia.

Nykyisessä toteutuksessa tuotteelle haetaan ainoastaan lopullinen hinta, joka voi olla tarjoushinta, mutta kauppa ei esitä sitä tarjoushintana. Kun tehdään yleispätevää toteutusta, niin asetettu tarjoushinta pitää tulla selvästi näkyville, jos sellainen on tuotteelle asetettu. Lisäksi promootiokuvan pitää olla kompaktimpi ja sen pitää erottua tuotelistauksen kategorialuettelon kuvista. Kuvassa 10 on esitetty tämä idea.

Kuva 10. Tuotekuva tarvitsee kaikki tiedot. Vasemmalla nykytila, oikealla tavoite.

5.2.2 Hallintaominaisuudet kauppiaalle

Nykyään promootiotuotteet asetetaan käytettäväksi kahdesta eri paikkaa eli tuoteominaisuuksien alasivuilta Yleinen ja Hinnat. Kuvassa 11 alla on esitetty tämä tilanne, kauppiaan pitää siis käydä läpi kaksi alavalikkoa, että tuotteen saa asetettua promootioksi.

Kuva 11. Nykyään promootiotuotteet pitää asettaa kahden eri valikon alta. Tavoitteena on saada kaikki tuotekohtaiset asetukset yhden valikon alle, kuten oikealla kuvassa.

Tavoitteena on laittaa kaikki promootioasetukset yhden valikon alle, ja se olisi nimeltään ”Promootioasetukset”. Promootiotuotteet toimivat muuttaman tuoteattribuutin varassa. Kun tehdään moduulia nämä tuoteattribuutit pitää luoda ohjelmallisesti. Samalla on mahdollisuus luoda nykyisen mainostekstin lisäksi muita valintoja kauppiaalle, jota kautta vaikuttaa promootiotuotteiden toimintaan.

Ominaisuudet, joita kauppiaan pitäisi pystyä muuttamaan promootiotuotteista, esitellään alla taulukossa 1.

Taulukko 1. Nykyisten ja tulevien ominaisuuksien yhteenveto

Tällä hetkellä toteutetut ominaisuudet
Promotiotuotteen mainosteksti
Sunnitellut tuotekohtaiset-asetukset
Päivämäärä loppu-alku, kun tuote on promootiossa (date-arvo)
Suunnitellut globaalit-asetukset
Promootiotuotteiden esitettävä määrä (numeerinen arvo)
Satunnainen vai järjestetty (alasvetovalikko)

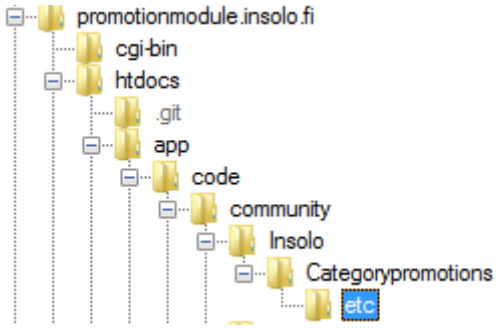
6 MODUULIN TOTEUTUS

Tässä kappaleessa käsitellään promootiotuotteet-moduulin toteuttaminen.

6.1 Moduulin tiedostorakenne ja kokoonpanotiedostot

Valitsin moduulin toteutettavaksi yhteisömoduulien alle. Tässä kohtaa nimi on Categorypromotions, koska se on tarpeeksi kuvaava. Kaikki hakemistot communitysta eteenpäin, ovat käsin luotuja moduulia varten. Categorypromotions-moduulin tiedostopolusta, joka esitellään kokonaisuudessaan kuvassa 12 alla, voidaan päätellä seuraavat asiat.

- Koodikirjasto (Codepool) = community eli yhteisö
- Nimiavaruus (Namespace) = Insolo eli yrityksen nimi Categorypromotions eli moduulin nimi.



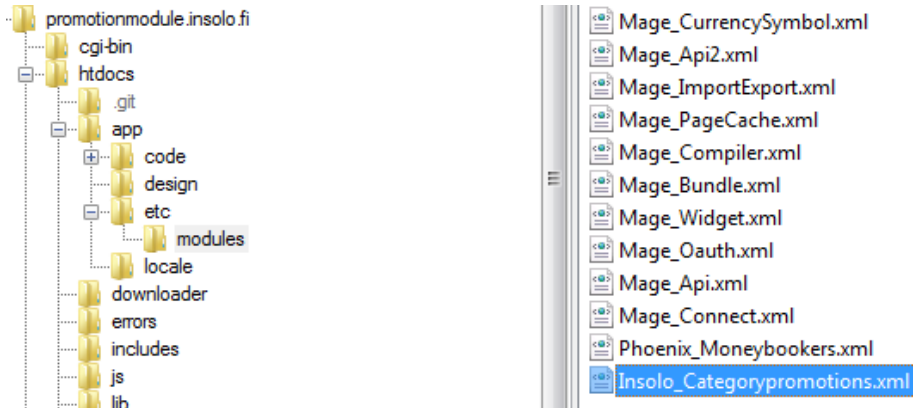
Kuva 12. Moduuli perustetaan communityn alle

Etc-hakemiston alle perustetaan kaksi xml-tiedostoa, jotka ovat siis moduulin kokoonpanotiedostot config.xml ja system.xml. Tässä vaiheessa system.xml jää tyhjäksi. Config.xml:n sisältö esitellään esimerkissä 14 alla.

