Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Anni Salo

Opinnäytetyö

PHP- ja J2ME-sovelluksen lokalisointi

Työn ohjaaja Petri Heliniemi
Työn tilaaja Mopedi Oy
Tampere 6/2009

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma

Tekijä	Anni Salo
Työn nimi	PHP- ja J2ME-sovelluksen lokalisointi
Sivumäärä	34
Valmistumisaika	kesäkuu 2009
Työn ohjaaja	Petri Heliniemi
Työn tilaaja	Mopedi Oy

TIIVISTELMÄ

Opinnäytetyö käsittelee PHP-sivuston ja J2ME-mobiilisovelluksen lokalisointia. Työn toimeksiantajana oli tamperelainen ohjelmistotalo Mopedi Oy, joka halusi sähköisiä sarjakuvia myyvään Eepos-palveluunsa englanninkielisen käännöksen. Lokalisoitava palvelu koostui PHP-sivustosta ja J2ME-mobiilisovelluksesta.

Työn tarkoituksena oli suunnitella ja toteuttaa PHP-sivuston ja J2ME-mobiilisovelluksen lokalisointi. Osana työtä selvitettiin myös lokalisoinnin taustaa ja mitä seikkoja tulisi ottaa huomioon web-sivuston ja mobiilisovelluksen lokalisoinnin suunnittelussa ja toteutuksessa. Työ keskittyi varsinkin erilaisiin tapoihin toteuttaa PHP-sivuston lokalisointi.

Työn lopputuloksena oli toimiva ratkaisu, jossa käytettiin tekstitiedostoja sivuston ja mobiilisovelluksen tekstisisällön säilyttämiseen.

Avainsanat lokalisointi, PHP, J2ME

TAMK University of Applied Sciences
Business Information Systems

Writer	Anni Salo
Thesis	PHP and J2ME application localisation
Pages	34
Graduation time	June 2009
Thesis Supervisor	Petri Heliniemi
Commissioning Company	Mopedi Oy

ABSTRACT

This thesis deals with the localisation of PHP and J2ME applications. The commissioning company, Mopedi Ltd of Tampere, Finland, wanted an English translation of Eepos, an online service that sells comics in electronic format. The service to be localised included a PHP site and a J2ME mobile application.

The purpose of this thesis was to design and create a way to localise PHP and J2ME applications. Another goal was to examine what factors should be taken into account in the localisation of web and mobile applications. The different ways to localise a PHP website were given emphasis.

As a result, the localisation of the site and the mobile application was achieved via the use of text files to store the text contents.

Keywords localisation, PHP, J2ME

1 Johdanto....5
2 Lokalisoinnin taustaa....7
2.1 Kielenkääntäminen....7
2.2 Merkistökoodaus....8
2.2.1 ASCII....9
2.2.2 ISO 8859-1....10
2.2.3 Windows-1252....10
2.2.4 UTF-8....10
2.2.5 Muita merkistökoodauksia....11
2.3 Kielikoodit....12
2.4 Palvelimen kielivalinta....13
2.5 Kielilinkit....14
2.5.1 Tekstilinkit....15
2.5.2 Kuvalinkit....15
3 Eepos-palvelun lokalisointi....17
3.1 PHP-sivusto....17
3.1.1 XML....18
3.1.2 PHP-tiedosto....19
3.1.3 Tekstitiedostot....20
3.1.4 Lokaalin vaihtaminen....24
3.1.5 Tekstien muokkaaminen....25
3.1.6 Kuvat....26
3.2 J2ME-mobiilisovellus....27
3.3 Jatkokehitys....31
4 Päätelmät....33
Lähteet....34

1 Johdanto

Idean opinnäytetyön aiheesta sain ollessani työharjoittelussa tamperelaisessa ohjelmistotalo Mopedissa. Harjoitteluni aikana yrityksessä kehitettiin Eepos-sarjakuvasivustoa, joka tarjoaa käyttäjille sarjakuvia sähköisessä muodossa. Yksi työtehtävistäni oli sivuston ja siihen liittyvän J2ME-mobiilisovelluksen kääntäminen englanniksi. Sivuston lokalisointi oli kuukausia jatkunut prosessi, jonka aikana kokeiltiin erilaisia tapoja toteuttaa kaksikielinen sivusto.

Tarkoituksena oli tutustua eri tapoihin, joilla PHP-sivustosta ja J2ME-sovelluksesta voi tehdä erikielisiä versioita. Luvussa 2 kerrotaan lokalisoinnin periaatteista ja lokalisointiin liittyvistä käytettävyysseikoista. Luvussa 3 esitellään kolme tapaa PHP-sivuston lokalisointiin. Lisäksi samassa luvussa esitellään yksi tapa, jolla voi tehdä monikielisen J2ME-mobiilisovelluksen. Työn tavoitteena oli myös selvittää, mitä asioita pitäisi ottaa huomioon, kun suunnittelee monikielistä sivustoa.

Harjoittelun päättymiseen mennessä sivusto ei vielä ollut valmis. Koska sivuston kääntäminen oli osa jatkuvaa kehitysprosessia, myös lokalisointi jäi epätäydelliseksi. Opinnäytetyön lopussa esitetään joitakin jatkokehitysideoita, jotka jäivät vielä toteuttamatta.

Lokalisointi on jatkuvasti ajankohtainen aihe. World Wide Web on nimensä mukaisesti maailmanlaajuinen verkko. World Internet Statsin (2009) mukaan Internet-käyttäjistä 29,1 % puhuu äidinkielenään englantia. Toisena on kiina, jota puhuu äidinkielenään 20,1 %. Kymmenen puhutuimman kielen joukossa ovat myös espanja (8,2 %), japani (5,9 %), ranska (4,6 %), portugali (4,5 %), saksa (4,1 %), arabia (2,6 %), venäjä (2,4 %) ja korea (2,3 %). Muiden kielten osuus on 16,2 %.

Englanti on Internetin käytetyin kieli, mutta se menettää valta-asemaansa, kun Internetin käyttäjien määrä kasvaa nopeasti. Vuosien 2000 ja 2008 välillä eniten kasvoi arabian-, venäjän- ja kiinankielisten Internet-käyttäjien määrä. Vaikka huomattava osa Internetin käyttäjistä osaa ainakin jotenkuten englantia, käyttää enemmistö Internetiä kuitenkin mieluiten äidinkielellään.

Suomenkielisen web-sivuston potentiaalisen käyttäjäkunnan koko on sama, kuin suomen kielen puhujien määrä maailmassa eli reilu viisi miljoonaa. Tarjoamalla käyttäjille kielivaihtoehtoja voi moninkertaistaa sivuston käyttäjien määrän. Jos kyseessä on kaupallinen sivusto, lokalisointi ja

internationalisointi lisää potentiaalisten asiakkaiden määrää huomattavasti. Toisaalta lokalisointi lisää tehtävän työn määrää, ja näin siitä aiheutuu lisäkustannuksia.

2 Lokalisoinnin taustaa

Lokalisointi (localisation, localization) ja internationalisointi (internationalisation, internationalization) tarkoittavat toimia, joilla sovellus sovitetaan eri kielien ja kulttuureiden mukaiseksi. Lokalisoinnista käytetään joskus myös termiä kotoistus ja internationalisoinnista termiä kansainvälistäminen. Englannissa kielessä käytetään myös lyhenteitä L10n ja i18n, missä ensimmäisen ja viimeisen kirjaimen välissä oleva numero tarkoittaa niiden välistä pois jätettävien kirjainten määrää.

Internationalisoinnilla tarkoitetaan sovelluksen muuttamista niin, että se ei ole sidottu mihinkään tiettyyn lokaaliin, vaan sitä voi käyttää melkein missä tahansa. Lokalisoinnin tavoite taas on kohdentaa tuote tietylle rajatulle käyttäjäryhmälle. Samasta tuotteesta voi olla useita erilaisia lokalisoituja versioita.

