Tallinna Ülikool

Informaatika Instituut

Daniel Labo

Adobe Flex – raamistik Flash'il põhinevate internetirakenduste koostamiseks

Bakalaureusetöö

Juhendaja: Jaagup Kippar

Autor: ..... "....." ..... 2008. a.
Juhendaja: ..... "....." ..... 2008. a.
Instituudi direktor: ..... "....." ..... 2008. a.

Tallinn 2008

Sisukord

SISSEJUHATUS **3**

1. RIA-D EHK RIKKASISULISED VEEBIRAKENDUSED **5**

1.1. RIA-DE JÄRJEST SUURENEV MÕJU 5

1.2. TRADITSIOONILISEST VEEBIST RIA-NI 6

1.3. RIKKASISULISTE VEEBIRAKENDUSTE ÄRILINE ASPEKT 7

1.4. RIA NÄIDIS – *PICNIK.COM* 8

2. ADOBE FLEX'I RAAMISTIK **9**

2.1. LIHTSAIM MXML-KEELNE RAKENDUS 9

2.2. MXML-I JA ACTIONSCRIPTI SEOS 12

2.3. FLEX'I RAKENDUSTE ÜLESEHITUSEST 12

2.4. MXML-I MÄRGENDITE ATRIBUUDI *ID* KASUTAMINE 15

2.5. XML-I NIMERUUMIDE KASUTAMINE 16

2.6. ANDMETE SIDUMINE ELEMENTIDE VAHEL 18

2.7. VÄLISTE ANDMEALLIKATEGA ÜHENDUMINE 19

2.8. ANDMETE VALIDEERIMINE 22

2.9. ASTMELISED LAADILEHED 23

3. IDEID BAKALAUREUSETÖÖ EDASIARENDAMISEKS **26**

KOKKUVÕTE **27**

ANNOTATION **28**

KASUTATUD KIRJANDUS **30**

LISAD **31**

1. NÄIDISVAADE RIA-ST *PICNIK.COM* 31

2. NÄIDISVAADE ARENDATAVAST FLEX'I RAKENDUSEST 32

Sissejuhatus

Kahekümne esimese sajandi lävepakul, märtsis 2002 tutvustas Ameerika Ühendriikide suurfirma, tarkvaratootja Macromedia Inc. esmakordselt RIA mõistet. RIA *(Rich Internet Application)* või „rikkasisuline veebirakendus" on selline veebirakendus, mis sarnaneb oma kasutajaliidese, funktsionaalsuse ja pakutavate võimaluste poolest tavalisele töölauarakendusele – installeeritavale programmile.

Antud bakalaureusetöö on koostatud eesmärgiga tutvustada RIA kontseptsiooni toetudes ühele võimalusterohkemale arendusraamistikule Adobe Flex'ile.

Võib öelda, et Adobe Flex on Eestis uus ja väheliikmelise arendajaskonnaga tehnoloogia. Maailmas on küll tänaseks päevaks loodud rohkesti Flex'il põhinevaid veebilahendusi, kuid enamik neist pole siinmail kuigi tuntud. Teostades internetiotsingut Eesti veebilehtedel, leiab äärmiselt vähe viiteid meil loodud süsteemidele, mis baseeruksid Flex'il, vähe sellest – eestikeelset infot tehnoloogia kui sellise kohta ei leidu praktiliselt üldse. Kompenseerimaks seda puudujääki, on käesoleva üliõpilastöö autor kirjutanud lühikese sissejuhatuse kaasaegse Interneti võimalusi veelgi avardavatesse Flex'i RIA-desse, mille töökeskkonnaks on üldlevinud ja operatsioonisüsteemist sõltumatu Adobe Flash Player.

Bakalaureusetöö on jaotatud kolme peatükki, mis üheskoos peaksid andma lugejale lühiülevaate käsitletavast raamistikust ning praktilise võimaluse tutvuda Flex'i tehnoloogia abil toimuva arendustegevusega.

Esimeses osas on avatud RIA mõiste tähendus ja sisu kõrvutades vastavaid rakendusi tavapärastel veebitehnoloogiatel baseeruvatega ning põhjendatud nende efektiivsust äriprotsessi edendajana, samuti on toodud näide ühest olemasolevast RIA-st. Teises peatükis on näidatud, kuidas Adobe Flex'i raamistiku pakutavad vahendid aitavad rikkasisulisi internetirakendusi lihtsasti koostada – on antud ülevaade olemasolevatest tarkvarakomponentidest, nende olulisematest kasutuspõhimõtetest. Viimases kolmandas peatükis on esitatud ideed antud lõputöö edasiarendamiseks tulevikus, põhilisemad nendest on RIA-sid tutvustava õppematerjali ja näitevaramu koostamine, aga samuti lihtsa, ent praktilise näiterakenduse loomine, mis demonstreeriks, kuivõrd kiiresti võimaldab Flex ehitada väga interaktiivseid ning sisu ja eesmärkide poolest suuresti erinevaid RIA-sid; koostamisel olev demorakendus näitab,

kuidas toimub Flex’is ühendumine serveripoolsete programmide ning andmeallikate, aga ka kolmanda osapoole pakutavate veebiteenustega.

Bakalaureusetöö autor tänab siinkohal juhendajat ja kõiki teisi, kes on käesoleva töö valmimisele kaasa aidanud.

1. RIA-d ehk rikkasisulised veebirakendused

Tarkvaraarenduslike lahenduste loomeprotsessis võib rääkida kahest omamoodi vastanduvast soovist või eesmärgist: vajadusest rikkalike võimalustega ning atraktiivsete kasutajaliideste järele, et suurendada sihtgrupi rahulolu, aga teisest küljest loomulikust nõudest vähendada maksimaalselt kulusid säilitades seejuures kasumlikkust. On ilmne, et innovatiivsed ja keerulisemad, aga kindlasti efektiivsemad lahendused maksavad rohkem.

Võib öelda, et turgu on asunud vallutama uut tüüpi infoesitusmeetodeid kasutavad rikkasisulised veebirakendused, mis on kasutajakogemuse poolest sarnased tüüpilistele klient-server-lahendustele, aga mille graafilised liidesed on kõrgelt interaktiivsed, suure jõudlusega ning visuaalselt esteetilised. RIA-de populaarsuse kasv jätkub, kuna kasutajate nõudmised RIA-de pakutavate võimaluste järele on suurenemas. Rakendustele soovitakse ligipääsu järjest enam sõltumatult füüsilisest asukohast või kasutatava arvuti konfiguratsioonist, juurdepääsu eelduseks on aga jätkuvalt võrguühenduse olemasolu, aga mitte operatsioonisüsteem või arvutisse installeeritud programmistik. Kuigi käesoleval ajajärgul räägitakse järjest rohkem ka niisugustest süsteemidest, mis vajavad oma tööks võrguühendust vaid kohati, mitte kogu aeg, võib RIA-sid käsitledes praegusel momendil tõdeda, et pideva aktiivse internetiühenduse olemasolu on igal juhul niisuguste rakenduste tulemusliku toimimise eelduseks.

RIA-sid on praegu ehitatud enamasti nelja tüüpi: Adobe Flash Player'il töötavad (Flash ja Flex), HTML-i ning Javascripti põhised AJAX-il baseeruvad *(Asynchronous Javascript and XML)*, Java *applet'*eid või ActiveX programme kasutavad ning Javal või .NET keeltel põhinevad erilahendused; viimase alla võiks paigutada ka Microsoft Silverlight'i, mis sarnaneb suuresti animeeritud Flash'ile.

1.1. RIA-de järjest suurenev mõju

Veebirakenduste sihtgrupp nõuab tänasel päeval oma kasutajakogemusest aina enam. Ammu on möödas staatiliste veebilehekülgede ajastu, mida iseloomustas interaktiivsuse vähesus. Tänapäevatrendidega kaasas käivad internetikasutajad ootavad, et veebirakendused töötaksid nagu töölauaprogrammid ehk reageeriksid nupuvajutustele koheselt. RIA-d pakuvad taolist funktsionaalsust ja veel enam – kombineerides reaalajas

toimuvaid mitmepoolseid suhtlusvõimalusi uuenduslike graafilis-visuaalsete mallidega, saavutatakse efekt, mille puhul mõjutatakse kasutajaid eelistama rakendusi, millele saab ligi olenemata füüsilisest asukohast võrguruumis ning mille pakutav töökeskkond on platvormist sõltumatult alati ühetaoline ning kasutaja poolt isikupärastatud.

