Tallinna Ülikool

Informaatika Instituut

Õppematerjal Silverlighti reaalaja rakenduse loomiseks.

Seminaritöö

Autor: Jens Kaspar Mikli

Juhendaja: Jaagup Kippar

Tallinn 2011

Sisukord

Sissejuhatus

Siinse töö eesmärgiks on luua ja lahti seletada *Silverlight* rakendus millel on võime reaalajas andmeid üle võrgu saata ja töödelda. Kuna *Silverlight* rakendused on mõeldud ennekõike toimima interaktiivse meedia rakendusena ilma mingi tagasidemeta serveri poolse koodiga, on tegu suhteliselt keeruline ettevõtmisega.

Silverlight on kasvava kasutusega standard mille ülevõtu üks peamisi takistusi on selle võõrasus võrreldes konkureeriva Adobe *Flash*-iga, seega võib teemat pidada aktuaalseks. Isiklikult tekkis huvi *Silverlighti* vastu soovist töötada lähemalt veebirakendustega ja juhendajal pakkus välja teemaks selle reaalaja rakenduste kohta õppematerjali luua.

Töö peamise sihtrühmana on silmas peetud teisi informaatika ala tudengeid ja teemaga vähe tuttavaid arendajaid. Töö jooksul toodetud õppematerjalist peaks piisama, et anda arusaamine selle kohta kuidas *Silverlight* võrgu kaudu andmeid edastab ja kasutab.

Õppematerjali täisväärtuslikuks kasutamiseks oleks soovitav mõningane eelnev tutvus *Silverlight* platvormiga, kuigi otsene vajadus puudub ei ole see ilmselt sobilik esimese omal käel loodud rakenduse koostamiseks. Soovitused sellistele õppematerjalidele võib leida esimesest peatükist: „Ülevaade olemasolevast materjalist.“

Üldised uurimisküsimused olid: Kas ja kuidas on võimalik luua *Silverlightis* reaalajas toimiv andmevahetus serveriga, mis on selle jaoks tõhusaim ja lihtsaim meetod ja kas selle kohta leidub õppematerjali? Kuidas saab seda laiendada ja kas see on kättesaadaval Eesti keeles?

Täpsem teema langes töö jooksul veebiteenustele mis on laialt kasutatav ja kergelt realiseeritav standard *Silvelightis* andmete jagamiseks.

Küsimused veebiteenuste kohta on kuidas need *Silverlightiga* ühenduvad, andmeid töötlevad ja kuidas nende kaudu luua kahe või enama *Silverlight* rakenduse vaheline side.

Tööeesmärkide saavutamiseks uurisin *Silverlighti* dokumentatsiooni ja ingliskeelseid õppematerjale, seejärel viisin läbi katseid ja lihtsustasin materjali, võtsin eeskujuks ka eelnevalt koostatud eestikeelsete materjalide vormistuse.

Tulemuseks genereeritud juhendit jagasin kaastudengitega ja kontrollisin arusaadavust.

Tänada sooviksin Jaagup Kippari abivalmis juhendamise ja Lauri Eliast mu suurimate lohakuste kätte näitamise eest.

Kasutatud võõrsõnad ja akronüümid

CLI(C*ommon Language Infrastructure*)- e. üldine keele-infrastruktuur, .NET-platvormi versioon, mis on platvormist sõltumatu ning võimaldab kolmandatel osapooltel välja töötada kompilaatoreid muudele (mitte-Microsoft'i) programmikeeltele ja käitusajakeskkondadele, mis ei kasuta Windows'i opsüsteemi. Sellistes CLI-põhistes süsteemides kasutatavat vahekeelt nimetatakse üldiseks vahekeeleks (CIL - *Common Intermediate Language*) erinevalt Microsoft'i vahekeelest MSIL (*Microsoft Intermediate Language)*(E-teatmik, 2012).

IDE (*Integrate Development Eviorment*)- e. integreeritud programmeerimiskeskkond Programmeerimisekeskkond, mis on rakendustarkvarasse sisse ehitatud. Harilikult koosneb integreeritud programmeerimiskeskkond lähtekoodi redaktorist, kompilaatorist või interpretaatorist või mõlemast ning programmeerimise automatiseerimise abivahenditest,millele enamasti lisandub ka silur (E-teatmik, 2012).

IIS(*Internet Information Service*)- e. Interneti infoserver Microsoft'i serverprogramm , mis on mõeldud suure hulga kasutajate vajaduste rahuldamiseks alates töögruppidest ja firmade intranettidest kuni ISP-deni, kes teenindavad päevas miljoneid pöördumisi saavaid veebisaite. IIS sisaldab veebikirjastamise tarkvara, kasutaja vajadustele kohandatavaid töövahendeid, võlureid , kohandatavaid võrguhalduse vahendeid, paindlikke administreerimisvariante ja analüüsivahendeid. IIS'i on lihtne kasutada dokumentide ja informatsiooni ühiskasutust firma intranetis või Internetis ning ühildub täielikult Windows NT Directory Service'iga. Konkurentideks on Apache ja Netscape Enterprise Server (esimene on ca 3 korda populaarsem ja teine 7 korda vähem levinud) (E-teatmik, 2012).

RIA(*Rich Internet Aplication*)- e. Rikkasiuline veebirakendus on veebirakendus mille eesmärgiks on pakkuda sarnast funktsionaalsust mida tavaliselt seostatakse töölauarakendustega. RIAd jagavad üldiselt andmete töötlemise serveri ja kliendi poolseks vähendades nõnda vajadust ühendust kurnata, kuulsaim tootja on Adobe poolt loodud *Flash* platvorm (Definition: Rich Internet Application (RIA),2012).

SOAP(*Simple Object Access Protocol*)- e, lihtne objektipöödusprotokoll Minimaalne komplekt kokkuleppeid programmide käivitamiseks XML'i abil üle HTTP. SOAP-protokoll kasutades ühe opsüsteemi (näit. Windows 2000) all töötav programm suhelda

mitte ainult teistes arvutites sama opsüsteemi all töötavate programmidega, vaid ka teiste opsüsteemide (näit. Linux) all töötavate programmidega. SOAP'i aluseks on sama strateegia, mis näit. CORBA'l või DCOM'il, kuid SOAP on märksa kergekaalulisem ja lihtsam kasutada. Ka suudab SOAP paremini läbida tulemüüre.