Esimerkki 14. Config.xml sisältö categorypromotions-moduulissa

```
<?xml version="1.0"?>
<config>
  <modules>
    <Insolo_Categorypromotions>
      <version>1.0.0</version>
    </Insolo_Categorypromotions>
  </modules>
</config>
```

Edelleen esimerkissä 14 config.xml -tiedostossa kerrotaan moduulin versio, nimi ja sen käyttämä nimiavaruus, joka on siis yrityksen nimi eli Insolo. Tämän jälkeen perustetaan toinen kokoonpanotiedosto moduulirekisteriin code/etc -hakemistoon, tämän sijainti ja nimi näytetään kuvassa 13.



Kuva 13. code/etc alla oleva moduulirekisteri

Etc/modules hakemiston alla Magento lukee kaikki xml-tiedostot ja moduulilla pitää olla täällä tiedosto olemassa, että Magento ymmärtää moduulin olevan olemassa ja lähtee hakemaan sen kokoonpanotiedostoista asetuksia, jotta Magento voi sijoittaa moduulin kauppaan. Tiedoston pitää olla muotoa Nimiavaruus_Moduulinimi.xml eli tässä tapauksessa se olisi Insolo_Categorypromotions.xml. Esimerkki 15 käsittelee tämän kokoonpanotiedoston sisällön.

Esimerkki 15. Moduulirekisterin xml-tiedoston sisältö

```
<config>
    <modules>
        <Insolo_Categorypromotions>
            <active>true</active>
            <codePool>community</codePool>
        </Insolo_Categorypromotions>
    </modules>
</config>
```

Tämä kertoo Magentolle, mistä Codepoolista (koodialtaasta) moduulin kokoonpanotiedostot löytyvät. Itse `<Insolo_Categorypromotions>` kohta kertoo moduulin nimen lisäksi moduulin nimiavaruuden, joka on Insolo.

6.2 Moduulin layout-päivitykset teemaan

Seuraavaksi luodaan moduulille oma template valmiiksi ja viitataan siihen moduulin kokoonpanotiedostossa config.xml. Tässä demonstroidaan siis sitä, miten moduulit ja kaupan teemaus toimivat yhdessä. Lisätään config.xml tiedoston `<config>` elementin sisään tieto frontendin layoutin päivittämisestä. Alla esimerkissä 16 esitellään uusi osa kokonaisuudessaan.

Esimerkki 16. Config.xml -tiedostossa kerrotaan mistä haetaan layoutin päivitykset

```
<frontend>
    <layout>
        <updates>
            <Categorypromotions>
                <file>categorypromotions.xml</file>
            </Categorypromotions>
        </updates>
    </layout>
</frontend>
```

Categorypromotions.xml:ssa on tiedot, mitä osaa Magenton teemasta päivitetään ja mikä on lisättävä template tähän paikkaan. Tämä categorypromotions.xml -tiedosto sijaitsee:
app/design/frontend/default/default/layout/ polussa. Esimerkissä 17 alla on categorypromotions.xml:n sisältö, jossa on kaupan teemaan lisättävä osa määriteltynä.

Esimerkki 17. Categorypromotions.xml jossa kerrotaan, mitä kaupan teemaan lisätään

```
<?xml version="1.0"?>
<layout>
<catalog_category_layered>

   <reference name="content">
        <block type="core/template" before="-"
name="categorypromotions" tem-
plate="insolo/categorypromotions/Categorypromotions.phtml"/
>
    </reference>

</catalog_category_layered>
</layout>
```

Alla puretaan tämän tiedosto.

`<layout>` Kerrotaan että käsitellään layoutia.

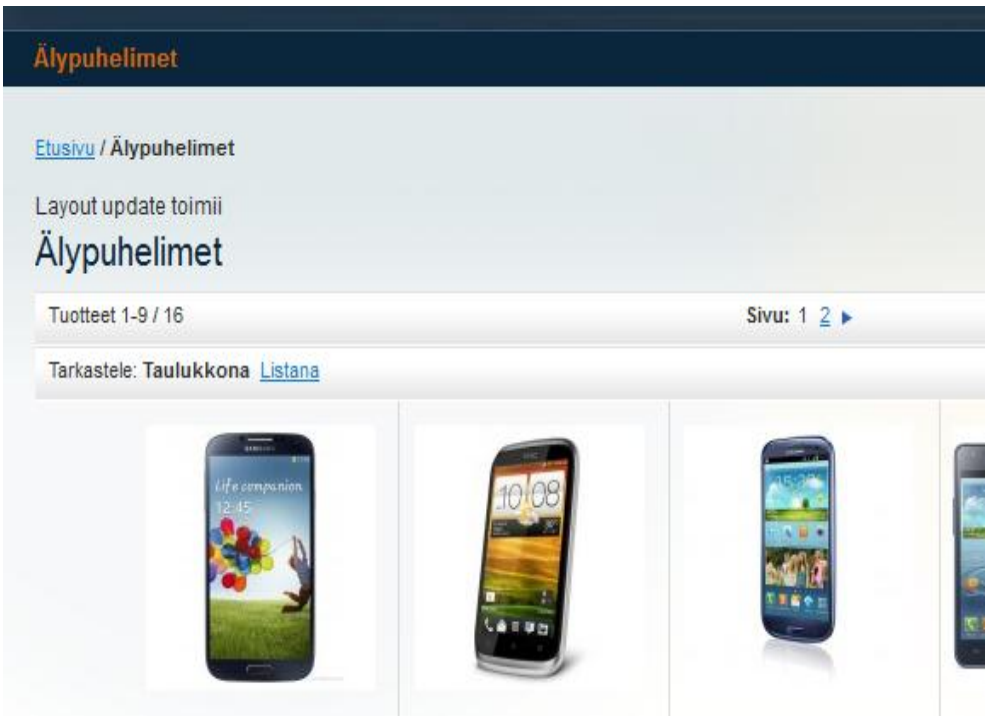
`<catalog_category_layered>`
On layoutin handle, johon otetaan kiinni, tässä tapauksessa kyseessä on katalogin kategoriasivu, jossa on käytössä layered eli kerroksittainen navigaatio. Tämä toimii testikaupassa oikein hyvin. Handle on tärkeä, koska kaikilla sivutyyppeillä on olemassa rakennelohko content, left tai right, mutta koska ne sijoitetaan eri handleihin, niin ne eivät esiinny jokaisessa rakennelohkossa, jonka nimi on content, left tai right.