Üldiselt saab väita, et RIA-d pakuvad tehnoloogiliselt keerukaid, aga uuenduslikke ja kaasahaaravaid võimalusi kasutajaga suhtlemiseks, töölauarakendustele iseloomulikke graafilisi komponente sisu esitamiseks ning praktilisi ja efektiivseid meetodeid andmeallikatega sidumiseks.

1.2. Traditsioonilisest veebist RIA-ni

Hoolimata veebi arhitektuuri puhul toimunud arengutest, ei suuda veebilehitseja pakutav harjumuspärane kasutajaliides ning pruugitavad tehnoloogiad nagu HTTP ja HTML seotuna mõninga skriptimise ja kujundamisega kompenseerida esilekerkinud puudujääke. Veeb oma praeguses vormis ei ole sobiv aseaine olemasolevatele klient-server-lahendustele, mis on oma olemuselt palju mitmekesisemad, jõudluse poolest efektiivsemad ja toimivad praktiliselt reaalajas. Kokkuvõtlikult saab tuua välja, et traditsiooniline veeb kätkeb endas järgmisi puudusi.

- Piirangud kasutajaga suhtlemisel. HTML oli algupäraselt mõeldud esitamaks staatilisi ning tekstimahukaid dokumente, mida iseloomustas vähene vajadus interaktiivsuse järele. Seetõttu puudub puhtal HTML-il põhineval esituskihil igasugune võimekus töötada kui lihtsaimgi klient-server-rakendus; igal juhul läheb vaja skriptimist. HTML ei võimalda kasutada eraldi aknaid infoesituseks, puudub „tiri ja aseta” *(drag and drop)* funktsionaalsus, ei ole võimalik suurema programmeerimiseta kasutada sorteeritavaid ning muul viisil kohandatavaid andmetabeleid, puudub võimalus kasutada dünaamiliste ajas muutuvate andmehulkade visualiseerimiseks graafikuid jms. Seega saab kannatada kasutaja produktiivsus.
- Vähene jõudlus. Veeb on disainitud pidades silmas seisunditeta interaktsiooni, see tähendab, et kasutajad suhtlevad veebisaidiga põhimõttel „klõpsa-oota-taaslae”. Niisugune meetod eeldab, et igasuguse muutuse esilekutsumiseks kasutajaliideses või mis tahes uue infopäringu läbiviimiseks tuleb taaslaadida terve veebikülg. On vaja teha mitmeid

ringkäike veebilehitseja ning -serveri vahel, millest tuleneb kokkuvõttes kasutaja produktiivsuse alanemine, liigne koormus serverile ja möödapääsmatu vajadus kiire ning usaldusväärse võrguühenduse järele, mis pole igal pool võimalik.

Seega võib järeldada, et piirangutest kasutajaga suhtlemisel ning esilekerkivatest jõudlusprobleemidest saab alguse harilike veebilehtede vähene jätkusuutlikkus, kuna ei ole võimalik efektiivselt realiseerida sündmuspõhiseid reaalajas aset leidvaid interaktsioone.

Neid puudujääke, mis on seotud veebi algse ülesehitusega, püüavad kompenseerida RIA-d, kus on võimalik mitmeid osapooli ühendav infovahetus – sündmuspõhised asünkroonsed meetodid suhtlemiseks.

1.3. Rikkasisuliste veebirakenduste äriline aspekt

RIA-d püüavad veebisaitide omanike tähelepanu, kuna kasutajatele meeldivad need – nad pakuvad niisuguseid võimalusi interaktsiooniks, mida HTML-il põhinev rakendus ei suuda. Kasutajate töö RIA-del on tulemuslik. Sellegipoolest kaaluvad paljud veebiäride omanikud, kas eelistada RIA-dele üleminekut, kuna mitmed nendest, kes juba rikkasisuliste veebirakenduste kasuks otsustanud on, näevad vaeva, et tõestada RIA-de tegelikku väärtust või nende kasumlikkust. Konservatiivsetele uurimismudelitele tuginedes saab siiski veenduda, et hästi ülesehitatud RIA-d õigustavad nende arendamisse suunatud investeeringuid ning vastavad tehnoloogiad võimaldavad toota piisavalt hämmastavaid lahendusi – seega loob nende kasutamine ettevõtte jaoks eeldusi olemaks edukas ka tulevikus. *[Forrester, 2007]*

Flex'il ning Ajax'il baseeruvate veebirakenduste populaarsuse kasv toob muutusi nendesse viisidesse, kuidas ettevõtted *online*'is äri teevad. Miks on RIA-d veebilehtede omanike silmis nii atraktiivsed? Esiteks seetõttu, et neid teatakse ja need meeldivad inimestele. 52% veebikasutajatest on proovinud niisuguseid kõrgelt interaktiivseid rakendusi nagu Google Maps või Zillow.com. Veelgi tähtsamat rolli mängib tõik, et suur enamik RIA-dega kokku puutunud inimestest väidavad, et need lahendused täiustavad nende veebikasutamiskogemust oluliselt. Teiseks saab tõdeda, et RIA-d aitavad kaasa äriliste tulemuste saavutamisele. Ettevõtted, kes on mõõtnud RIA-de mõju oma tegevusele, väidavad, et need vastavad nõudmistele ning isegi ületavad ootusi. Kui valdkonna entusiastlikud pioneerid RIA-de algusaegadel need esmakordselt kasutusele

võtsid ja tuntumaks tegid, on järgnenud ka mitmed suuremad firmad, kes otsivad rikkasisulistest internetirakendustest tulevikuväljavaateid ka enda tegevusele. *[Forrester, 2007]*

1.4. RIA näidis – *Picnik.com*

Demonstreerimaks Adobe Flash'i platvormil töötavat rikkasisulist veebirakendust, märgib autor üht suhteliselt innovatiivset lahendust, mis kujutab endast brauseripõhist fototöötlustarkvara. Nimeks on tal Picnik ja on kättesaadav aadressilt *http://www.picnik.com*. Tegu on niisuguse süsteemiga, mis laseb kasutajatel laadida üles oma fotosid ja töödelda neid täielikult veebipõhiselt. Süsteemis on olemas enamjaolt kõik tavakasutajal vajaminevad funktsioonid piltide redigeerimiseks alates lihtsast lõikamisest, teravdamisest ja värvitasakaalu muutmisest kuni kõikvõimalike efektide ning teksti lisamise, punasilmsuse vähendamise ja raamistamiseni. Ühe siinseski töös kasutatud illustratsiooni kohandamist Picnik'u abil kujutab ekraanipilt lisas 1.

2. Adobe Flex'i raamistik

Andmaks ülevaadet RIA-de praktilisest koostamisest, on käesolevas peatükis kokku võetud, milliseid võimalusi pakub rikkasisuliste veebirakenduste koostamiseks Macromedia loodud arendusplatvorm, millel nüüd nimeks Adobe Flex.

Adobe Flex (töö kirjutamise hetkel aprillis 2008 uusima versiooniga nr. 3) on RIA-de koostamiseks loodud tehnoloogia või raamistik, milles asetleidva arendustegevuse lõplikuks tulemiks on kompileeritud failina standardsel Flash Player'il töötav *swf*-animatsioon. Flex'is arendamiseks vajaminev tarkvara Adobe Flex SDK *(Software Development Kit)* on tänaseks päevaks avatud lähtekoodiga ja kõigile tasuta kättesaadav. Selle tarkvara puhul on tegemist MXML-i/Actionscripti kompilaatoriga. Käesolevas bakalaureusetöös on näiterakenduste kompileerimiseks kasutatud justnimelt seda vabatarkvaralist SDK-d. Flex'is arendamise produktiivsuse tõstmiseks on küll võimalik kasutada Adobe vastavat integreeritud arenduskeskkonda (Flex Builder), kuid selle puhul on tegemist kommertsprogrammiga, mis ei ole Flex'i rakenduste kompileerimiseks hädavajalik.

Adobe Flex 3 tarkvara (Flex Builder ja SDK) on allalaetav aadressilt *http://www.adobe.com/cfusion/entitlement/index.cfm?e=flex3email*. Flex'i rakenduste käivitamiseks on vajalik Adobe Flash Player alates versioonist 9.

Antud peatükis ongi esitatud lühike ülevaade sellest, kuidas kasutada MXML-i ja Actionscripti Flex'i rakenduste koostamiseks ning kuidas saab lähtekoodi sisaldavaid faile Flash'i *swf*-animatsioonideks kompileerida.

2.1. Lihtsaim MXML-keelne rakendus

MXML on XML-vormingus *(Extensible Markup Language)* keel, mida kasutatakse Flex'is kasutajaliideste disainimiseks. Ühtlasi on võimalik MXML-i abil defineerida ka Flex'i rakenduste mittevisuaalseid aspekte nagu näiteks serveripoolseid andmeallikaid ning infovahetuse loogikat sama rakenduse kasutajaliidese erinevate komponentide vahel.