SOAP 1.1 on defineeritud World Wide Web Consortium'i poolt (E-teatmik, 2012)
Socket- IP-protokollil põhinevas võrgus, näiteks Internetis, nimetatakse antud hosti sokliks sidekanali otspunktina toimivat IP-aadressist ja pordinumbrist koosnevat identifikaatorit (Cisco määrang). Sageli loetakse sokli määrangusse kuuluvaks ka kasutatav sideprotokoll (harilikult TCP või UDP) (E-teatmik, 2012).

HTML(*HyperText Markup Language*)- e. Enimlevinud kodeerimissüsteem (tekstivorming) veebidokumentide loomiseks. HTML koodid ehk märgendid määravad ära selle, kuidas veebileht arvutiekraanil välja näeb(E-teatmik, 2012).

HTTP*(*HyperText Transfer Protocol)- e. hüperteksti edastusprotokoll TCP/IP klient-server protokoll HTML-dokumentide vahetamiseks veebis ehk lihtsamalt öeldes andmevahetusprotokoll, mida kasutatakse Internetis dokumentide vahetamiseks (E-teatmik, 2012).

WCF-(*Windows Communication Foundation*)-e. Windows kommunikatsiooni fond on raamistik teenuse-orienteeritud rakenduste loomiseks. WCF kasutades saab saata andmeid asünkroonsete teadetena ühelt teenuse lõpppunktilt teisele. Teenuse lõpppunkt võib olla alaliselt kättesaadav teenus mida hostib IIS või rakendus. Lõpppunkt võib olla teenuse klient mis pärib andmeid teenuse lõpppunktilt. Sõnum võib olla lihtne nt. üks sõna või täht XMLina vormistatud, või keeruline nt. binaarne andmejada (Microsoft Development Network, 2012).

XML (*eXtensible Markup Language*)- e. laiendatav märgistuskeel XML on suvaliste andmete struktureerimiseks mõeldud märgistuskeel, mis loodi eemärgiga võtta see veebis kasutusele HTML'i asemel. XML kasutab andmetüüpide (nimetatakse "skeemideks") defineerimiseks dokumenditüübi definitsioone (DTD), mis on pärit SGML'i dokumendikesksest lähenemisest. XML on osutunud väga edukaks ärivahelistes stsenaariumides ja seda kasutatakse järjest rohkem andmevahetuses dokumentide vahetamise asemel (E-teatmik, 2012).

XAML (*eXtensible Application Markup Language)*- XML põhine deklaratiivne keel. Mida rakendatakse .NET raamistiku programmitöö mudelis, XAML lihtsustab kasutajaliidese loomist .NET raamistikus. Andes võimaluse luua UI elemendid deklaratiivses XAML märgistuskeeles ja eraldades UI definitsiooni loogika. Nii saavad UI kallal töötada samaaegselt erinevead osapooled (Microsoft Development Network, 2012).

1.Ülevaade olemasolevast materjalist

Silverlight kohta leidub küllalt laialdaselt õppematerjale, sealhulgas ka eesti keeles.
Silverlighti tutvustavaks materjaliks võiks pidada ENETA veebistuudiumit:
http://www.eneta.ee/oppimine/veebistuudium/Lehed/veebidisain.aspx
Ingliskeelse alusmaterjali võib leida *Silverlight quickstartist*:
http://www.silverlight.net/learn/overview/getting-started

Laialdasemalt on käsitletud *Silverlighti* teemat Jaana Tomsoni ajaveebist „Hõbevalge nagu Silverlight“
http://h6bevalge.riiul.com/
Mitmeid koodinäiteid ja õpetusi võib leida ingliskeelsest keelsest kommunist „Silverlight Show“
http://www.silverlightshow.net/

Siiski leidub vähe õppematerjale mis tegeleks otseselt veebipõhise reaalaja rakenduse loomisega Eesti keeles vist ainult ENETA näidises.

Töös leiduv rakendus on koostatud Codeprojecti näitel:
http://www.codeproject.com/KB/silverlight/AccessWebService.aspx
Kusjuures olulisel määral kattub "hello world" rakenduse loome protsess. Tuvastus meetod on omaloomeline.

Silverlight ingliskeelne dokumentatsioon:
http://msdn.microsoft.com/en-us/library/bb404700%28v=vs.95%29.aspx

2.Tehnoloogi tutvustus

Silverlight on Micrososfti poolt loodud RIA arendusplatvorm mis pakub võimalusi toota interaktiivseid kliendipoolseid meediarakendusi ja kõrgekvaliteetseid videoülekandeid. *Silverlight* standard on ehitatud .Net raamistikuga ühilduvaiks ja sarnaneb funktsionaalsuselt konkureerivate Adobe *Flashi* ja *Flexiga. Silverlight* on mugava arhitektuuriga ja mitmekülgne tarkvara mida on siiski kogenematutel arendajatel veidi raskem rakendada kui viimati mainitud konkurente. Töös om rakendatud *Silverlight* versiooni neli (Silverlight FAQ,2011).

Silverlighti peamiseks arenduskeskkonnaks on Microsofti poolt pakutav *Visual Studio* IDE toodete grupp. Siinse õppematerjali näidisrakenduse koostamisel on kasutatud *Visual Web Developer 2010 Expressi* mis on ilmarahata kättesaadava Microsofti arendajate kodulehelt. Võimalik et siinne lihtsakoeline juhend toimib ka varasemate versioonidega kuid neid pole töö raames katsetatud, kindlasti kõlbab ka tasuline *Visual Studio* 2010 variant. Visual Studio toetab CLI-kaudu erinevaid programmeerimiskeeli, töös kasutatakse Visual C#.