`<reference name="content">`
Tämä on osa sivua, jonne sisältö sijoitetaan eli mihin rakennelohkoon viitataan. Handle on siis jo määritelty, nyt määritellään, mihin osaan template sijoitetaan.

```
<block type="core/template" before="-"
name="categorypromotions" tem-
plate="insolo/categorypromotions/Categorypromotions.phtml"/
>
```

Kerrotaan teeman lisättävän lohkon tyyppi, sijainti ja järjestys muihin loh koihin nähden. Template sijaitsee polussa:
app/design/frontend/default/default/template

Templaten sisältö on testitarkoitusta varten yksinkertainen ”Layout update toimii” -tekstin tuominen esille. Kuvassa 14 näytetään, miten kaupan teema muuttuu.



Kuva 14. Testataan layout-päivitysten toimivuus

Nyt jos poistetataan aktiivisen teeman catalog.xml, tiedostosta kaikki tiedot siitä, miltä katalogin pitäisi näyttää, niin jäljelle jäisi pelkästään teksti ”Layout update toimii”. Kuvassa 15 näytetään tämä tilanne. Tämä johtuu siitä, että koska kaupan oletuksena lisäämät moduulit on poistettu, jäljelle jää juuri kauppaan lisätyn moduulin muutokset.

Kuva 15. Teeman catalog.xml:sta on poistettu kaikki tiedot, mutta moduulin layoutin päivitykset säilyvät.

Vaikka Magenton perusteemaan tehtäisiin muutoksia, niin moduulin tekemät muutokset ovat silti aina erillään niistä muutoksista, mitä teemaan tehdään. Nyt siis Categorypromotions-moduuli on siinä vaiheessa, että Magento löytää sen, hakee sille layoutin päivitykset ja esittää Moduulin määrittelemän template-tiedoston kaupassa.

6.3 Moduulin globaalit asetukset

Luodaan moduulille valmiita globaaleja asetuksia, joita kauppias pystyy hallintapuolella vaihtamaan. Nämä ovat asetuksia, jotka vaikuttavat koko moduulin toimintaan pelkän tuotteen sijaan. Tarkoituksena on toteuttaa seuraavat toiminnallisuudet globaalien asetusten kautta. Nämä asetukset on listattu Taulukossa 2.

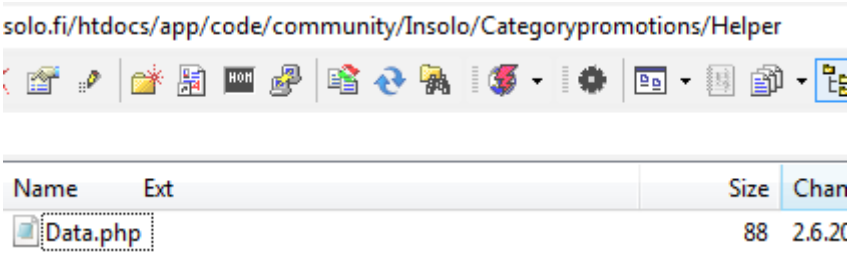
Taulukko 2. Moduulin globaalit asetukset

Toteutettavat gloobalit asetukset moduulille	
Asetuksen nimi	Asetuksen tyyppi
Promootiotuotteiden esitettävä	Numeerinen arvo
Satunnainen vai järjestetty	Alasvetovalikko
Mainostekstit näkyvillä	Kyllä/Ei

Moduulille luotavat globaalit asetukset määritellään moduulin system.xml-tiedostossa. Ennen kuin aletaan kirjoittamaan system.xml:aan tarvittavia hallintapuolen asetuksia, pitää perustaa moduulille Helperi ja kertoa järjestelmälle, mistä se löydetään. Magento olettaa, että kaikilla moduuleilla, jotka haluavat perustaa hallintapuolelle kontrolleja, on olemassa Helper. Ilman määriteltyä Helperiä saadaan vain virheviesti.

6.3.1 Helperin perustaminen ja sen osoittaminen kokoonpanotiedostossa

Luodaan hakemisto moduulin Helper:ille, jonne perustetaan tiedosto Data.php. Alla olevasta kuvasta 16 selviää polku.



Kuva 16. Helperin luominen moduulille

Itse Data.php:n sisältö on yksinkertainen. Se vain laajentaa Magenton olemassa olevaa ydin Helper-luokkaa. Alla esimerkissä 18 tiedoston sisältö.

Esimerkki 18. Data.php laajentaa Magenton ydin Helper-luokkaa

```
<?php
class Insolo_Categorypromotions_Helper_Data extends
Mage_Core_Helper_Abstract
{
}
```

Määritelty luokka on muotoa Codepool_Nimiavaruus_MVC-Osa. Eli tässä tapauksessa Insolo on codepool. Nimiavaruus on moduulin nimi eli Categorypromotions ja haettava MVC:n osa on Helper. Helperin alta halutaan Data.php. Tämän jälkeen lisätään moduulin kokoonpanotiedostoon config.xml, tieto siitä, mistä Magento löytää helperit. Esimerkissä 19 näytetään lisätty kohta config.xml -tiedostoon.