Kielenkääntäminen on olennainen osa lokalisointia. Muita lokalisoinnissa huomioon otettavia asioita ovat muun muassa merkistöt, valuutta, mittayksiköt ja paikalliset tavat.

2.1 Kielenkääntäminen

Kielenkääntäminen on lokalisoinnin olennainen ja näkyvin osa. Usein lokalisoinnilla tarkoitetaankin vain tekstisisällön kääntämistä toiselle kielelle. Yleinen käytäntö on erottaa tekstisisältö koodista. Tämä helpottaa ylläpidettävyyttä.

Kielenkääntämisessä lähdekielellä tarkoitetaan tekstin alkuperäistä kieltä, ja kohdekielellä kieltä, jolle teksti käännetään. Ammattikääntäjät kääntävät yleensä vain omalle äidinkielelleen. Kääntäjä on usein joku ulkopuolinen henkilö, joka ei ymmärrä ohjelmakoodia. Kääntäjän työ on helpompaa ja selvempää, jos hänellä on käännettävänä yksinkertainen tekstitiedosto, eikä vaikkapa PHP-tiedosto, jossa käännettävät tekstit ovat PHP-koodin seassa. Myös tästä syystä onkin hyvä pitää tekstit erillään koodista.

Internetissä on monia käännöskoneita, joiden taso vaihtelee suuresti. Monet niistä kääntävät sanan kerrallaan, minkä seurauksena käännetty teksti on usein lukukelvotonta. Jotkut käännöskoneet, kuten Google Translate, osaavat kääntää jopa lauserakenteita, ja tuloksena voi olla sujuvaakin kieltä. Mikään käännöskone ei kuitenkaan vielä korvaa kääntäjää, joka puhuu lähdekieltä ja kohdekieltä sujuvasti.

Käännösten laatu Internetissä vaihtelee suuresti. Riippuu sivujen tekijästä, kuinka tärkeänä hän pitää käännöksen laatua. Harrastelijasivun ylläpitäjä ei ehkä välitä, vaikka sivun englanninkielinen käännös ei ole täydellinen. Sen sijaan kaupallisen tai virallisen sivuston uskottavuus kärsii, jos sen kieli on kömpelöä. Vieraalle kielelle tehty käännös olisi hyvä ainakin tarkistuttaa kohdekielen natiivipuhujalla. Käännöksen huono laatu karkottaa käyttäjiä. Äidinkielenään muuta kuin englantia puhuvat käyttäjät saattavat myös käyttää mieluummin alkuperäistä englanninkielistä tuotetta, jos heidän omalle äidinkielelleen tehty käännös on huonolaatuinen. Toisaalta syynä vieraskielisessä versiossa pitäytymiseen voi olla myös se, että alkuperäinen versio on käyttäjille ennastaan tuttu.

2.2 Merkistökoodaus

Merkistö (character set) on kokoelma kirjaimia ja symboleja, joita käytetään kielen kirjoittamiseen. Merkistökoodaus (character encoding) määrittää merkistön merkeille koodipaikat, eli miten erilaiset bittiyhdistelmät tulkitaan merkeiksi.

Merkistökoodaus ei määrää kirjoitusmerkin ulkoasua. Merkin esitysmuoto eli glyyfi riippuu käytössä olevasta kirjasintyypistä eli fontista. Fontti on joukko glyyfejä. Eri fonttien esitysmuodot samasta merkistä voivat olla hyvinkin eri näköisiä. Kuviossa 1 näytetään saman koodipaikan erilaisia glyyfejä.

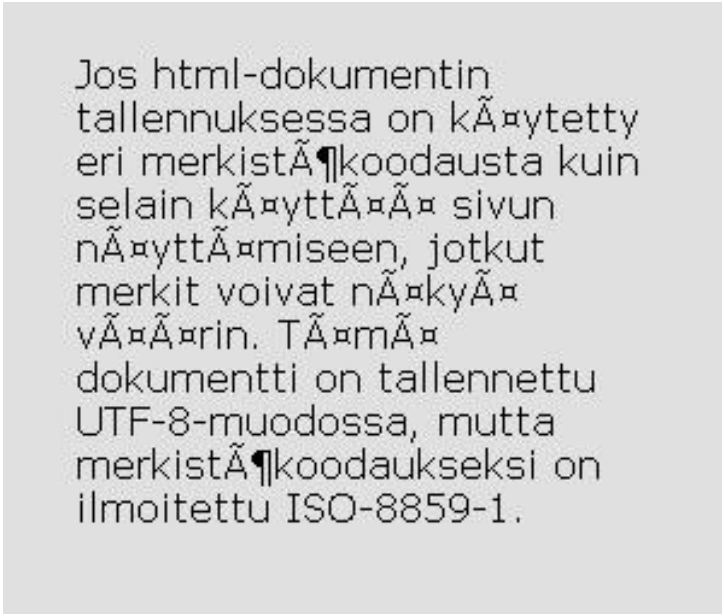
Kuvio 1: Saman koodipaikan erilaisia glyyfejä

Intenet-sivulla käytettävä merkistökoodaus määritellään HTML-dokumentin otsikkotietojen meta-elementissä:

```
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
```

Rivillä kerrotaan, että sivun sisältö on HTML:ää, ja että sivulla käytetään UTF-8-merkistökoodausta.

HTML-tiedosto täytyy tallentaa käyttäen samaa merkistökoodausta kuin otsikkotiedoissa ilmoitetaan. Jos tallennuksessa käytetty ja otsikkotiedoissa ilmoitettu merkistökoodaus eivät ole samat, web-selaimet saattavat näyttää osan sivun merkeistä väärin (kuvio 2). Sama ongelma ilmenee myös, jos otsikkotiedoista puuttuu tieto dokumentissa käytetystä merkistökoodauksesta. Tällöin selain käyttää sille asetettua oletusmerkistökoodausta. Merkistökoodausristiriitojen välttämiseksi kaikissa samaan sovellukseen liittyvissä tiedostoissa on hyvä käyttää samaa merkistökoodausta.



Kuvio 2: Väärä merkistökoodaus HTML-sivulla

2.2.1 ASCII

1960-luvulla kehitetty seitsemänbittinen ASCII-merkistö sisältää 128 merkkiä. Osa merkistöstä on varattu kontrollimerkeille, joten varsinaisia kirjoitusmerkkejä on alle 100. ASCII-merkistö onkin

varsin suppea, ja soveltuu lähinnä englannin kirjoittamiseen. Koska esimerkiksi Ä/ä- ja Ö/ö-kirjaimet eivät sisälly ASCII-merkistöön, kehitettiin siitä Euroopassa uusia versioita, joissa jotkin erikoismerkit korvattiin tarvittavilla kansallisilla merkeillä. Uudemmissa merkistöissä 128 ensimmäistä merkkiä ovat yleensä samat kuin ASCIIssa, eli ne ovat ASCII-yhteensopivia.

2.2.2 ISO 8859-1

Vuonna 1987 standardoitu ISO 8859-1 on kahdeksanbittinen ASCII-merkistön laajennos, joka tunnetaan myös nimellä Latin1. Sen ensimmäiset 128 merkkiä ovat samat kuin ASCII-merkistössä. ISO 8859-1 sisältää suuren osan länsieurooppalaisissa kielissä käytetyistä merkeistä. Suomen kieltä ISO 8859-1 tukee lähes täydellisesti. Siitä puuttuu joissakin suomalaisissa lainasanoissa ja venäläisten nimien translitteroinnissa käytetyt kirjaimet Š, š, Ž ja ž. Nämä kirjaimet ja lisäksi euron merkki € ovat osa uudempaa ISO 8859-15 -merkistöä.

2.2.3 Windows-1252

Windows-1252 on Windows-käyttöjärjestelmässä käytettävä merkistö. Se sisältää kaikki ISO 8859-15 -merkit, mutta niiden koodipaikat eivät aina ole samat.

Windows-1252 tunnetaan harhaanjohtavasti myös nimellä ANSI-merkistö, vaikka se ei olekaan ANSIn eli American National Standards Instituten hyväksymä standardi.

