LAUREA
AMMATTIKORKEAKOULU
Uuden edellä

Ajax tiedostonsiirto – lisäosan toteutus MODX:lle

Tenhunen Henri

2013 Laurea Leppävaara

Laurea ammattikorkeakoulu
Laurea Leppävaara

Ajax tiedostonsiirto - lisäosan toteutus MODX:lle

Tenhunen Henri
Tietojenkäsittelyn koulutusohjelma
Opinnäytetyö
Joulukuu, 2013

Laurea ammattikorkeakoulu **Tiivistelmä**
Laurea Leppävaara
Tietojenkäsittelyn koulutusohjelma

Tenhunen Henri

Ajax tiedostonsiirto - lisäosan toteutus MODX:lle

Vuosi 2013 Sivumäärä 71

Tämän toiminnallisen opinnäytetyön tavoitteena oli suunnitella ja toteuttaa Ajax-tekniikkaa hyödyntävä tiedostonsiirto - lisäosa MODX – sisällönhallintajärjestelmälle.

Työn alkuperäinen tarkoitus oli laatia mediatoimisto Sivudesign Ky:n asiakkaan www-sivuille tiedostonsiirtolomake, jota kautta käyttäjät voisivat lähettää kuvatiedostoja asynkronisesti palvelimelle. Toteutuksen jälkeen syntyi idea jatkokehittää työstä uudenlainen versio, jota voisi mahdollisesti hyödyntää myös tulevaisuuden asiakasprojekteissa.

Työnantajayritys käytti MODX – sisällönhallintajärjestelmää Internet-sivujen toteuttamiseen. Päämääränä oli paketoida tiedostonsiirtoskripti MODX:n ominaisuuksia hyödyntäväksi, helppokäyttöiseksi, toimivaksi ja tietoturvalliseksi lisäosaksi, jonka voisi asentaa vaivattomasti MODX:n pakettienhallinnan kautta. Työ toteutui määritysten mukaisesti. Projektista syntyi sivutuotteena avoimen lähdekoodin MODX Extra, joka on vapaasti ladattavissa MODX:n kotisivulta.

Opinnäytetyön alkuvaiheessa tarkastellaan projektin kannalta keskeisimpiä käsitteitä sekä kuvataan työn taustaa. Tämän jälkeen opinnäytetyö painottuu asiakasprojektin ja jatkokehitysprojektin tekniseen analyysiin. Työn lopussa esitetään jatkokehittämisen tulokset sekä johtopäätökset koko hankkeesta.

Asiasanat MODX, Ajax, sisällönhallintajärjestelmä, CMS, lisäosa, tiedostonsiirto.

Laurea University of Applied Sciences **Abstract**
Laurea Leppävaara
Bachelor's Degree Programme in Business Information Technology

Tenhunen Henri

Implementing Ajax file transfer Extra for MODX

Year 2013 Pages 71

The objective of this functional thesis was to design and actualize a data transfer accessory which utilizes AJAX technology for the MODX content management system.

The original purpose was to devise a data transfer form that the users of a certain site could use to upload pictures to a web site server. After the implementation a need emerged to further develop the data transfer form to a more accessible version that could also be used for the future customer projects.

The employing company used the MODX content management system to create web sites. The objective of the project was to packet the data transfer script into a more user friendly, efficient and secure accessory that could be installed effortlessly through the MODX package management. The project produced an open source code byproduct called MODX Extra. It is freely downloadable from the MODX homepage.

The initial section of the thesis paper examines the essential terms and describes the background of the project. The thesis continues by focusing on the technical analysis of the customer and byproduct projects. The final section presents the conclusions and the results of further developments.

Keywords MODX, Ajax, content management system, CMS, Extra, file transfer

Sisällys

1 Johdanto 6
2 Käytetyt tekniikat ja ohjelmointikielet 8
2.1 MODX 8
2.2 JavaScript 9
2.3 PHP 10
2.4 MySQL 11
2.5 Ajax 12
3 Tiedostonsiirtoskriptin toteutus asiakkaalle 14
3.1 Projektin vaatimukset 14
3.2 Suunnittelu ja implementointi 15
3.3 Lopputulos 17
4 Jatkokehittäminen (AjaxTransfer) 18
4.1 Toiminnan kuvaus ja tavoitteet 18
4.2 Visiointi ja toteutus 20
4.2.1 Lisäosan käynnistysprosessi 21
4.2.2 Selainpuolen toiminnallisuus 23
4.2.3 Palvelinpuolen toiminnallisuus 26
4.3 Toimivuuden testaus 29
4.4 Tuotos 32
4.4.1 Asennusohjeet 33
4.4.2 Käyttöohjeet 34
5 Yhteenveto ja johtopäätökset 37
Lähteet 38
Kuviot 40
Liitteet 41

1 Johdanto

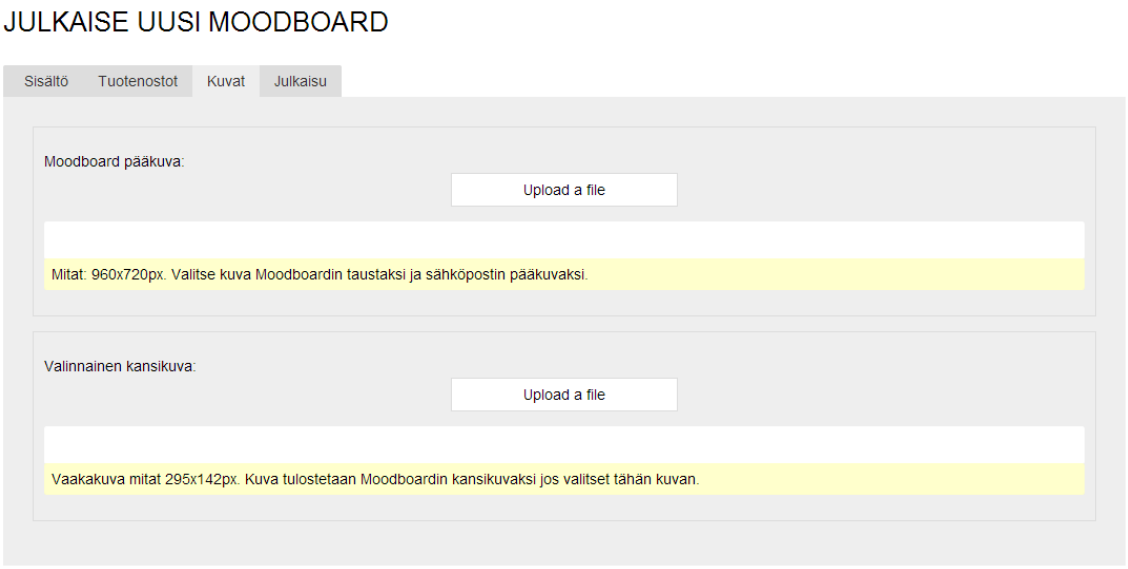
Viime aikoina sisällönhallintajärjestelmät ovat muuttaneet verkkosivujen toteuttamista ja muokkaamista merkittävästi. CMS-järjestelmien avulla yritysten www-sisällön ylläpidosta on tullut helppoa ja nopeaa, mikä selittääkin sisällönhallintajärjestelmien voimakkaan kasvun. Www-sisällönhallinnasta hyötyvät sekä asiakkaat, että web-suunnittelijat. Asiakkaat säästävät verkkosivujen ylläpitokustannuksissa, sillä he voivat CMS-järjestelmän kautta itse päivittää ja luoda uutta sisältöä verkkosivuilleen. Web-suunnittelijoille dynaamisten Internet-sivujen luominen helpottuu huomattavasti muun muuassa CMS-järjestelmien kattavien ja monipuolisten lisäosavalikoimien ansiosta.

Tämän toiminnallisen opinnäytetyön lähtökohtana oli toteuttaa Sivudesign Ky:n asiakkaan www-sivuille tiedostonsiirtolomake, jota kautta käyttäjät voisivat lähettää kuvatiedostoja asynkronisesti palvelimelle. Työ toteutettiin onnistuneesti keväällä 2013 sovitun aikataulun mukaisesti. Asiakasprojektin jälkeen tiedostonsiirtoskriptistä jatkokehitettiin avoimen lähdekoodin verkkosovellus MODX:lle.

Työnantajayritys käytti Internet-sivujen toteuttamisessa MODX Revolution -sisällönhallintajärjestelmää. MODX:n vahvuutena voidaan pitää sitä, että se tarjoaa kehittäjille vapaata ja luovaa muokattavuutta. Vaikka MODX:n käyttäjäyhteisö onkin laaja ja aktiivinen, lisäosien määrän suhteen MODX kalpenee Drupalin ja Joomlan rinnalla. Opinnäytetyön kirjoitushetkellä MODX:lle on julkaistu 474 lisäosaa, kun taas Drupalille on kehitetty 14,085 moduulia. Tästä johtuen koettiin hyödylliseksi kehittää tiedostonsiirtoskriptistä vapaasti ladattava lisäosa, sillä Ajax tekniikkaa hyödyntävää tiedostonsiirto -lisäosaa ei ollut kukaan vielä tehnyt MODX:lle. Tavoitteena oli toteuttaa lisäosasta helppokäyttöinen, toimiva ja tietoturvallinen kokonaispaketti.

Opinnäytetyö käsittelee kahta hanketta: asiakasprojektia (kuvio 1) sekä jatkokehitysprojektia (kuviot 2 ja 3). Asiakasprojekti on räätälöity työnantajan ja asiakkaan tarpeita vastaaviksi. Jatkokehitysprojekti on sen sijaan ammatillista kasvua edistävä ohjelmointiprojekti, joka on suunniteltu yleisempään käyttöön.

Seuraavassa luvussa esitetään opinnäytetyön toteutuksessa käytettyjä tekniikoita sekä ohjelmointikieliä. Luvussa 3 käydään läpi asiakasprojektin vaatimusmäärittelyä, suunnittelua, toteutusta sekä paljastetaan työn lopputulos. Luku 4 kattaa jatkokehitysprojektin kaikki vaiheet suunnittelusta lopputuotoksen esittelyyn. Viimeisessä luvussa tarkastellaan molempia projekteja yleisesti sekä pohditaan toteutuivatko tavoitteet suunnitelmien mukaisesti.



Kuvio 1, Asiakasprojekti

```
<!DOCTYPE html>
<html>
<head>
<title>AjaxTransfer</title>
</head>
<body>

[[!AjaxTransfer? &path=`assets/images/` &max_size=`10000000`]]

</body>
</html>
```

Kuvio 2, Sovelluksen käyttöönottaminen. Esimerkissä lisäosaan viittaava PHP-pala upotetaan HTML-sivun lähdekoodin sekaan.

Kuvio 3, PHP-pala tulostuu www-sivulle tiedostonsiirtolomakkeeksi

2 Käytetyt tekniikat ja ohjelmointikielet

Tässä luvussa käsitellään tiivistetysti opinnäytetyön kannalta oleellisimpia ohjelmistoja, tekniikoita sekä ohjelmointikieliä. Web-sivujen luomisessa on käytetty perinteisten muotoilukielien lisäksi MODX -sisällönhallintajärjestelmää. Hankkeen selainpuolen toiminnallisuudet on toteutettu JavaScriptillä. Palvelinpuolen toiminnot sekä lisäosan asennusskriptit on kirjoitettu PHP-ohjelmointikielellä. Projektin tietokantayhteydet hoitaa relaatiotietokantaohjelmisto MySQL. Työssä käytetyt muotoilukielet HTML ja CSS jätetään esittämättä, koska ne eivät ole tämän projektin kannalta keskeisimpiä tekniikoita.

2.1 MODX

Nykyään Internet-sivuja kehitetään yhä yleisemmin sisällönhallintajärjestelmien avulla. Sisällönhallintajärjestelmä, lyhennettynä CMS (Content Management System) on ohjelma, joka mahdollistaa sisällön julkaisun ja muokkauksen käyttöliittymän kautta. (Kohan 2010.) Erilaisia sisällönhallintajärjestelmiä on useita. Tämän toiminnallisen opinnäytetyön toteutuksessa on käytetty WWW-sisällönhallintajärjestelmä MODX Revolutionia.

Ilmaisessa ja lähdekoodiltaan avoimessa MODX:ssä yhdistyvät sisällönhallintajärjestelmä sekä PHP-kehyssovellus (framework). Edistyneemmät käyttäjät suosivat järjestelmää erityisesti sen joustavan muokattavuuden ansiosta. MODX on suunniteltu siten, että sivuston kehittäjät voivat vapaasti räätälöidä ja muuttaa web-sivua, ohjauspaneelia tai jopa järjestelmän koodia. (Thrash 2013.) Muihin sisällönhallintajärjestelmiin nähden MODX ei ehkä ole aloittelijaystävällisimmästä päästä, mutta järjestelmän pääperiaatteiden sisäistämisen jälkeen se muuttuu yllättävän helppokäyttöiseksi. (Ray 2012).

Käytännössä MODX:n toiminnallisuus nojautuu tietokantoihin. Järjestelmä säilyttää web-sivujen dataa tietokannassa. Kun käyttäjä vierailee MODX:llä toteutetulla kotisivulla, MODX hakee sivun tietokannasta ja lähettää sen nähtäväksi käyttäjän selaimelle. Monet käyttäjät tekevät kotisivuja perehtymättä ollenkaan tietokantayhteyksiin. MODX:n hyödyt tulevat kuitenkin parhaiten esille, kun oppii tallentamaan ja lataamaan tietoa hyödyntäen järjestelmän tietokantaa. (Ray 2012.)

MODX:ää kontrolloidaan sen hallintapaneelin kautta. Näin käyttäjä pystyy muokkaamaan kotisivujaan mistä tahansa ilman, että tarvitsisi käyttää erillistä editoria tai FTP-yhteyttä. Sivujen hallinnoimiseen vaaditaan vain Internet-yhteydellä varustettu tietokone sekä käyttäjätunnus MODX:n hallintapaneeliin. (Ray 2012.)

Web-sivuja rakennetaan MODX:n objekteilla, jotka koostuvat resursseista ja elementeistä.

Resurssit ovat joko dokumentteja, weblinkkejä, symlinkkejä tai staattisia resursseja. (MODX Docs n.d.) Yleensä resursseja käytetään kuitenkin dokumentteina, eli perinteisinä web-sivuina. Resursseihin voi liittää dynaamisia elementtejä, joihin kuuluvat sivupohjat (template), sivupohjan muuttujat (template variable), HTML-palat (chunk) sekä PHP-palat (snippet). (Ray 2012.)

Yleensä sivupohjia käytetään elementteinä, jotka määrittävät sivun layoutin ja ulkoasun. Sivupohjaan voi esimerkiksi sisällyttää sivun bannerin, footerin, menun tai muuta vastaavaa rakennetta, joiden halutaan näkyvän jokaisissa tai tietyntyyppisissä sivuissa. Sivupohjat mahdollistavat sen, että web-sivujen rakenteen ja sisällön voi jakaa eri kokonaisuuksiin. (MODX Docs 2013d.)

Sivupohjan muuttuja (lyhennös TV) on sivupohjaan liitoksissa oleva kustomoitu kenttä. MODX:n resursseilla on tietyt oletuskentät, kuten sivuotsikko, sisältö, kuvaus ja niin edelleen. Käyttäjä voi halutessaan laajentaa resurssien attribuutteja lisäämällä sivuilleen kustomoituja kenttiä, jotka sisältävät esimerkiksi asiakasdataa, kuvia tai muuta lisäsisältöä. Sivupohjan muuttujaa kutsutaan tagilla [[*sivupohjan_muuttujan_nimi]] ja se sijoitetaan sivun HTML-lähdekoodiin. (MODX Docs 2013c.) Kun käyttäjä vierailee sivulla (missä TV tai muu elementti on käytössä), hänen selain lähettää palvelimelle pyynnön sivusta, jolloin MODX hakee sivun tietokannasta ja kokoaa siihen liitetyt elementit mukaan. Tämän jälkeen MODX lähettää sivun käyttäjän selaimelle. (Ray 2012.)

HTML-pala on resurssiin liitettävä osa, joka voi sisältää tekstiä, html-koodia, Javascript-koodia, kuvia tai mitä tahansa Internet-sivuille soveltuvaa sisältöä (Ray 2012). HTML-palaan ei voi suoraan kirjoittaa PHP-koodia, mutta PHP-palojen liittäminen HTML-palaan onnistuu. HTML-paloja voi sijoittaa ylätunnisteisiin, alatunnisteisiin, sivupohjiin tai sivun sisällön sekaan. Kutsuminen toteutuu tagilla [[$html_palan_nimi]]. (MODX Docs 2013a.)

MODX:stä saa paljon enemmän irti, jos osaa ohjelmoida PHP-ohjelmointikielellä. MODX:n PHP-palat mahdollistavat dynaamisten PHP-skriptien suorittamisen sivuilla. Niitä käytetään samalla tavalla, kuten muitakin elementtejä, eli lisäämällä web-sivun sisälle tagi, joka viittaa elementtiin. PHP-paloja kutsutaan tagilla [[php_palan_nimi]]. (Ray 2012.) MODX:lle löytyy runsaasti valmiita snippettejä ja niitä voi ladata lisää MODX:n pakettienhallinnan kautta (kuten tämän toiminnallisen opinnäytetyön tuotoksen).

2.2 JavaScript

Netscapen kehittämä JavaScript on Web-ympäristössä käytettävä ohjelmointikieli, tarkalleen ottaen skriptikieli. JavaScriptillä voi lisätä web-sivuille selainpuolen skriptejä, kontrolloida

selainta, kommunikoida asynkronisesti sekä muokata web-sivun sisältöä dynaamisesti (Flanagan, Ferguson 2006). Suurin osa verkkosivuilla esiintyvistä dynaamisista visuaalisista efekteistä sekä interaktiivisista toiminnoista on ohjelmoitu JavaScriptillä. (JavaScripter.net. n.d.)