Esimerkki 19. Config.xml kokoonpanotiedoston lisäys, jolla kerrotaan Magentolle mistä moduulin Helperit -löytyvät.

```
<global>
    <helpers>
        <categorypromotions>
            <class>Insolo_Categorypromotions_Helper</class>
       </categorypromotions>
    </helpers>
</global>
```

6.3.2 Moduulin globaalien asetusten luominen system.xml -tiedostoon

System.xml on tiedosto, jonne kirjoitetaan kaikki päivitykset hallintapuolen käyttöliittymään. Kaikki päivitykset system.xml –tiedostoon pitää sijoittaa `<config>` elementin sisään. Kaupan hallintapuolelle luodaan Categorypromotionsia varten oma valikko lisäämällä Esimerkki 20:n mukainen kohta system.xml:aan.

Esimerkki 20. System.xml, jolla lisätään valikko hallintapuolelle

```
<tabs>
<categorypromotionsconfig translate="label" mod-
ule="categorypromotions">
            <label>Insolo Categorypromotions</label>
            <sort_order>3</sort_order>
        </categorypromotionsconfig>
</tabs>
```

`<tabs>` Elementissä määritellään otsikko, jonka alle moduulin valikot tulevat. Tab on siis sitova elementti, jonka alle valikot sijoitetaan.

`<categorypromotionsconfig translate="label" module="categorypromotions">` Tätä käytetään uuden osion merkitsemiseen, elementin nimi voi olla periaatteessa mikä vain, mutta moduulin nimi pitää olla oikein.

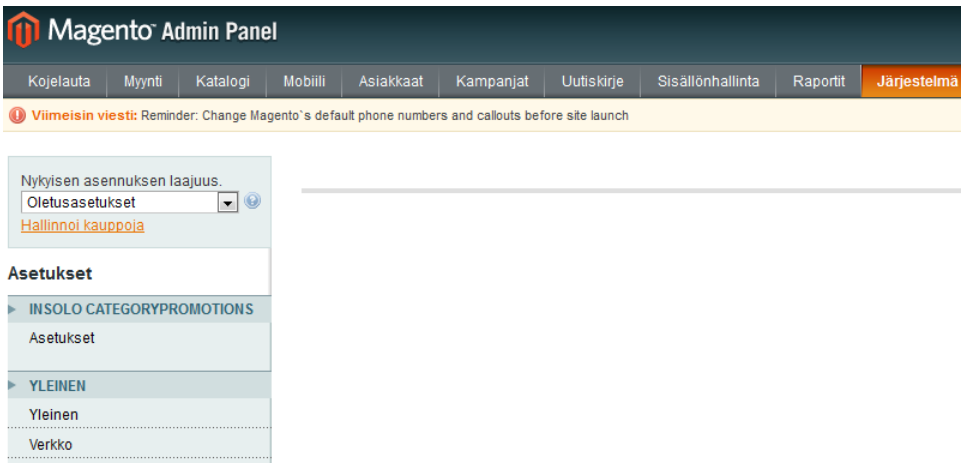
`<label>` Elementissä määritellään, millä nimellä tämä pääotsikko löytyy hallintapuolesta.
`<sort_order>` Miten valikko sijoittuu muiden valikkojen joukossa.

Nyt päävalikko, jonka alle asetukset sijoitetaan, on valmis. Asetukset sijoitetaan `<sections>` elementin sisään. Tämä elementti perustetaan xml-tiedostoon sen jälkeen, kun `<tabs>` elementti on suljettu. Esimerkissä 12 käsitellään Asetukset-valikon perustaminen.

Esimerkki 21. Tässä perustetaan system.xml -tiedostoon asetukset-valikko

```
<sections>
<categorypromotions_options translate="label" mod-
ule="categorypromotions">
      <label>Asetukset</label>
      <tab>categorypromotionsconfig</tab>
      <frontend_type>text</frontend_type>
      <sort_order>1000</sort_order>
      <show_in_default>1</show_in_default>
      <show_in_website>1</show_in_website>
      <show_in_store>1</show_in_store>
</categorypromotions_options>
```

Aivan kuten tabeissakin tässäkin elementin nimi pitää olla yksilöivä ja osoittaa oikeaan moduuliin. Myös se päävalikko eli tab, johon tämä sektio viittaa, pitää olla oikein. Eli tässä tapauksessa viitataan juuri luotuun päävalikkoon elementissä `<tab>categorypromotionsconfig</tab>` alla kuvassa 17 näkyy, miltä valikko näyttää.



Kuva 17. Hallintapuolen näkymä.

Tämän jälkeen siirrytään luomaan ryhmiä, joiden sisälle sijoitetaan itse valikot. Ryhmät sijoitetaan `<categorypromotions_options>` elementin sisälle. Tähän moduuliin tarvitaan vain yksi ryhmä. Alla esimerkissä 22 luodaan tämä ryhmä.

Esimerkki 22. Luodaan ryhmä asetuksien alle system.xml -tiedostoon

```
<groups>
      <asetukset translate="label">
          <label>Kategoriatuotenostojen asetukset</label>
          <frontend_type>text</frontend_type>
          <sort_order>1</sort_order>
          <show_in_default>1</show_in_default>
          <show_in_website>1</show_in_website>
          <show_in_store>1</show_in_store>
      </asetukset>
</groups>
```

Jälleen kerran elementin nimi on vapaavalintainen, mutta se kannattaa pitää mahdollisimman yksilöivänä, tässä tapauksessa nimeämiskäytännöllä ei ole niin suurta merkitystä, koska on vain yksi valikko. Saman `<groups>` elementin sisälle sijoitetaan asetuksia, joita moduuli käyttää. Nämä asetukset menevät `<fields>` elementin sisälle. Esimerkissä 23 alla yksi promootiotuotteet-moduulin asetuksista.