2.2.4 UTF-8

Unicode on merkistöstandardi, jonka on tarkoitus kattaa kaikki maailman kielissä käytetyt merkit. Unicode kattaa kaikki maailmassa käytössä olevat kirjoitusjärjestelmät. Unicodessa on koodipaikkoja yli miljoona, kun 7-bittisissä merkistöissä niitä on 128 ja 8-bittisissä 256. Unicoden koo-

daamiseen käytetään kolmea eri koodaustapaa: UTF-8, UTF-16 ja UTF-32, joissa yhden merkin koodaamiseen käytetään yhdestä neljään tavua. Unicode-merkistöön kuuluu käytössä olevien kielten merkkien lisäksi mm. historiallisia kirjoitusmerkkejä kuten riimukirjaimia.

Googlen indeksoimilla sivuilla Unicode ohitti ASCII:n ja länsieurooppalaiset merkistökoodaukset (Windows-1252 ja ISO-8859-1) joulukuussa 2007 (Davis, 2008).

2.2.5 Muita merkistökoodauksia

Unicodea voi käyttää kaikkien maailman kielten kirjoittamiseen. Muut merkistökoodaukset ovat paljon rajallisempia. Yleisesti käytössä oleva ISO 8859-1 ei esimerkiksi sovellu venäjän kielen kirjoittamiseen. Venäjän ja muiden kyrillisiä kirjaimia käyttävien kielten kirjoittamiseen on omat merkistökoodauksensa, muiden muassa ISO 8859-5 ja KOI8-R. Muita ISO 8859 -standardeja ovat muun muassa ISO 8859-6 (arabia), ISO 8859-7 (kreikka) ja ISO 8859-8 (heprea).

Kiinan, japanin ja korean kirjoittamista varten on yritetty luoda yhteinen Unicode-merkistö. Japanin kanji- ja korean hanja-merkit pohjautuvat kiinan hanzi-merkkeihin, mutta niiden ulkoasu poikkeaa jonkin verran toisistaan. Koska Unicode koodaa grafeemeja, ei glyyfejä, eli se ei määrää merkin ulkoasua, voi saman merkin näyttää erinäköisenä eri kielillä. Han-yhdistämistä ei kuitenkaan ole otettu hyvin vastaan Itä-Aasiassa. Japanissa käytetään yleisesti JIS-, Kiinassa Guobiao- ja Taiwanissa Big5-merkistökoodauksia.

Korean kielellä on oma aakkostonsa, hangul, jolla sitä yleensä kirjoitetaan. Hangul kirjoitetaan tavuittain, jotka koostuvat 24 kirjaimesta. Unicodessa on määritelty kaksi tapaa kirjoittaa hangulia. Ensimmäinen tapa on koodata jokainen tavu erikseen. Näitä tavuja on yli 11 000. Toinen tapa on koodata kirjaimet erikseen ja yhdistää niistä tavuja.

2.3 Kielikoodit

Kielikoodia käytetään dokumentin kielen ilmoittamiseen. Kielikoodit on määritelty kansainvälisissä standardeissa ISO 639-1 (kaksikirjaimiset koodit) ja ISO 639-2 (kolmikirjaimiset koodit).

HTML-dokumentin kieli määritellään html-elementin sisällä:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fi">
```

Kielikoodit eroavat monissa tapauksissa maatunnuksista. Suomen maatunnus ja suomen kielen kielikoodi ovat molemmat "fi", mutta monien muiden maiden ja kielien kohdalla näin ei ole. Esimerkiksi Ruotsin maatunnus on "se", mutta ruotsin kielikoodi on "sv". Kielikoodi "se" taas tarkoittaa pohjoissaamea. Myös monissa muissa tapauksissa maan ja sen valtakielen koodit eroavat toisistaan (taulukko 1).

Taulukko 1: Esimerkkejä tapauksista, joissa maan maatunnus ja maan valtakielen kielikoodi eroavat toisistaan (IANA — Root Zone Database, 2009; ISO 639-2 Language Code List, 2009)

maa	**maatunnus**	**kieli**	**kielikoodi (ISO 639-1)**	**huomautuksia**
Japani	jp	japani	ja	
Kiina	cn	kiina	zh	
Kreikka	gr	kreikka	el	
Ruotsi	se	ruotsi	sv	kielikoodi se = pohjoissaame maatunnus sv = El Salvador
Serbia	rs	serbia	sr	maatunnus sr = Surinam
Tanska	dk	tanska	da	
Viro	ee	viro	et	kielikoodi ee = ewe maatunnus et = Etiopia

Yhdistämällä kielikoodi ja maatunnus voidaan ilmaista tarkoitettavan kielen kyseisessä maassa puhuttua muotoa. Esimerkiksi sv-se tarkoittaa Ruotsissa puhuttavaa ruotsia eli riikinruotsia, kun taas sv-fi tarkoittaa suomenruotsia.

2.4 Palvelimen kielivalinta

Sisältöneuvottelu (content negotiation) on yksi tapa, jolla palvelin voi tehdä kielivalinnan. Käyttäjä voi asettaa Internet-selaimen asetuksissa kielet siihen järjestykseen, missä hän haluaa sivut mieluiten esitettävän (kuvio 3). Normaalit kielikoodit ovat kaksikirjaimisia, mutta monilla kielillä on myös pidempi koodi, johon on liitetty maakoodi. Esimerkiksi en-gb tarkoittaa brittienglantia, en-us amerikanenglantia ja en mitä tahansa englantia.

Kuvio 3: Kieliasetukset Mozilla Firefox 3.0.10 -selaimessa

Selain lähettää kieliasetukset palvelimelle osana HTTP-pyyntöä. Jos asetuksissa kieliksi on valittu suomi, brittienglanti, amerikanenglanti ja englanti, selaimen lähettämässä HTTP-pyynnössä ne määritellään Accept-Language-kentässä:

```
Accept-Language: fi,en-gb,en-us,en
```

Kielille voidaan antaa laatuarvo 0:n ja 1:n väliltä:

```
Accept-Language: fi;q=1, en-gb;q=0.8, en-us;q=0.7, en;q=0.6
```

Tämä tarkoittaa sitä, että suomi on mieluisin kieli ja englannin eri versiot käyvät myös, mutta niillä on hieman pienempi painoarvo. Jos pyydetystä sivusta on palvelimella erikielisiä versioita, kieliasetuksia käytetään kieliversion valinnassa. Laatuarvojen takia palvelin voi tarjota englanninkielistä versiota suomenkielisen sijaan, jos käännösten laaduissa on eroja.

Kielivalinnan perusteena voi olla myös käyttäjän Internet-osoitteen maatunnus. Tässä tavassa on kuitenkin ongelmansa, sillä maatunnus ei takaa, että käyttäjä osaisi kyseisen maan valtakieltä. Suomessakin on paljon Internetin käyttäjiä, joiden Internet-osoitteen maatunnus on .fi, mutta heidän suosimansa kieli on suomen sijaan jokin muu, esimerkiksi ruotsi.

2.5 Kielilinkit

Moni Internetin käyttäjä ei koske selaimen kieliasetuksiin. Tästä syystä tai vaikkapa hakukoneen linkkiä seuraamalla, käyttäjä voi päätyä sivulle, jonka kieltä hän ei ymmärrä. Jos sivusta on olemassa erikielisiä versioita, käyttäjää ei pidä pakottaa tiettyyn kieliversioon, vaan muihin kieliin on hyvä tarjota linkit.

Kielilinkkien tarjoamisessa käyttäjälle on eri vaihtoehtoja: aloitussivu, jolla käyttäjän pitää ensin valita haluamansa kieli, kielivalikko pääsivulla tai kielivalikko jokaisella sivulla. Jakob Nielsen (2000, 324–325) suosittelee, että erillistä kielenvalintasivua käytettäisiin vain siinä tapauksessa, että sivustolla ei ole selvää oletuskieltä, jota useimmat käyttäjät käyttäisivät. Kielilinkkien tarjoaminen oikeilla sisältösivuilla vähentää turhaa klikkailua. Kielilinkkien yhteydessä tulisi myös tehdä selväksi, tarjotaanko kyseisellä kielellä kaikki sivut ja niiden sisältö vai vain osa sivuston sisällöstä, esimerkiksi pelkkä yrityksen esittely.

Kielilinkkien sijoittaminen sivun alkuun helpottaa niiden havaitsemista. Sivun oikea yläkulma on melko vakiintunut paikka kielilinkeille.

2.5.1 Tekstilinkit

Kielivaihtoehtojen esittämiselle on monia tapoja. Linkit eri kieliin voivat olla tekstiä tai kuvia. Tekstilinkeissä on monia vaihtoehtoa sille, minkäkielisenä eri kielilinkit näytetään. Esimerkkinä voidaan käyttää sivustoa, josta on versiot suomen, ruotsin, englannin ja saksan kielillä.

Linkkitekstivaihtoehtoja:
"ruotsi", "englanti", "saksa" (kielten nimet sen hetkisellä käytössä olevalla kielellä)
"svenska", "English", "Deutsch" (kielten nimet käännettynä kyseisille kielille)
"Swedish", "English", "German" (kielten englanninkieliset nimet)
"sv", "en", "de" (kielikoodit)

Käyttäjän kannalta paras vaihtoehto on käyttää tapaa, jossa kieli ilmoitetaan käännettynä kyseiselle kielelle. Tärkeintä on, että kielilinkkejä tarvitsevat käyttäjät löytävät helposti oman kielensä, vaikka muut käyttäjät eivät tunnistaisikaan kaikkia vieraskielisten linkkien kieliä. On hyvinkin mahdollista, että suomenkieliselle sivulle eksynyt ruotsalainen ei välttämättä tiedä, mitä "ruotsi" tarkoittaa. Kielikoodien käyttö voi olla perusteltavaa tilanteessa, jossa kielilinkkien pitää mahtua pieneen tilaan.

2.5.2 Kuvalinkit

Tekstilinkkien sijaan linkit eri kieliin voivat olla myös kuvia. Yleisin tapa on käyttää lippuja. Lipun käyttö kielen symbolina on kuitenkin ongelmallista. Monia kieliä puhutaan useammassa kuin yhdessä maassa, ja monissa maissa puhutaan useampaa kuin yhtä kieltä. Esimerkiksi englantia puhutaan useissa maissa, joista tunnetuimpia ovat Yhdistynyt kuningaskunta ja Yhdysvallat. Yhdistyneen kuningaskunnan lippua käytetään yleisesti merkitsemään englannin kieltä, vaikka sivujen kieli olisikin amerikanenglantia.

Monikielisissä maissa, esimerkiksi Suomessa, lippujen käyttö kielen symbolina voi olla ongelmallista. Monikansallisen yrityksen web-sivuilla lippujen käyttö saattaa olla hämmentävää, koska käyttäjä voi luulla, että lippulinkki osoittaa sivuille, joissa kerrotaan yrityksen toiminnasta kysei-

sessä maassa. Maiden ja kielien sekoittaminen voi olla myös harhaanjohtavaa, jos sivuston ei ole tarkoitus eri kieliversioista huolimatta palvella oman kotimaansa ulkopuolelta tulevia käyttäjiä. Esimerkiksi verkkokaupan tulisi selvästi kertoa maksuvaihtoehdoista ja siitä, toimittaako se tuotteita ulkomaille. Jos maksuvaihtoehtoina ovat vain suomalaiset verkkopankit, se tulisi kertoa käyttäjälle ennen kuin hän on ehtinyt lisätä tuotteita ostoskoriin.

3 Eepos-palvelun lokalisointi

Eepos on Mopedi Oy:n vielä kehitteillä oleva palvelu, jossa käyttäjät pystyvät ostamaan sarjakuvia sähköisessä muodossa. Palvelua voi käyttää tietokoneella tai mobiililaitteella. Sarjakuvia tarjotaan aluksi suomen- ja englanninkielisinä. Tämän vuoksi palvelukin haluttiin tarjota useammalla kielellä. Palveluun kuuluu PHP-sivusto ja J2ME-mobiilisovellus. Työharjoitteluni aikana käänsin sekä PHP-sivuston että mobiilisovelluksen suomesta englanniksi. Lokalisointi oli kuukausia kestävä jatkuva prosessi, joka jatkui, kun palveluun tuli uusia ominaisuuksia.

3.1 PHP-sivusto

Eepos-sivusto on Java EE -pohjalle rakennettu PHP-sivusto. Alkuperäisellä sivustolla sivujen suomenkieliset tekstit oli sijoitettu suoraan koodin joukkoon. Monikielisen sivuston toteutuksessa on kaksi vaihtoehtoa: joko jokaiselle kielelle tehdään omat erilliset sivunsa tai sisältö erotetaan koodista ja sivun sisältöä muutetaan kielen mukaan. Eepos-sivustolla haluttiin käyttää jälkimmäistä vaihtoehtoa, eli eri kielille ei luoda erillisiä PHP-sivuja, vaan kunkin PHP-sivun sisältö määräytyy käyttäjän kielen mukaan. Ylläpidettävyyden kannalta erilliset kieliversiot ovat hankalammat, koska sivuille tehtävät muutokset täytyy tehdä jokaiseen kieliversioon erikseen.

PHP-sivujen lokalisointiin on monia tapoja. Eepoksen kääntäminen oli jatkuva prosessi, jonka aikana ehdittiin kokeilla useita eri lokalisointimenetelmiä. Pohjimmainen idea pysyi samana kaikissa menetelmissä:

•Käyttäjän kielivalintaa säilytetään $locale_-muuttujassa.

•PHP-sivulla jokaisen näytettävän tekstin kohdalla kutsutaan tekstinhakufunktiota.

•Tekstinhakufunktio hakee halutun tekstin erillisestä tiedostosta.

•Kaikilla sivun teksteillä on jonkinlainen yksilöivä tunnus, jonka perusteella tekstit erotellaan toisistaan ja jota käytetään tekstinhakufunktiossa.

3.1.1 XML

Eepoksen tekstit oli alustavasti suunniteltu tallennettavan XML-tiedostoihin. XML-tiedostoja käytettiin Eepoksessa muutenkin väliaikaisten tietojen tallennukseen. Ensimmäinen lokalisointikokeilu tehtiin siis XML-tiedostoilla. Kielitiedostot nimettiin `$locale_`-muuttujan mahdollisten arvojen mukaan: eng.xml ja fin.xml. Kielitiedostossa jokainen sivuston tekstinpätkä oli oman <tag>-elementin sisällä. Jokainen <tag>-elementti yksilöitiin id-nimisellä attribuutilla, joka sai arvokseen viisinumeroisen kokonaisluvun (00001, 00002, 00003, …):

```
<?xml version="1.0" encoding="UTF-8" ?>
<tekstit>
            <tag id="00001">
                        Etusivu
            </tag>
            <tag id="00002">
                        Selaa sarjakuvia
            </tag>
</tekstit>
```

XML-tiedostojen käsittelyyn on olemassa valmiit PHP-funktiot, joita käytettiin tekstinhakufunktiossa. Käytössä olevaa kieltä vastaavan XML-tiedoston sisältö luettiin `$xml`-muuttujaan PHP:n simplexml_load_file-funktiolla. Tekstinhakufunktion foreach-silmukassa käytiin läpi kaikki XML-tiedoston <tag>-elementit ja palautettiin sen elementin sisältö, jonka id-attribuutti vastasi funktion kutsussa käytettyä arvoa:

```
function getLocText($tagi)
{
            $xml = simplexml_load_file("locale/".$locale_.".xml");
                        foreach ($xml->tag as $tag) {
                        if($tag['id'] == $tagi) {
                                    return trim(encThis($tag));
                        }
            }
}
```

XML-tiedostossa tekstit olivat luettavuuden vuoksi omilla riveillään. Rivinvaihto lisäsi merkkijonon alkuun yhden välilyönnin, joka poistettiin trim-funktiolla. Tekstin koodaukseen käytettiin omaa encThis-funktiota.

Esimerkki getLocText-funktion kutsusta php-koodin seassa:

```
print '<h1>'.getLocText("00001").'</h1>';
```

Tässä esimerkissä funktiokutsu hakee XML-tiedostosta <tag>-elementin, jonka id-attribuutin arvo on 00001, ja palauttaa elementin sisällön, eli tässä tapauksessa merkkijonon "Etusivu".

Numeeriset tunnukset vaikeuttivat kuitenkin koodin lukemista, ja siksi ne nimettiin uudelleen. Tagin id:hen lisättiin etuliite, joka selvensi mihin sivuston aihealueeseen tagin sisältö kuului. Esimerkiksi "frontpage_01" kertoi, että teksti kuului etusivulle, ja "user_01", että teksti liittyi käyttäjätietoihin. Näin koodin luettavuus parani.