Nii nagu ka HTML *(Hypertext Markup Language)*, võimaldab MXML kasutajaliidese struktuuri deklaratiivselt kirjeldada. Need keeled on nii mõneski mõttes sarnased, sellegipoolest on MXML paremini või enam struktureeritud ning tema

märgendisüsteem on palju mitmekesisem. Näiteks on MXML-is olemas vahendid kirjeldamaks sorteeritavaid Exceli-laadseid andmetabeleid, puustruktuure, vahekaartidega *(tabs)* või akordioni-laadseid navigatsioonisüsteeme, erisuguseid menüüsid, aga samuti mittevisuaalseid komponente, mis näiteks võimaldavad luua ühendusi erinevate veebiteenustega. Lisaks on võimalik defineerida kasutajaliidestes mitmesuguste animatsioonide kasutamist, aga seegi pole veel kõik; nagu XML-ile kui laiendatavale märgendikeelele omane, saab nimetatud funktsionaalsust edasi arendada luues juurde kohandatud ja isikupärastatud elemente.

Üks suurimaid erinevusi MXML-i ja HTML-i vahel on see, et MXML-is kirjutatud rakendused kompileeritakse Flash Player'i taasesitatavasse *swf*-vormingusse, mis on oma olemuselt enamasti oluliselt dünaamilisem ja väljanägemiselt esteetilisem kui tavapärane HTML-dokument oma kõikvõimalike laiendustega.

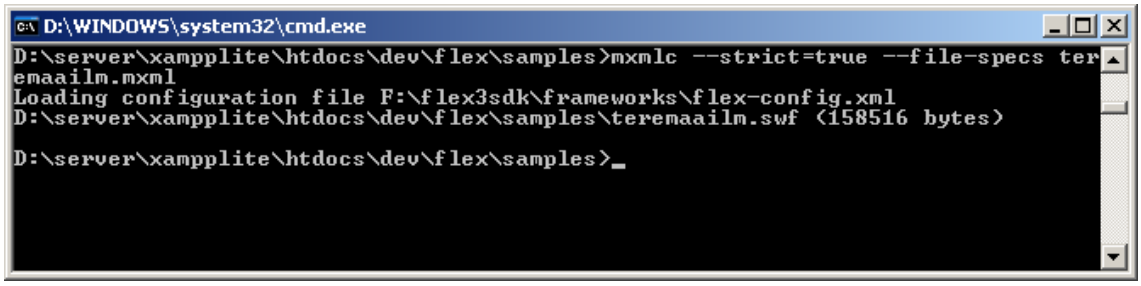
MXML-rakenduste lähtekood on võimalik kirjutada nii ühteainsasse faili kui ka jagada koodi mitmete failide vahel.

Kuna *mxml*-failide puhul on tegemist täiesti tavaliste *xml*-failidega, on nende koostamiseks võimalik kasutada suvalist lihtsat tekstiredaktorit, näiteks Windows Notepad'i. Järgmisena koostamegi lihtsaima MXML-rakenduse, niinimetatud *„hello world"* programmi.

```
<?xml version="1.0"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml">
  <mx:Panel title="Minu esimene rakendus">
    <mx:Label text="Tere maailm!" />
  </mx:Panel>
</mx:Application>
```

Antud minirakendus sisaldabki üksnes <*mx:Application*> märgendit ning kahte alamelementi – <*mx:Panel*> ja <*mx:Label*>. Esimene nendest märgenditest defineerib rakenduse tüübi, see on MXML-keelsetes programmides alati juurelemendiks. Teine element *Panel* fikseerib paneeli, millel on pealkiri ja ala sisu jaoks. Kolmas märgend *Label* on lihtne konteiner teksti hoidmiseks.

Pärast MXML-i kirjutamist tuleb lähtekood *swf*-animatsiooniks kompileerida; see toimub Flex'i SDK-sse kuuluva MXML-i/Actionscripti kompilaatoriga *mxmlc*, mille kasutamine toimub käsurealt viisil, nagu on näidatud joonisel 1.



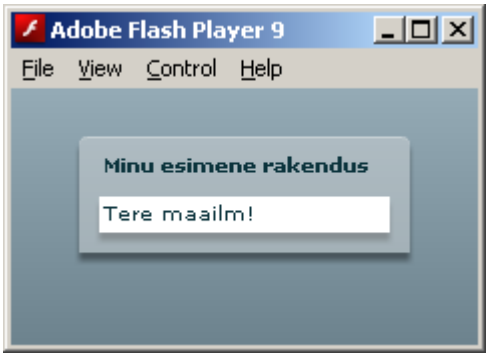
Joonis 1. mxml-faili kompileerimine swf-animatsiooniks

Siin on MXML-i kompileerimiseks kasutatud järgmist käsku.

```
mxmlc --strict=true --file-specs teremaailm.mxml
```

mxmlc on kompilaatorprogramm, millele on edastatud kaks võtit – *strict* ja *file-specs*. Need võtmed näitavad vastavalt, kas me kasutame kompileerimisel ranget meetodit (mis ootab meie kirjutatud koodilt pisut enam) ning millist *mxml*-faili parajasti kompileerime. Antud töös me keerukamaid viise MXML-i kompileerimiseks ei vaatle; nimetatud käsk on optimaalne kompileerimaks väga mitmesuguste omadustega rakendusi.

Kompileerimise tulemusena salvestatakse Flash Player'is esitatav *swf*-fail. Seda kujutab joonis 2.



Joonis 2. „Hello world" programmi SWF-animatsioon

Analoogiliselt HTML-iga saab erinevate atribuutide abil seadistada ka MXML-i elemente, misjärel muutub nende esitatava komponendi väljanägemine või käitumuslik aspekt.

2.2. MXML-i ja Actionscripti seos

Actionscript, mida kasutatakse Flex'i (ka Flash'i) rakenduste programmilise loogika kirjeldamiseks, on objektorienteeritud keel. MXML-keele ja Actionscripti klasside vahel on seos. Adobe ehitas Flex'i kui Actionscripti klasside kataloogi või raamatukogu *(library)*. See klasside süsteem sisaldab väga erisuguseid vahendeid rakenduse ülesehitamiseks – erinevaid konteinereid ja kontroller-komponente, klasse andmeallikatega ühendamiseks ning kõigeks muuks, mida kõnealune raamistik RIA-de koostamiseks võimaldab.

MXML-märgenditele vastavad omad Actionscripti klassid või nende klasside atribuudid *(properties)*. Flex'i kompilaator vaatab läbi arendaja koostatud MXML-i ning koostab selle põhjal *swf*-faili, mis sisaldab vastavaid Actionscripti objekte. Näiteks on Flex'is olemas nupu *(button)* klass, mis defineerib nuppu kujutava kontroller-komponendi. Järgnevalt on esitatud nupu definitsioon.

```
<mx:Button label="Saada ära" />
```

Kui defineerida MXML-is mingi komponent, siis tegelikult tekitatakse tema klassile vastav objekt. Antud rida MXML-is kujutab nupu objekti loomist, millele on omistatud atribuut *label* (tekst, mis asetatakse nupule) ning väärtustatud viimane sõnega „saada ära".

Actionscripti igat klassi ning selle klassiga vastavuses olevat MXML-i märgendit nimetatakse kokkuleppeliselt sarnasel viisil – kasutades nimede märkimisel järgmist analoogiat: klasside nimed algavad suurte tähtedega ja suurtähed eraldavad klassi nimes ka sõnu, mis sellesse kuuluda võivad; näiteks on olemas Actionscripti klass nimega *TextInput* ja sellele vastab sama nimega MXML-i märgend *(TextInput)*. Igale MXML-märgendi atribuudile vastab tema objekti atribuut, aga võib vastata ka objektile rakendatud stiil või mõni sündmuse kuular *(event listener)*.

2.3. Flex'i rakenduste ülesehitusest

MXML-i rakenduse võib koostada ühes või mitmes lähtekoodifailis. Tavaliselt luuakse põhifail, mis sisaldab <*mx:Application*> märgendit ning lisatakse sellele failile vajaduse järgi teisi, mis võivad olla kirjutatud nii MXML-is kui ka Actionscriptis või on kombineeritud neist kahest keelest.

Harilikult jagatakse kogu Flex'i rakendus funktsionaalselt terviklikesse osadesse või moodulitesse, millest igaüks vastutab eraldiseisva tegevuse läbiviimise eest. Kui rakendus niimoodi moodulitesse grupeerida, on sellest palju kasu; mõned eelised on toodud järgnevalt.