Esimerkki 23. Esimerkki asetuksesta joka sijoitetaan <groups> elementin sisälle

```
<sekoita translate="label">
     <label>Sekoita tuotteet</label>
     <frontend_type>select</frontend_type>
<source_model>adminhtml/system_config_source_yesno</source_
model>
     <sort_order>0</sort_order>
     <show_in_default>1</show_in_default>
     <show_in_website>1</show_in_website>
     <show_in_store>1</show_in_store>
     <comment>Arvo tuotteiden jarjestys</comment>
</sekoita>
```

`<frontend_type>` Kohtaan kerrotaan, mitä tyyppiä luotava valikko on, tässä tapauksessa select eli pudotusvalikko.

`<source_model>` Koska valikko on tyyppiä select, siinä olevat arvot pitää esitäyttää. Tähän voitaisiin ohjelmoida itse valintoja, mutta Magentossa tulee mukana mm. maat, kuukaudet ja yksinkertaiset Yes/No -valinnat.

`adminhtml/system_config_source_yesno` tässä on valittu Magenton sisäänrakennetut Kyllä/Ei -valinnat. Kun kaikki asetukset on luotu system.xml:aan, näkymä hallinnassa on kuvan 18 mukainen.

Kuva 18. Moduulin globaalit asetukset

Koodia ei tarvitse kirjoittaa erikseen arvojen tallentamista varten vaan Magentoon sisäänrakennetut metodit hoitavat asian. Moduulille suunitellut globaalit asetukset on nyt toteutettu hallintapuolelle ja on aika kutsua niitä frontendin koodissa. Palataan Categorypromotionsin view-templateen, johon lisätään esimerkin 24 sisältö.

Esimerkki 24. Kutsutaan juuri luotuja asetuksia view-templatessa että ne näkyvät asiakasnäkymässä

```
Sekoita: <? echo Ma-
ge::getStoreConfig('categorypromotions_options/asetukset/se
koita');?><br>
n-Tuotetta: <? echo Ma-
ge::getStoreConfig('categorypromotions_options/asetukset/tu
otteita'); ?><br>
Mainostekstit: <? echo
Mage::getStoreConfig('categorypromotions_options/asetukset/
mainostekstit');?><br>
```

getStoreConfig on metodi, jota käytetään globaalille Mage-objektille. Polku, mitä kautta asetukset haetaan, noudattaa samaa logiikkaa, mitä käytettiin valikkojen luomiseen. Alla olevassa taulukossa pilkotaan tämä osiin, mitä metodilla haetaan, ja mikä osa vastaa mitäkin juuri luotua valikkoa.

section_name	group_name	field_name
categorypromotions_options	asetukset	tuotteita

Globaalit asetukset ovat nyt siinä kunnossa, että niitä voidaan käyttää itse moduulin toiminnallisuuksien ohjelmoimisessa. Kuvassa 19 näytetään, miltä moduulin hallintapuolten asetusten haku näyttää asiakasnäkymässä.

Kuva 19. Frontend näkymä, johon on haettu moduulin asetukset.

6.4 Tuoteattribuuttien perustaminen ohjelmallisesti

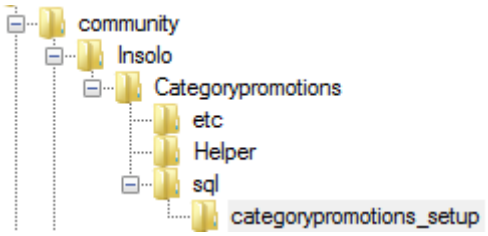
Ei-modulaarisessa toteutuksessa tuoteattribuutit perustettiin käsin. Kun toteutus siirretään modulaariseen muotoon, samat attribuutit pitää perustaa ohjelmallisesti. Tarvittavat attribuutit ovat Taulukossa 3, suluissa attribuutin tyyppi.

Taulukko 3. Ohjelmallisesti perustettavat tuoteattribuutit, ja attribuutin tyyppi

Ohjelmallisesti perustettavat tuoteattribuutit	
Attribuutin nimi	Attribuutin tyyppi
Promootiotuote	Yes/No -valinta
Promootio alku ja loppu pvm	Päivämäärän valinta
Mainosteksti	Teksti

6.4.1 SQL-asennusskripta

Moduulille lisätään uusi kansio nimeltä sql, jossa on moduulin sql-asennusskriptit. Sql-kansion alle perustetaan uusi kansio, joka on nimeltään categorypromotions_setup. Tämä kansio osoitetaan myöhemmin moduulin kokoonpanotiedostossa, että Magento osaa ajaa asennusskriptan. Alla kuvassa 20 moduulin päivitetty tiedostorakenne.



Kuva 20. Moduulin tiedostorakenne SQL-asennusskriptin sisältävän kansion kanssa

Tähän kansioon perustetaan uusi php-tiedosto, joka on itse asennusskripta. Tiedosto on nimeltään mysql4-install-1.0.0.php, jossa versionumero pitää olla sama kuin moduulin versio eli tässä tapauksessa 1.0.0. Seuraavaksi selvitetään asennusskriptan tärkeimmät kohdat.

`$installer = $this;` Asennusskripta alkaa tällä, tässä luodaan Setup Resource Class-luokka. Jokainen viittaus asennuskriptassa $this viittaa objektiin, joka luodaan tässä luokassa.

`$setup = new Mage_Eav_Model_Entity_Setup('core_setup');` Tässä luodaan instanssi Mage_Eav_Model_Entity_Setup-luokasta.