JavaScriptin kirjava historia juontaa juurensa vuoteen 1995, jolloin se oli lähinnä harrastelijoiden suosima ohjelmointikieli. Sillä tuotettiin vähemmän hyödyllisiä skriptejä, kuten animoituja perhosia, jotka seurasivat käyttäjän kursoria www-sivulla. Tuohon aikaan JavaScript ei toiminut kunnolla kaikilla selaimilla, mutta nykyään asiat ovat toisin, kun nykyajan selaimet (Firefox, Safari, Chrome, Opera ja IE) ovat standardisoineet menetelmät kuinka JavaScriptiä käsitellään. (McFarland 2011, 11-17.) Tosin vieläkin on olemassa selainten välillä tietynlaisia yhteensopimattomuusongelmia, kuten tämän opinnäytetyön jatkokehitysprojektin aikana kävi ilmi.

JavaScriptillä ohjelmoidaan siten, että koodi kirjoitetaan HTML-tiedostoon <script> -tagien sisälle tai erilliseen tiedostoon, joka liitetään HTML-dokumenttiin. Näistä jälkimmäinen on suotavampi tapa, jos koodia kertyy runsaasti. Alla oleva HTML-koodin sekaan upotettu yksinkertaistettu esimerkkikoodi tulostaisi sivulle tekstiä paragrafin sisälle (kuvio 4).

```
<html>
<head>
<script>

        document.write("<p>Tervehdys Maailma!</p>");

</script>
</head>
<body>
</body>
</html>
```

Kuvio 4, JavaScriptin syntaksi

2.3 PHP

PHP, eli Hypertext Preprocessor on avoimen lähdekoodin ohjelmointikieli, joka soveltuu etupäässä Web-sovelluskehitykseen (PHP n.df). Syntaksiltaan C:ta ja Perliä muistuttava PHP toimii siten, että käyttäjän ladatessa sivua, PHP -prosessorimoduulilla varustettu palvelin tulkitsee sivuston koodin ja tuottaa tuloksena web-sivun käyttäjän selaimelle (PHP n.de). PHP:lle on kehitetty laaja luokkakirjasto, joka pitää sisällään useita hyödyllisiä funktioita (PHP n.dc).

Komentosarjakielen kehitys alkoi vuonna 1994, jolloin tanskalais-grönlantilainen Rasmus Lerdorf laati kotisivuilleen kokoelman C-kielisiä CGI-skriptejä. Sen jälkeen PHP:tä on

jatkokehitetty Andi Gutmasin ja Zeev Suraskin toimesta. He uudelleenkirjoittivat lähdekoodin lähes kokonaan ja loivat siihen ytimen, joka tukee kolmansien osapuolten ohjelmointirajapintoja. (PHP n.dd.) Opinnäytetyön kirjoittamishetkellä PHP:n uusin versio on 5.5.5, joka on julkaistu lokakuussa 2013. Nykyään PHP on asennettu arviolta yli 244 miljoonaan Internet-sivustoon (PHP 2013).

PHP:n komennot voi sisällyttää suoraan HTML-sivun sisälle. Tällöin dokumenttiin tulee merkitä PHP:n tiedostotunniste, jotta palvelin osaisi käsitellä tiedostoa PHP-koodina. Kuvio 5 havainnollistaa kuinka PHP-koodin upotus HTML-dokumentin sekaan tapahtuu.

```
<html>
<head></head>
<body>

<?php
        echo "Tervehdys Maailma!";
?>

</body>
</html>
```

Kuvio 5, PHP:n syntaksi

2.4 MySQL

MySQL on maailman laajimmin käytetty avoimen lähdekoodin SQL-tietokannan hallintajärjestelmä. Sillä voi lisätä, hakea ja prosessoida dataa tietokannasta. Nimensä mukaisesti MySQL tukee Structured Query Language kyselykieltä, joka on yleisin standardoitu tietokantojen kyselykieli. (MySQL Developer Zone 2013.)

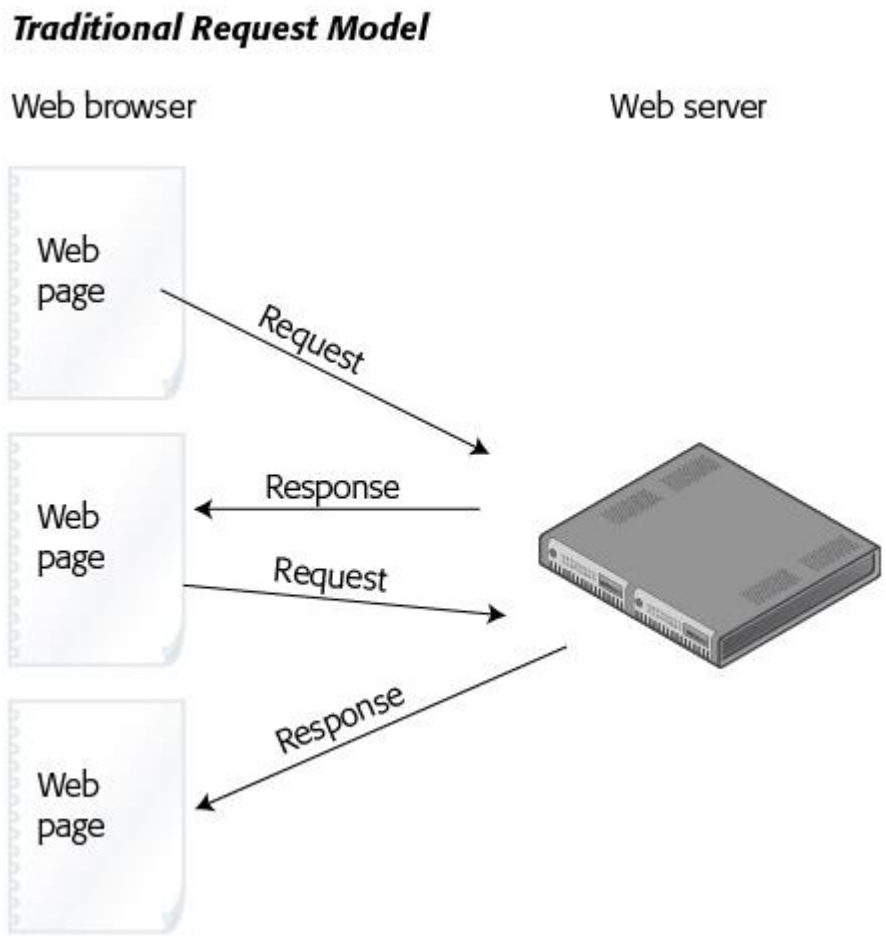
MySQL:n tietokannat ovat relationaalisia, joissa dataa säilytetään erillisissä tauluissa sen sijaan, että kaikki data säilöttäisiin yhteen paikkaan. Relaatiotietokannat toimivat siten, että tietokannan taulujen tiedot yhdistetään toisiinsa toisen taulun avaimella. MySQL:ää kuvaillaan helppokäyttöiseksi, nopeaksi, luotettavaksi ja sitä käytetään kaikenkokoisissa palvelinympäristöissä. (MySQL Developer Zone 2013.)

MySQL-tietokanta luotiin vuonna 1995 ruotsalaisen David Axmarkin sekä suomalaisen Micheal Wideniuksen toimesta. Alun perin tietokantaohjelmisto oli tarkoitettu ruotsalaisen konsultointiyritys MySQL AB:n käyttöön. (Moisio 2008.) Sun Microsystems osti yrityksen vuonna 2008 (Lehtinen 2008). Vuosi jälkeenpäin MySQL siirtyi taas uuteen omistukseen, kun ohjelmistoyritys Oracle Corporation osti Sun Microsystemsin (Noponen 2009). Nykyään Oracle Corporations kehittää, tukee ja levittää MySQL:ää (MySQL Developer Zone 2013).

2.5 Ajax

Ajax (Asynchronous Javascript and XML) sisältää web-sovelluskehityksen tekniikoita, joilla pyritään tuomaan web-sovelluksiin vuorovaikutuksellisuutta, nopeutta ja käytettävyyttä. Tarkalleen ottaen tekniikoiden yhdistelmä koostuu HTML sekä CSS -muotoilukielistä, Document Object Model -rajapinnasta, XMLHttpRequest-objektista, XML -merkintäkielestä tai JSON tiedostonsiirtomuodosta. JavaScriptiä käytetään näiden tekniikoiden yhteensitomiseen. (Garrett 2005.)

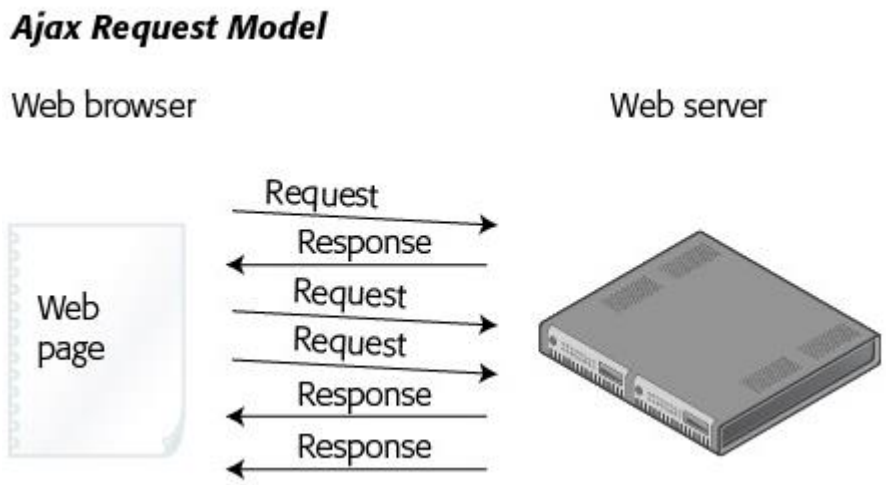
Ajaxin toimintatapa perustuu siihen, että selainohjelma vaihtaa dataa palvelimen kanssa taustalla ilman verkkosivun uudelleenlataamista. Perinteisesti selainohjelma kommunikoi palvelimen kanssa pyynnöillä ja vastauksilla, jolloin web-sivu ladataan aina uudestaan (kuvio 6) (McFarland 2011, 787).



Kuvio 6, Perinteinen HTTP-pyyntö

Ajaxissa selainohjelma pyytää vain uutta informaatiota. Palvelin palauttaa pyydetyn datan, jolloin esimerkiksi web-sivun sisältö tai ulkoasu päivittyy. Uutta sivua ei siis ladata, vaan

kerran ladattu web-sivu päivitetään Javascriptin toiminnoilla. (kuvio 7) (McFarland 2011, 787.)



Kuvio 7, Ajax -pyyntö

Ajaxia käytettäessä yhdistyvät selainpuolen (front end) ja palvelinpuolen (back end) ohjelmointi. Selaimelta siirretään tietoa palvelimelle joko POST tai GET tyyppisillä HTTP-pyynnöillä. Palvelin käsittelee datan ja palauttaa selaimelle vastauksen tapahtumasta. Ajax mahdollistaa esimerkiksi sen, että verkkosivulle pystyy kirjautumaan sisään ilman, että sivulta tarvitsee poistua tai, että sitä tarvitsee päivittää uudestaan. (McFarland 2011, 784-785.)

3 Tiedostonsiirtoskriptin toteutus asiakkaalle

Opinnäytetyön hanke sai alkunsa keväällä 2013 työharjoittelujakson aikana syntyneen projektin tiimoilta. Työ on toteutettu mediatoimisto Sivudesignille. Tehtäväni keskittyi front end sekä back end puolien ohjelmointiin. Työnantaja puolestaan vastasi www-sivun ulkoasusta, suunnittelusta ja toteutuksesta.

Asiakasprojektin toiminnallisuutta tarkastellaan yleisestä näkökulmasta, eikä työn kaikkiin yksityiskohtiin syvennytä kovinkaan tarkasti. Asiakasprojekti on ikään kuin alustus varsinaiseen jatkokehitysprojektiin, jota opinnäytetyö enimmäkseen käsittelee. Seuraavaksi käydään läpi työn vaatimusmäärittelyt, toteutusvaihe ja viimeisessä osiossa tarkastellaan työn lopputulosta.

3.1 Projektin vaatimukset

Työnantajayrityksen asiakkaan www-sivuille oli tarkoitus tehdä osio, missä käyttäjä voi lähettää kuvatiedoston omalta koneelta palvelimeen. Ideana oli, että käyttäjä pystyisi menetelmällä päivittämään sivustonsa moodboardin pääkuvan tai valinnaisen kansikuvan.

Projektin alkuvaiheessa suunniteltiin yhdessä työnantajan kanssa tärkeimmät toiminnallisuudet, jotka tiedostonsiirtoskriptin tulisi pitää sisällään. Tiedostonsiirtoskripti oli pienehkö osa suurta kokonaisuutta. Tästä syystä vaatimusmäärittely oli laadittu lyhyeksi ja ytimekkääksi.

Hanke tuli toteuttaa JavaScript sekä PHP – ohjelmointikielillä. Käytettäväksi sisällönhallintajärjestelmäksi oli valittu MODX. Tiedostonsiirtoskriptiä ei ollut tarkoitus tehdä kertakäyttöiseksi, vaan pikemminkin mallipohjaksi tulevaisuutta varten. Tiedostonsiirtomoduuli palvelee työnantajan asiakkaita. Kohderyhmälle täytyi saada käytettävyydeltään toimiva, laadukas ja moderni työkalu, jolla kuvien lähettäminen olisi vaivatonta.

Tiedostonsiirtoskriptin täytyi hyödyntää MODX Revolutionin ominaisuuksia seuraavalla tavalla:

1. Tallennetun kuvatiedoston polku päivittyy MODx:n sivupohjan muuttujaan (template variable).
2. Skriptin toiminnallisuus kirjoitetaan PHP-paloihin (snippet) sekä HTML-palaan (chunk).
3. Skriptiä kutsutaan HTML-palalla määrittämällä parametreiksi tiedostopolku sekä kuvan leveys ja korkeus.

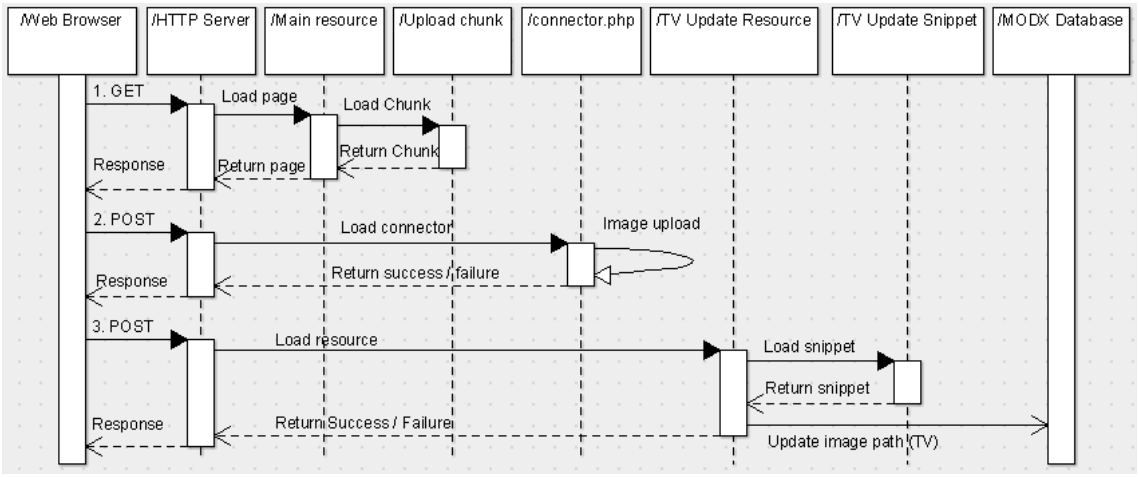
Tiedostonsiirtoskripti oli toteutettava Ajax web-sovelluskehityksen tekniikoilla, jotta tiedostonsiirrosta tulisi dynaamista sekä vuorovaikutuksellista. Kuvien lähettäminen täytyisi olla mahdollista perinteisen tavan lisäksi myös ”raahaus & pudotus” –menetelmällä. Teknisten ominaisuuksien tulisi olla yhteensopivaa kaikkien käytetyimpien selainten kanssa (Internet Explorer, Firefox, Opera, Chrome ja Safari).

3.2 Suunnittelu ja implementointi

Projektin suunnitelmavaiheessa otettiin huomioon vaatimusmäärittelyjen lisäksi aiemmat vastaavat hankkeet, joiden vahvuuksia ja heikkouksia kartoitettiin.

Skripti on rakennettu siten, että web-selainpuolen toiminnallisuus on rajattu HTML-palaan ja web-palvelinpuolen koodit on määritetty PHP-palaan. Näiden vuorovaikutusta kuvataan UML-sekvenssikaaviolla. Asiakasprojektin kiireisen aikataulun vuoksi tiedostonsiirrossa on jouduttu käyttämään Fine uploaderia, joka on erillinen Javascript liitännäinen. Itsenäisesti toteutettu tiedostonsiirtomoduuli on kehitetty varsinaiseen jatkokehitysprojektiin, jota tarkastellaan luvussa 4.

UML-sekvenssikaavio havainnollistaa asiakasprojektin teknisen toiminnallisuuden eri vaiheet (kuvio 8). Sekvenssikaavion pääkohdat on jaettu kolmeen eri HTTP-pyyntöön. Seuraavaksi selvennetään hieman mitä näiden HTTP-pyyntöjen aikana oikein tapahtuu.