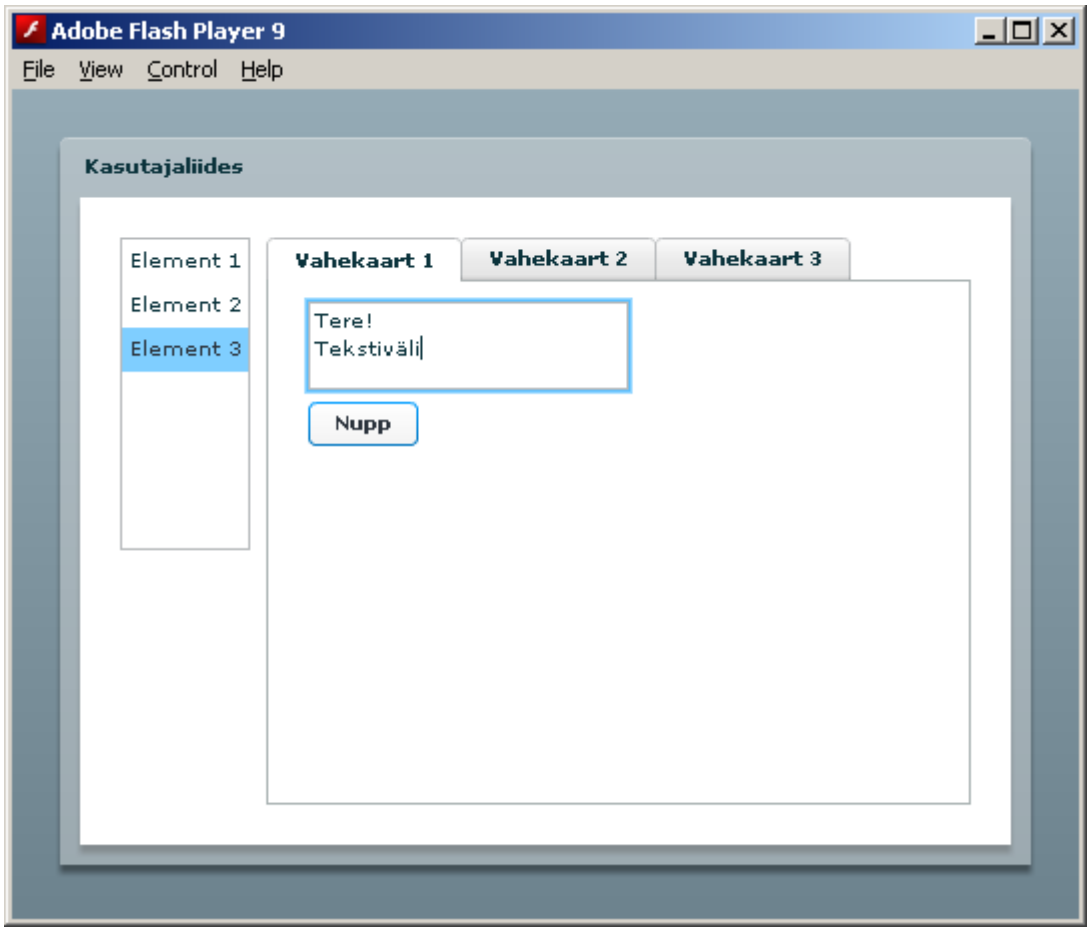
- Arendustegevuse hõlbustamine. Erinevad arendajad või arendusmeeskonnad saavad töötada erinevate moodulitega, mis on teineteisest sõltumatud.
- Moodulite taaskasutatavus. Erinevate rakenduste puhul on võimalik kasutada samu mooduleid, mis vähendab vajadust sarnast koodi mitu korda kirjutada – see säästab aega ja vaeva.
- Rakenduste hõlpsam hooldatavus. Kui komponendid on üksteisest isoleeritud, on ka vigade leidmine ja parandamine kogu rakenduses lihtsam ning kogu süsteemi on kokkuvõttes kergem hallata.

MXML-is asetleidev arendustegevus on analoogiline iteratiivne protsess, nagu HTML-i, JSP *(Java Server Pages)* või näiteks ASP *(Active Server Pages)* puhul. Protsess koosneb lähtekoodi kirjutamisest, koodi kompileerimisest (võib piirduda ühe-kahe klahvivajutusega), tulemuse viseerimisest veebibrauseris või Flash Player'is ning toodud sammude kordamisest.

Flex'i rakenduse kasutajaliides koostatakse enamasti konteiner- ning kontrollerkomponentidest. Konteinerid sisaldavad sarnaselt HTML-iga teisi konteinereid või kontroller-elemente. Lihtsaks konteineriks on *VBox*, mis järjestab oma alamelemendid vertikaalses suunas. Elementaarset kasutajaliidest kujutavas näites joonisel 3 on demonstreeritud ka paari kontroller-elementi – nimekirja, tekstiala ja nuppu. Kõigepealt toome vastavat kasutajaliidest defineeriva koodi.

```
    <?xml version="1.0"?>
    <mx:Application xmlns:mx="http://www.adobe.com/2006/mxml">
      <mx:Panel title="Kasutajaliides" paddingTop="20"
paddingBottom="20" paddingLeft="20" paddingRight="20">
        <mx:HBox>
          <mx:List>
            <mx:dataProvider><mx:Array>
                <mx:String>Element 1</mx:String>
                ...
            </mx:Array></mx:dataProvider>
          </mx:List>
```

```
          <mx:TabNavigator>
            <mx:VBox paddingLeft="20" label="Vahekaart 1" width="350"
height="250">
              <mx:TextArea text="Tere!" />
              <mx:Button label="Nupp" />
            </mx:VBox>
            <mx:VBox paddingLeft="20" label="Vahekaart 2" width="350"
height="250">
            </mx:VBox>
            <mx:VBox paddingLeft="20" label="Vahekaart 3" width="350"
height="250">
            </mx:VBox>
          </mx:TabNavigator>
        </mx:HBox>
      </mx:Panel>
    </mx:Application>
```



Joonis 3. Lihtne kasutajaliides nimekirjast ja vahekaart-navigaatorist

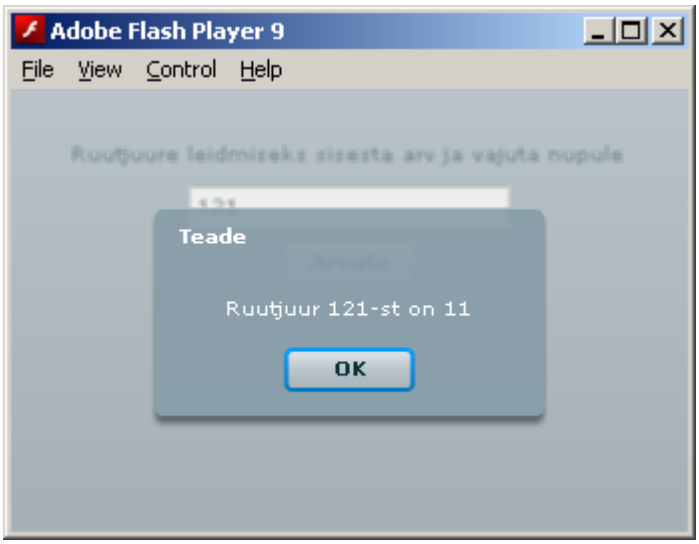
Antud näites on vasakpoolne nimekiri ning vahekaart-navigaator paigutatud horisontaalselt kõrvuti, sest nad mõlemad asuvad *HBox* konteineris. Tekstiväli ning nupuke paiknevad üksteise järel vertikaalselt, kuna asuvad *VBox* konteineris. Vastavalt kombineerides saab paigutada kõikvõimalikke kasutajaliidese elemente, et ehitada üles suhteliselt keerukaid struktuure, mis peaksid rahuldama ka mahukamatele rakendustele esitatud nõudmisi.