`$installer->startSetup();` Tällä alustetaan SQL-yhteys tietokannan päivittämistä varten.

```
$setup->addAttributeGroup('catalog_product', 'Default',
'Promootioasetukset', 1000);
```

Tämän jälkeen käytetään luokan metodia `addAttributeGroup()`, jolla luodaan uusi ryhmä, johon tuoteattribuutit sijoitetaan. Metodille annetut parametrit käsitellään alla taulukossa 4.

Taulukko 4. Luokan metodin osat avattuna

addAttributeGroup osat selitettynä	
catalog_product	Ryhmä koskee tuotteita, se voisi koskea esim.kategorioita
Default	Liitetään tuoteattribuuttien kokoelmaan ”Default” eli oletuskokonaisuuteen.
Promootioasetukset	Ryhmän nimi
1000	Ryhmän järjestysnumero, kun Magento listaa ryhmät tuoteasetuksissa, mitä pienempi numero sitä korkeammalla promootioasetukset näkyvät.

Tämän jälkeen käsitellään tuoteattribuutin perustaminen. Kyseessä on Kyllä/Ei -attribuutti, jolla kerrotaan moduulille, onko tuote promootiotuote. Alla esimerkki 25 näyttää tämän tuoteattribuutin ohjelmallisen perustamisen.

Esimerkki 25. Esimerkki

```
$setup->addAttribute('catalog_product', 'is_promo', array(
    'group'         => 'Promootioasetukset',
    'input' => 'boolean',
    'type' => 'int',
    'label' => 'Promootiotuote',
    'backend' => '',
    'source' => 'eav/entity_attribute_source_table',
    'visible'       => 1,
    'required'      => 0,
    'user_defined' => 1,
    'searchable' => 1,
    'filterable' => 0,
    'comparable'    => 1,
    'position' =>1,
    'note' =>'Nostetaanko tuote erikseen listattavaksi',
    'visible_on_front' => 1,
    'visible_in_advanced_search'  => 0,
    'is_html_allowed_on_front' => 0,
    'global'        =>
Mage_Catalog_Model_Resource_Eav_Attribute::SCOPE_GLOBAL,
));
```

Käsitellään alla tätä asennutiedoston kohtaa. Käytetään luokan metodia `addAttribute` , jolla lisätään itse tuoteattribuutti. Annetut kaksi parametria kertovat, että tuoteattribuutti koskee katalogin tuotteita ja attribuutin nimi on is_promo. Käsitellään annetuista parametreista tärkeimmät Taulukko 5:ssä.

```
$setup->addAttribute('catalog_product', 'is_promo',
```

Taulukko 5.Tuoteattribuutin ohjelmallinen perustaminen osissa

Tuoteattribuutin ohjelmallinen perustaminen osissa	
`'group' => 'Promootioasetukset',`	Attribuutti kuuluu ”Promootioasetukset” ryhmään, joka luotiin asennusskriptin alussa.
`'input' => 'boolean',`	Syöttötapa on 1 tai 0 eli Yes / No.
`'type' => 'int',`	Tulos on 1 tai 0.
`'label' => 'Promootiotuote',`	Tällä nimellä attribuutti näkyy hallintapuolella.
`'source'=>'eav/ entity_attribute_source_table',`	Koska kyseessä on dropdown-valikko, vaihtoehdoille pitää hakea arvot eli source. Tässä tapauksessa arvot ovat Kyllä/Ei. Arvot haetaan Magenton sisäänrakennetusta sourcesta.
`'position' =>1,`	Attribuutin järjestys ryhmän sisällä, tällä is_promo attribuutti tulee ensimmäiseksi.
`'note'=>'Nostetaanko tuote erikseen listattavaksi',`	Opasteksti attribuutin kentälle.

Kun kaikki halutut attribuutit on kirjoitettu asennusskriptaan, lopetetaan asennusskripta `$installer->endSetup();` metodilla, tämä sulkee SQL-yhteyden.

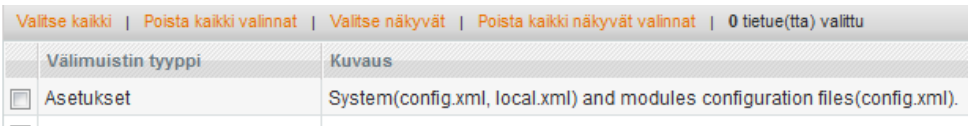
6.4.2 Päivitetään kokoonpanotiedosto ja kaupan välimuisti

Kokoonpanotiedostossa config.xml kirjoitetaan, mistä Magento ajaa moduulin SQL-asennusskriptan. Tämä tieto lisätään config.xml:ssa `<global>` elementin sisään. Esimerkissä 26 esitellään määrittelytapa.

Esimerkki 26. Config.xml johon määritetty SQL-asennuskriptan sijainti

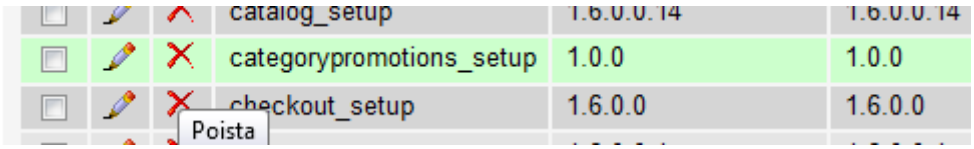
```
<resources>
    <categorypromotions_setup>
        <setup>
          <module>Insolo_Categorypromotions</module>
        </setup>
     </categorypromotions_setup>
 </resources>
```

Tämän jälkeen kirjaudutaan Magenton hallintapuolelle. Mennään välimuistin hallintaan, josta ajetaan uudelleen Järjestelmä -> Välimuistin hallinnasta Magenton välimuistit. Kuvassa 21 näytetään tämä päivitettävä välimuisti.