Kuvio 8, Asiakasprojekti

1. Alussa käyttäjä siirtyy www-sivulle, jota kautta hän voisi lähettää kuvatiedoston moodboardiin. Ennen sivun latausta web-selain lähettää palvelimelle HTTP GET-pyynnön. WWW-palvelin vastaanottaa pyynnön tarkistaakseen löytyykö tiedostojärjestelmästä GET-pyynnön tiedostoa. (Djce n.d.) Oletetaan, että käyttäjä hakee tiedostoa nimeltä kuvanlahetys.html. Kyseinen tiedosto on MODX:n resurssi,

joka hakee kaikki projektin Javascript – ja HTML-koodit erillisestä HTML-palasta. Tähän HTML-palaan viitataan kuvanlahetys.html – tiedostoon upotetulla kutsulla:

[[$tvImgUpdate? &path=`assets/images/tvupload/` &width=`200` &height=`200`]]

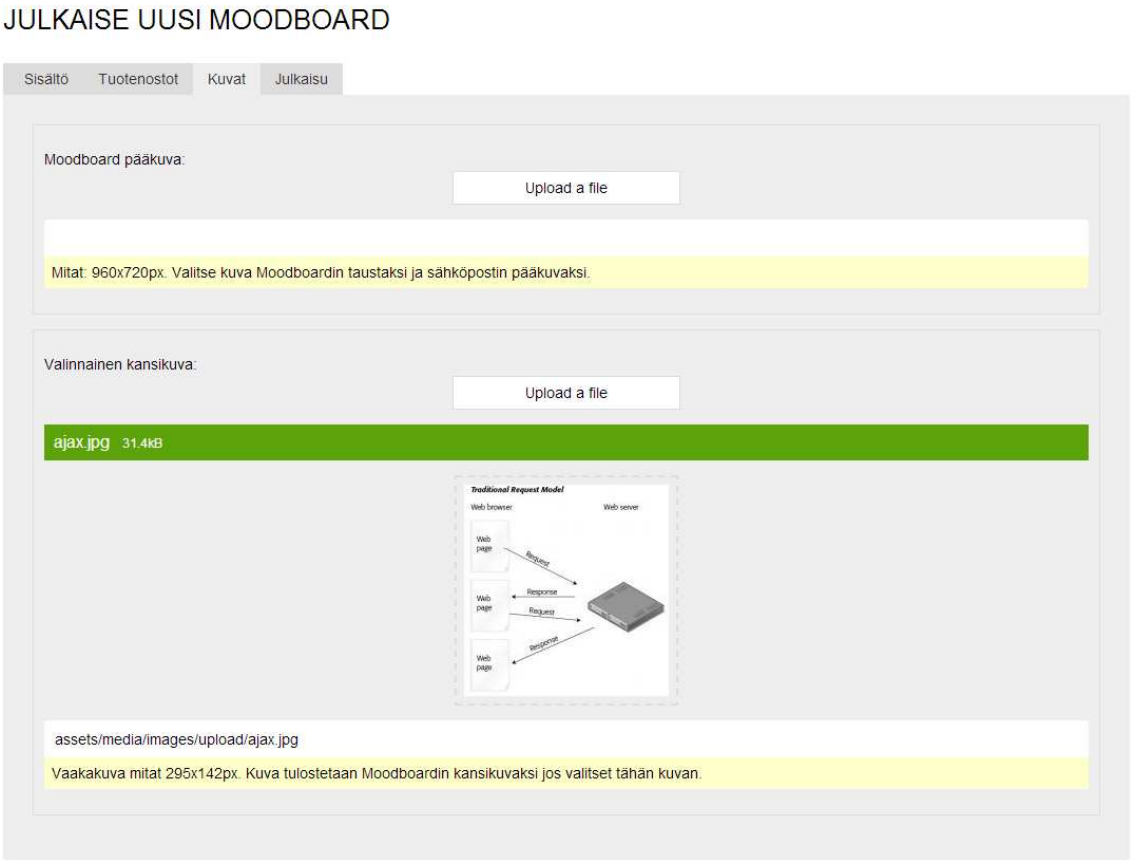
Esimerkissä tvImgUpdate on viitattavan HTML-palan nimi. Parametreiksi on asetettu tiedostopolku (kohde palvelimen tiedostojärjestelmässä, jonne lähetetyt kuvat menevät) sekä kuvan leveys ja korkeus (ei muuta kuvatiedoston kokoa, mutta määrittää kuinka tilavana kuva näkyy www-sivulla). Äskeisessä malliesimerkissä lähetettävä kuvatiedosto kopioituisi käyttäjän koneelta palvelimen hakemistoon assets/images/tvupload/ ja käyttäjä näkisi lähetetyn kuvan selaimessaan 200x200 pikselin kokoisena. HTML-pala, eli tvImgUpdate sisältää tiedoston lähetykseen ja tallentamiseen liittyvät selainpuolen funktiot. Kun MODX:n resurssiin on yhdistetty HTML-pala, HTTP palvelinohjelma vastaa selaimen GET-pyyntöön lähettämällä tälle resurssin (kuvanlahetys.html) sisällön.

2. Käyttäjä valitsee koneeltaan kuvatiedoston, jonka hän tallentaa palvelimeen. Tiedoston tallentamisessa on käytetty Fine uploaderia (Fine Uploader 2013). Tämän JavaScript liitännäisen tiedostonsiirtoon liittyvät funktiot on sisällytetty HTML-palaan sekä sen palvelinpuolen toiminnot määritetty erilliseen connector.php tiedostoon. Kun käyttäjä on valinnut tiedoston, web-selain lähettää POST-pyynnön HTTP-palvelimelle. POST-pyyntö pitää sisällään kuvatiedoston sekä muita tietoja, joita Fine uploader käyttää tiedoston tallentamiseen. Lähetetty tiedosto tallentuu palvelimelle, mikäli se on oikean tyyppinen ja kokoinen. Prosessin jälkeen palvelin vastaa web-selaimelle talletuksen onnistumisesta. Jos kaikki meni niin kuin pitikin, JavaScript funktio lisää kuvan esikatseltavaksi www-sivulle. Kuvan koko muunnetaan phpthumbOf – lisäosalla (MODX 2013). PhpthumbOf käyttää kuvan koon rajaamisessa samoja arvoja, mitä HTML-palan parametreihin on asetettu (&width=`200` &height=`200`).

3. Lopulta käyttäjä voi tallentaa lähetetyn kuvan moodboardin pääkuvaksi tai kansikuvaksi. Teknisesti tämä on toteutettu siten, että kuvatiedoston URI-osoite tallentuu MODX:n sivupohjan muuttujaan. ”Tallenna” – nappia painaessa käyttäjän web-selain lähettää HTTP-palvelimelle POST-pyynnön, joka sisältää kuvatiedoston URI-osoitteen. Palvelin vastaanottaa pyynnön ja lataa tarvittavan resurssin ja PHP-palan, joka sisältää tallentamiseen liittyvän skriptin. Kuvatiedoston URI-osoite päivittyy MODX:n tietokantaan, jonka jälkeen palvelin vastaa web-selaimelle onnistuneesta prosessista.

3.3 Lopputulos

Työ toteutettiin mediatoimisto Sivudesignin asiakkaalle aikataulun mukaisesti keväällä 2013. Kuvio 9 demonstroi lopputuotosta asiakkaan näkökulmasta (tallennus/esikatselu ja peruutus -painikkeet eivät mahtuneet kuvaan).



Kuvio 9, Asiakasprojektin lopputulos

Projektin vaatimukset täyttyivät onnistuneesti ja kaikki asiakkaalle räätälöidyt toiminnallisuudet toimivat niin kuin pitääkin. Työn lopputulos poikkesi kuitenkin hiukan suunnitellusta, sillä työnantaja sisällytti samalle sivulle muitakin toiminnallisuuksia (kuviossa 9 näkyvät sisältö, tuotenostot ja julkaisu -välilehdet). Näiden yhdistäminen samalle sivulle muutti hiukan alkuperäisen suunnitelman rakennetta. Päätoiminnallisuus on kuitenkin pysynyt täysin samana.

Yhtenä tavoitteena oli tehdä tiedostonsiirtoskriptistä helppokäyttöinen snippetti, eli PHP-pala. Työnantajaa ajatellen PHP-palaa olisi kätevää hyödyntää tulevaisuuden asiakasprojekteissa. Tämä tavoite ei kuitenkaan toteutunut, koska kiireisen aikataulun vuoksi työssä jouduttiin käyttämään Fine uploaderia, joka toimii vain staattisessa muodossa. Fine uploaderia käyttävää skripti on melko työlästä asentaa MODX:n ympäristöön, koska

palvelinpuolen PHP-sivu täytyisi luoda erikseen kopio ja liitä – menetelmällä, sekä tiettyjä asetuksia joutuisi määrittelemään selain – ja palvelinpuolen koodiin. Tästä kehkeytyikin suunnitelma jatkokehittää työstä Ajax web-sovelluskehityksen tekniikalla varustettu tiedostonsiirtoskripti, joka olisi helppo asentaa MODX package managementin kautta ja helppo ottaa käyttöön www-sivuille yhden PHP-palan kutsulla.

4 Jatkokehittäminen (AjaxTransfer)

Asiakasprojektia toteutettaessa kävi ilmi, että MODX:lle ei löytynyt Ajax – tekniikalla varustettua tiedostonsiirto – lisäosaa. Tästä johtuen tuntuikin luontevalta jalostaa työharjoittelussa opitut asiat verkkosovellukseksi, jota kuka tahansa MODX:n käyttäjä voisi käyttää verkkosivullaan. MODX:lle löytyi tavallisia tiedostonsiirtoon tarkoitettuja lisäosia. Tehtävänä oli vertailla näiden lisäosien ominaisuuksia, poimia niistä parhaimmat puolet ja pyrkiä tekemään tiedostonsiirto -lisäosa, joka hyödyntäisi Ajaxia.

Kehitysprojekti on jatkoa asiakasprojektille. Se on ikään kuin viimeistelty versio asiakasprojektista. Asiakasprojekti ja jatkokehitysprojekti eroavat siten, että asiakasprojekti on räätälöity enemmän työnantajan ja asiakkaan tarpeita vastaaviksi, kun taas jatkokehitysprojekti on suunniteltu yleisempään käyttöön. Lisäksi jatkokehitysprojekti ei keskity pelkästään kuvatiedostojen lähetykseen, vaan sallitun tiedostotyypin voi sivuston kehittäjä itse määrittää.

Kehitysprojektin työnimikkeeksi valittiin AjaxTransfer. Jatkokehitysprojektilla ei ole tilaajaa, vaan se on kehitetty lähinnä kiinnostuksesta aihetta kohtaan. Pääasiassa on koettu, että projektin toteutus kehittäisi tehokkaasti ammatillista kasvua. Vastasin itsekseen koko projektin suunnittelusta, ohjelmoinnista sekä testaamisesta.

Hankkeen kohderyhmäksi on suunnattu työnantaja sekä tavalliset MODX:n käyttäjät. Lisäosa on tarkoitettu niille web-kehittäjille, jotka tarvitsevat sivuilleen Ajax-tiedostonsiirtolomakkeen. Kaikilla ei ole kuitenkaan aikaa, osaamista tai kiinnostusta ohjelmoida tiedostonsiirtoskriptiä. Lisäosalla pyritään helpottamaan hieman heidän elämää.

4.1 Toiminnan kuvaus ja tavoitteet

Tavoitteena oli tehdä MODX:lle asennettava vapaan lähdekoodin Ajax tiedostonsiirto -lisäosa, jolla voisi lähettää asynkronisesti tiedostoja palvelimelle ylläpitäjän määritysten mukaisesti. Tarkoitus oli myös julkaista lisäosa MODx:n sivuille, jotta kuka tahansa voisi ladata sen, muokata sitä tai antaa työstä palautetta.

Pyrkimyksenä oli saada sovelluksesta helppokäyttöinen, toimiva ja tietoturvallinen. Helppokäyttöisyydellä tarkoitetaan sitä, että sovelluksen voisi asentaa vaivattomasti MODX:n package managementin kautta ja ottaa lisäosan käyttöön www-sivulle yhdellä PHP-palan pyynnöllä:

[[!AjaxTransfer? &path=`path/to/your/files/`]]

PHP-palan pyyntöön on pystyttävä asettamaan tiedostonsiirtoon liittyviä asetuksia, kuten lähetettyjen tiedostojen kohdekansio sekä tiedoston kokoon ja muotoon liittyvät rajoitukset. PHP-palan on tarkoitus tulostaa www-sivulle tiedostonsiirtolomake.

Toimivuudella tähdätään siihen, että kaikki algoritmit suorittavat käskysarjat tavoitteiden mukaisesti ja koodi on yhteensopivaa tunnetuimpien selainten kanssa. Arkkitehtuurivaatimuksiltaan lisäosan täytyy tukea MySQL – tietokantaohjelmistoa sekä sen pitää toimia MODX:n versioissa 2.2.x. Lisäosa ei saa vaatia käyttäjältä erillisiä ohjelmistokirjastoja tai lisäosia.

Ohjelmistovaatimuksen sykli on määritetty seuraavanlaisesti:

1. Sovellus saa syötteitä www-sivulle upotetun PHP-palan parametreista, jotka on määritetty sisällönhallintajärjestelmään sisäänkirjautuneen käyttäjän/adminin toimesta. Syötteet ovat tiedostonsiirtoon tai tietoturvaan liittyviä asetuksia.
2. Syötteet vaikuttavat sovelluksen toimintaan. Asetukset tallentuvat tietokantaan.
3. Sovellus tulostaa www-sivulle tiedostonsiirtolomakkeen.
4. Käyttäjät lähettävät tiedostonsiirtolomakkeen kautta tiedostoja palvelimelle käsiteltäväksi.
5. Sovellus tarkistuttaa tiedostot syötettyjen asetusten mukaisesti ja siirtää ne asetuksissa määritettyyn kohdehakemistoon. Sivunlatauksen jälkeen sykli alkaa alusta. (Paakki 2011.)

Sovelluksen tietoturva ansaitsee erityistä huomiota, sillä tiedostonsiirtolomakkeisiin liittyy merkittäviä tietoturvauhkia (Calin 2009). Laatuvaatimuksena lisäosan on taattava tiedostonsiirron luottamuksellisuus, eheys, käytettävyys, luotettavuus sekä suorituskyky. Toisin sanoen, lisäosan tulee tarkistaa käyttäjien lähettämien tiedostojen oikeellisuus, lisäosaa saa hallita vain siihen valtuutetut käyttäjät, tiedostonsiirtolomakkeen on oltava aina käytettävissä kun sitä kutsutaan PHP-palalla, tiedostonsiirrossa ei saa tapahtua odottamattomia virheitä sekä sovelluksen suorituskyky on oltava tehokasta.

Lisäosan kehittämisestä ei synny kuluja, eikä työlle ei aseteta takarajaa. Se on niin sanottu ikuisuusprojekti, jota kehitetään ja parannellaan kiinnostuksen mukaan niin kauan, kun lisäosalle riittää kysyntää.

4.2 Visiointi ja toteutus

Sovelluksen päärakenne koostuu kahdesta PHP-palasta ja yhdestä HTML-palasta. Applikaation käynnistystiedosto, nimeltään AjaxTransfer on PHP-pala (Liite 7), johon viittaamalla sovellus ladataan käyttöön www-sivulle. Selainpuolen koodit (HTML ja Javascript) on kirjoitettu HTML-palaan ajaxtransfer.chunk.tpl (Liite 6) ja palvelinpuolen toiminnot ovat PHP-palassa AjaxTransfer_Server (Liite 8). Tiedostonsiirron tietoturvaan liittyvät tarkastustoimenpiteet on toteutettu varmuuden vuoksi sekä selain – että palvelinpuolen osiin.

Tiedostonsiirtoskripti on paketoitu Extraksi MODX:n virallisen ohjeistuksen mukaisesti (MODX Docs 2013b). Lisäosaan kertyi kymmenen asennustiedostoa, mutta niiden toiminnallisuutta ei tarkastella tässä opinnäytetyössä.

Web-sivuston ylläpitäjän ei tarvitse välittää muusta kuin sovelluksen lataavasta PHP-palasta, jonka viitteeseen hän syöttää parametrina omat asetukset. Näihin asetuksiin kuuluvat ID-numero, tiedostopolku, esto satunnaisen tiedostonimen generoimiselle, suurin sallittu tiedoston koko, sallitut tiedostopäätteet sekä sallitut tiedostotyypit. Asetuksista vain tiedostopolku on pakollista määrittää.

ID-numerolla viitataan tietokantataulun riviin, jonne asetukset talletetaan. ID-numero tulee määrittää, mikäli samassa MODX -järjestelmässä käytetään useampaa kuin yhtä tiedostonsiirtolomaketta. Jos ID-numeron jättää tyhjäksi, sovellus käyttää oletusnumeroa 1.

Tiedostopolku on kohde palvelimen tiedostojärjestelmässä, jonne sivuston ylläpitäjä haluaa sijoittaa käyttäjien lähettämät tiedostot. Tiedostopolun hakemiston käyttöoikeuksien tulee sallia sovellukselle kirjoitusoikeudet, jotta tiedoston siirto olisi mahdollista.

Tiedostoille luodaan tietoturvasyistä oletusarvoisesti satunnainen nimi. Tämän voi kuitenkin halutessaan estää syöttämällä parametrin ”random_name” arvoksi ”false”.

Palvelunestohyökkäysten välttämiseksi sovellus siirtää vain alle viiden megan tiedostoja. Kyseisen oletusasetuksen voi muuttaa parametrilla &max_size = x, jossa x on suurimman sallitun tiedoston koko bitteinä (esimerkiksi 7000000 bittiä on 6.6 megabittiä). Asetuksella &max_size=`unlimited` voidaan lähettää niin suuria tiedostoja, kuin PHP:n asetukset vain sallivat.