```
print '<h1>'.getLocText("frontpage_01").'</h1>';
```

Esimerkin koodissa ”frontpage_01” kertoo lukijalle, että funktiokutsun paikalle tuleva teksti liittyy etusivuun. Aiempi ”00001” ei kerro lukijalle mitään.

3.1.2 PHP-tiedosto

XML-tiedostoista siirryttiin PHP-tiedostoihin. PHP-tiedostojen käyttö XML-tiedostojen sijaan yksinkertaisti sekä kielitiedostoja että tekstinhakufunktiota. PHP-tiedostoissa fin.php ja eng.php oli molemmissa sisältönä taulukko. Taulukko oli sisällöltään sama kuin aiempi vastaava XML-tiedosto. Taulukossa oli määritelty avain–arvo-pareja, joissa avaimena oli XML-tiedoston <tag>-elementin id-attribuutti ja arvona sitä vastaava tekstisisältö:

```
$texts = array (
           "links_01" => "Etusivu";
           "links_02" => "Selaa sarjakuvia";
);
```

Tekstinhakufunktio selkiytyi php-taulukon käytön myötä. Funktiossa otettiin ensin käyttäjän kieltä vastaava php-tiedosto käyttöön. Seuraavaksi taulukosta haetaan parametrina saatua $tagi-muuttujaa vastaava teksti:

```
function getLocText($tagi)
{
           $file = "locale/".$locale_.".php";
           include $file;
           return $texts[$tagi];
}
```

Seuraavaksi tageja (eli taulukon avaimia) selvennettiin vielä enemmän, eli ne muuttuivat selväsanaisiksi, esimerkiksi "frontpage_title", "user_password". Näin koodin luettavuus parani entisestään ja tekstitiedostossa samaan osioon liittyvät tekstit erottuivat selvemmin.

```
print '<h1>'.getLocText("frontpage_title").'</h1>';
```

Uusien selväsanaisten tagien ansiosta lähdekoodi on huomattavasti luettavampaa kuin aiemmin. Esimerkin tagi ”frontpage_title” kertoo sivun koodajalle selkeästi, että kyseiseen kohtaan haettava teksti on etusivun otsikko.

3.1.3 Tekstitiedostot

Lopulta php-taulukoista siirryttiin tekstitiedostoihin. Näin sivuston tekstisisältö pidetään selvästi erillään ohjelmakoodista.

Lopullisessa versiossa tageilla on etuliitteet, joista tunnistaa, mihin sivuston osaan ne kuuluvat. Etuliitteitä ovat esimerkiksi "frontpage" etusivun teksteille, "user" käyttäjään liittyville teksteille ja "links" navigointipalkin linkkiteksteille. Nämä etuliitteet helpottavat tekstitiedostojen käyttöä, kun kaikki samaan alueeseen kuuluvat tekstit ovat yhdessä. Tekstit eivät kuitenkaan välttämättä pysy yhdessä, jos ylläpitäjä muuttaa tagien nimiä Eepoksen tekstieditorissa. Tagin nimen muuttaminen aiheuttaa sen, että vanha tagi–teksti-pari poistetaan ja uusi muokattu tagi–teksti-pari lisätään tekstitiedoston loppuun.

Jokaisella kielellä on oma tekstitiedosto, jossa sivuston tekstit säilytetään. Tekstitiedostossa on ensin määritetty tunnistetagi, jolla jokainen sivuston teksti yksilöidään. Tagi erotetaan tekstistä kolmella yhtäsuuruusmerkillä (===). Tagi–teksti-parit erotetaan toisistaan kolmella pystyviivalla eli

putkimerkillä (|||). Erotinmerkkejä tarvitaan silloin, kun tiedoston sisältö luetaan php-taulukkoon. Tagit ovat samat kaikissa erikielisissä tekstitiedostoissa.

fin.txt:

```
links_frontpage===Etusivu|||
links_login===Kirjaudu|||
links_browse===Selaa sarjakuvia|||
```

eng.txt:

```
links_frontpage===Front page|||
links_login===Log in|||
links_browse===Browse comics|||
```

Vaikka tavoitteena oli pitää teksti ja koodi erillään toisistaan, on tekstitiedostoissa jonkin verran html-koodia, esimerkiksi linkkejä.

Toisin kuin XML- ja PHP-tiedostojen kohdalla, tekstitiedostosta ei voi suoraan hakea tunnustagia vastaavaa tekstiä. Tiedoston käsittelyyn tarvittiin uusi funktio, readLangTexts, jossa tiedoston sisältö pilkotaan erotinmerkkien kohdalta explode-funktiota käyttämällä. Aluksi tiedoston teksti pilkotaan jokaisen |||-erotinmerkin kohdalta taulukon alkioiksi. Seuraavaksi taulukon jokainen alkio käydään läpi foreach-silmukassa ja jaetaan avain–arvo-pariksi ===-erotinmerkin kohdalta. Funktio luo tiedoston sisällöstä PHP-taulukon, jota käsitellään tekstinhakufunktiossa kuten ennenkin:

```
function readLangTexts ($loc) {

            global $langTexts;
            global $locale_;

            $langArray = array();

            $filecontents = file_get_contents("locale/".$loc.".txt");

            $texts = explode("|||", $filecontents);

            foreach($texts as $text)
            {
                        $line = explode("===", $text);

                        $key = trim($line[0]);
                        $value = trim($line[1]);

                        $langArray[$key] = $value;
            }
```

```
            return $langArray;
}
```

getLocText-funktiota kutsutaan jokaisella sivulla monta kertaa. PHP-koodin joukossa funktion kutsu on yksinkertaista. HTML-koodissa tarvitaan PHP-lohko, jossa funktiota kutsutaan:

```
<h2><?php print getLocText("partners_title")?></h2>
```

Funktio saa parametrina tagin, jota vastaava teksti haetaan taulukosta. Aluksi tarkistetaan, että käytössä on oikea kielitaulukko vertaamalla $lastlocale- ja $locale_-muuttujia. Jos vanha ja nykyinen kieli eivät ole samat, tai jos käytössä ei ole kielitaulukkoa, kutsutaan readLangTexts-funktiota, joka palauttaa nykyistä kieltä vastaavan kielitaulukon. Jos vanha ja uusi kieli ovat samat, luetaan tekstit olemassa olevasta vanhasta taulukosta.