Kuva 21. Kun tämä välimuisti on päivitetty moduulin asennusskripta päivittää tietokannan ja uudet tuoteattribuutit on lisätty kauppaan.

Magento ajaa asennusskriptan vain kerran. Jos moduulin asennusskriptaan tulee muutoksia, niin Magenton tietokannasta pitää nollata merkintä, jotta Magento ajaa asennusskriptan uudelleen. Tämä merkintä sijaitsee core_resource –taulussa. Toinen vaihtoehto on, että kasvattaa moduulin versionumeroa, jolloin asennusskripta ajetaan uudelleen, mutta tämä käytäntö on parempi jättää oikeille päivityksille. Kuvassa 22 näytetään taulun tietue, joka pitää poistaa, että asennuskripta suoritetaan uudelleen.



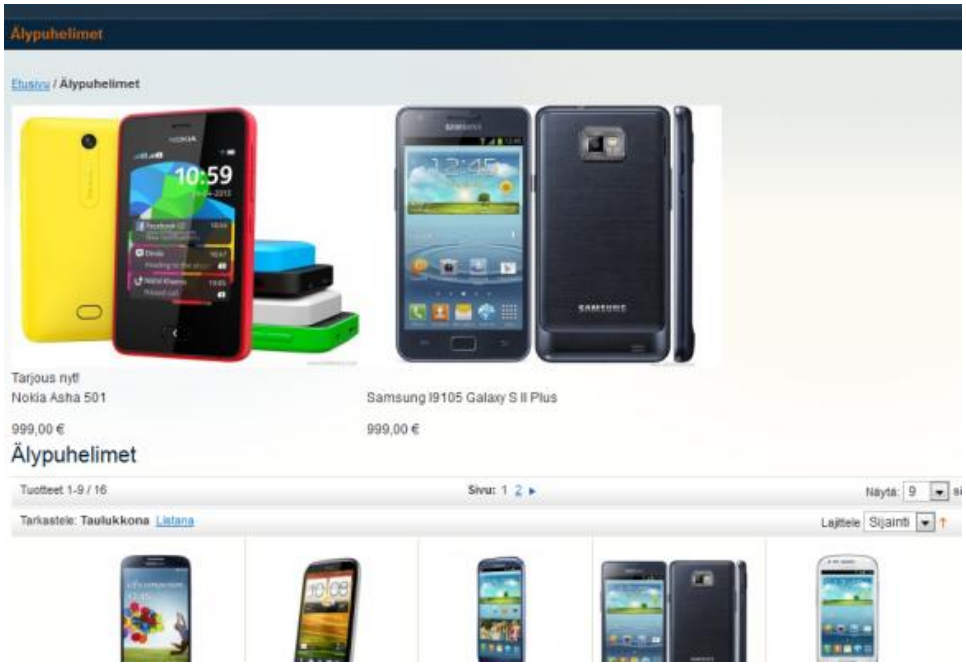
Kuva 22. Jotta asennusskriptan voi ajaa uudelleen on tämä merkintä poistettava.

Kun tuoteattribuuttien asennuskripta on suoritettu, hallintapuolelle on luotu kaikki moduulin tarvitsemat tuoteattribuutit. Kuvassa 23 näytetään kaikki asennuskriptan luomat tuoteattribuutit.

Kuva 23. Kun moduulin sql-asennusskripta on ajettu, tuoteattribuutit on luotu ohjelmallisesti ja sijoitettu ”Promootioasetukset” ryhmään.

6.5 Asiakkaalle näkyvä view-tiedosto

Ilman mitään lisämuutoksia view-tiedostoon moduuli näyttää asiakkaalle Kuvan 24 mukaiselta. Moduulista asiakkaalle näkyvä puoli ei ole merkittävästi muuttunut, koska kaupan layouttiin tai asetteluun ladattava template jossa on promootiotuotteet hakeva koodi, on sama mikä ei-modulaarisessakin toteutuksessa.



Kuva 24. Näin moduuli näkyy asiakkaalle

7 YHTEENVETO

7.1 Saavutetut edut

Koska valikot ja tuoteattribuutit luodaan ohjelmallisesti, mitään sekaannuksia ei tule, kun promootiotuotteet-toiminnallisuutta asennetaan toiseen kauppaan. Manuaalinen työ rajoittuu asennustiedostojen raahaamiseen ja tiputtamiseen palvelimelle. Myöskään varsinaista asennusdokumentaatiota ei tarvitse kirjoittaa.

Modulaarisessa muodossa kaikki toiminnot ovat riippumattomia Magenton teematiedostoista tai ytimen toiminnasta. Ei-modulaarisessa toteutuksessa tuotenostojen toiminta oli sidottu kauppakohtaisesti kaupan teematietoihin. Tämä tarkoittaa sitä, että jos ei-modulaarinen asennus halutaan siirtää toiseen kauppaan, se vaatii kohdekaupan teematiedostojen muokkaamista, johon tuotenostot halutaan. Modulaarisessa muodossa kaupan teema voi olla mikä vain.

7.2 Haasteet

Suurin haaste oli tehdä tätä työtä oman työn ohessa. Tiesin moduuleista opinnäytetyötäni aloittaessa paljon, mutta en ollut vielä tehnyt omaa moduuliani täysin alusta loppuun. Olin käyttänyt aiemmin valmiita ohjelmia moduulien luomiseen. Tässä työssä päätin, että en käytä moduulien luomiseen tarkoitettua Magentolle tehtyä lisäohjelmaa, koska minun piti tietää kaikki mahdollisimman yksityiskohtaisesti.