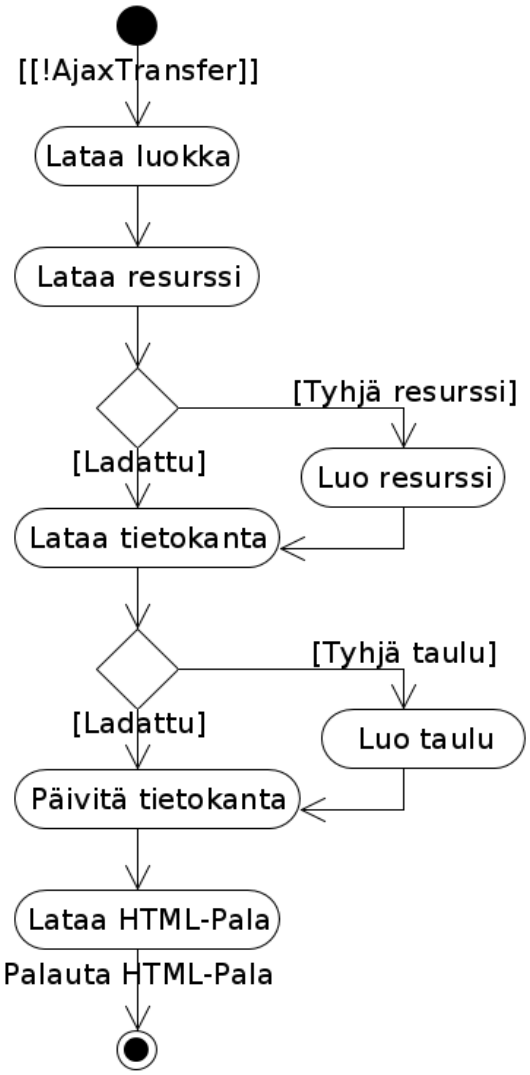
Ylläpitäjän tulee voida rajata minkälaisia tiedostoja käyttäjät voivat lähettää palvelimelle. Sallittuja tiedostotyyppejä voi asettaa niiden päätteen sekä Internet media typen (MIME) mukaan. Esimerkkinä parametrit &allowed_extensions=`jpg,png` ja &allowed_mimes=`image/jpeg, image/png` sallivat lähetettäväksi vain JPG –ja PNG tyyppiset kuvatiedostot. Sallittuja tiedostotyyppejä ei ole pakollista rajata. Tiedostotyyppeihin perustuvan tietoturvatarkastuksen voi ohittaa asetuksilla &allowed_extensions=`unlimited` ja &allowed_mimes=`unlimited`.

Seuraavaksi analysoidaan sovelluksen päätoimintoja. Sovellus koostuu useasta eri tiedostosta, joista suurin osa on lisäosan asennusskriptejä. Tekninen analyysi painottuu kolmeen eri osaan (Liittet 6-8).

4.2.1 Lisäosan käynnistysprosessi

Ensimmäinen PHP-pala AjaxTransfer käynnistää sovelluksen. Tämän PHP-palan pääfunktio painottuu tietokannan taulun luomiseen tai päivittämiseen, sekä MODX Ajax yhdistäjän (connector) lataamiseen tai tarvittaessa luomiseen. Ajax yhdistäjä on tavallinen MODX:n PHP-tyyppinen resurssi, joka hakee sovelluksen palvelinpuolen metodit dynaamisesta PHP-palasta AjaxTransfer_Server. Käynnistysprosessin jälkeen applikaatio lataa sovelluksen selainpuolen koodit ja palauttaa ne [[!AjaxTransfer]] -kutsun tilalle.

Alla näkyvä UML aktiviteettikaavio havainnollistaa AjaxTransfer PHP-palan toiminnallisuudet vaihe kerrallaan (kuvio 10). Seuraavaksi näitä prosessin eri osia tarkastellaan hieman yksityiskohtaisemmin.



Kuvio 10, AjaxTransfer PHP-pala

Sovellus käynnistyy, kun AjaxTransfer PHP-pala ladataan. AjaxTransferia viittaava PHP-pala upotetaan HTML-koodin sekaan:

[[!AjaxTransfer? &path=`path/to/your/files/`]]

Ensimmäiseksi sovellus lataa lisäosan luokan instanssin sekä asettaa lisäosan tiedostopolun MODX:n tietokantayhteyttä varten (MODX Docs 2013b).

Tässä projektissa AJAX – tekniikan käyttö on toteutettu siten, että palvelinpuolen toiminnallisuudet ladataan erilliseen PHP-tyyppiseen resurssiin. Tämä resurssi luodaan lisäosan asennuksen yhteydessä. Sama toimenpide löytyy myös sovelluksen käynnistysprosessista siltä varalta, jos MODX ei pystynyt jostain syystä luomaan resurssia

asennusvaiheessa (tai jos joku on poistanut resurssin). Resurssi asetetaan näkymättömäksi, niin etteivät MODX:n käyttäjät näe sivua hallintapaneelissa. Sen sisällöksi tulee pelkästään kutsu PHP-palaan nimeltä AjaxTransfer_Server, joka sisältää tiedostonsiirtoon liittyvät toimenpiteet.

Lisäosa tallentaa pääkäyttäjien määrittämät asetukset MODX:n tietokantaan. AjaxTransferin tietokantataulu oletusasetuksineen luodaan jo asennusvaiheessa. Uutta taulua tuskin tarvitsee luoda sovelluksen käynnistyessä, mutta se tehdään varmuuden varalta, jos taulua ei jostain syystä löydykään.

Lisäosan tietokantataulun rivi koostuu neljästä kentästä: id, path, random_name, max_size, allowed_extensions ja allowed_mimes (kuvio 11). Jokaisella rivillä on yksilöllinen avain, eli ID. Kyseisellä avaimella erotetaan tiedostonsiirtolomakkeet toisistaan, mikäli MODX järjestelmässä on käytössä enemmän kuin yksi tiedostonsiirtolomake. AjaxTransferin PHP-palaan syötetyt asetukset tallentuvat tietokantaan. Path, eli tiedostopolku on ainoa pakollinen täytettävä kenttä. Muut kohdat voi halutessaan jättää tyhjäksi, mikä toisaalta ei ole tietoturvasyistä suositeltavaa.

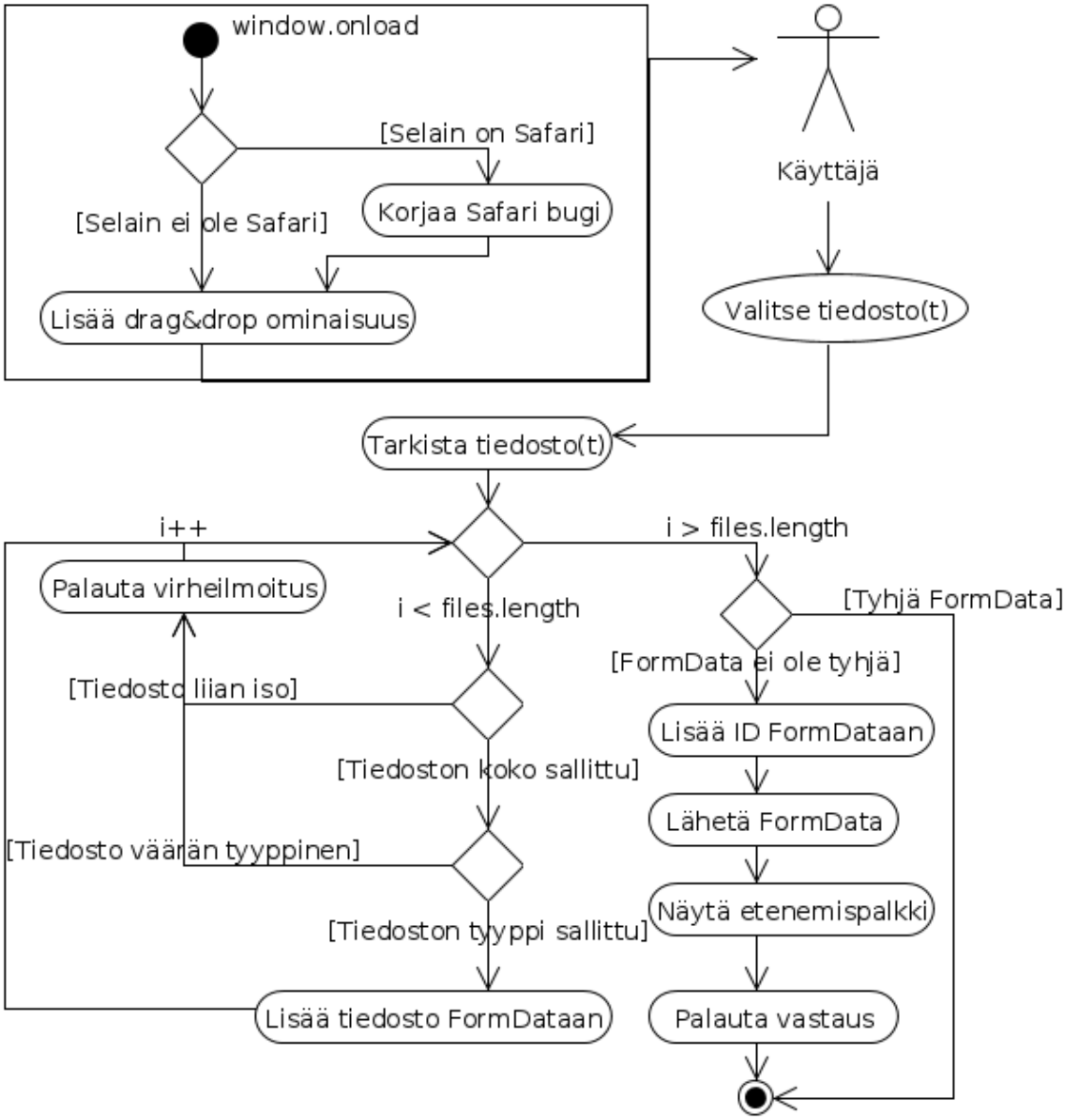
	id	path	random_name	max_size	allowed_extensions	allowed_mimes
	5724	assets/documents/		6000000	pdf	application/pdf
	26541	assets/images/		6000000	jpg,png,gif	image/jpeg,image/png,image/gif
	335315	assets/stuff/	false	unlimited	unlimited	unlimited

Kuvio 11, Esimerkki lisäosan tietokantatauluista

Sovellus lataa HTML-palan, joka sisältää kaikki selainpuolen toiminnallisuudet. HTML-palaan välittyy osa pääkäyttäjän asettamista parametreista, joita tarvitaan selainpuolen toiminnoissa. Lopulta HTML-palan kaikki sisältö haetaan [[!AjaxTransfer]] -kutsun tilalle.

4.2.2 Selainpuolen toiminnallisuus

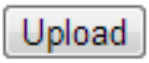
Sovelluksen selainpuolen koodit (HTML ja Javascript) on kirjoitettu HTML-palaan ajaxtransfer.chunk.tpl. Tämä noin 250 koodirivin pituinen HTML-pala tulostuu www-selaimeen yhdeksi minimalistiseksi painikkeeksi, jota klikkaamalla (tai raahaamalla tiedostoja painiketta kohti) käyttäjä voi lähettää tiedostoja palvelimeen. Kuten seuraavasta UML:n aktiviteettikaavion sekä käyttötapauskaavion kombinaatiosta voi hahmottaa, selainpuolen pääfunktio on vastaanottaa käyttäjän valitsemat tiedostot, tarkistuttaa niiden koko ja tyyppi ja lähettää ne palvelimelle prosessoitavaksi (kuvio 12).



Kuvio 12, Selainpuolen toiminnallisuus

Sovelluksen selainpuolen toiminnot käynnistyvät web-sivun latautumisen jälkeen Javascriptin window.onload – funktiolla. Kyseinen käynnistysfunktio lisää sovellukseen drag and drop – ominaisuuden, joka mahdollistaa tiedostojen ”raahaamisen” web-lomakkeeseen (Kalla 2010). Työtä toteutettaessa kävi ilmi, että tiedostonsiirtolomakkeen tyyliasetukset eivät toimi Applen Safari – selaimella. Lisäksi HTML5:n mukana tullut <progress> -tagi ei näy Safarilla. Tästä johtuen käynnistysfunktioon täytyi lisätä ehtolause, joka suorittaa viankorjaus metodin, mikäli käyttäjän www-selain osoittautuu Safariksi.

Normaalisti lomake-elementti näkyy käyttäjälle yhtenä yksinkertaisena painikkeena (kuvio 13).



Kuvio 13, Normaali näkymä lomakkeesta

Safarilla toiminnallisuus poikkeaa siten, että tiedostot valitaan esikatseltavaksi ennen varsinaista lähettämistä (kuvio 14).

Kuvio 14, Näkymä lomakkeesta Safari – selaimella katsottuna

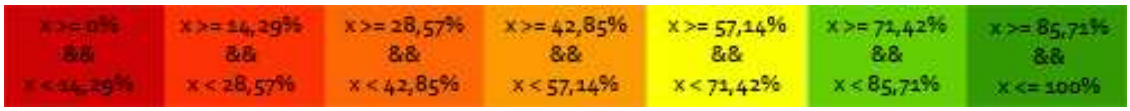
Oletetaan, että seuraavaksi käyttäjä valitsee yhden tiedoston koneeltaan lähetettäväksi palvelimelle (AjaxTransferilla voi lähettää kerralla niin monta tiedostoa, kuin palvelimen PHP:n asetukset sallivat). Sen seurauksena tiedostonsiirtoon liittyvät metodit aktivoituvat. Ennen lähetystä koodi tarkistaa onko syötetty tiedosto oikean kokoinen ja tyyppinen vertailemalla sen ominaisuuksia ylläpitäjän määrittämiin asetuksiin. HTML-palan funktio forbiddenSize(file) vertaa ylittääkö valittu tiedosto sallitun tiedostokoon rajan. Maksimirajan oletusarvoksi tulee 5000000 tavua (noin 5Mb), mikäli sivuston ylläpitäjä jättää parametrin ’max_size’ tyhjäksi.

Tietoturvamenetelmän funktio forbiddenType(file) puolestaan tarkistaa valitun tiedoston Internet media typen (MIME). MIME-tyyppi on standardoitu tunniste, joka kertoo sisällön tiedostomuodon. Parametriin määritettyjä sallittuja tiedostotyyppejä voi olla useita, jonka vuoksi tarkistus on toteutettu siten, että ensin sallitut mime-tyypit erotellaan alkioiksi array-taulukkoon. Taulukon alkioita vertaillaan yksi kerrallaan lähetettävän tiedoston MIME-tyyppiin. Funktio palauttaa arvon true, mikäli vertailun haku ei löydä lähetettävän tiedoston MIME-tyyppiä taulukon alkioista. Muussa tapauksessa vertailtu tyyppi on sallittu ja tiedosto voidaan lähettää palvelimelle käsiteltäväksi.

Tietoturvatarkastuksen jälkeen www-sivulle tulee virheilmoitus, jos käyttäjän valitsema tiedosto ilmenee liian isoksi tai väärän tyyppiseksi. Tällöin tiedostoa ei lähetetä Ajax-pyyntönä ollenkaan. Muussa tapauksessa tiedosto liitetään FormData – objektiin, johon tulee mukaan myös tietokantataulun rivin ID-numero. FormData lähetetään HTTP Ajax – pyynnön mukana käsiteltäväksi palvelimelle.

Tiedostosiirron aikana palvelin palauttaa asynkronisesti selaimelle tietoja siirron etenemisestä. Näistä tiedoista muodostetaan prosentuaalinen arvo siitä, kuinka paljon FormData -objektia on lähetetty. Prosenttimäärä määrittyy HTML5:n progress tagin arvoksi.

Progress tagi näkyy käyttäjälle tiedostonsiirron etenemispalkkina. Tarkoituksena oli ohjelmoida etenemispalkki JavaScriptillä sellaiseksi, että tiedostonsiirroon etenemisprosentti määrittäisi etenemispalkin värin (kuvio 15). Esimerkiksi kun tiedostoa on siirretty noin puolet, etenemispalkki näkyisi käyttäjälle oranssina. Teoriassa yksinkertaiselta vaikuttava idea oli käytännössä vaikeaa toteuttaa, koska tietyt selaimet käyttävät HTML5:n etenemispalkkiin omia muotoiluasetuksia, joita ei pysty muuttamaan dynaamisesti JavaScriptillä. Parasta olisikin toteuttaa kokonaan erilainen etenemispalkki CSS – muotoilukielen avulla, mutta siihen ei enää valitettavasti riittänyt aikaa.