Töö seisukohalt on vaja esiteks defineerida reaalaja rakendus. Ka *Silverlighti* poolt loodavaid tavalisi interaktiivseid meediarakendusi võiks pidada mõnevõrra reaalaja rakendusteks kuid sisuliselt on siiski tegu staatilise koodiga mis peale allalaadimist töötab brauseris nagu tavapärane töölauarakendus, töö raames on üritatud saavutada reaalaegset andmesidet kahe või enama rakenduse vahel.

Reaalaja rakenduse tootmiseks oli tarvis luua meetod kuidas andmeid saaks veebi kaudu edastada ja vastu võtta. Valik andmesideme loomiseks langes veebiteenustele. Kindlasti ei ole veebiteenused ideaalsed loomaks kiireimad või elegantseimaid süsteeme kuid nende lihtsus ja laialdane kasutus teevad nad algajaile ahvatlevaks ning lihtsamate ülesannete jaoks sobivaimaks. Peamiseks puudujäägiks võib pidada interneti jaoks mõeldud raamistike olekuta (*stateless*) disaini, mis teeb keerukaks individuaalsete klientide tuvastamise. Kuid nagu dokumentatsioonis pidevalt korratakse on selline võrgu asjade loomulik seisund (Webucator, 2011).

Seevastu keerukam kuid kiirem lahendus on kasutada madalama taseme *Silverlighti socketid*. Siiski on veebiteenused arendajate seas populaarsem meetod oma suhteliselt kerge ja mitmekülgse kasutusviisi tõttu.

2.1 Veebiteenused

Veebiteenused on tarkvararakendus mida võib veebi kaudu tuvastada ja rakendada kasutades XMLi ja standard võrguprotokolle. Veebiteenused loodi kolme peamise probleemi lahendamiseks mille alla kannatasid distributiivsed programmid milleks olid: tulemüüri läbipääs, keerukus ja koostalitlusvõime. Veebiteenused viib läbi teatud ettemääratud funktsiooni kasutavad andmete transpordiks sama HTTP protokolli mida rakendavad kõik veebiühendused, tehes neist universaalselt vastuvõetava Internetiga opereerivate süsteemide lahendades ka tulemüürid lubavad tavaliselt liiklusel toimida läbi HTTP pordi. Seega on veebiteenused enne kõike koostalitusvõimeline erineva tarkavaraga, samas jaotudes loogiliselt ja funktsionaalselt piirates keerukust.

XML küljendus ja Tehnoloogiad nagu UDDI (*Universal Description, Discovery, and Integration*) ja ebXML registrid, lubavad rakendusi dünaamilistel veebiteenuste kohta infot koguda kattes tuvastamise osa meie definitsioonist. Veebiteenuste sõnumi süntaks on kirjeldatud WSDLis (*Web Service Definition Language*).

Kui rääkida veebiteenustest mõeldaks ennekõike SOAPi mis on see osa tarkvarast mida välised rakendused tarbivad. SOAP *(Simple Object Access Protocol*) on XMLil põhinev sõnumi protokoll. Seda rakendavad ennekõike äriettevõtted ja sellele on ülesehitatud ka siinsed näidisrakendused.

SOAP on süntaks XML sõnumite saatmiseks ja vastuvõtmise HTMLi kaudu. See annab standard keele sidumaks rakendusi ja teenuseid. Rakendus saadab SOAP taotluse veebiteenusele , ning veebiteenus vastab midagi.

Kusjuures on arendatud ka „uus“ veebiteenuste standard REST (*REpresentational State Transfer*). RESTis on igal teenusel meetodil eraldi URI ja ühendus toimib läbi HTTP päiste. Seda kasutavad ennekõike uuemad võrgurakendused nagu Twitter.(Web Service tutorial,2011).

Valik langes SOAPil kuna standard on laiemas kaustuses loodetavasti on siinses õppematerjalis uuritav meetod ka ülekantav ka uuematele tehnoloogiatele.

Lisaks tuli tööjooksul valid erinevate veebiteenuste raamistike ja teostuste vahel, sealjuures ASP.NET veebiteenus olid esimene valik kuna nende arendus oli *Visual Web Developeriga* integreeritud. Sealjuures tuleb meeles pidada, et WCF on Microsofti poolt arendatud olema viimast asendavaks tehnoloogiaks ja on kindlasti palju võimsam. Õppematerjali koostamisel ei ole see aga tingimata pluss kuna tööprotsessi lihtsustamine toimib vaid ühes suunas, seega on võimalik .asmx formaadist WCF üle minna hulga kergemini kui vastupidi.

Veel jäid tööst kõrvale teised protokollid nagu JSON (*JavaScript Object Notation*), mis ei mahtunud tööaega kuid mida võiks samuti reaalajarakenduse huvides uurida.

3.Juhend

Eesolevas juhendis üritan lahti seletada kuidas luua reaalaja rakendust kasutades *Silverlighti* ja veebiteenust.

Reaalaaja rakendust defineeritakse kui neid mis toimuvad kasutaja jaoks kohesena tunduva aja jooksul, nii et tavaliselt mitte pikema kui sekundid kestva viivitusega.

Kuna *Silverlightis* puudub meetod mis lubaks luua püsivat võrguandmesidet kasutame veebiteenust vahekomponendina ja loome mängu kus kaks *Silverlight* rakendust mis on loonud ühenduse sama veebiteenusega saavad omavahel suhelda mängijate jaoks nähtamatu ajavahemiku jooksul.