En pitänyt Magento module creatoria mainitsemisen arvoisena tässä opinnäytetyössä, koska sen käyttäminen Magenton moduulien ymmärtämiseen ei ole läheskään niin tuottavaa kuin Magento-ohjelmoijien oppaiden seuraaminen.

7.3 Jatkokehitys

Promootiotuotteet-moduulin potentiaali myytäväksi moduuliksi on vielä selvittämättä. Tiedossa on, että tämä on haluttu ominaisuus, tätä on asiakas pyytänyt. Mahdollisuus on, että jos moduuli laitetaan esille, se kehittäisi kysyntää.

Tämän opinnäytetyön valmistuessa moduulissa ei ole kaikkia siihen tarvittavia ominaisuuksia, mutta kaikki opinnäytetyön kannalta olennainen on implementoitu ja dokumentoitu. Loput viimeisteltävät asiat eivät ole opinnäytetyössä mainitsemisen arvoisia, koska ne ovat lähinnä CSS-muotoiluja ja Magenton perusominaisuuksien käyttöä.

LÄHTEET

Internet-lähteet

A History of Magento – Orangecollarmedia.com. Viitattu 8.4.2013
http://orangecollarmedia.com/a-history-of-magento

Anatomy of a Magento extension – Mandagreen.com. Viitattu 13.5.2013
http://mandagreen.com/anatomy-of-a-magento-extension/

Compare Magento Products – Magentocommerce.com. Viitattu 27.3.2013.
http://www.magentocommerce.com/product/overview-compare

Config.xml Reference - Magentocommerce.com. Viitattu 19.5.2013.
http://www.magentocommerce.com/wiki/5_-_modules_and_development/reference/module_config.xml

Creating Magento Connect Extension Package – Magentocommerce.com. Viitattu 25.3.2013.
http://www.magentocommerce.com/wiki/7_-_magento_connect/creating_magento_connect_extension_package

Custom Module with Custom Database Table – Magentocommerce.com. Viitattu 25.3.2013.
http://www.magentocommerce.com/wiki/5_-_modules_and_development/0_-_module_development_in_magento/custom_module_with_custom_database_table

Get Ready for Magento Certified Developer Exam. Magento Module Structure. Viitattu 12.5.2013
http://blog.belvg.com/magento-certification-module-structure.html

Installing Custom Attributes with Your Module – Magentocommerce.com viitattu 10.5.2013
http://www.magentocommerce.com/wiki/5_-_modules_and_development/0_-_module_development_in_magento/installing_custom_attributes_with_your_module

Intro to Layouts, Magento Designer's Guide – Magentocommerce.com viitattu 4.5.2013
http://www.magentocommerce.com/design_guide/articles/intro-to-layouts

Layout-update tietoja. Viitattu 25.5.2013
http://magebase.com/magento-tutorials/demystifying-magentos-layout-xml-part-1/

Magento Architecture for a Simple Module. randallhook.com viitattu 12.5.2013
http://randallhook.com/2009/08/24/magento-architecture-for-a-simple-module/

Magento Design Terminologies. Magentocommerce.com viitattu 12.5.2013
http://www.magentocommerce.com/design_guide/articles/magento-design-terminologies4

Magento for Developers: Part 1 - Introduction to Magento - Magentocommerce.com. Viitattu 26.3.2013.
http://www.magentocommerce.com/knowledge-base/entry/magento-for-dev-part-1-introduction-to-magento

Magento for Developers: Part 5 - Magento Models and ORM Basics. Viitattu 13.5.2013
http://www.magentocommerce.com/knowledge-base/entry/magento-for-dev-part-5-magento-models-and-orm-basics

Magento Module Creator – Silksoftware.com. Viitattu 25.3.2013.
http://www.silksoftware.com/magento-module-creator/#.UZCj5bWnryi

Magento's Architecture: Part 1 – Packtpub.com. Viitattu 8.4.2013
http://www.packtpub.com/article/magentos-architecture-part1

Model-View-Controller – msdn.microsoft.com. Viitattu 4.5.2013
http://msdn.microsoft.com/en-us/library/ff649643.aspx

Packaging a Magento Extension – Magentocommerce.com. Viitattu 25.3.2013.
http://www.magentocommerce.com/wiki/7_-_magento_connect/packaging_a_magento_extension

Simple Example of MVC (Model View Controller) Design Pattern for Abstraction – Codeproject.com. Viitattu 4.5.2013
http://www.codeproject.com/Articles/25057/Simple-Example-of-MVC-Model-View-Controller-Design

The Basics Of Creating A Magento Module – Smashingmagazine.com. Viitattu 10.5.2013
http://coding.smashingmagazine.com/2012/03/01/basics-creating-magento-module/

Understanding Model-View-Controller – codinghorror.com. Viitattu 8.4.2013
http://www.codinghorror.com/blog/2008/05/understanding-model-view-controller.html

Using Collections in Magento. Magentocommerce.com. Viitattu 14.5.2013

http://www.magentocommerce.com/wiki/1_-_installation_and_configuration/using_collections_in_magento

XML Structure for admin configurations. Viitattu 2.6.2013 http://www.magentocommerce.com/wiki/5_-_modules_and_development/admin/xml_structure_for_admin_configurations

Zend Framework & MVC Introduction – Framework.zend.com. Viitattu 8.4.2013 http://framework.zend.com/manual/1.12/en/learning.quickstart.intro.html

10 Compelling Reasons to Use Zend Framework – Net.tutsplus.com. Viitattu 8.4.2013 http://net.tutsplus.com/tutorials/php/10-compelling-reasons-to-use-zend-framew