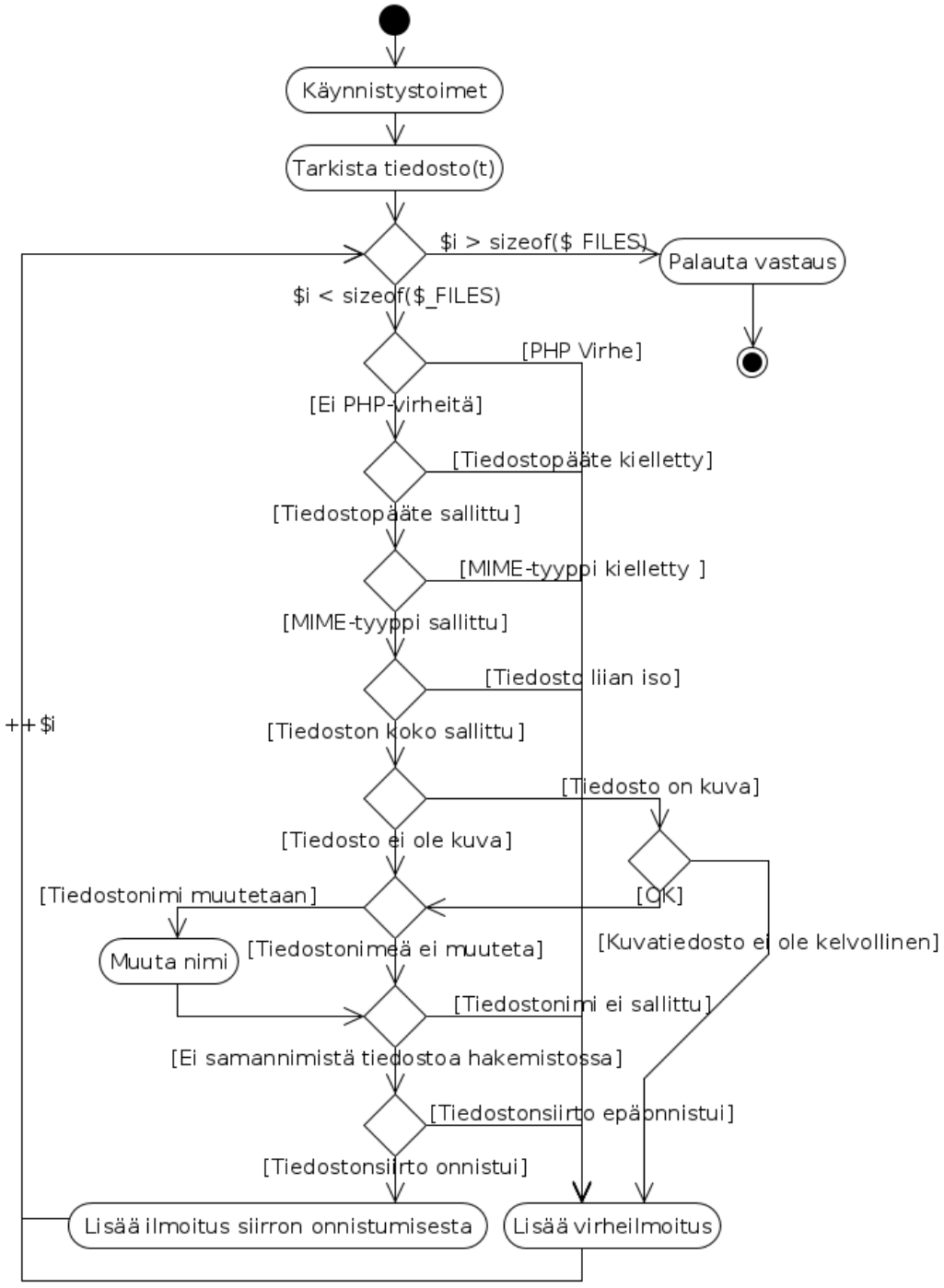
Kuvio 15, Etenemispalkin värit eri prosenttimäärille. X = Tiedostonsiirron etenemisprosentti

Kun serveri on prosessoinut lähetetyn tiedoston, se palauttaa vastauksen www-sivulle siirron onnistumisesta ilman, että koko web-sivua tarvitsisi ladata uudestaan.

4.2.3 Palvelinpuolen toiminnallisuus

Tiedostojen talletus palvelimen tietokantaan on määritetty PHP-palaan AjaxTransfer_Server. Sovelluksen palvelinpuolen toimintojen tarkoitus on vastaanottaa käyttäjän lähettämät tiedostot, verifioida niiden oikeellisuus ja lopulta siirtää tiedostot sivuston ylläpitäjän määrittämään tiedostopolkuun. Tietoturvaan on pyritty kokoamaan parhaita käytäntöjä useasta eri lähteestä. Turvaamattomat tiedostonsiirtoskriptit ovat suuri riski palvelimen tietoturvalle. Kaikkea tietoturvaan liittyviä riskejä ei voi estää pelkästään lisäosan palvelinpuolen toiminnoilla. Esimerkiksi Dos-hyökkäysten varalta palvelimen ylläpitäjän täytyy tiukentaa PHP:n ja HTTP-palvelinohjelman oletusasetuksia.

Seuraava UML aktiviteettikaavio havainnollistaa sovelluksen palvelinpuolen tiedostonsiirtoon sekä tietoturvaan liittyvät toimenpiteet (kuvio 16).



Kuvio 16, Palvelinpuolen toiminnallisuus

Alussa, kun käyttäjä on lähettänyt web-selaimen kautta esimerkiksi yhden tiedoston, PHP vastaanottaa POST-pyynnön multipart/form-data muotoisen sisällön, luo tästä väliaikaisen tiedoston satunnaisella tiedostonimellä ja siirtää sen väliaikaiseen kansioon (esimerkiksi /var/tmp/php6yXOVs). (Calin 2009.) POST-pyynnön mukana tulee myös id-numero, jolla haetaan MODX:n tietokannasta käyttäjän laittamat asetukset, kuten tiedostopolku, maksimikoko ja sallitut tyypit. Tämän jälkeen tiedostolle tehdään useavaiheinen

tietoturva/virheiden tarkistus. Tietoturvatarkastuksen toteuttamisessa on osittain käytetty apuna Tidy designin laatimaa PHP secure file upload scriptiä (Tidy designs 2012).

Ensiksi sovellus tarkistaa PHP:n virheilmoitukset. Tiedostonsiirto epäonnistuu, mikäli PHP:n superglobaali muuttuja $_FILES['userfile']['error'] on jotain muuta kuin 0 (PHP n.db). Esimerkiksi jos kyseinen muuttuja palauttaa arvon 1, tällöin käyttäjän lähettämän tiedoston koko on suurempi mitä palvelimen PHP:n asetukset sallivat. Oletusarvoisesti PHP suostuu siirtämään maksimissaan vain 2 megatavun kokoisia tiedostoja (PHP n.da). Se ei nykyään riitä, jos käyttäjät lähettävät esimerkiksi kameralla otettuja pakkaamattomia kuvia. PHP:n asetuksia joutuukin usein muuttamaan sivuston tarpeiden mukaisesti.

Toisessa vaiheessa lähetetyn tiedoston päätettä verrataan sallittuihin tiedostopäätteisiin. Ylläpitäjän spesifioidut sallitut tiedostopäätteet haetaan MODX:n tietokannasta. Tiedostonsiirto ei mene läpi, jos lähetetyn tiedoston päätettä ei löydy sallituista tiedostopäätteistä. Todellisuudessa tämä vaihe tietoturvatarkastuksesta on kuitenkin kierrettävissä melko yksinkertaisella tavalla. Oletetaan, että käyttäjä lähettää tiedoston kuva.jpg ja ylläpitäjä on sallinut jpg –päätteet. Tällöin tiedosto lähetetään palvelimelle kuten pitääkin. Mutta entä jos hakkeri naamioi haittaohjelman sisältävän PHP-tiedoston muotoon JPG? Eli tiedostolle luodaan kaksoispääte ja se nimetään esimerkiksi muotoon haittaohjelma.php.jpg. Kyseisen tiedoston pystyy lähettämään palvelimelle, koska PHP luulee tiedoston olevan kuva. (Daniweb 2012.) Tämä yksinkertainen haavoittuvuustesti on osoittanut, että pelkästään tiedostopäätteeseen perustuvaan validointiin ei ole luottamista. Tietoturvalliseen tiedostonsiirtoon tarvitaan muitakin validointimenetelmiä.

Tiedoston muodon saa tarkemmin selville sen MIME-tyypin perusteella. Seuraavassa vaiheessa sovellus hakee lähetetyn tiedoston MIME-tyypin ja tarkistaa onko tyyppi sallittu vai ei. Menetelmässä käytetään samaa periaatetta mitä tiedostopäätteen vertailussa, eli tiedostonsiirto keskeytyy, mikäli lähetetyn tiedoston MIME-tyyppiä ei löydy sallituista MIME-tyypeistä. MIME-tyypin vertailu on toteutettu kahdella eri tavalla, joista sovellus suorittaa jommankumman. Tiedoston tyyppi tarkistetaan perinteisellä menetelmällä, jos palvelimen PHP:n versio osoittautuu vanhentuneeksi (alle 5.3.0). Tässä tapauksessa tiedoston tyyppi saadaan PHP:n superglobaalista muuttujasta $_FILES['file']['type']. Valitettavasti kyseinen metodi ei ole riittävä, koska sen voi ohittaa muuttamalla tiedoston päätteen. Tieto MIME-tyypistä saadaan selaimista, mutta tietoon ei ole luottamista, koska suurin osa selaimista määrittelee MIME-tyypin tiedoston päätteen perusteella. Siksi on tarvittu toimivampi menetelmä määrittää tiedoston tyyppi. Tällä hetkellä suositeltava tapa on käyttää PECL:n laajennusta nimeltä Fileinfo. Kyseinen laajennus selvittää tiedoston tyypin tutkimalla tiedoston muutamia ensimmäisiä tavuja, jotka ovat niin sanottuja ”magic bytejä”. Nämä magic bytet ovat ikään kuin allekirjoituksia, jotka yhdistetään tiedoston tyyppeihin.

Sovelluksessa MIME-tyyppi tarkistetaan fileinfolla, mikäli palvelimen PHP:n versio on päivitetty (fileinfo tulee automaattisesti PHP 5.3 – version mukana). (Nallan 2009.)

Tietoturvatarkastuksen neljäs vaihe selvittää lähetetyn tiedoston koon ja vertaa sitä sallittuun kokoon. Tiedoston koko saadaan selville PHP:n globaalista supermuuttujasta $_FILES['userfile']['size']. Tiedostoa ei tallenneta palvelimelle, jos sen koko on sallittua arvoa suurempi.

Viides vaihe hyödyntää PHP:n funktiota getimagesize(), mikäli lähetetty tiedosto on päätteeltään kuvatiedosto. Tätä funktiota käytetään pyrkimyksenä selvittää onko kuva oikeasti kuva vai löytyykö lähdekoodista esimerkiksi PHP-koodia. Toiminnolla saadaan selville kuvan leveys ja korkeus. (Calin 2009.) Vaikka menetelmä lisääkin tietoturvaa, on hyvin todennäköistä, että edistyneemmät hakkerit osaavat kiertää getimagesize() funktion (Ha.ckers 2007).

Oletusarvoisesti AjaxTransfer muuttaa tietoturvasyistä tiedostojen nimet satunnaisiksi. Eli käyttäjän lähettämä kuva.jpg voisi muuttua vaikkapa muotoon 0521933001382156141.jpg. Menettelytavalla paikataan kaksoispäätteeseen liittyvä haavoittuvuus. Kaksoispääte häviää, kun tiedoston nimi muutetaan. Esimerkiksi hakkerin lähettämä tiedosto haittaohjelma.php.jpg on uudelleennimeämisen jälkeen 0704048001382978612.jpg. (Daniweb 2012.) Joissakin tilanteissa tiedostojen vanhat nimet halutaan säilyttää, jonka vuoksi sivuston ylläpitäjät voivat halutessaan asettaa AjaxTransferin satunnaisnimeämisen pois päältä.

Viimeiseksi vahvistetaan, ettei samannimistä tiedostoa löydy palvelimelta (varsin epätodennäköistä jos tiedostojen satunnaisnimeäminen on päällä). Tiedosto siirretään väliaikaisesta kansiosta lopulliseen kohteeseen, mikäli se on oikean tyyppinen ja kokoinen eikä tiedostonsiirrossa ilmennyt virheitä. Lopulta PHP palauttaa vastauksen tiedostonsiirron onnistumisesta selaimelle.

4.3 Toimivuuden testaus

Ohjelmointivaiheen suorittamisen jälkeen jatkokehitysprojektin tuotosta on testattu eri menetelmin, jotta voitaisiin osoittaa, että lisäosa täyttää tekniset vaatimukset. Testaamisella pyritään löytämään ja korjaamaan sovelluksessa ilmenevät viat. Käytännössä kaikkia mahdollisia vikoja on hankala korjata, sillä ohjelmistot toimivat eri tavalla eri tilanteissa ja ympäristöissä.

Ohjelmistojen testaamista varten on kehitetty useita erilaisia menetelmiä (kuten esimerkiksi black box -testaukset), joilla pyritään löytämään sovelluksista virheitä. Ajansäästösyistä AjaxTransferin testaus painottuu vain tärkeimpien osa-alueiden tarkasteluun, joita ovat:

- Ohjelmointivirheet
- Tietoturva-aukot
- Yhteensopivuusongelmat (eri selaimet sekä palvelinympäristöt)
- Suorituskyky (hyvin lyhyesti)

Testiprosessissa oli tärkeää löytää työstä ohjelmointivirheitä sekä selainten välisiä yhteensopivuusongelmia. Projektin suunnitteluvaiheessa oli tiedossa, että lisäosa tulee toimimaan vain uusimmissa Internet-selaimissa (kuvio 17). Tämä johtuu siitä, koska lisäosan selainpuolen toiminnoissa on käytetty suhteellisen tuoretta FormData – rajapintaa, jota vanhemmat selaimet eivät tue. Nimensä mukaisesti FormData objektia käytetään pääasiassa silloin, kun halutaan siirtää lomakkeiden sisältämää dataa. (Mozilla Developer Network 2013.)

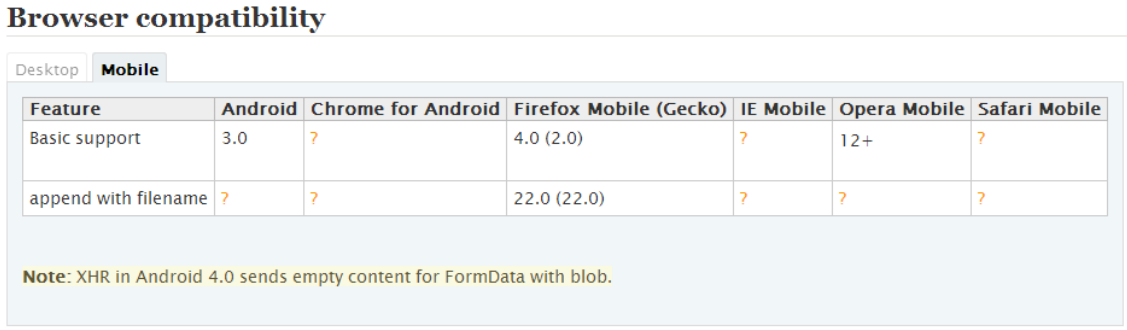
Browser compatibility

Desktop | Mobile

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari
Basic support	7+	4.0 (2.0)	10+	12+	5+
append with filename	(Yes)	22.0 (22.0)	?	?	?

Kuvio 17, FormDatan yhteensopivuus käytetyimpien selainten välillä

Testauksia ei ole ajettu mobiiliselaimilla, koska ne eivät vielä tue FormDataa kovinkaan hyvin (kuvio 18). FormData täytyisi vaihtaa johonkin toiseen menetelmään, jotta lisäosa toimisi muillakin selaimilla.



Browser compatibility

Desktop | Mobile

Feature	Android	Chrome for Android	Firefox Mobile (Gecko)	IE Mobile	Opera Mobile	Safari Mobile
Basic support	3.0	?	4.0 (2.0)	?	12+	?
append with filename	?	?	22.0 (22.0)	?	?	?

Note: XHR in Android 4.0 sends empty content for FormData with blob.

Kuvio 18, FormDatan yhteensopivuus mobiiliselainten välillä

Testausympäristö on rajoitettu niihin selaimiin, jotka tukevat FormDataa. Lisäosan toimivuutta on kokeiltu useaan otteeseen Google Chromella (v.30), Mozilla Firefoxilla (v.25), Apple Safarilla (v.5), Internet Explorerilla (v.10) sekä Operalla (v.18). Sovellusta täytyy testata vielä selainten vanhemmilla versioilla (Chrome 7, Firefox 4.0 ja Opera 12). Testatuilla selaimella on tarkistettu, että sovellus toimii oletetusti. Tässä on käytetty apuna selainten sisäisiä kehitystyökaluja. Selaimilla on kokeiltu muun muuassa tiedostojen siirtämistä, drag & drop – ominaisuutta sekä yleistä toimivuutta. Testitulosten mukaan lisäosa on yhteensopiva ainakin kaikkien tunnetuimpien selainten uusimpien versioiden kanssa. Safarilla näkymä on hieman erilainen verrattuna muihin selaimiin, mutta se ei vaikuta tiedostonsiirron toimivuuteen.

Lisäosan asentamista MODX:lle on kokeiltu kahdessa eri palvelinympäristössä. Onnistunut asennus edellyttää, että MODX luo asennuksen aikana lisäosalle tietokantataulun sekä resurssin Ajax-yhdistäjää varten. Lisäosa tulee pystyä asentamaan MODX:n pakettienhallinnan kautta. Molemmissa palvelinympäristössä Extran asennus onnistui vaatimusten mukaisesti.

Sovelluksen koodi on tarkastettu monta kertaa läpi ohjelmointivirheiden varalta. Virheitä ei ole löytynyt, mutta se ei tarkoita sitä, että koodi olisi täysin virheetöntä. Tarvittaisiin joku kokenut ohjelmoija arvioimaan koodia, jotta saataisiin objektiivisempi käsitys aiheesta. Koodista on pyritty tekemään yksinkertaista, toimivaa sekä riippumatonta erillisistä ohjelmistokirjastoista tai lisäosista.

Tiedostosiirron suorituskykyä on testattu lähettämällä isoja tiedostoja sekä useita pienempiä tiedostoja jatkuvaan syöttötahtiin. Toistaiseksi tiedostonsiirrossa ei ole ilmennyt ongelmia.

Haastavinta oli saada tiedostonsiirrosta mahdollisimman tietoturvallista. Lisäosan palvelinpuolen PHP-palaan kootut tämänhetkiset parhaaksi todetut tietoturvakäytännöt eivät riitä sellaisenaan. Mukaan tarvitaan joukko muitakin menettelytapoja, jotta tiedostonsiirrosta tulisi tarpeeksi turvallista. Palvelimen ylläpitäjän on syytä esimerkiksi erottaa web-sivujen juurikansio erilleen kohdehakemistosta, jossa lähetettyjä tiedostoja säilytetään. Kohdehakemisto tarvitsee vain luku/kirjoitusoikeudet, eli suoritusoikeudet on otettava pois päältä. Joitain HTTP-palvelinohjelmiston sekä PHP:n oletusasetuksia on syytä muuttaa. Esimerkiksi PHP:n asetuksista pystyy rajaamaan maksimimäärän sille, kuinka monta tiedostoa voi lähettää kerralla. Internetistä löytyy runsaasti palvelimen tietoturvaan liittyviä ohjeita, joita kannattaa soveltaa sivuston tarpeiden mukaisesti.