2.4. MXML-i märgendite atribuudi *id* kasutamine

Mõningate eranditega on igal MXML-i märgendil atribuut *id*, mis peab olema unikaalne kogu MXML-faili lõikes. Selle atribuudi kaudu on võimalik vastavatele elementidele Actionscriptis, aga ka teiste elementide juurest viidata. Järgmiseks praktiliseks näiteks on rakendus, mis kasutab ruutjuure arvutamiseks Actionscripti funktsiooni ning tulemuse väljastamiseks teatist *(alert)*. Actionscript pääseb sisestatud numbri juurde tekstivälja atribuudi *id* kaudu. Vastav ekraanipilt on kujutatud joonisel 4.

```
    <?xml version="1.0"?>
    <mx:Application xmlns:mx="http://www.adobe.com/2006/mxml">
      <mx:Script>
        <![CDATA[
        import mx.controls.Alert;
        public function juuri():void {
          var arv:int = parseInt(juuritav.text);
          var teade:String;
          if (arv >= 0) {
            teade = "Ruutjuur " + arv.toString() + "-st on " +
Math.sqrt(arv).toString();
          }
          else {
            teade = "Sisesta positiivne arv!";
          }
          Alert.show(teade, "Teade");
        }
        ]]>
      </mx:Script>
      <mx:Text text="Ruutjuure leidmiseks sisesta arv ja vajuta nupule"
/>
      <mx:TextInput id="juuritav" />
      <mx:Button label="Arvuta" click="juuri()" />
```

```
</mx:Application>
```



Joonis 4. Ruutjuurekalkulaator. Actionscripti tarvitamise näidis

Antud rakenduse puhul genereerib MXML-i kompilaator avaliku muutuja nimega *arv*, mis viitab temaga seotud tekstiväljale identifikaatoriga *juuritav*. Selle tekstivälja atribuutidele saab nüüd muutuja kaudu ligi igast Actionscripti klassist või skriptiplokist.

Samuti demonstreerib käesolev näide, kuidas Flex'i rakendusi juhitakse tihti kasutajapoolseid tegevusi jälgides – selleks otstarbeks ongi loodud nn. sündmuskuularid, mis võimaldavad reageerida näiteks hiireklõpsudele või klahvivajutustele. Meie näites vajutab kasutaja nupule ning talle näidatakse juurimistehte tulemust. Hiireklõpsul käivitatakse Actionscripti funktsioon, mis on üles ehitatud kombineerides vajalikke meetodeid sisestatud arvu väärtuse positiivsuse kontrollimiseks ning juurimistehte sooritamiseks, aga ka tulemuse väljastamiseks rakenduses teatisena.

2.5. XML-i nimeruumide kasutamine

XML-dokumentides on märgendid tavaliselt seotud nimeruumidega. Nimeruumid võimaldavad samas dokumendis kasutada enam kui ühte komplekti märgendeid. Kasutatavatele nimeruumidele viitamine toimub MXML-is elemendi *Application* atribuudi *xmlns* kaudu. Konkreetsesse nimeruumi kuuluvaid märgendeid identifitseeritakse eesliitega, näiteks vaikimisi kasutatava MXML-i nimeruumi eesliiteks on *mx*, mis kajastub ka kõigi elementide puhul, mida käesolevas töös on

demonstreeritud. Erinevate nimeruumide kasutamine loob eelduse isikupärastatud komponentide loomiseks, mis võivad olla jaotatud paljude erinevate MXML-rakenduste vahel. Arendajal tuleb kirjutada talle vajalik komponent, ütleme, et selleks on Eesti kõigi linnade nimekiri, mille definitsioon koosneb tegelikult mitmete MXML-märgendite kombinatsioonist ja andmetest; kui seda nimekirja soovitakse kasutada mitmetes MXML-rakendustes, on viimastes võimalik viidata antud nimekirjale väga lühidalt, s.t lisada vastav nimekiri lühikese deklaratsiooniga – nt. järgmisel viisil.

```
<?xml version="1.0"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
xmlns:komponendid="komponendid.*">
  <komponendid:Linnad />
</mx:Application>
```

See on põhirakendus. Salvestame ta näiteks nimega „eestilinnad.mxml". Järgmisena koostame komponendi, mis sisaldab vajaminevat linnade nimekirja.

```
<?xml version="1.0"?>
<mx:ComboBox xmlns:mx="http://www.adobe.com/2006/mxml">
  <mx:dataProvider>
    <mx:String>Abja-Paluoja</mx:String>
    <mx:String>Antsla</mx:String>
    <mx:String>Elva</mx:String>
    <mx:String>Haapsalu</mx:String>
    <mx:String>...</mx:String>
  </mx:dataProvider>
</mx:ComboBox>
```

See on meie isikupärastatud komponent – antud juhul *combobox*, mis on sarnane element HTML-i ripploendiga. See komponent tuleb salvestada kataloogi „komponendid" nimega „Linnad.mxml".

Nagu lähtekoodist nähtub, on põhirakenduses defineeritud nimeruum *komponendid*, mille väärtuseks on kataloogitee vajalike komponentide juurde, antud juhul *komponendid.**. See kirjutusviis ütleb MXML-i kompilaatorile, et failis „eestilinnad.mxml" saab kasutada komponente, mis asuvad kataloogis „komponendid". Meie näites lisatakse põhirakendusse nimekiri linnadest, mis on defineeritud antud kataloogi salvestatud failis „Linnad.mxml". Tuleb panna tähele, et komponendi

failinimi vastab nimele, millega teda põhirakenduses kasutatakse. Joonisel 5 on toodud illustratsioon põhirakendusest.

Joonis 5. Eraldi nimeruumi ja komponendi kasutamine MXML-rakenduses

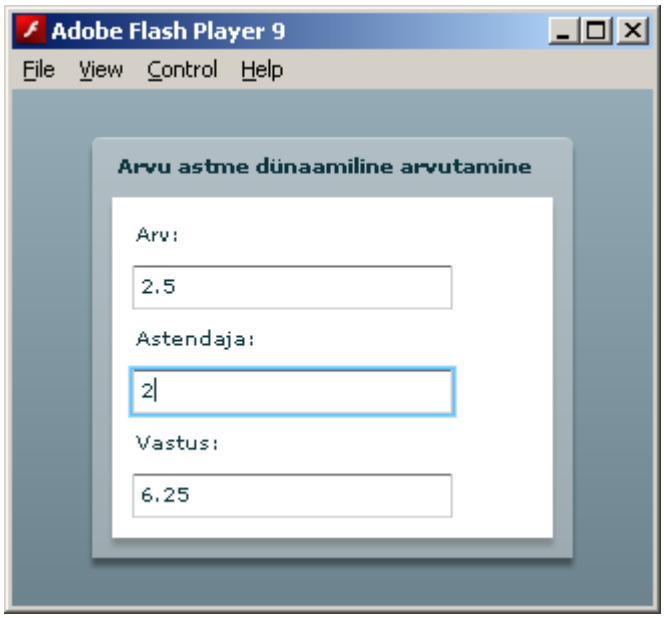
Sellist mooduliteks jagamise meetodit tarvitades saame kirjutada kui tahes palju komponente, mis on nimeruumide abil ühtsetel alustel kasutatavad kõigis rakendustes, kus nende järele vajadus tekib.

2.6. Andmete sidumine elementide vahel

Flex'is on väga lihtsalt realiseeritud võimalus siduda erinevate elementide vahel mitmesuguseid andmeid, näiteks kasutaja sisestatud infot või märgendite atribuutide väärtusi. Selleks otstarbeks kasutatakse atribuudi väärtuse defineerimisel loogelisi sulge viitega mõnele andmeallikale, näiteks mõnda sisestusvälja kirjutatud tekstile. Järgmises näites loetakse kahte sisestusvälja kirjutatud arvud, teostatakse nendega astendamistehe ning kuvatakse vastus vastavalt hetkel olemasolevatele lähteandmetele.

```
    <?xml version="1.0"?>
    <mx:Application xmlns:mx="http://www.adobe.com/2006/mxml">
      <mx:Panel paddingTop="10" paddingBottom="10" paddingLeft="10"
paddingRight="10" title="Arvu astme dünaamiline arvutamine">
        <mx:Label text="Arv:" />
        <mx:TextInput id="arv" />
        <mx:Label text="Astendaja:" />
        <mx:TextInput id="astendaja" />
        <mx:Label text="Vastus:" />
        <mx:TextInput text="{Math.pow(parseFloat(arv.text),
parseFloat(astendaja.text)).toString()}" />
      </mx:Panel>
    </mx:Application>
```

Nagu antud lähtekoodist selgub, võib elemendi atribuudi väärtustamisel kasutada koheselt Actionscripti meetodeid ilma eraldi funktsioone kirjutamata. Rakenduse ekraanipilt on esitatud joonisel 6.