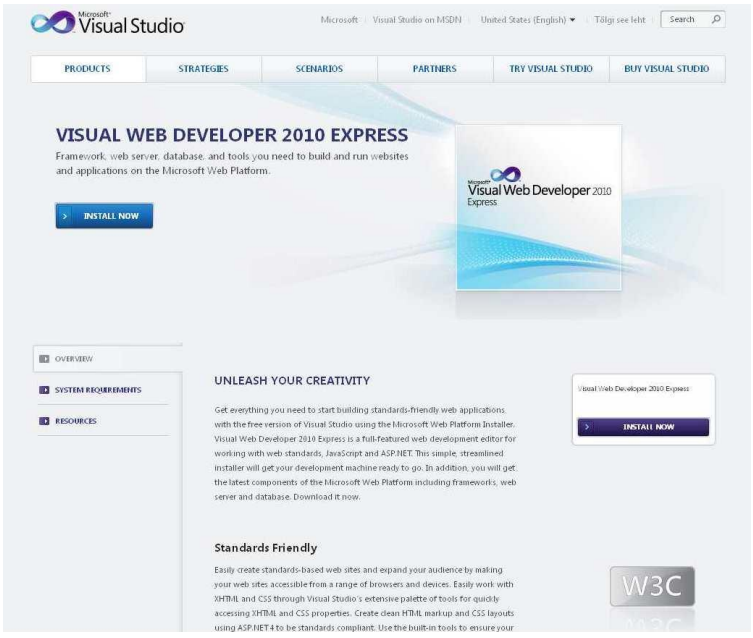
Sarnast koodi võib kasutada ka näiteks jututoa ja keerulisemate rakenduste loomisel ja kuna veebiteenused on mitmekülgsed ja on võimelised andmeida jagama pea kõigi võrgutehnoloogiatega avardab see võimalusi veelgi.

3.1 Visual Studio paigaldamine.

Visual Studio on Microsofti IDE mille ekspress variant on tasuta. Minge *Visual Web Developer 2010 Express* kodulehele aadressil:

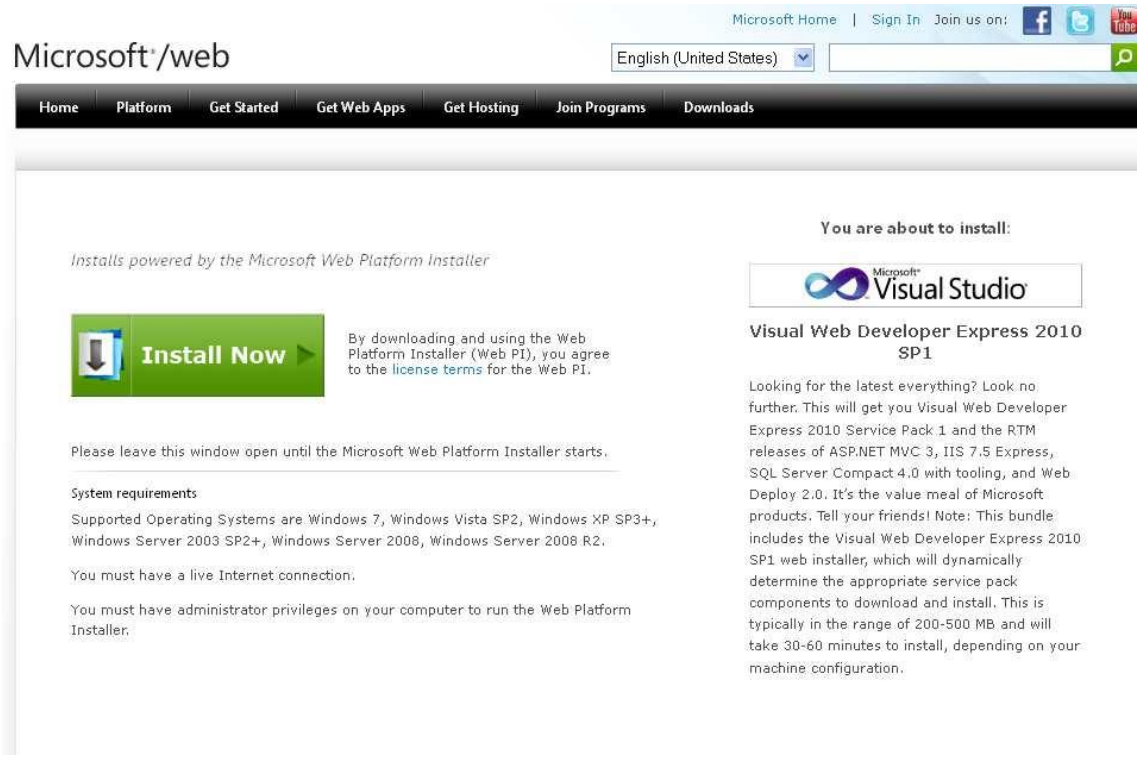
http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-web-developer-express

ja valige sealt *Install Now* (Joonis 1) ja valige ilmuval aknal Install Visual *Web Developer 2010.*



Joonis 1 *Visual Web Developer Express* allalaadimise lehekülg

Juhul kui te varem Microsofti tooteid ei ole veebi kaudu allalaadinud palutakse teil ka installeerida *Microsoft Web Platformi* (Joonis 2).



Joonis 2 *Microsoft Web Platform* allalaadimise lehekülg

Seejärel tõmbab arvuti tavaliselt tunni aja jooksul vajalikud tööliidesed alla ja installatsioon ei tohiks palju kasutajapoolset osalemist nõuda. Lisainfo saamiseks tasuks külastada Visual Stuudio veebilehekülge.
Süsteeminõuded on operatsioonisüsteem üks järgnevatest:

- Windows XP (x86) Service Pack 3 – kõik peale Starter Editioni
- Windows Vista (x86 & x64) Service Pack 2 - all kõik peale Starter Edition
- Windows 7 (x86 & x64)

Soovitav riistvara:

- 1.6GHz või kiirem protsessor
- 1 GB (32 Bit) või 2 GB (64 Bit) RAM (512 MB enam virtuaalsel masinal)
- 3GB vaba kettaruumi
- 5400 RPM kõvakettaajam
- DirectX 9 võimeline video kaart 1024 x 768 või kõrgemal resolutsioonil
- DVD-ROM ajam

(Visual Web Developer 2010 Express: System Requierments, 2010)