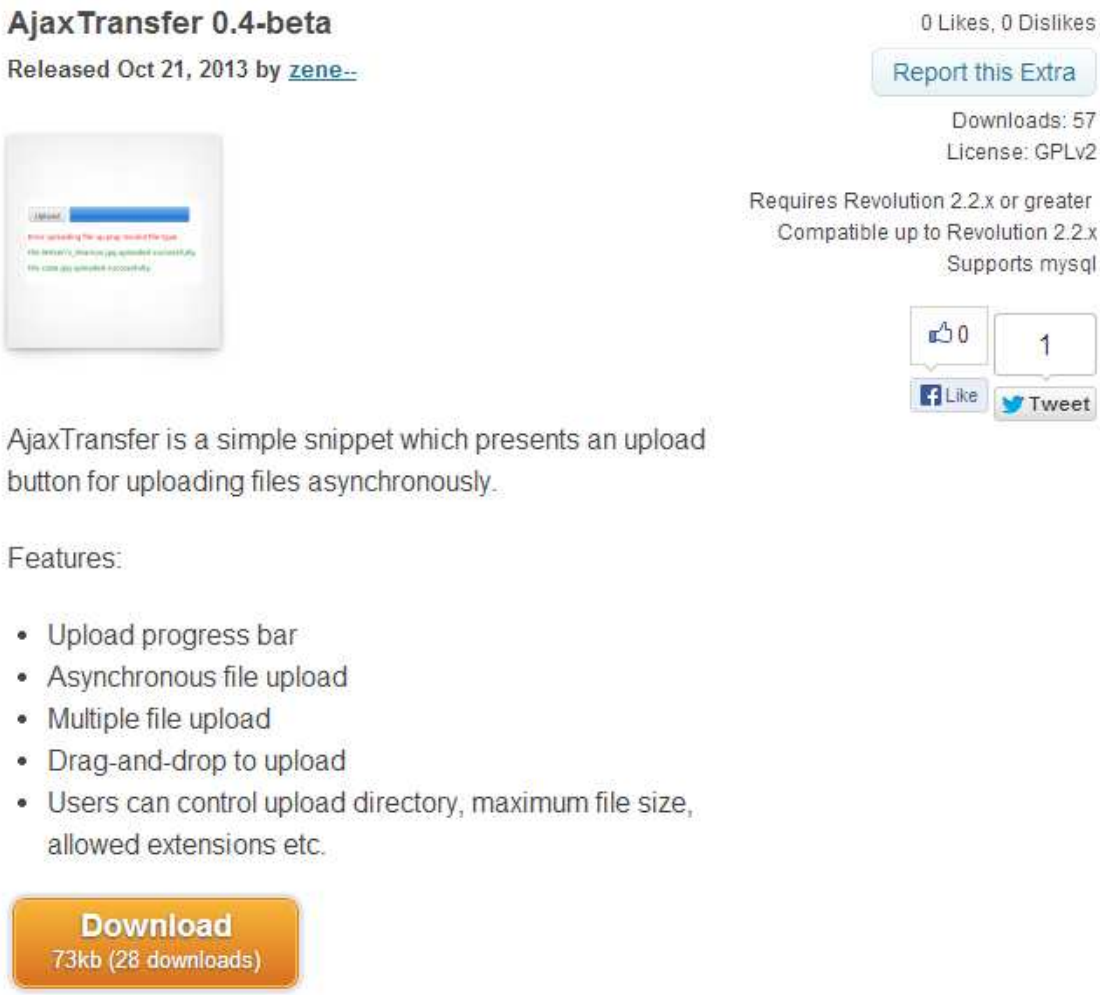
Opinnäytetyön kirjoitushetkellä lisäosa käyttää vielä Ajax-yhdistäjänä MODX:n PHP-muotoista resurssia. Tälläistä front end tyyppisen yhdistäjän käyttöä ei voi valitettavasti suojata MODX:n käyttöoikeuksien perusteella. Loogisempi ratkaisu olisikin käyttää MODX:n back end -

tyyppistä Ajax-yhdistäjää. Alun perin lisäosan yhdistäjästä suunniteltiin back end – tyyppistä, mutta sen toteutuksessa ilmeni ongelmia, jonka vuoksi suunnitelmat vaihtuivat. Front end - tyyppisen yhdistäjään liittyvä vika huomattiin liian myöhään, eikä sitä enää ehditty korjata opinnäytetyön kirjoittamisvaiheessa. Siihen liittyy tietynlainen tietoturva-aukko, mikäli sivustolla on käytössä useita eri tiedostonsiirtolomakkeita. Jokainen tiedostonsiirtolomake käyttää eri ID-numeroa. ID-numero lähetetään selaimesta palvelimelle POST-pyynnön mukana, mikä on helppoa väärentää kenenkä tahansa toimesta (Stackoverflow 2011). Oletetaan, että verkkosivulla on kaksi tiedostonsiirtolomaketta, joista ensimmäinen vastaanottaa vain kuvatiedostoja ja toinen dokumentteja. Hakkeri pystyisi lähettämään kuvatiedostoja dokumenttien sijaan sekä dokumentteja kuvien sijaan vaihtamalla POST-pyynnön ID-numeroa (hakkerin täytyisi tietää tai arvata toisen lomakkeen ID-numero). Tämä ei ole kuitenkaan merkittävä tietoturvauhka, jos ID-numerot määrittäisi vaikeasti arvattaviksi, esimerkiksi pitkiksi lukusarjoiksi. Ratkaisua voidaan silti pitää väliaikaisena, koska tarjolla on back end - tyyppinen parempi käytäntö. Tarkoituksena onkin korjata tilanne pian sillä tavalla, että seuraava versio lisäosasta käyttäisi kustomoitua MODX:n back end -tyyppistä yhdistäjää. Ennen uutta versiota käyttäjä voi halutessaan parantaa lisäosan tietoturvaa ainakin kahdella tavalla. MODX:llä pystyy suojaamaan resursseja käyttöoikeuksien mukaan. Toisin sanoen, lisäosan tietoturvan luottamuksellisuutta voi parantaa siten, että sallii pääsyoikeuden AjaxTransferia käyttävälle sivulle esimerkiksi vain tietyille MODX:n käyttäjille. Toinen tapa on vaihtaa Ajax-yhdistäjää, mutta se vaatii jo käyttäjää näkemään hieman vaivaa.

Lokakuusta 2013 lähtien AjaxTransfer on ollut vapaasti ladattavissa MODX:n kotisivulta. Työ läpäisi MODX:n moderaattorin arviointiprosessin. Lisäosa on vielä betatestausvaiheessa. Tarkoituksena on viimeistellä työtä lähitulevaisuudessa sekä kerätä palautetta MODX:n käyttäjiltä.

4.4 Tuotos

Työ valmistui syksyllä 2013, mutta sen korjailu ja viimeistely jatkuu sitä mukaa kuin virheitä löytyy ja uusia ideoita tulee mieleen. Lisäosa löytyy MODX:n verkkosivulta http://modx.com/extras/package/ajaxuploader (kuvio 19).



Kuvio 19, modx.com/extras/package/ajaxuploader

Alustavasti jatkokehitysprojektin tavoitteet on saavutettu onnistuneesti, vaikka työ onkin vielä kehitysvaiheessa. Lisäosasta tuli MODX:n käyttäjille helposti asennettava sekä helppokäyttöinen, eikä se vaadi erillisiä ohjelmistokirjastoja tai muita lisäosia toimiakseen.

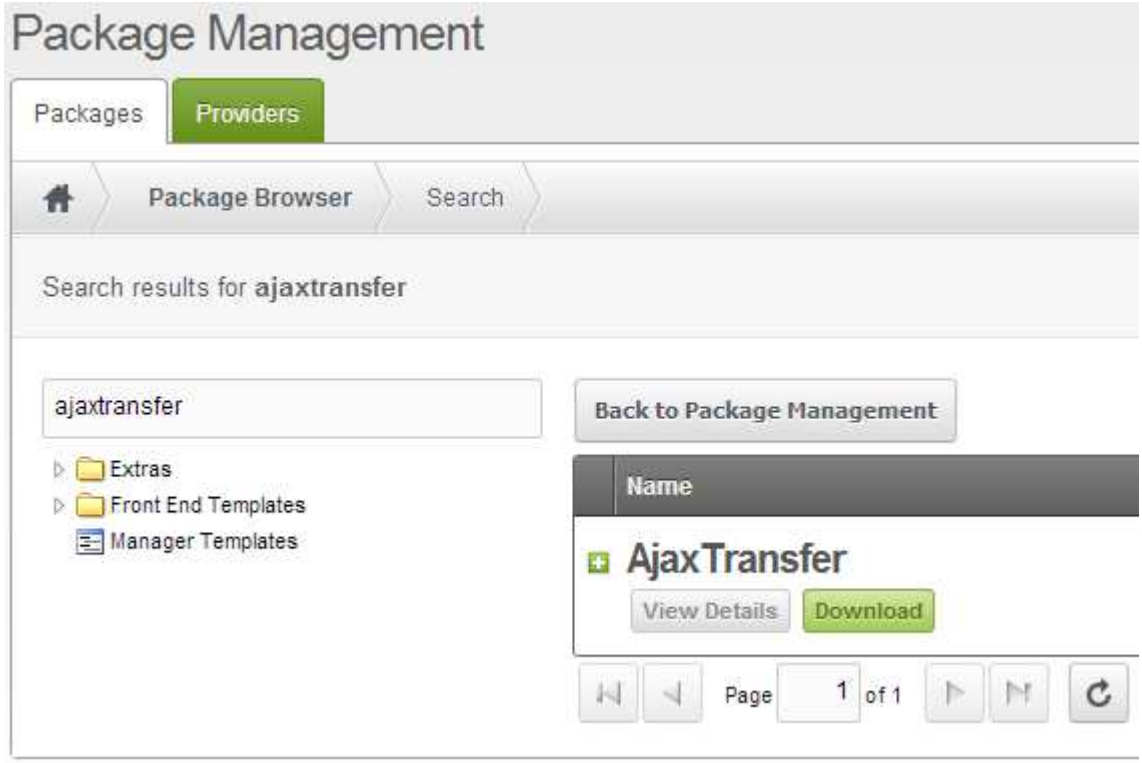
4.4.1 Asennusohjeet

AjaxTransfer vaatii toimiakseen MODX – sisällönhallintajärjestelmän version 2.2.x tai uudemman (saattaa toimia vanhemmillakin versioilla, mutta tätä ei ole testattu). Lisäosa tukee MySQL – tietokantaohjelmistoa.

Extran asennus tapahtuu MODX:n Package Management – osion kautta, joka löytyy System – välilehden alta (kuvio 20).

Kuvio 20, Asennusohjeet

Package Managementissa valitaan kohta ”Download Extras”, jolloin aukeaa näkymä uusimmista ja suosituimmista Extroista (kuvio 21). Toiminnallinen opinnäytetyöni pitäisi löytyä hakusanalla ”ajaxtransfer”.



Kuvio 21, Asennusohjeet 2

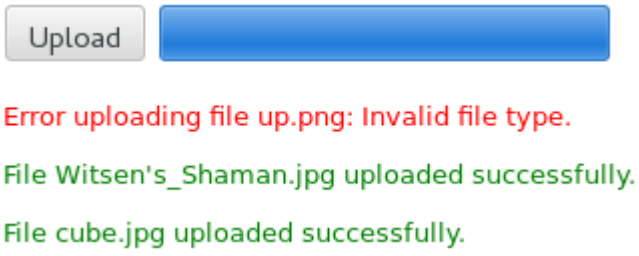
Extra ladataan kohdasta ”Download”, jonka jälkeen se asennetaan Package Managementin kautta parilla hiiren klikkauksella. AjaxTransferin asennustiedoston, eli ”transport packagen” voi myös ladata manuaalisesti MODX:n kotisivuilta. Asennustiedosto on siirrettävä MODX:n hakemistoon /core/packages/ ja se asennetaan Package Managementin kautta.

4.4.2 Käyttöohjeet

Asennetun Extran käyttöönotto tapahtuu yksinkertaisesti upottamalla AjaxTransferiin viittaava PHP-pala parametrineen MODX:n sivun HTML-lähdekoodiin. PHP-palan on oltava <body> - tagien sisällä.

```
<html>
<head>
<title>AjaxTransfer</title>
</head>
<body>
[[!AjaxTransfer? &path=`assets/images/`]]
</body>
</html>
```

Yllä oleva yksinkertaistettu esimerkki HTML-koodista ja PHP-palan käytöstä tulostaa web-sivulle painikkeen, jota painamalla käyttäjä valitsee tietokoneeltaan tiedoston tai tiedostoja lähetettäväksi palvelimen hakemistoon assets/images/. Tiedostoja voi lähettää myös raahaamalla niitä toisesta ikkunasta/kansiosta tiputettavaksi Upload – painikkeen päälle. Tiedostojen siirron aikana ruudulle ilmestyy etenemispalkki. Ilmoitukset siirron onnistumisesta ilmestyvät asynkronisesti lähetyspainikkeen ja etenemispalkin alapuolelle (kuvio 22). Web-suunnittelijoilla on yleensä tapana muuttaa elementtien värejä, asettelua ja muotoilua heidän tekemiinsä sivujen ulkoasuun sointuvaksi. AjaxTransferin ulkoasua voi muuttaa tiedostosta ajaxtransfer.chunk.tpl.



Kuvio 22, AjaxTransfer

Tietoturvaa parantavia asetuksia on suositeltava käyttää. Seuraava esimerkki PHP-palasta sallisi lähetettäväksi vain alle 9.5 MB kokoiset JPG, GIF – ja PNG – kuvatiedostot.

```
[[!AjaxTransfer?
&path=`assets/images/`
&allowed_extensions=`jpg,jpeg,png,gif`
&allowed_mimes=`image/jpeg,image/png,image/gif`
&max_size=`10000000`
```

]]

AjaxTransferin käyttö edellyttää, että ylläpitäjän määrittämään palvelimen kohdehakemistoon sallitaan tiedostonsiirtoa varten tarvittavat kirjoitusoikeudet.

5 Yhteenveto ja johtopäätökset

Opinnäytetyön projektin lähtökohtana oli toteuttaa työnantajayrityksen asiakkaan kotisivulle Ajax – tekniikalla varustettu tiedostonsiirtolomake. Tämän jälkeen hankkeesta jatkokehitettiin MODX:lle asennettava lisäosa.

Asiakasprojekti toteutettiin onnistuneesti aikataulun mukaisesti keväällä 2013. Työn tavoitteet tulivat toteen ja asiakas sai kotisivulleen tiedostonsiirtolomakkeen, jolla voi lähettää kuvia moodboardiin. Tosin kiireellisen aikataulun takia asiakasprojektista ei syntynyt työnantajalle kovinkaan helposti asennettavaa tiedostonsiirtoskriptiä. Työn täytyi olla niin sanotusti mobiilimpi, jotta sitä voisi käyttää vaivattomasti tulevaisuuden hankkeissa. Tiedostonsiirtoskriptiä ei ollut kuitenkaan edes välttämätöntä jatkokehittää, sillä asiakasprojekti oli jo takana ja työnantajalla oli uudet hankkeet mielessä. Itsestään asiakasprojekti oli sen verran työläs, että jo siitä olisi voinut tehdä toiminnallisen opinnäytetyön. Näistä asioista huolimatta halusin kuitenkin tehdä työn loppuun asti ihan vain siksi, koska aihe oli mielenkiintoinen ja se kehitti ammatillista kasvuani.

Työn jatkokehittäminen oli hyvin opettavaista. Projekti perehdytti tehokkaasti ohjelmoimaan PHP:llä sekä JavaScriptillä. Haastavinta oli toteuttaa tiedostonsiirtoskriptistä itsenäinen MODX Extra, koska lisäosan tietokantayhteyden kanssa ilmeni pitkään ongelmia. Tämän lisäksi tietoturvaan liittyvät kysymykset aiheuttivat hieman harmaita hiuksia. Jälkeenpäin ajateltuna joitakin asioita olisi voinut tehdä toisella tavalla. Onneksi kaikki tähän asti löydetyt erehdykset ovat kuitenkin helposti oikaistavissa. Koska aikaansaannos on avoimen lähdekoodin projekti, sen täytyy olla kirjoitettu mahdollisimman laadukkaasti ja loogisesti. Kirjoitushetkellä koodin kommentoimisessa on vielä parantamisen varaa. Jotkut koodin metodit saattavat vaikuttaa kryptiseltä, koska olen jättänyt ne kiireen takia kommentoimatta.

Olen tyytyväinen jatkokehittämisen lopputulokseen. Asetetut tavoitteet on saavutettu ja lisäosaa voi tarvittaessa käyttää myöhemmissä asiakasprojekteissa. AjaxTransfer läpäisi MODX:n moderaattorin arviointiprosessin ja työ julkaistiin MODX:n kotisivulle ladattavaksi. Marraskuussa 2013 lisäosaa on ladattu 60 kertaa eikä työstä ole toistaiseksi annettu palautetta.

Lisäosan päätoiminnallisuudet on toteutettu, mutta tuote on vielä beta-vaiheessa. Opinnäytetyössä aiemmin mainitut puutteet on syytä korjata. Tilaston mukaan joka päivä joku lataa AjaxTransferin ja mahdollisesti lisää sen nettisivulleen. Tämä lisää vastuuta parantaa työtä jatkuvasti. Lisäosaa on tarkoitus kehittää niin kauan, kun sille riittää kysyntää.

Lähteet

Calin Bogadan. 2009. Why File Upload Forms are a Major Security Threat. Viitattu 14.11.2013. http://www.acunetix.com/websitesecurity/upload-forms-threat/

Daniweb. 2012. Prevent double extension upload in php. Viitattu 19.11.2013. http://www.daniweb.com/web-development/php/threads/438614/prevent-double-extension-upload-in-php

Djce. n.d. What's in an HTTP request?. Viitattu 14.11.2013. http://djce.org.uk/dumprequest

Fine Uploader. 2013. Introducing Fine Uploader. Viitattu 14.11.2013. http://fineuploader.com/

Flanagan, David; Ferguson, Paula (2006). JavaScript: The Definitive Guide. 5. painos. O'Reilly & Associates. ISBN 0-596-10199-6.