Joonis 6. Vastuse arvutamisel võetakse sisendandmed teistelt tekstiväljadelt

2.7. Väliste andmeallikatega ühendumine

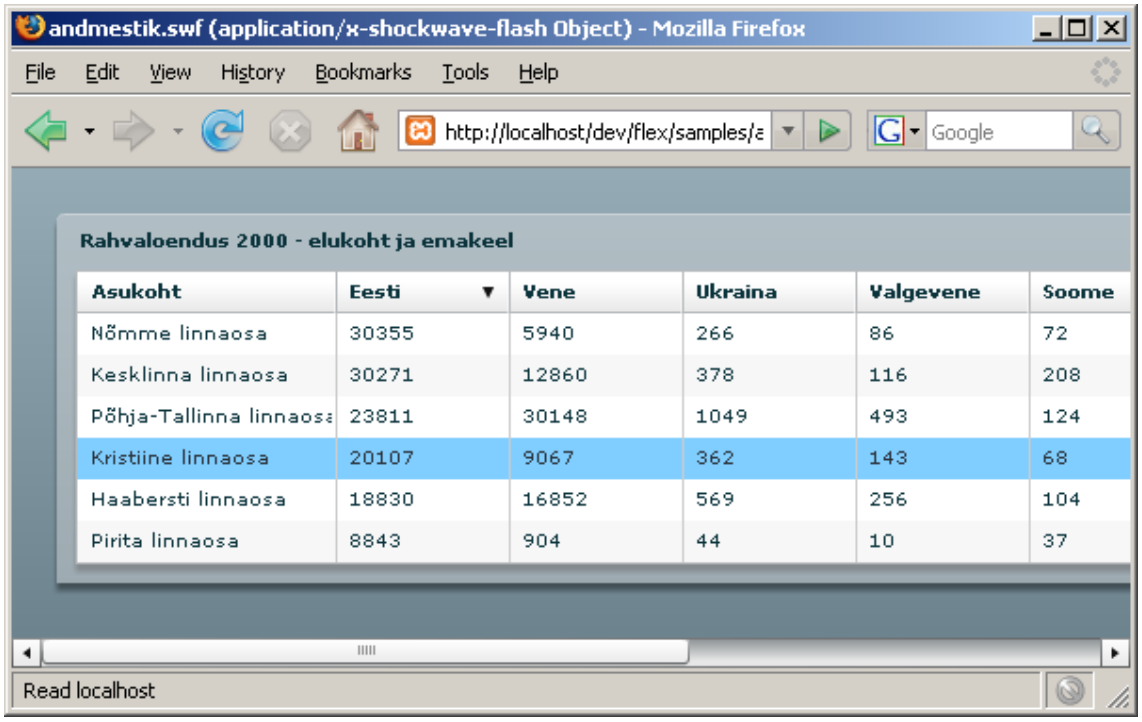
Väga oluliseks tuleb pidada Flex'i rakenduste võimet ühenduda eemalolevate andmeallikatega, mis teeb võimalikuks asünkroonse infovahetuse kasutaja veebibrauseris töötava rakenduse ning serveri vahel. Niisuguse ühenduse loomine võimaldab serverist andmeid lugeda, aga neid ka sinna saata. Sellise funktsionaalsuse realiseerimiseks on Flex'is kasutusel RPC *(Remote Procedure Call)* teenused. Flex on disainitud olemaks suutlik ühenduma mitut tüüpi RPC teenustega. Näiteks saab ühenduda SOAP-il *(Simple Object Access Protocol)* põhinevate veebiteenustega, lugeda HTTP protokolli kaudu harilikke XML-dokumente või suhelda serveripoolsete skriptidega, mille ülesandeks on väljastada või koguda andmeid. Järgmises näites on demonstreeritud Flex'i *HTTPService* komponendi pakutavaid võimalusi XML-vormingus andmete lugemiseks veebiserveris paiknevast failist.

```
    <?xml version="1.0"?>
    <mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
creationComplete="loe_xml()">
      <mx:Script>
        <![CDATA[
```

```
        public function loe_xml():void {
          loe_fail.send();
        }
        ]]>
      </mx:Script>
      <mx:HTTPService id="loe_fail" url="emakeel.xml"
resultFormat="object" />
      <mx:Panel title="Rahvaloendus 2000 - elukoht ja emakeel">
        <mx:DataGrid width="100%"
dataProvider="{loe_fail.lastResult.rida}">
          <mx:columns>
            <mx:DataGridColumn dataField="lahter0" headerText="Asukoht"
/>
            <mx:DataGridColumn dataField="lahter1" headerText="Eesti"
/>
            <mx:DataGridColumn dataField="lahter2" headerText="Vene" />
            <mx:DataGridColumn dataField="lahter3" headerText="Ukraina"
/>
            <mx:DataGridColumn dataField="lahter4"
headerText="Valgevene" />
            <mx:DataGridColumn dataField="lahter5" headerText="Soome"
/>
            <mx:DataGridColumn dataField="lahter6" headerText="Läti" />
            <mx:DataGridColumn dataField="lahter7" headerText="Muu
keel" />
            <mx:DataGridColumn dataField="lahter8" headerText="Keel
teadmata" />
          </mx:columns>
        </mx:DataGrid>
      </mx:Panel>
    </mx:Application>
```

Selle programmi ülesandeks on veebiserverisse pöördudes lugeda seal paikneva XML-faili sisu ning kuvada see sorteeritavas Exceli-laadses andmetabelis *(datagrid)*. Serveris paiknev XML-struktuur võib olla dünaamiliselt genereeritud mõne andmebaasirakenduse poolt; XML on sedavõrd universaalne vorming, et väga tihti kasutataksegi just seda andmekogumite esitamiseks. Flex'is on olemas väga mitmekülgsed ja mugavad vahendid XML-kujul andmetega manipuleerimiseks. Joonisel 7 toodud ekraanipildil on kujutatud antud koodile vastav rakendus, sel korral

peame teda domonstreerima veebiserveri kaudu, kuna andmete sisselugemiseks on vajalik HTTP-päringu sooritamine. Samuti on vajalik märkida, et Flex'i rakendus suudab ilma keerukama spetsiaalseadistuseta ühenduda üksnes sellesse veebiserverisse, kust rakendust ennast kliendile serveeritakse. Siin on tegemist turvalisusküsimusega.



Joonis 7. Sorteeritava andmetabeli kuvamine XML-faili alusel

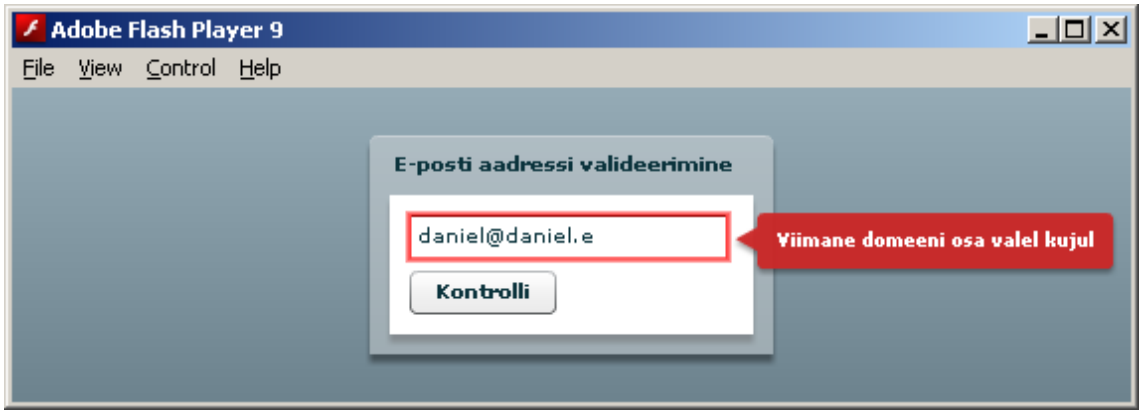
Toome ka põgusa näite kasutatud XML-andmefailist, et kõrvutada infostruktuuri rakenduses ning tegelikus allikas.

```
<rida>
  <lahter0>Tallinn</lahter0>
  <lahter1>209957</lahter1>
  <lahter2>173119</lahter2>
  <lahter3>5747</lahter3>
  ...
</rida>
<rida>
  <lahter0>Haabersti linnaosa</lahter0>
  <lahter1>18830</lahter1>
  <lahter2>16852</lahter2>
  ...
</rida>
...
```