3.2 „*Hello World*“ Veebiteenus

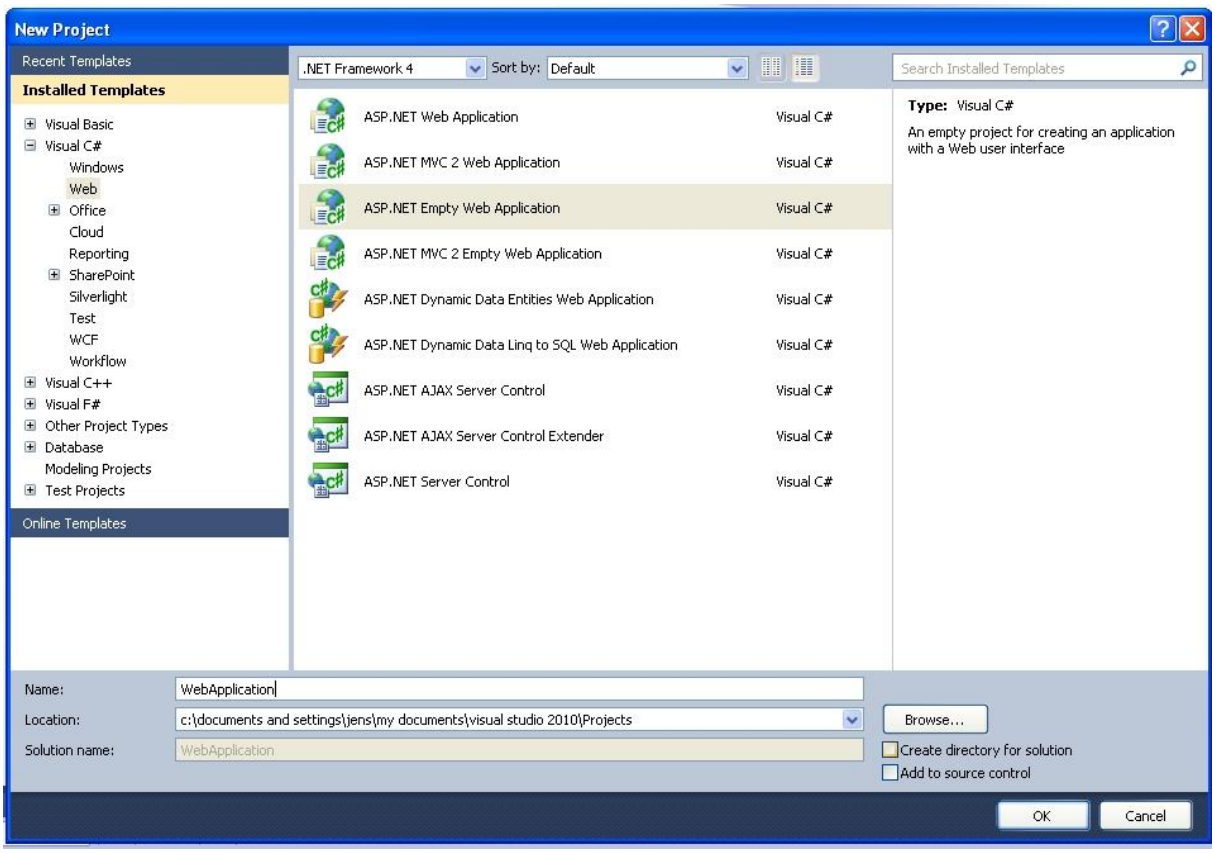
Alguses loome veebiteenuse, mis vastab päringule lihtsa „*Hello World*!“ meetodiga. Rakendus koosneb kahest komponendist: Veebiteenus ja klient.

Veebiteenuse loomine ei tohiks olla nõudlik kuna „*Hello World*“ sätted on *Visual Studios* vaikimisi juba ette määratud.

Selleks tuleb minna menüüst:

„*File-New Project*“

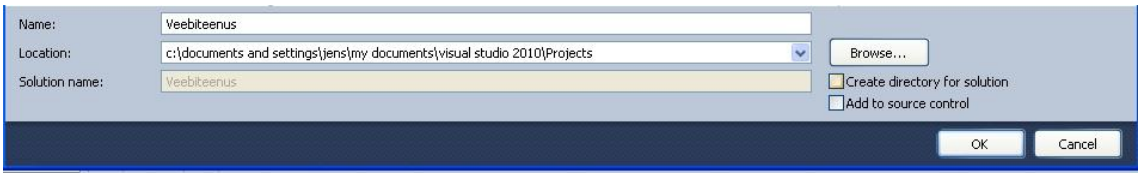
Millele vajutades ilmub järgnev aken (Joonis 3).