Garrett Jesse James. 2005. Ajax: A New Approach to Web Applications. Viitattu 14.11.2013. https://web.archive.org/web/20080702075113/http://www.adaptivepath.com/ideas/essays/archives/000385.php

Ha.ckers. 2007. Passing Malicious PHP Through getimagesize(). Viitattu 19.11.2013. http://ha.ckers.org/blog/20070604/passing-malicious-php-through-getimagesize/

JavaScripter.net. n.d. What Is JavaScript?. Viitattu 13.11.2013. http://www.javascripter.net/faq/whatisja.htm

Kalla Riyad. 2010. HTML5 Drag and Drop Upload and File API Tutorial. Viitattu 14.11.2013. http://www.thebuzzmedia.com/html5-drag-and-drop-and-file-api-tutorial/

Kohan Bernard. 2010. What is a Content Management System (CMS)?. Viitattu 13.11.2013. http://www.comentum.com/what-is-cms-content-management-system.html

McFarland David Sawyer. 2011. JavaScript & jQuery: The Missing Manual. 2. painos. O'Reilly Media, Inc.

MODX. 2013. phpThumbOf 1.4.0-pl. Viitattu 27.11.2013. http://modx.com/extras/package/phpthumbof

MODX Docs. 2013. Chunks. Viitattu 14.11.2013. http://rtfm.modx.com/revolution/2.x/making-sites-with-modx/structuring-your-site/chunks

MODX Docs. 2013. Developing an Extra in MODX Revolution. Viitattu 14.11.2013. http://rtfm.modx.com/revolution/2.x/case-studies-and-tutorials/developing-an-extra-in-modx-revolution

MODX Docs. n.d. Structuring Your Site. Viitattu 13.11.2013. http://rtfm.modx.com/revolution/2.x/making-sites-with-modx/structuring-your-site

MODX Docs 2013. Template Variables. Viitattu 13.11.2013. http://rtfm.modx.com/revolution/2.x/making-sites-with-modx/customizing-content/template-variables

MODX Docs. 2013. Templates. Viitattu 13.11.2013. http://rtfm.modx.com/revolution/2.x/making-sites-with-modx/structuring-your-site/templates

Moisio Aleksi. 2008. MySQL – suomalainen menestystarina. Viitattu 14.11.2013. http://www.itviikko.fi/ratkaisut/2008/01/16/mysql--suomalainen-menestystarina/20081483/7

Mozilla Developer Network. 2013. FormData. Viitattu 15.11.2013. https://developer.mozilla.org/en-US/docs/Web/API/FormData

MySQL Developer Zone. 2013. What is MySQL?. Viitattu 13.11.2013. http://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html

Nalla Kishore. 2009. Smart File Type Detection Using PHP. Viitattu 19.11.2013. http://designshack.net/articles/php-articles/smart-file-type-detection-using-php/

Noponen Sami. 2009. Oracle teki jättikaupan – nyt ostettiin Sun. Viitattu 14.11.2013. http://www.taloussanomat.fi/talous/2009/04/20/oracle-teki-jattikaupan-nyt-ostettiin-sun/200910061/133

Paakki Jukka. 2011. Ohjelmistojen vaatimusmäärittely. Viitattu 17.11.2013. http://www.cs.helsinki.fi/u/paakki/Vaatimus-11-Luentokalvot-1.pdf

PHP n.d. Description of core php.ini directives. Viitattu 14.11.2013. http://www.php.net/manual/en/ini.core.php#ini.upload-max-filesize

PHP n.d. Error Messages Explained. Viitattu 14.11.2013. http://php.net/manual/en/features.file-upload.errors.php

PHP. n.d. Extension List/Categorization. Viitattu 13.11.2013. http://www.php.net/manual/en/extensions.alphabetical.php

PHP. n.d. History of PHP. Viitattu 13.11.2013. http://us3.php.net/manual/en/history.php.php

PHP. 2013. Usage Stats for January 2013. Viitattu 13.11.2013. http://www.php.net/usage.php

PHP. n.d. What can PHP do?. Viitattu 13.11.2013. http://www.php.net/manual/en/intro-whatcando.php

PHP. n.d. What is PHP?. Viitattu 13.11.2013. http://www.php.net/manual/en/intro-whatis.php

Ray Bob. 2012. The MODX Content Management System. Viitattu 13.11.2013. http://bobsguides.com/modx.html

Stackoverflow. 2011. How to manually fire HTTP Post Requests with Firefox or Chrome. Viitattu 14.11.2013. http://stackoverflow.com/questions/4797534/how-to-manually-fire-http-post-requests-with-firefox-or-chrome

Thrash Ryan. 2013. The History, Present and Future of MODX. Viitattu 13.11.2013. http://modx.com/company/media-center/background/

Tidy designs. 2012. PHP Secure File Upload Script. Viitattu 19.11.2013. http://www.tidy-designs.co.uk/website-development/php-secure-file-upload-script/

Kuviot

Kuvio 1, Asiakasprojekti 7
Kuvio 2, Sovelluksen käyttöönottaminen. Esimerkissä lisäosaan viittaava PHP-pala upotetaan HTML-sivun lähdekoodin sekaan. 7
Kuvio 3, PHP-pala tulostuu www-sivulle tiedostonsiirtolomakkeeksi 7
Kuvio 4, JavaScriptin syntaksi 10
Kuvio 5, PHP:n syntaksi 11
Kuvio 6, Perinteinen HTTP-pyyntö 12
Kuvio 7, Ajax -pyyntö 13
Kuvio 8, Asiakasprojekti 15
Kuvio 9, Asiakasprojektin lopputulos 17
Kuvio 10, AjaxTransfer PHP-pala 22
Kuvio 11, Esimerkki lisäosan tietokantatauluista 23
Kuvio 12, Selainpuolen toiminnallisuus 24
Kuvio 13, Normaali näkymä lomakkeesta 25
Kuvio 14, Näkymä lomakkeesta Safari – selaimella katsottuna 25
Kuvio 15, Etenemispalkin värit eri prosenttimäärille. X = Tiedostonsiirron etenemisprosentti 26
Kuvio 16, Palvelinpuolen toiminnallisuus 27
Kuvio 17, FormDatan yhteensopivuus käytetyimpien selainten välillä 30
Kuvio 18, FormDatan yhteensopivuus mobiiliselainten välillä 30
Kuvio 19, modx.com/extras/package/ajaxuploader 33
Kuvio 20, Asennusohjeet 34
Kuvio 21, Asennusohjeet 2 34
Kuvio 22, AjaxTransfer 35

Liitteet

Liite 1: build.config.php 42
Liite 2: build.schema.php 43
Liite 3: build.transport.php 45
Liite 4: transport.snippets.php 49
Liite 5: resolve.tables.php 50
Liite 6: ajaxtransfer.chunk.tpl 51
Liite 7: snippet.ajaxtransfer.php 58
Liite 8: snippet.ajaxtransfer_server.php 60
Liite 9: ajaxtr.class.php 64
Liite 10: ajaxtr.map.inc.php 65
Liite 11: ajaxtr.class.php (2) 67
Liite 12: ajaxtransfer.class.php 68
Liite 13: ajaxtransfer.mysql.schema.xml 71

Liite 1: build.config.php

```
<?php
/**
 *
 * @package ajaxtransfer
 * @subpackage build
 */
//define('MODX_BASE_PATH', dirname(dirname(dirname(dirname(__FILE__)))) . '/modx/');
define('MODX_BASE_PATH', dirname(dirname(dirname(dirname(__FILE__)))) . '/www/modx/');
define('MODX_CORE_PATH', MODX_BASE_PATH . 'core/');
define('MODX_MANAGER_PATH', MODX_BASE_PATH . 'manager/');
define('MODX_CONNECTORS_PATH', MODX_BASE_PATH . 'connectors/');
define('MODX_ASSETS_PATH', MODX_BASE_PATH . 'assets/');

define('MODX_BASE_URL','/modx/');
define('MODX_CORE_URL', MODX_BASE_URL . 'core/');
define('MODX_MANAGER_URL', MODX_BASE_URL . 'manager/');
define('MODX_CONNECTORS_URL', MODX_BASE_URL . 'connectors/');
define('MODX_ASSETS_URL', MODX_BASE_URL . 'assets/');
```

Liite 2: build.schema.php

```
<?php
/**
 * Build Schema script
 *
 * @package ajaxtransfer
 * @subpackage build
 */
$mtime = microtime();
$mtime = explode(" ", $mtime);
$mtime = $mtime[1] + $mtime[0];
$tstart = $mtime;
set_time_limit(0);

require_once dirname(__FILE__).'/build.config.php';
include_once MODX_CORE_PATH . 'model/modx/modx.class.php';
$modx= new modX();
$modx->initialize('mgr');
$modx->loadClass('transport.modPackageBuilder','',false, true);
echo '<pre>'; /* used for nice formatting of log messages */
$modx->setLogLevel(modX::LOG_LEVEL_INFO);
$modx->setLogTarget('ECHO');

$root = dirname(dirname(__FILE__)).'/';
$sources = array(
    'root' => $root,
    'core' => $root.'core/components/ajaxtransfer/',
    'model' => $root.'core/components/ajaxtransfer/model/',
    'schema' => $root.'core/components/ajaxtransfer/model/schema/',
    'schema_file' =>
$root.'core/components/ajaxtransfer/model/schema/ajaxtransfer.mysql.schema.xml',
    'assets' => $root.'assets/components/ajaxtransfer/',
);
$manager= $modx->getManager();
$generator= $manager->getGenerator();

if (!is_dir($sources['model'])) {
    $modx->log(modX::LOG_LEVEL_ERROR,'Model directory not found!');
```

```
    die();
}
if (!file_exists($sources['schema_file'])) {
    $modx->log(modX::LOG_LEVEL_ERROR,'Schema file not found!');
    die();
}
$generator->parseSchema($sources['schema_file'],$sources['model']);

$mtime= microtime();
$mtime= explode(" ", $mtime);
$mtime= $mtime[1] + $mtime[0];
$tend= $mtime;
$totalTime= ($tend - $tstart);
$totalTime= sprintf("%2.4f s", $totalTime);

echo "\nExecution time: {$totalTime}\n";

exit ();
```

Liite 3: build.transport.php

```
<?php
/**
 * AjaxTransfer build script
 *
 * @package ajaxtransfer
 * @subpackage build
 */
$tstart = explode(' ', microtime());
$tstart = $tstart[1] + $tstart[0];
set_time_limit(0);

/* define package names */
define('PKG_NAME','AjaxTransfer');
define('PKG_NAME_LOWER','ajaxtransfer');
define('PKG_VERSION','0.3');
define('PKG_RELEASE','beta');

/* define build paths */
$root = dirname(dirname(__FILE__)).'/';
$sources = array(
    'root' => $root,
    'build' => $root . '_build/',
    'data' => $root . '_build/data/',
    'resolvers' => $root . '_build/resolvers/',
    'chunks' => $root.'core/components/'.PKG_NAME_LOWER.'/chunks/',
    'lexicon' => $root . 'core/components/'.PKG_NAME_LOWER.'/lexicon/',
    'docs' => $root.'core/components/'.PKG_NAME_LOWER.'/docs/',
    'elements' => $root.'core/components/'.PKG_NAME_LOWER.'/elements/',
    'source_assets' => $root.'assets/components/'.PKG_NAME_LOWER,
    'source_core' => $root.'core/components/'.PKG_NAME_LOWER,
);
unset($root);

/* override with your own defines here (see build.config.sample.php) */
require_once $sources['build'] . 'build.config.php';
require_once MODX_CORE_PATH . 'model/modx/modx.class.php';
```

```
$modx= new modX();
$modx->initialize('mgr');
echo '<pre>'; /* used for nice formatting of log messages */
$modx->setLogLevel(modX::LOG_LEVEL_INFO);
$modx->setLogTarget('ECHO');

$modx->loadClass('transport.modPackageBuilder','',false, true);
$builder = new modPackageBuilder($modx);
$builder->createPackage(PKG_NAME_LOWER,PKG_VERSION,PKG_RELEASE);
$builder-
>registerNamespace(PKG_NAME_LOWER,false,true,'{core_path}components/'.PKG_NAME_LOW
ER.'/');

/* create category */
$category= $modx->newObject('modCategory');
$category->set('id',1);
$category->set('category',PKG_NAME);

/* add snippets */
$modx->log(modX::LOG_LEVEL_INFO,'Packaging in snippets...');
$snippets = include $sources['data'].'transport.snippets.php';
if (empty($snippets)) $modx->log(modX::LOG_LEVEL_ERROR,'Could not package in snippets.');
$category->addMany($snippets);

/* create category vehicle */
$attr = array(
    xPDOTransport::UNIQUE_KEY => 'category',
    xPDOTransport::PRESERVE_KEYS => false,
    xPDOTransport::UPDATE_OBJECT => true,
    xPDOTransport::RELATED_OBJECTS => true,
    xPDOTransport::RELATED_OBJECT_ATTRIBUTES => array (
        'Snippets' => array(
            xPDOTransport::PRESERVE_KEYS => false,
            xPDOTransport::UPDATE_OBJECT => true,
            xPDOTransport::UNIQUE_KEY => 'name',
        ),
    ),
);
```

```
$vehicle = $builder->createVehicle($category,$attr);

$modx->log(modX::LOG_LEVEL_INFO,'Adding file resolvers to category...');
$vehicle->resolve('file',array(
    'source' => $sources['source_assets'],
    'target' => "return MODX_ASSETS_PATH . 'components/';",
));
$vehicle->resolve('file',array(
    'source' => $sources['source_core'],
    'target' => "return MODX_CORE_PATH . 'components/';",
));
$builder->putVehicle($vehicle);

$modx->log(modX::LOG_LEVEL_INFO,'Adding in PHP resolvers...');
$vehicle->resolve('php',array(
    'source' => $sources['resolvers'] . 'resolve.tables.php',
));
$builder->putVehicle($vehicle);
//unset($vehicle,$menu);

/* now pack in the license file, readme and setup options */
$modx->log(modX::LOG_LEVEL_INFO,'Adding package attributes and setup options...');
$builder->setPackageAttributes(array(
    'license' => file_get_contents($sources['docs'] . 'license.txt'),
    'readme' => file_get_contents($sources['docs'] . 'readme.txt'),
    'changelog' => file_get_contents($sources['docs'] . 'changelog.txt'),
//    'setup-options' => array(
//        'source' => $sources['build'].'setup.options.php',
//    ),
));


$resourceAlias = "AjaxTransferConnector";
$ajaxConnector = $modx->getObject('modResource', array('alias'=>$resourceAlias));

if($ajaxConnector == null){

                // Create a new ajax connector resource
                $ajaxConnector = $modx->newObject('modResource');
```

```
                $ajaxConnector->set('alias', 'AjaxTransferConnector');
                $ajaxConnector->set('contentType', 'text/php');
                $ajaxConnector->set('pagetitle', 'AjaxTransferConnector');
                $ajaxConnector->set('richtext', 0);
                $ajaxConnector->set('cacheable', 0);
                $ajaxConnector->set('content', '[[!AjaxTransfer_Server]]');
                $ajaxConnector->set('show_in_tree', '0');
                $ajaxConnector->set('published', 1);
                $ajaxConnector->save();

}

/* zip up package */
$modx->log(modX::LOG_LEVEL_INFO,'Packing up transport package zip...');
$builder->pack();

$tend= explode(" ", microtime());
$tend= $tend[1] + $tend[0];
$totalTime= sprintf("%2.4f s",($tend - $tstart));
$modx->log(modX::LOG_LEVEL_INFO,"\n<br />Package Built.<br />\nExecution time:
{$totalTime}\n");
exit ();
```

Liite 4: transport.snippets.php

```
<?php
/**
 * @package ajaxtransfer
 * @subpackage build
 */
function getSnippetContent($filename) {
    $o = file_get_contents($filename);
    $o = trim(str_replace(array('<?php','?>'),'',$o));
    return $o;
}
$snippets = array();

/* course snippets */
$snippets[1]= $modx->newObject('modSnippet');
$snippets[1]->fromArray(array(
    'id' => 1,
    'name' => 'AjaxTransfer',
    'description' => 'Ajax File Upload Form',
    'snippet' => getSnippetContent($sources['elements'].'snippets/snippet.ajaxtransfer.php'),
),'',true,true);
$properties = include $sources['data'].'properties/properties.ajaxtransfer.php';
$snippets[1]->setProperties($properties);
//unset($properties);

$snippets[2]= $modx->newObject('modSnippet');
$snippets[2]->fromArray(array(
    'id' => 2,
    'name' => 'AjaxTransfer_Server',
    'description' => 'Server side code for AjaxTransfer',
    'snippet' =>
getSnippetContent($sources['source_core'].'/elements/snippets/snippet.ajaxtransfer_server.p
hp'),
));

return $snippets;
```

Liite 5: resolve.tables.php

```
<?php
/**
 * Resolve creating custom db tables during install.
 *
 * @package ajaxtransfer
 * @subpackage build
 */
if ($object->xpdo) {
    switch ($options[xPDOTransport::PACKAGE_ACTION]) {
        case xPDOTransport::ACTION_INSTALL:
            $modx =& $object->xpdo;
            $modelPath = $modx->getOption('ajaxtransfer.core_path',null,$modx-
>getOption('core_path').'components/ajaxtransfer/').'model/';
            $modx->addPackage('ajaxtransfer',$modelPath);

            $manager = $modx->getManager();

            $manager->createObjectContainer('AjaxTr');

            break;
        case xPDOTransport::ACTION_UPGRADE:
            break;
    }
}

return true;
```