2.8. Andmete valideerimine

Väga hästi on Flex'is realiseeritud tihti vajaminevate andmevalideerimiste teostamine. Nimelt on selle jaoks olemas standardsed komponendid, aga luua saab ka isikupärastatuid – näiteks niisuguseid, mis kasutavad regulaaravaldisi, et teha kindlaks, kas sõne muster *(string pattern)* vastab sellele esitatud nõudmistele. Järgmine näide demonstreerib Flex'is saadavaloleva valmiskomponendi – e-posti aadressi validaatori kasutamist. Validaator kontrollib nupuvajutusel e-posti aadressi struktuuri ning annab kasutajale kohest tagasisidet, kui selle kuju ei ole kooskõlas nõutavaga. Pärast lähtekoodi joonisel 8 on kujutatud ekraanipilt.

```
    <?xml version="1.0"?>
    <mx:Application xmlns:mx="http://www.adobe.com/2006/mxml">
      <mx:Script>
        import mx.controls.Alert;
      </mx:Script>
      <mx:EmailValidator source="{epost}" property="text"
      trigger="{nupp}" triggerEvent="click"
      valid="Alert.show('E-posti aadress on korrektne!')"
      invalidCharError="Keelatud sümbolid"
      invalidDomainError= "Viimane domeeni osa valel kujul"
      invalidIPDomainError="IP-vormis domeen valel kujul"
      invalidPeriodsInDomainError="Domeen sisaldab järjestikuseid
punkte"
      missingAtSignError="@ puudub"
      missingPeriodInDomainError="Domeenis puudub punkt"
      missingUsernameError="Puudub kasutajatunnus"
      tooManyAtSignsError="@ märke peab olema üks" />
      <mx:Panel title="E-posti aadressi valideerimine" paddingTop="10"
paddingLeft="10" paddingRight="10" paddingBottom="10">
      <mx:TextInput id="epost" />
      <mx:Button id="nupp" label="Kontrolli" />
      </mx:Panel>
    </mx:Application>
```



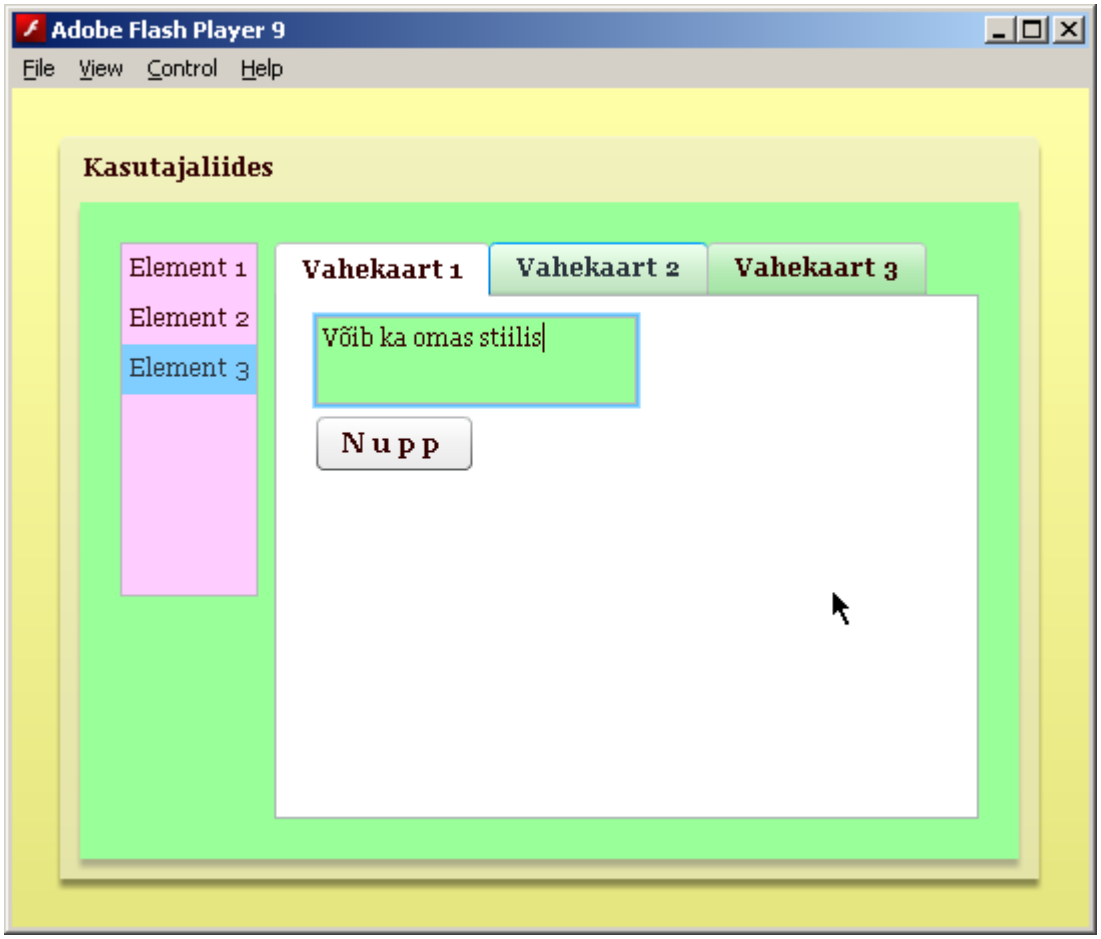
Joonis 8. E-posti aadressi valideerimine

2.9. Astmelised laadilehed

Flex'i komponentide väljanägemise isikupärastamiseks on võimalik kasutada standardset CSS-i laadilehestikku *(Cascading Style Sheets)*. Selleks otstarbeks tarvitatakse märgendit *<mx:Style>*, mille alla kirjutatakse vastavad stiilimäärangud või viited välistele CSS-failidele. Nimetatud märgend peab järgnema koheselt juurelemendile *Application*. Üksikutele komponentidele saab stiile rakendada klassiselektori kaudu (analoogiliselt HTML-iga), aga kõigi teatud tüüpi komponentide jaoks kasutatakse tüübiselektorit. Järgmises näites on demonstreeritud mõlemat tüüpi selektorite kasutamist; illustreerib joonis 9. Rakendus ise on joonisel 3 toodud kasutajaliidese stiliseeritud versioon.

```
<?xml version="1.0"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml">
  <mx:Style>
    Application {
      background-color: #ffff99;
      font-family: Georgia, Garamond, Arial, Helvetica, Sans-Serif;
      color: #330000;
      font-size: 12pt;
    }
    Panel, TextArea {
      background-color: #99ff99;
    }
    List {
      background-color: #ffccff;
    }
    .saatmine {
```

```
          font-size: 14pt;
          letter-spacing: 3px;
        }
      </mx:Style>
      <mx:Panel title="Kasutajaliides" paddingTop="20"
paddingBottom="20" paddingLeft="20" paddingRight="20">
        <mx:HBox>
          <mx:List>
            <mx:dataProvider>
              <mx:Array>
                <mx:String>Element 1</mx:String>
                <mx:String>Element 2</mx:String>
                <mx:String>Element 3</mx:String>
              </mx:Array>
            </mx:dataProvider>
          </mx:List>
          <mx:TabNavigator>
            <mx:VBox paddingLeft="20" label="Vahekaart 1" width="350"
height="250">
              <mx:TextArea text="Tere!" />
              <mx:Button styleName="saatmine" label="Nupp" />
            </mx:VBox>
            <mx:VBox paddingLeft="20" label="Vahekaart 2" width="350"
height="250">
            </mx:VBox>
            <mx:VBox paddingLeft="20" label="Vahekaart 3" width="350"
height="250">
            </mx:VBox>
          </mx:TabNavigator>
        </mx:HBox>
      </mx:Panel>
    </mx:Application>
```



Joonis 9. Stiliseeritud kasutajaliides

3. Ideid bakalaureusetöö edasiarendamiseks

Mais 2008 valmib näidisrakendust ning algset õppematerjali esitlev veebikülg, mille eesmärgiks on tutvustada RIA kontseptsiooni just Adobe Flex'i pakutavate võimaluste valguses. Nimetatud veebikülg demonstreerib, kuivõrd interaktiivseid ja sisult mitmekesiseid RIA-sid aitab luua käsitletav arendusraamistik võrdlemisi vähese vaevaga. Veebipäeviku tüüpi lehekülg on Internetist kättesaadav aadressil *http://www.daniel.ee/bak*. Üks ekraanipilt arendatavast näidisrakendusest on esitatud lisas 2.

Milliseid funktsioone demorakendus täpsemalt sisaldab, saab uurida mainitud lehelt. Olgu veelkord märgitud, et Flex'i *swf*-animatsioonid töötavad Flash Player'il alates selle versioonist 9. Vastav tarkvara on operatsioonisüsteemist sõltumatult allalaetav Adobe veebilehelt *http://www.adobe.com*.

Näidisrakendusega samalt leheküljelt on kättesaadavad ka kõik antud töös esitatud näited – nii koodina kui ka kompileeritud kujul, aga samuti kokkupakituna allalaadimiseks.

Tulevikus on autori sooviks luua praktilisi näiteid koos selgitava tekstiosaga nii palju juurde, et nendest võiks koostada mahukama veebilehestiku, mida oleks sobiv kasutada kõrg-, ent miks mitte ka keskkoolides RIA kontseptsiooni tutvustava õppematerjalina.