Joonis 3 New Project Empty Web Application

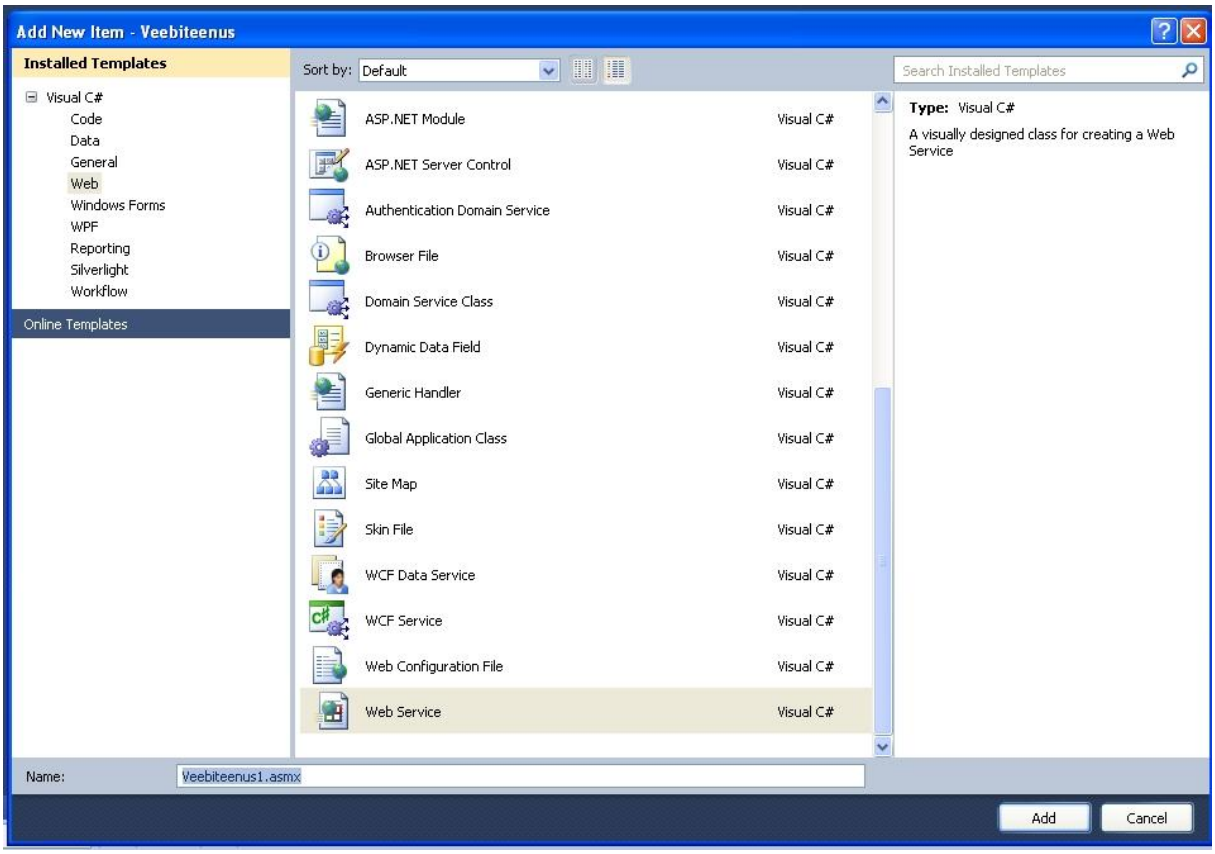
Sealt valige parempoolses nimekirjas: Programmerimis keeleks, antud juhendi tarbeks „*Visual C#*“, „*Web*“ ja seejärel „*ASP.NET Empty Web Application*.“ Allpool nimetage projekt „Veebiteenus.“

Eemaldades sealjuures kindlasti linnuke „*Create directory for solution*“ (Joonis 4) kõrvalt, selle paigale jätmine teeb meie seadistuse hiljem keerulisemaks.



Joonis 4 New Project all: Nimi-Veebiteenus

Tekkis tühi veebileht millele me saame lisada veebiteenuse. Vajutada parempoolses aknas nimega „*Solution Explorer*“ paremklõpsuga projekti nimele ja vali rippmenüüst: *„Add-New Item“* ja ilmuvast aknast:“*Web-WebService“*(Joonis 5).



Joonis 5 Add New Item: Veebiteenus1

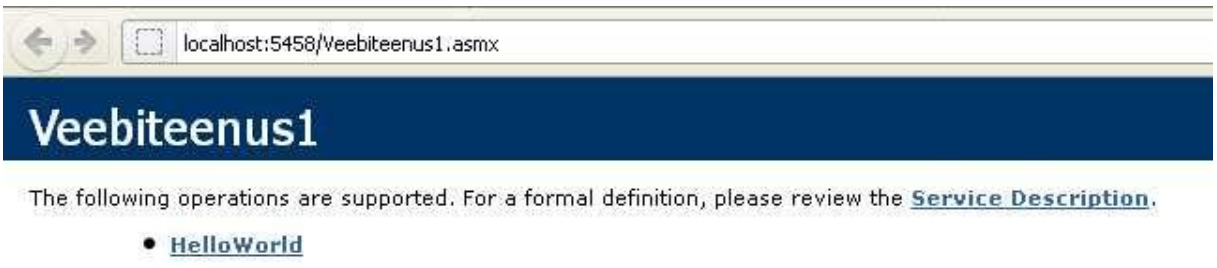
Anname sellele nimeks: „Veebiteenus1“(Koodinäide1).

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;

namespace Veebiteenus1
{
    /// <summary>
    /// Summary description for Veebiteenus1
    /// </summary>
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]
    // To allow this Web Service to be called from script, using
ASP.NET AJAX, uncomment the following line.
    // [System.Web.Script.Services.ScriptService]
    public class Veebiteenus1 : System.Web.Services.WebService
    {