Liite 6: ajaxtransfer.chunk.tpl

```
<script type="text/javascript">

window.onload = function() {

		progressBar.style.visibility='hidden';
		var isSafari =
Object.prototype.toString.call(window.HTMLElement).indexOf('Constructor') > 0;

		if(isSafari){
			safariBugFix();
		}

		var dropzone = document.getElementById("dropbox");

		dropzone.ondragover = dropzone.ondragenter = function(event) {

			event.stopPropagation();
			event.preventDefault();

		}

		dropzone.ondrop = function(event) {

			event.stopPropagation();
			event.preventDefault();
			var filesArray = event.dataTransfer.files;
			createFD(filesArray);

		}
}

function safariBugFix(){

		document.getElementById("files").setAttribute("style","");
		document.getElementById("files").setAttribute("onchange","");
		document.getElementById("uploadButton").setAttribute("onclick","createFD();")
;
```

```
		var elem = document.getElementById("progressBar");
		elem.style.width="0px";

}

function createFD(files) {

		var maxSize = "[[!+max_size]]";
		var allowedMimes = "[[!+allowed_mimes]]";

		if (files == undefined) {

				var input = document.getElementById('files');
				var formData = checkFiles(input.files);

		}

		else {
				var formData = checkFiles(files);
		}

		// 7 = empty formData

		if(formData == 7){
				return false;
		}

		formData.append('id', '[[!+id]]');

		sendFile(formData);
		return false;

}

function checkFiles(files){

		document.getElementById('error').innerHTML = "";
		var formData = new FormData();
		var allowedMimes = "[[!+allowed_mimes]]";
```

```
                var emptyFD = 7;
                var errors = 0;
                var uploadFile = 0;

                for (var i=0; i < files.length; i++) {

                if(forbiddenSize(files[i])){

                var response = "Error uploading file " + files[i].name + ": File is too big.";
                errors = errors + 1;
                appendError(response);
                continue;

                }

                if (allowedMimes != ""){

                if(forbiddenType(files[i])){
                var response = "Error uploading file " + files[i].name + ": Invalid file type.";
                errors = errors + 1;
                appendError(response);
                continue;
                }

                }

                emptyFD = 8;
                uploadFile = i - errors;
                formData.append("files" + uploadFile, files[i]);

        }

                if(emptyFD == 7){
                                return emptyFD;
                }

                return formData;
}
```

```
function forbiddenSize(file){

		var size = file.size;
		var allowedSize = "[[!+max_size]]";

		if(allowedSize == ""){
			allowedSize = 5000000;
		}

		if(size > allowedSize){
			return true;
		}

		return false;

}

function forbiddenType(file){

		var type = file.type;
		var mimes = "[[!+allowed_mimes]]";
		var result = mimes.split(",");
		var errors = 0;

		for(i = 0; i < result.length; i++){
			if(type.search(result[i]) == -1){
				errors = errors + 1;
			}
		}

	if (errors < result.length){
		return false;
	}

		return true;

}

function appendError(response){
```

```
                var elem = document.getElementById("error");
                elem.innerHTML += response + '<p></p>';
                elem.style.color = "Red";
                elem.style.fontSize="small";

}

function _(el){
                return document.getElementById(el);
}

function sendFile(formData){

                var ajax = new XMLHttpRequest();
                progressBar.style.visibility='visible';
                ajax.upload.addEventListener("progress", progressHandler, false);
                ajax.addEventListener("load", completeHandler, false);
                ajax.addEventListener("error", errorHandler, false);
                ajax.addEventListener("abort", abortHandler, false);
                ajax.open("POST", "[[[!~[[!+connectorID]]]]");
                ajax.send(formData);

}

function progressHandler(event){

                var percent = (event.loaded / event.total) * 100; _("progressBar").value =
                Math.round(percent);
                _("status").innerHTML = Math.round(percent)+"% uploaded... please wait";
                _("status").style.color = "Black";
 }

function completeHandler(event){

                document.getElementById('status').innerHTML = "";
                var response = event.target.responseText.split("\n");

                for(i = 0; i < response.length; i++){
```

```
                var checkError = response[i].slice(0,5);
                if (checkError == "Error"){
                var elem = document.getElementById("status");
                elem.innerHTML += '<span style="color:red">' + response[i] + '</span><p></p>';
                elem.style.color = "Red";
                                elem.style.fontSize="small";
                }
                else{
                var elem = document.getElementById("status");
                elem.innerHTML += '<span style="color:green">' + response[i] +
'</span><p></p>';
                elem.style.color = "Green";

                elem.style.fontSize="small";
                }
}

                _("progressBar").value = 100;
 }

function errorHandler(event){
                _("status").innerHTML = "Upload Failed";
}

function abortHandler(event){
                _("status").innerHTML = "Upload Aborted";
}

</script>
<div id="dropbox">
<form method="post" enctype="multipart/form-data" name="AjaxTransferForm"
id="AjaxTransfer">
<input id="uploadButton" style="height: 30px;vertical-align:top;" type="button" value="Upload"
onclick="document.getElementById('files').click();" /></input>
<progress id="progressBar" value="0" max="100" style="width:225px;height:28px;vertical-
align:top;"></progress>
<input name="files[]" id="files" type="file" style="display: none;" onchange="createFD();"
multiple="true" /></input>
</form>
```

```
</div><br>
<div id="error"></div><div id="status"></div>
```

Liite 7: snippet.ajaxtransfer.php

```
<?php
/**
 * @package ajaxtransfer
 */
$ajaxtr = $modx->getService('ajaxtransfer','AjaxTransfer',$modx-
>getOption('ajaxtransfer.core_path',null,$modx-
>getOption('core_path').'components/ajaxtransfer/').'model/ajaxtransfer/',$scriptProperties);
if (!($ajaxtr instanceof AjaxTransfer)) return '';

// Load ajax connector resource

$resourceAlias = "AjaxTransferConnector";
$ajaxConnector = $modx->getObject('modResource', array('alias'=>$resourceAlias));

if($ajaxConnector == null){

                // Create a new ajax connector resource
                $ajaxConnector = $modx->newObject('modResource');
                $ajaxConnector->set('alias', 'AjaxTransferConnector');
                $ajaxConnector->set('contentType', 'text/php');
                $ajaxConnector->set('pagetitle', 'AjaxTransferConnector');
                $ajaxConnector->set('richtext', 0);
                $ajaxConnector->set('cacheable', 0);
                $ajaxConnector->set('content', '[[!AjaxTransfer_Server]]');
                $ajaxConnector->set('show_in_tree', '0');
                $ajaxConnector->set('published', 1);
                $ajaxConnector->save();

}

// Get ajax connector ID

$connectorID = $ajaxConnector->get('id');

if($id == null){
                $id = 1;
}
```

```
// Set parameters to database

$updateDB = $modx->getObject('AjaxTr', array('id'=>$id));

if($updateDB == null){
                $updateDB = $modx->newObject('AjaxTr',array('id' => $id));
}

$updateDB->set('path',$path);
$updateDB->set('random_name',$random_name);
$updateDB->set('max_size',$max_size);
$updateDB->set('allowed_extensions',$allowed_extensions);
$updateDB->set('allowed_mimes',$allowed_mimes);
$updateDB->save();

$uploadForm = $ajaxtr->getChunk('ajaxtransfer',array('id' => $id, 'path' => $path,'max_size' =>
$max_size,'allowed_mimes' => $allowed_mimes, 'connectorID' => $connectorID));

return $uploadForm;
```

Liite 8: snippet.ajaxtransfer_server.php

```
<?php

$ajaxtr = $modx->getService('ajaxtransfer','AjaxTransfer',$modx-
>getOption('ajaxtransfer.core_path',null,$modx-
>getOption('core_path').'components/ajaxtransfer/').'model/ajaxtransfer/',$scriptProperties);
if (!($ajaxtr instanceof AjaxTransfer)) return '';

$id = $_REQUEST['id'];

if($id == null){
		$id = 1;
}

$ajaxTR = $modx->getObject('AjaxTr', array('id'=>$id));

$tempPath = $ajaxTR->get('path');
$random_name = $ajaxTR->get('random_name');
$max_size = $ajaxTR->get('max_size');
$allowed_extensions = $ajaxTR->get('allowed_extensions');
$allowed_mimes = $ajaxTR->get('allowed_mimes');

$whitelist_ext = array();
$whitelist_type = array();
$images_ext =
array('jpg','jpeg','gif','png','tiff','bmp','webp','ppm','pgm','pbm','pnm','psd','psp','jxr','hdp','wdp');

$errors = array();
$succeeded = array();

// File default max size. 5000000 = 4.7MB

if ($max_size == null) {
		$max_size = 5000000;
}

if (isset($allowed_extensions)) {
```

```
                $whitelist_ext = explode(",", $allowed_extensions);
}

if (isset($allowed_mimes)) {
                $whitelist_type = explode(",", $allowed_mimes);
}

$path = $modx->config['base_path'] . $tempPath;

if (!$path) {
                return "Error: Please specify a valid upload path.";
}

for($i = 0; $i < sizeof($_FILES); $i++) {

                $file_field = "files".$i;
                $file_info = pathinfo($_FILES[$file_field]['name']);
                $name = $file_info['filename'];
                $ext = $file_info['extension'];
                $ext = strtolower($ext);

                if ($_FILES[$file_field]['error']){
                if($_FILES[$file_field]['error'] == 1){
                $errors[$i] = "Error uploading file $name.$ext : The uploaded file exceeds the
upload_max_filesize directive in php.ini";
                }
                else {
                $errors[$i] = "Error uploading file $name.$ext : PHP Error code
$_FILES[$file_field]['error']";
                }
                continue;
                }

                if (!in_array($ext, $whitelist_ext)) {
                $errors[$i] = "Error uploading file $name.$ext : Invalid file extension.";
                continue;
                }

                if (version_compare(phpversion(), '5.3.0', '<')) {
```

```
                if (!in_array($_FILES[$file_field]["type"], $whitelist_type)) {
                $errors[$i] = "Error uploading file $name.$ext : Invalid file type.";
                continue;
                }
                }

                else {
                $finfo = finfo_open(FILEINFO_MIME_TYPE);
                $mime = finfo_file($finfo, $_FILES[$file_field]['tmp_name']);

                if (!in_array($mime, $whitelist_type)) {
                $errors[$i] = "Error uploading file $name.$ext : Invalid file type";
                continue;
                }
                }

                if ($_FILES[$file_field]["size"] > $max_size) {
                $errors[$i] = "Error uploading file $name.$ext : File is too big.";
                continue;
                }

                if (in_array($ext, $images_ext)) {
                if (!getimagesize($_FILES[$file_field]['tmp_name'])) {
                $errors[$i] = "Error uploading file $name.$ext : Uploaded file is not a valid
image.";
                continue;
                }
                }

        if ($random_name != "false") {

                // Generate random filename
                $tmp = str_replace(array('.',' '), array(","), microtime());

                if (!$tmp || $tmp == '') {
                $errors[$i] = "Error uploading file $name.$ext : File must have a name.";
                }

                $newname = $tmp.'.'.$ext;
```

```
    }

else {
        $newname = $name.'.'.$ext;
}

//Tarkista löytyykö tiedostoa jo serveriltä

    if (file_exists($path.$newname)) {
$errors[$i] = "Error uploading file $name.$ext : A file with this name already exists.";
continue;
    }

if (move_uploaded_file($_FILES[$file_field]['tmp_name'], $path.$newname)) {
$succeeded[$i] = "File $name.$ext uploaded successfully.";
}

else {
$errors[$i] = "Error uploading file $name.$ext : Server error.";
}

}

$merged = array_merge($errors, $succeeded);
$output = implode("\n", $merged);
return $output;
```

Liite 9: ajaxtr.class.php

```
<?php
require_once (strtr(realpath(dirname(dirname(__FILE__))), '\\', '/') . '/ajaxtr.class.php');
class AjaxTr_mysql extends AjaxTr {
    public function __construct(& $xpdo) {
        parent :: __construct($xpdo);
    }
}
?>
```

Liite 10: ajaxtr.map.inc.php

```
<?php
$xpdo_meta_map['AjaxTr']= array (
 'package' => 'ajaxtransfer',
 'table' => 'ajaxtransfer',
 'fields' =>
 array (
  'path' => '',
  'random_name' => '',
  'max_size' => 5000000,
  'allowed_extensions' => '',
  'allowed_mimes' => '',
 ),
 'fieldMeta' =>
 array (
  'path' =>
  array (
   'dbtype' => 'text',
   'phptype' => 'string',
   'null' => false,
   'default' => '',
  ),
  'random_name' =>
  array (
   'dbtype' => 'varchar',
   'phptype' => 'string',
   'precision' => '255',
   'default' => '',
  ),
  'max_size' =>
  array (
   'dbtype' => 'text',
   'phptype' => 'string',
   'default' => '5000000',
  ),
  'allowed_extensions' =>
  array (
   'dbtype' => 'text',
```

```
      'phptype' => 'string',
      'default' => '',
    ),
    'allowed_mimes' =>
    array (
      'dbtype' => 'text',
      'phptype' => 'string',
      'default' => '',
    ),
  ),
);
```

Liite 11: ajaxtr.class.php (2)

```
<?php
class AjaxTr extends xPDOSimpleObject {
    public function __construct(& $xpdo) {
        parent :: __construct($xpdo);
    }

    /**
     * Overrides xPDOObject::save to add edited/created auto-filling fields
     *
     * {@inheritDoc}
     */
    public function save($cacheFlag = null) {
        /* set createdon/editedon */
        $now = new DateTime();
        $type = $this->isNew() ? 'created' : 'edited';
        $this->set($type.'on',$now->format('Y-m-d h:i:s'));
        if ($this->xpdo instanceof modX && $this->xpdo->user) {
            $this->set($type.'by',$this->xpdo->user->get('id'));
        }

        return parent :: save($cacheFlag);
    }
}
```

Liite 12: ajaxtransfer.class.php

```
<?php
/**
 * @package ajaxtransfer
 */
class AjaxTransfer {
    /**
     * Constructs the AjaxTransfer object
     *
     * @param modX &$modx A reference to the modX object
     * @param array $config An array of configuration options
     */
    function __construct(modX &$modx,array $config = array()) {
        $this->modx =& $modx;

        $basePath = $this->modx->getOption('ajaxtransfer.core_path',$config,$this->modx->getOption('core_path').'components/ajaxtransfer/');
        $assetsUrl = $this->modx->getOption('ajaxtransfer.assets_url',$config,$this->modx->getOption('assets_url').'components/ajaxtransfer/');
        $this->config = array_merge(array(
            'basePath' => $basePath,
            'corePath' => $basePath,
            'modelPath' => $basePath.'model/',
            'processorsPath' => $basePath.'processors/',
            'templatesPath' => $basePath.'templates/',
            'chunksPath' => $basePath.'elements/chunks/',
            'jsUrl' => $assetsUrl.'js/',
            'cssUrl' => $assetsUrl.'css/',
            'assetsUrl' => $assetsUrl,
            'connectorUrl' => $assetsUrl.'connector.php',
        ),$config);

        $this->modx->addPackage('ajaxtransfer',$this->config['modelPath']);
    }

    /**
     * Initializes the class into the proper context
     *
```

```
     * @access public
     * @param string $ctx
     */
    public function initialize($ctx = 'web') {
        switch ($ctx) {
            case 'mgr':
                $this->modx->lexicon->load('ajaxtransfer:default');

                if (!$this->modx->loadClass('ajaxtransferControllerRequest',$this-
>config['modelPath'].'ajaxtransfer/request/',true,true)) {
                    return 'Could not load controller request handler.';
                }
                $this->request = new ajaxtransferControllerRequest($this);
                return $this->request->handleRequest();
            break;
        }
        return true;
    }

    /**
     * Gets a Chunk and caches it; also falls back to file-based templates
     * for easier debugging.
     *
     * @access public
     * @param string $name The name of the Chunk
     * @param array $properties The properties for the Chunk
     * @return string The processed content of the Chunk
     */
    public function getChunk($name,$properties = array()) {
        $chunk = null;
        if (!isset($this->chunks[$name])) {
            $chunk = $this->_getTplChunk($name);
            if (empty($chunk)) {
                $chunk = $this->modx->getObject('modChunk',array('name' => $name));
                if ($chunk == false) return false;
            }
            $this->chunks[$name] = $chunk->getContent();
        } else {
            $o = $this->chunks[$name];
```

```
            $chunk = $this->modx->newObject('modChunk');
            $chunk->setContent($o);
        }
        $chunk->setCacheable(false);
        return $chunk->process($properties);
    }
    /**
     * Returns a modChunk object from a template file.
     *
     * @access private
     * @param string $name The name of the Chunk. Will parse to name.$postfix
     * @param string $postfix The default postfix to search for chunks at.
     * @return modChunk/boolean Returns the modChunk object if found, otherwise
     * false.
     */
    private function _getTplChunk($name,$postfix = '.chunk.tpl') {
        $chunk = false;
        $f = $this->config['chunksPath'].strtolower($name).$postfix;
        if (file_exists($f)) {
            $o = file_get_contents($f);
            $chunk = $this->modx->newObject('modChunk');
            $chunk->set('name',$name);
            $chunk->setContent($o);
        }
        return $chunk;
    }
}
```

Liite 13: ajaxtransfer.mysql.schema.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<model package="ajaxtransfer" baseClass="xPDOObject" platform="mysql"
defaultEngine="MyISAM">
   <object class="AjaxTr" table="ajaxtransfer" extends="xPDOSimpleObject">
      <field key="path" dbtype="text" phptype="string" null="false" default=""/>
                <field key="random_name" dbtype="varchar" precision="255" phptype="string"
default=""/>
                <field key="max_size" dbtype="text" phptype="string" default="5000000" />
                <field key="allowed_extensions" dbtype="text" phptype="string" default=""/>
                <field key="allowed_mimes" dbtype="text" phptype="string" default=""/>
   </object>
</model>
```