```
function getLocText($tagi)
{
            global $langTexts;
            global $lastLocale;
            global $locale_;

            if ( $lastLocale != $locale_ ||
                        is_null($langTexts) ) {
                        $langTexts = readLangTexts($locale_);
                        $lastLocale = $locale_;
            }

            if(array_key_exists($tagi, $langTexts))
            {
                        return $langTexts[$tagi];
            }
            else
                        return $tagi;

}
```

Parametrina saatua tagia etsitään kielitaulukosta. Jos tagi löytyy, palautetaan sitä vastaava teksti. Jos tagia ei löydy taulukosta, palautetaan tagi. Tällöin PHP-sivulla näkee helposti, jos joku tekstikäännös puuttuu.

Esimerkiksi getLocText-funktiota kutsutaan etusivun koodissa:

```
print '<h1>'.getLocText("frontpage_title").'</h1>';
```

Jos tekstinhakufunktio getLocText löytää ”frontpage_title”-tagia vastaavan tekstin, se sijoitetaan funktiokutsun paikalle:

```
<h1>Sarjakuvat matkapuhelimeen</h1>
```

PHP-sivulla kielitiedostosta haettu teksti ”Sarjakuvat matkapuhelimeen” näkyy sivun otsikkona (kuvio 4).

Kuvio 4: Kielitiedostosta haettuja tekstejä

Jos getLocText-tekstinhakufunktio ei löydä tagia vastaavaa tekstiä kielitiedostosta, PHP-sivulla näytetään tagin nimi. Kuviossa 5 näytetään, kuinka puuttuva teksti näkyy PHP-sivulla. Sivun muut tekstit on haettu onnistuneesti getLocText-funktion avulla kielitiedostosta, mutta sivun otsikkoa ei löytynyt kielitiedostosta. Otsikon paikalla näytetään tagi, joka pitäisi lisätä kielitiedostoon.

Kuvio 5: Kielitiedostosta puuttuva teksti

Osa sivuston sisällöstä on toteutettu PHP:n heredoc-syntaksin avulla. Heredoc-syntaksia käytetään varsinkin luomaan pitkiä, valmiiksi muotoiltuja merkkijonoja, joissa on esimerkiksi sisennyksiä. Merkkijonon sisällä voi tulostaa muuttujia, mutta ei voi kutsua funktioita. Tässä tilanteessa käytetään apuna PHP:n create_function-funktiota:

```
$getLocText = create_function('$tagi', 'return getLocText($tagi);');
```

create_function-funktio luo uuden anonyymin funktion, jota voidaan kutsua $getLocText-muuttujan kautta myös heredocin sisällä. Anonyymin funktion ensimmäinen argumentti on luotavan funktion parametri ja toinen argumentti luotavan funktion koodi. Tässä tapauksessa luodaan anonyymi funktio, joka saa parametrikseen $tagi-muuttujan, ja joka vuorostaan kutsuu getLocText-funktiota $tagi-muuttujalla ja palauttaa sen arvon.

Näin anonyymia funktiota kutsutaan heredoc-syntaksin sisällä hakemaan uloskirjautumislinkin teksti linkkilistaan:

```
$templates_["links_loggeduser"] .= <<<EOD
<li><a href="{$vars_["logout"]}">{$getLocText("user_logout")}</a></li>
EOD;
```

3.1.4 Lokaalin vaihtaminen

Eepos-sivuston oletuskieli on suomi. Käyttäjä voi vaihtaa sivuston kieltä kielilinkeistä, jotka näkyvät jokaisella sivulla. Kielivaihtoehdot esitettiin harjoitteluni aikana linkitettyinä Suomen ja Yhdistyneen kuningaskunnan lippuina. Ehdotin lippujen muuttamista suomi- ja English-tekstilinkeiksi, mutta harjoitteluni päättyessä lopullista päätöstä lippujen ja tekstilinkkien välillä ei ollut tehty.

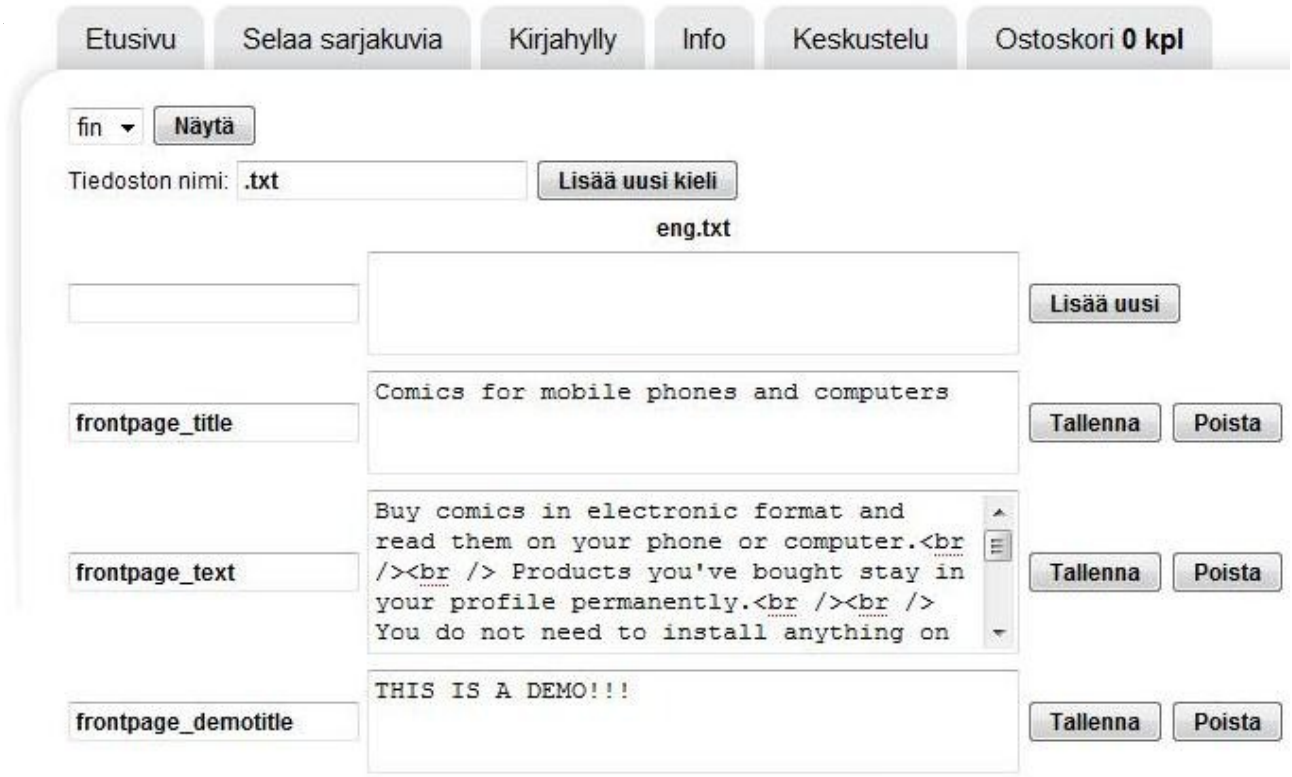
Jos käyttäjä valitsee sivustolla kielilinkkiä klikkaamalla toisen kielen, uusi lokaali tallentuu $locale_-muuttujaan. Lokaalin vaihtuminen tarkistetaan $lastLocale-muuttujan avulla. Vanhan kielen tallentamiseen tarkoitettu $lastLocale-muuttuja on aluksi tyhjä. Kun käyttäjä vaihtaa kieltä, vanha kieli tallennetaan $lastLocale-muuttujaan. Aina kun getLocText-tekstinhakufunktiota kutsu-

taan, verrataan vanhaa lokaalia nykyiseen lokaaliin. Jos vanha ja uusi lokaali eivät ole samat, kielitaulukko ladataan uudestaan. Muussa tapauksessa tekstit luetaan olemassa olevasta vanhasta taulukosta.

3.1.5 Tekstien muokkaaminen

Eepos-sivuston admin-käyttäjällä on käyttöliittymässä linkkejä, joiden kautta pääsee sivuston hallintatyökaluihin. Yksi näistä on tekstieditori, jolla voi muokata sivuston tekstejä (kuvio 6). Tekstieditorissa näytetään koko kielitiedoston sisältö taulukossa, jota ylläpitäjä voi muokata. Oletuksena editorissa näytetään englanninkieliset tekstit. Muunkieliset tekstit saa näkyviin valitsemalla haluamansa kielen editorin yllä olevasta alasvetovalikosta.

Editorilla voi muokata kielitageja ja niitä vastaavia tekstejä. Lisäksi editorilla voi lisätä uusia tagi–teksti-pareja ja niitä voi myös poistaa. Editorilla voi myös lisätä palveluun kokonaan uuden kielen. Jokainen tagi–teksti-pari on omassa lomakkeessaan, jolla on omat Tallenna- ja Poista-painikkeet. Kun Tallenna- tai Poista-painiketta painetaan, lomake käsitellään updateText-funktiossa. Funktiossa tehdään halutut muutokset kielitaulukkoon, ja lopuksi muunnetaan taulukko tekstimuotoon. Päivitetyt tekstit tallennetaan vanhojen tilalle tekstitiedostoon.



Kuvio 6: Eepoksen tekstieditori

3.1.6 Kuvat

Muutamissa Eepos-sivuston kuvissa on tekstiä, ja näiden tapausten kohdalla tarvitaan eri kielille omat versiot kuvista. Suurin osa kuvista on kuitenkin samoja jokaisella kielellä. Vaikka kuva olisi sama, sen alt-teksti on kuitenkin yleensä käännettävä eri kielille. Alt-teksti on kuvan selitys, joka näytetään, kun kuvaa ei voi jostain syystä näyttää. Monissa Eepos-sivuston kuvissa alt-teksti on tarpeeton ja se on jätetty tyhjäksi. Käännöksen tarvitsevat alt-tekstit on tallennettu samoihin kieli-tiedostoihin kuin muutkin sivuston tekstit. Alt-tekstien tunnistetagit alkavat img-merkkijonolla, jotta ne erottuvat normaaleista kieliteksteistä.

3.2 J2ME-mobiilisovellus

Eepos-MIDlet on J2ME-pohjainen mobiilisovellus, jonka kautta käyttäjä voi kirjautua Eepos-palveluun. MIDlet koostuu neljästä luokasta: Eepos, joka on sovelluksen pääluokka, EeposCanvas, jonka avulla luodaan sovelluksen käyttöliittymä, WordParser, jota käytetään tekstin rivitykseen ja Locale, jolla toteutetaan eri kieliversioiden tekstien käsittely. Alunperin MIDlet oli vain suomenkielinen ja siihen toivottiin kielenvaihtomahdollisuutta.

Mobiilisovelluksen lokalisoinnissa on kaksi vaihtoehtoa: joko eri kielille tehdään erilliset sovellukset, tai saman sovelluksen sisältöä vaihdetaan kielen mukaan. Eepos-MIDletistä päätettiin tehdä yksi sovellus, joka toimii sekä suomen- että englanninkielisenä sen sijaan, että eri kielille olisi tehty erilliset sovellukset. MIDletiin toivottiin toimintoa, jossa käyttäjälle tarjotaan mahdollisuus kielen vaihtoon.

Monikielisen MIDletin toteutuksessa on joitakin vaihtoehtoja. Erikielisten tekstien tallentamiseen voi käyttää:

•pääluokan muuttujia

•erillistä luokkaa

•erillistä rajapintaa

•erillistä tekstitiedostoa.

PHP-sivuston tapaisesti Eepos-mobiilisovelluksen tekstit on erotettu ohjelmakoodista. Tekstejä säilytetään täysin erillään ohjelmakoodista erillisessä resurssitiedostossa locale.txt. Sekä suomen- että englanninkieliset tekstit ovat samassa tekstitiedostossa. Tekstitiedostoa käsitellään erillisessä Locale-luokassa.