        [WebMethod]
        public string HelloWorld()
        {
            return "Hello World";
        }
    }
}
```

Koodinäide 1. Veebiteenus1 ja „*Hello World*“ meetod

Nagu näha on veebiteenuse loomisel juba deklareeritud *Web Method „Hello World“* mis vastab päringule samasisulise stringiga. Sellest meile alguseks ka piisab, hiljem me vaatame kuidas kirjutada ka keerulisemaid teenuseid. Vajuta Ctrl-F5, et teenus brauseris käivitada.

Tähelepanu! Visual stuudio tekitab tavaliselt testserveri mis käivitatakse *debuggeriga*, viimane aga ei luba teenusele ligipääsu välistelt rakendustelt, kuna me arendame neid mitu tükki koos on soovitatav käivitada rakendus ilmtingimata Ctrl-F5ga, mis ehitab rakenduse ilma *debuggerit* käivtmata.

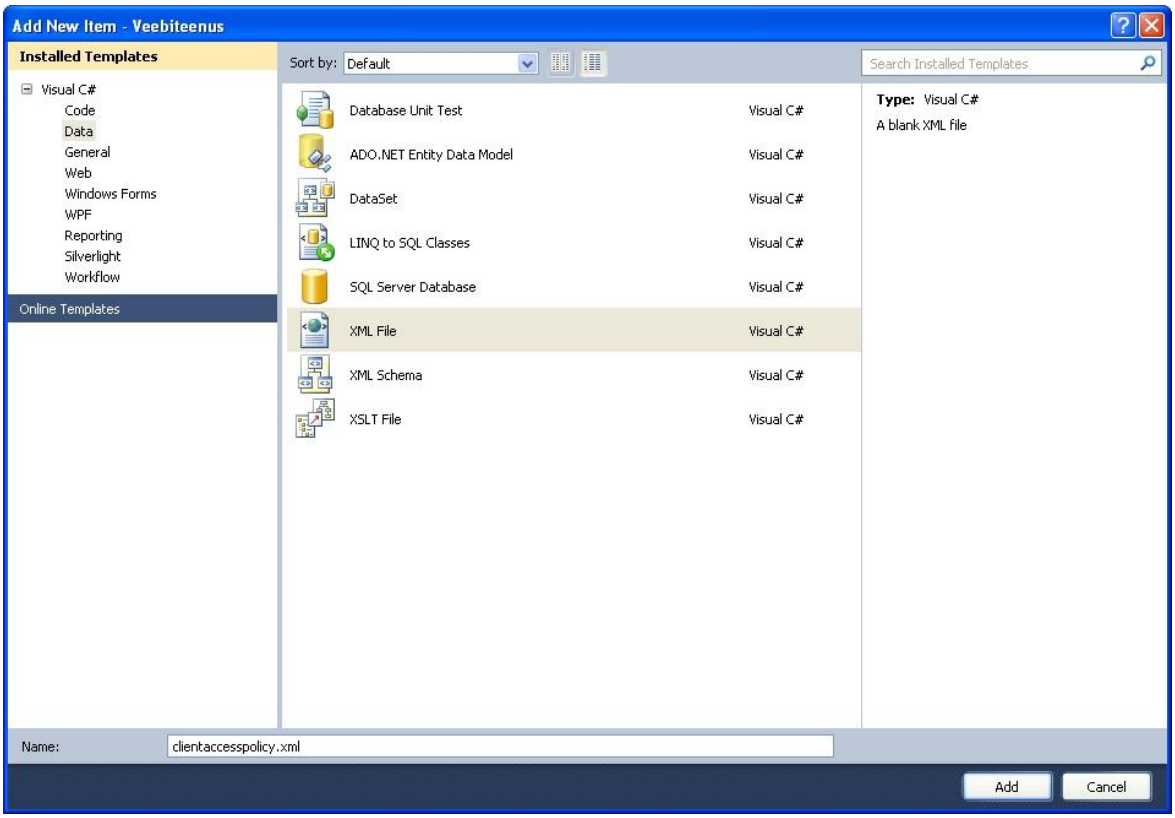


Joonis 6 Aadresss ja port

Siit saad teada mis porti *Visual Studio* on testserveri loonud (Joonis 6). Hiljem kasutame seda aadressi *Silverlight* kliendis teenusele viitamiseks. Näha on ka teenuse poolt pakutavad avalikud meetodid ja näited nende päringute kohta.

Kuna *Visual Studio* käsitleb erinevaid porte kui eri domeene ja kuna meil on kavas sama teenusega ühendada mitu projekti mis hakkavad paiknema eri testserveritel tuleb meil *Silverlight* turvanõuete kohaselt lisada veebiteenuse algkausta (*root directory*) fail nimega *clientaccesspolicy.xml* või *crossdomain.xml* . Kui *Silverlight* ei leia päringu toimepanekul esimest otsib ta teiste, leides mitte kumbagi ei lähe päring läbi.

Klõpsa taas „Solution Explorer“ aknas projekti nimele ja võta samast „Add-New Item“ menüüst seekord: „Data- XML File“ ja nimeta ümber *clientaccesspolicy.xml*-iks (Joonis 7).



Joonis 7 clientaccesspolicy.xml

Tekkinud XML faili lisa koodinäide 2.

```
<?xml version="1.0" encoding="utf-8"?>
<access-policy>
  <cross-domain-access>
    <policy>
      <allow-from http-request-headers="SOAPAction">
        <domain uri="*"/>
        <!--"*" tähistab, et kõikidest domeenidest tulevad päringud võetakse
vastu-->
      </allow-from>
      <grant-to>
        <resource path="/" include-subpaths="true"/>
        <!--Määrab millistele teenuse ressurssidele on antud domeenist tulevatel
päringutel ligipääsuõigus antud juhul kõigile.-->
      </grant-to>
    </policy>
  </cross-domain-access>
</access-policy>
```

Koodinäide 2. ***clientaccesspolicy.xml***

Võid lisada ka faili *crossdomain.xml* sisuga koodinäide 3, kuid see ei ole ilmtingimata tarvilik. Kui *Silverlight* ei leia teenuse algkaustast faili clientaccesspolicy.xml otsib ta selle asemel crossdomain.xml-i. Ei kumbagi leides rakendus ei suhtle veebiteenusega.

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE cross-domain-policy SYSTEM
"http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
  <allow-http-request-headers-from
   domain="*" headers="SOAPAction,Content-Type"/>
</cross-domain-policy>
```

Koodinäide 3. ***crossdomain.xml***

Ennem edasi liikumist veenduge veelkord, et mõlemad failid paikneksid justnimelt algkaustas (Code Project, 2011).

3.3 *Silverlight* kliendi loomine

Nüüd loome *Silverlight* kliendi mis seda teenust kasutab.

Käivita *Visual Studio* uus instants (Joonis 8).