Kokkuvõte

Jaanuaris-veebruaris 2008 käesoleva bakalaureusetöö koostamiseks ideid otsides olid autori teadmised rikkasisuliste veebirakenduste *(RIA – Rich Internet Application)* loomiseks kasutatavatest vahenditest võrdlemisi kesised, samuti polnud omandatud praktilisi kogemusi ühegi niisuguse veebiarendusraamistikuga töötamiseks, millisest antud töös ülevaade esitati.

Alanud sajandi esimese aastakümne lõpp on toonud interneti-maastikule RIA mõiste – need on töölauaprogrammi tüüpi „uusima aja veebirakendused", mille kasutajaliideste ning programmiloogika kirjeldamiseks ei kasutata niivõrd palju traditsioonilisi vahendeid nagu HTML-i või teisi varemloodud tehnoloogiaid, vaid tihti pigem klient-server-rakendusi ehitada võimaldavaid kompileeritavaid keeli, millest üht käsikäes töötavat paari, Adobe MXML-i ning Actionscripti käsitleti käesolevas lõputöös.

Võrreldes veebi viieteist aasta taguste algusaegadega on internetikasutajate nõudmised tänaseks päevaks oluliselt kõrgenenud, samamoodi on mitmekesistunud saadavalolevate arendusvahendite valik. Ühed neist sobivad paremini loomaks lahendusi, mis töötavad levinumates veebilehitsejates, teised esitavad kasutaja arvuti konfiguratsioonile kõrgemaid nõudmisi. Viimase hulka kuulub ka Adobe Flex, milles toimuva arendustegevuse põhimõtetele keskendus käesolev bakalaureusetöö.

Uute veebilahenduste kasutajasõbralikkus ning kohati hämmastav dünaamilisus, mis tõstavad neid kogeva inimese produktiivsust interneti kasutamisel, on RIA kontseptsioonis eesmärgid, mille saavutamiseks kergeid kompromisse ei tehta. Adobe Flex pole ehk väga lihtne, aga on ääretult võimas autorsüsteem mõjuvate veebirakenduste loomiseks, mis kestaksid ajas kauem ning tõenäoliselt ületaksid neile pandud lootusi paljukordselt.

Lõputöö autorile andis tehtud uurimustöö arvestataval määral uusi teadmisi tänapäeval olemasolevatest veebiraamistikest, mida on küll praegusajal Eestis vähem kasutatud, ent millel on tulevikku ka aastate pärast, kui jõuavad auditooriumini lahendused, millised võib võtta koondnimetuse Web 3.0 alla.

Annotation

Adobe Flex – The Framework for Developing Rich Internet Applications on Flash Platform

BSc thesis by Daniel Labo

When author of this thesis gathered ideas about subject of research, he knew quite few about Rich Internet Applications' development techniques that have gained so much popularity and prominence these days the same way as applications itself. These applications are similar to conventional desktop programs but do not require to be installed onto users' computers. Web 2.0 is everywhere, but author assumes that majority of Internet users have not made themselves clear what are these new RIAs that provide so efficient, entertaining and engaging user experience.

Author was not the exception. When he studied about RIAs and corresponding development software, he actually learned much new he wasn't familiar with before and had practical chance to write many introductive sample applications using Adobe Flex 3 as the development framework. This technology was first introduced few years ago by Macromedia Inc. as convenient set of tools that would be perfect for building RIAs that are based on robust and proven client-server architecture.

In 2008 Adobe Flex framework is rebuilt and can be used by anyone with no charge if using the SDK. This software development kit that was used to compile all samples included in this thesis is now open-source.

The first decade of 21-st century has introduced the "Internet applications of new age", simply – RIAs. When software developers build these applications, traditional technologies such as HTML are less used. If mentioning Flex, developers use its languages MXML and Actionscript to define applications' user interfaces and business logic, respectively. Tools Flex provides are on the contrary much more powerful although probably not so easy to use as HTML with Javascript.

When comparing Internet users' needs and expectations now and fifteen years ago, the picture is so radically different. Of course, the set of technologies available to build these web applications is much more diverse, similarly. Some technologies work better when there is need for wider compatibility with web browsers, other tools are

innovative, but using them results in drawbacks as considerable number of users in target audiences may not have proper computer configuration to view rich content the applications provide. Flex applications use Flash Player 9 as its platform, but it is sufficiently widespread.

User friendliness, aesthetical visual aspects and cutting edge interactivity of RIAs enhance users' productivity like no conventional websites. Uncompromising movement towards these goals is on of the key concepts in development process of Rich Internet Applications.

Adobe Flex may not be the framework that's very easy to learn, but it's powerful and effective for developing extensive set of highly interactive database driven web applications.

Writing this bachelor's thesis gave author lot new knowledge about RIAs and web developing frameworks used in present. Although they may not be used very extensively in Estonia these days, author believes that this situation is changing. World is on doorstep of Web 3.0.

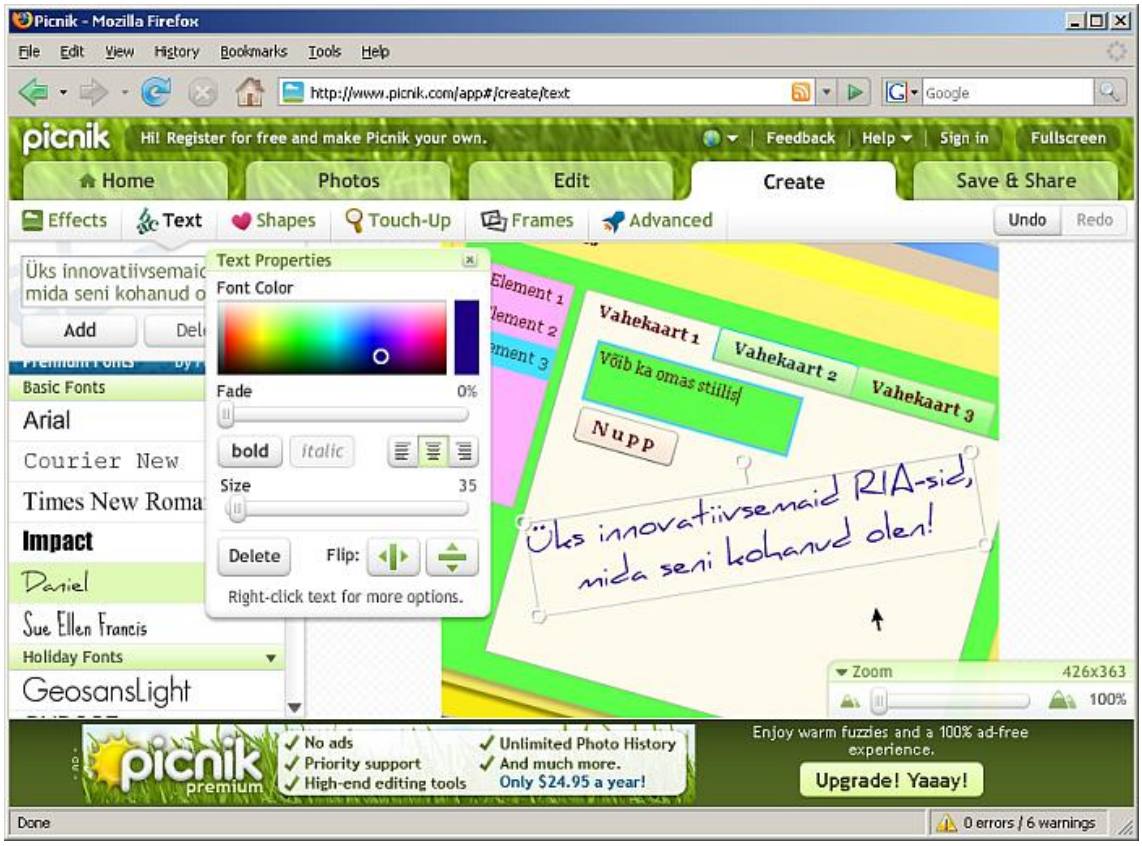
Kasutatud kirjandus

[Forrester, 2007] Rogowski, R. (2007) “The Business Case For Rich Internet Applications”
http://www.adobe.com/enterprise/pdfs/Forrester_RRogowski_BusCase_for_RIAs3_07.pdf

Adobe Inc. “Flex 3 Developer’s Guide”
http://livedocs.adobe.com/flex/3/html/help.html?content=Part2_DevApps_1.html

Lisad

1. Näidisvaade RIA-st *Picnik.com*



2. Näidisvaade arendatavast Flex'i rakendusest