```
savetext_en=Save
usernametext_en=Username
passwordtext_en=Password
...
savetext_fi=Tallenna
usernametext_fi=Tunnus
```

```
passwordtext_fi=Salasana
...
```

MIDletissä käytetään RecordStore-tietovarastoa käyttäjätietojen säilyttämiseen. RecordStore on tietuekanta, joka tallentaa pysyvää tietoa mobiililaitteen muistiin. RecordStoressa säilytetään käyttäjän Eepos-tunnuksen ja salasanan lisäksi myös tietoa sovelluksen kielestä. MIDletin käynnistyksen yhteydessä RecordStoresta haetaan tieto kielestä, jota käyttäjä on edellisellä kerralla käyttänyt.

Kun Eepos-MIDlet käynnistetään, tarkistetaan, onko sovelluksessa tietovarastona käytettävään RecordStoreen tallennettu tietoa kielestä, jota käyttäjä on edellisellä kerralla käyttänyt. Jos RecordStoressa ei ole tallennettuna käyttäjän kieltä, mobiililaitteen kieli selvitetään `System.getProperty ("microedition.locale")`-metodilla. Metodi palauttaa mobiililaitteen kielikoodin, esimerkiksi en-US tai fi-FI. MIDletin String-tyyppinen muuttuja `locale` saa arvokseen metodin palauttaman merkkijonon kaksi ensimmäistä merkkiä. Jos sovellus ei tue mobiililaitteen kieltä, localeksi asetetaan en (englanti). Kieli tallennetaan RecordStoreen.

Kun kieli on valittu, tekstitiedoston sisältö luetaan Locale-luokassa. Tekstitiedosto on tallennettu UTF-8-muodossa, joten lukemisen yhteydessä tekstin merkistökoodaukseksi pitää määritellä UTF-8. Tämä on mahdollista vain käyttämällä InputStreamReader-luokkaa, joka saa yhdeksi paramet-rikseen tekstin merkistökoodauksen:

```
InputStream is = this.getClass().getResourceAsStream("/locale.txt");
Reader stream = new InputStreamReader(is, "UTF-8");
```

Tagi ja sitä vastaava teksti tallennetaan avain–arvo-parina Hashtable-taulukkoon siten, että kunkin rivin =-erotinmerkkiä edeltävä osuus on avaimena ja erotinmerkkiä seuraava osuus avainta vastaavana arvona.

Pääluokasta kutsutaan Locale-luokan getString-metodia. Metodikutsussa annetaan parametrina haluttua tekstiä vastaava tagi:

```
l = new Locale();
l.getString("savetext_en");
```

Metodi hakee taulukosta parametrina annetun avaimen ja palauttaa sitä vastaavan arvon pääluokkaan.

Koska Commandia, eli käyttöliittymän toimintoa, ei voi jälkeenpäin muokata, tarvitaan jokaisesta kommennosta oma versio jokaisella sovelluksen kielellä. MIDletin käynnistyksen yhteydessä kaikista Commandeista luodaan sekä suomen- että englanninkieliset versiot:

```
Command save_en = new Command(Locale.getString("savetext_en"), Command.ITEM,
1);
Command save_fi = new Command(Locale.getString("savetext_fi"), Command.ITEM,
1);
```

Jokaisen Commandin ensimmäinen parametri on valikossa näytettävä komennon teksti. Valikkotekstit nimetään hakemalla Locale-luokasta tarvittava teksti. Esimerkissä luodaan englannin- ja suomenkieliset käyttäjätunnuksen tallennuskomennot.

Commandit lisätään Hashtable-taulukkoon, jotta niihin päästään myöhemmin helposti käsiksi. Taulukossa on avain–arvo-pareina jokaisen Command-olion nimi String-merkkijonona ja sitä vastaava Command-olio:

```
Hashtable locObjects = new Hashtable();
locObjects.put("save_en", save_en);
locObjects.put("save_fi", save_fi);
```

Tämän jälkeen tarvittava Command-olio voidaan hakea locObjects-taulukosta syntaksilla `(Command)locObjects.get("save_" + locale)` eli halutun Command-olion nimi täydennetään `locale`-muuttujaan tallennetun kielen avulla.

Muita käyttöliittymäelementtejä voi muokata jälkikäteen, joten niistä luodaan vain aloituskielen mukaiset versiot. Esimerkiksi salasanakentän luonnin yhteydessä kentän otsikoksi asetetaan aloituskielen mukainen teksti:

```
TextField passwordField = new TextField(Locale.getString("passwordtext_" +
locale), "", 20, TextField.PASSWORD);
```

Kun kaikki käyttöliittymäkomponentit on luotu, kutsutaan setLocale-metodia:

```
public void setLocale() {

            if(!oldloc.equals("")) { // Jos oldloc ei ole tyhjä eli
                                     // kieltä vaihdettu
                        // Poistetaan vanhat komennot.
                        // Nimetään tunnus- ja salasanakenttien otsikot
                        // uudestaan.
            }
            // Asetetaan sovelluksen tekstit.
            // Asetetaan uudet komennot.
}
```

Metodissa tutkitaan ensin oldloc-muuttujan arvo. Metodin if-lohko suoritetaan vain, jos oldloc-muuttujalla on arvo. MIDletin käynnistyksen yhteydessä oldloc-muuttujan arvo asetetaan tyhjäksi, joten kun setLocale-funktioon saavutaan ensimmäistä kertaa, if-lohko sivuutetaan. Käynnistyksen yhteydessä suoritetaan vain if-lohkon jälkeen tuleva koodi eli sovellukset tekstien ja komentojen asettaminen.

Sovelluksen kieltä vaihdetaan valitsemalla kielenvaihto sovelluksen valikosta (kuvio 7). Kun käyttäjä valitsee kielen, oldloc-muuttujaan asetetaan vanha kieli ja locale-muuttujaan käyttäjän valitsema uusi kieli.

Kuvio 7: Mobiilisovelluksen kielenvaihtovalikko

Uutta kieltä verrataan vanhaan kieleen. Jos kielet eivät ole samat, kutsutaan setLocale-metodia. Kielenvaihdon kautta metodiin tultaessa oldloc-muuttuja ei ole tyhjä, joten if-lohko suoritetaan, eli

kaikki vanhaa kieltä vastaavat Commandit poistetaan ja tunnus- ja salasanakenttien otsikot vaihdetaan. Tämän jälkeen haetaan uutta kieltä vastaavat kommennot taulukosta ja asetetaan ne sovelluksen näkymiin. Metodissa vaihdetaan myös sovelluksen muut tekstit, joita ei tarvitse ensin erikseen poistaa. Kun tekstit on vaihdettu, uusi kieli tallennetaan RecordStoreen.

Eepos-MIDletillä otetaan yhteys Eepos-sivustoon. Kun käyttäjä valitsee sivustolle siirtymisen, mobiililaitteen Internet-selain avataan. Suunnitelmissa oli lisätä URL-osoitteen yhdeksi parametriksi MIDletin kieli, jota käytettäisiin sivuston kielen asetukseen ja sivusto näytetään käyttäjälle vastaavalla kielellä. Tämä jäi vielä toteuttamatta, sillä MIDletissä käytetään kaksikirjaimisia kielitunnuksia fi ja en, ja sivustossa käytetään pitempiä muotoja fin ja eng. Näiden yhteensovittaminen jäi vielä avoimeksi.

3.3 Jatkokehitys

Koska Eepos-palvelu ei ollut valmis työharjoitteluni päättyessä, sen lokalisointikaan ei ollut vielä valmis. Lokalisointi on jatkuva prosessi, joka jatkuu koko sovelluksen kehityksen ajan.

Eepos-sivustoa ja mobiilisovellusta ei lokalisoitu juurikaan muuten kuin kielen osalta. Lokalisointi kuitenkin käsittää muutakin kuin tekstisisällön kääntämisen toiselle kielelle. Avoimeksi jäi muun muassa se, onko Suomen ulkopuolelta tulevalla käyttäjällä ylipäätään mahdollisuutta ostaa Eepoksen tarjoamia tuotteita.

Käytettävyyden osalta huomiota voisi kiinnittää kielilinkkeihin. Harjoitteluni päättyessä käytössä oli lippulinkit, joihin liittyvistä ongelmista kerrottiin tämän työn luvussa 2.5.2. Lippujen sijaan linkeissä tulisi käyttää kielten nimiä. Koska Eepos-palvelussa tarjotaan erikielisiä sarjakuvia, käyttäjälle voi olla myös epäselvää tarkoittaako kielilinkki sivustolla käytettävää kieltä vai sarjakuvien kieltä.

Kun kielinä on vain suomi ja englanti, ei ole niinkään väliä, mitä merkistökoodausta käytetään. Jos palveluun halutaan kuitenkin lisätä muita kieliä, joissa esimerkiksi käytetään jotain muuta kuin latinalaista kirjoitusjärjestelmää, tähän voisi varautua käyttämällä Unicodea. Unicode mahdollistaa

kaikkien maailman kielten käyttämisen sivustolla. Vaikkei palveluun lisättäisikään Unicodea vaativia kieliversioita, voisi kuvitella, että sivustolle haluttaisiin tulevaisuudessa lisätä japaninkielistä tekstiä, koska Eepos-palvelu tarjoaa käyttäjille japanilaista sarjakuvaa eli mangaa.

Koska Eepos-sivuston ja mobiilisovelluksen englanninkieliset käännökset on tehnyt henkilö, jolle englanti on vieras kieli, käännösten laatu tulisi tarkistuttaa käyttäjällä, joka puhuu englantia äidinkielenään. Pienemmässä mittakaavassa huomiota tarvitsivat oman osallistumiseni päättyessä esimerkiksi sellaiset yksityiskohdat kuin päivämäärien esitysmuoto ja kielestä riippuvainen desimaalierotin, joka on suomessa pilkku, mutta englannissa piste.

4 Päätelmät

Kaikki kokeillut lokalisointitavat olivat toimivia ratkaisuja. Kaikki kolme kokeiltua lokalisointitapaa olivat perusperiaatteiltaan hyvin samanlaisia. Tekstitiedostoratkaisussa sivuston koodi ja sisältö on erotettu selvimmin toisistaan. Tekstitiedostojen käyttö hankaloitui huomattavasti, kun tiedostoja haluttiin muokata Eepos-sovelluksen kautta. Tekstienmuokkausmahdollisuus lisättiin sovellukseen vasta, kun lokalisoinnissa oli päädytty lopulliseen tekstitiedostoratkaisuun. Voi olla, että XML- tai PHP-tiedostojen muokkaus olisi yksinkertaisempaa.

Lokalisoinnin perusteellisella suunnittelulla olisi voitu keskittyä yhteen huolella valittuun lokalisointimenetelmään. Tavasta toiseen siirtymistä ei kuitenkaan koettu kovin työlääksi. Eri kokeilujen kautta saatiin selvä kuva eri ratkaisujen hyvistä ja huonoista puolista. Alussa käytössä olleet numeeriset tagit haittasivat koodin luettavuutta. Selväkieliset tagit paransivat luettavuutta huomattavasti, mutta niiden huonoksi puoleksi voitaisiin lukea hankaluus keksiä oikeasti kuvaavia ja yksilöiviä tunnuksia.

Mobiilisovelluksen lokalisointi suunniteltiin ja toteutettiin sen jälkeen, kun PHP-sivuston lokalisointiin oli valittu lopullinen toteutustapa. Samaa periaatetta pitää sisältö erillään koodista käytettiin myös mobiilisovelluksen lokalisoinnissa. Koska kyseessä oli eri ohjelmointikielet, mobiilisovelluksen lokalisointia ei voinut kuitenkaan tehdä suoraan PHP-sivuston mallin mukaan. Kielenvaihdon toteutus oli yllättävän hankalaa, eikä lopputulos ole ehkä paras mahdollinen ratkaisu, vaikka se toimiva onkin.

Lähteet

Davis, Mark 2008. Official Google Blog: Moving to Unicode 5.1.
[www-sivu] [viitattu 17.5.2009]
http://googleblog.blogspot.com/2008/05/moving-to-unicode-51.html

IANA — Root Zone Database.
[www-sivu] [viitattu 4.6.2009]
http://www.iana.org/domains/root/db/

ISO 639-2 Language Code List.
[www-sivu] [viitattu 4.6.2009]
http://www.loc.gov/standards/iso639-2/php/code_list.php

Nielsen, Jakob 2000. Designing Web Usability: The Practice of Simplicity. Indianapolis: New Riders Publishing

World Internet Stats.
[www-sivu] [viitattu 17.5.2009]
http://www.internetworldstats.com/stats7.htm