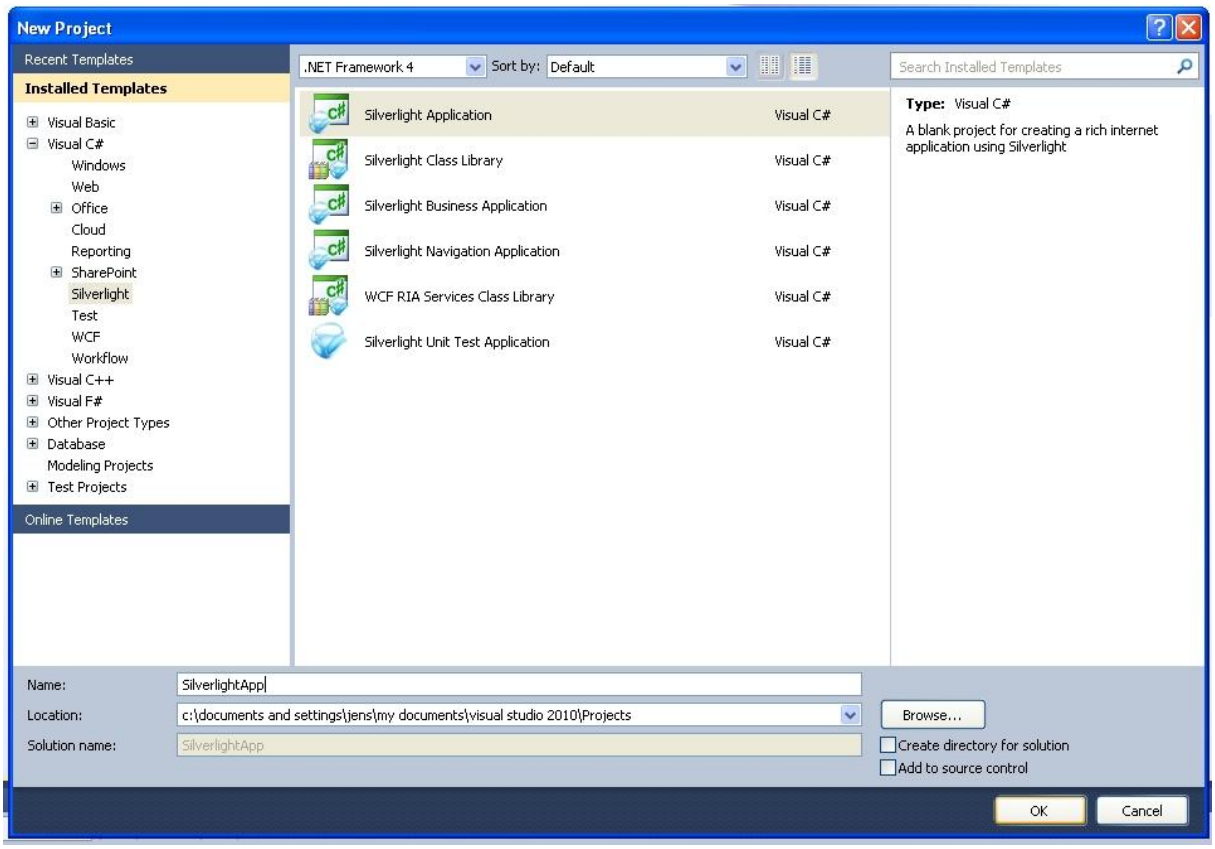
Joonis 8 Kaks instantsi Web Developeri menüüribas

Nii saame ehitada ühel masinal kaks või enam klienti ja need seejärel veebiteenusega ühendada. See meetod ei ole veel ilmtingimata tarvilik *Silverlight* pääseb ligi ka samal testserveril asuvale veebiteenusele kuid kahe eri *Silverlight* rakendusega läheb asi keerulisemaks.

Kliendi loomisel on kaks tähtsamat sammu:

- Luua kliendipoolne loogika sealhulgas ka kujundus
- Luua proksiklass mille kaudu saada ligipääsu veebiteenusele

Ava menüüst: *„File“-„New Project“-„Visual C#“ –„Silverlight“- „Silverlight Application“*(Joonis 9).



Joonis 9 New Project: SilverlightApp

Paneme sellele nimeks„ SilverlightApp“.

Joonis 10 Hosti veebilehel

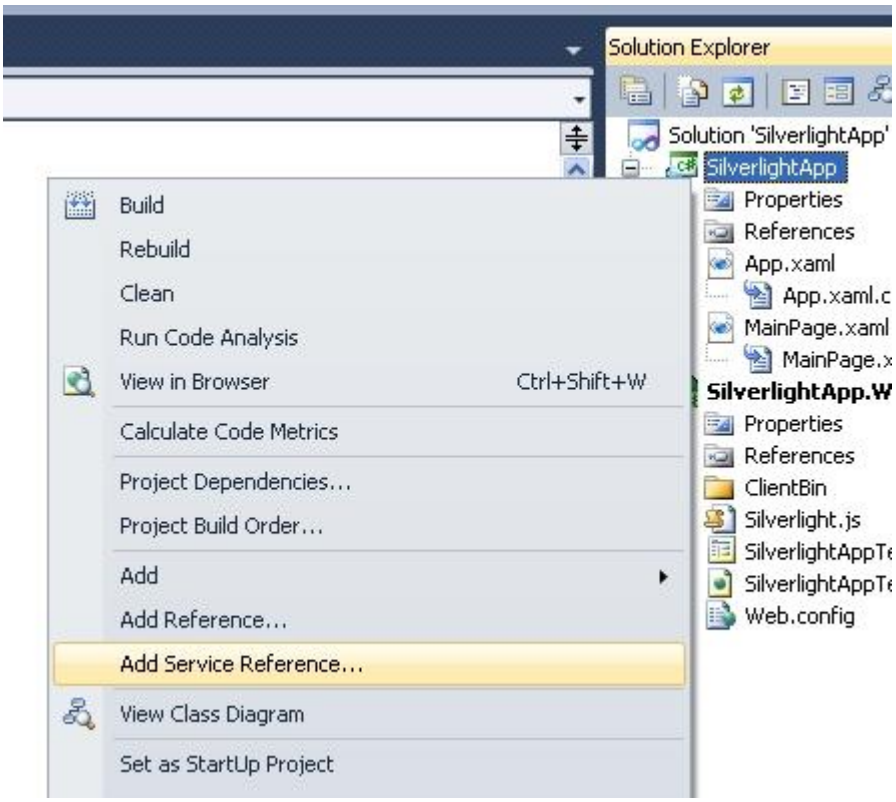
Avaneb aken millel saab valida kas paralleelselt *Silverlight*iga arendada ka seda hostivat veebilehte. Hiljem arendame ka seda, seega jätame asjad siin nii nagu nad ilmuvad

(Joonis 10). Kui sätteid erinevad, siis pange linnuke „*Host the Silverlight application in a new Web site*“ kõrvale.

Nüüd on meil ka *Silverlighti* rakendus, proovime luua ühendust veebiteenusega.

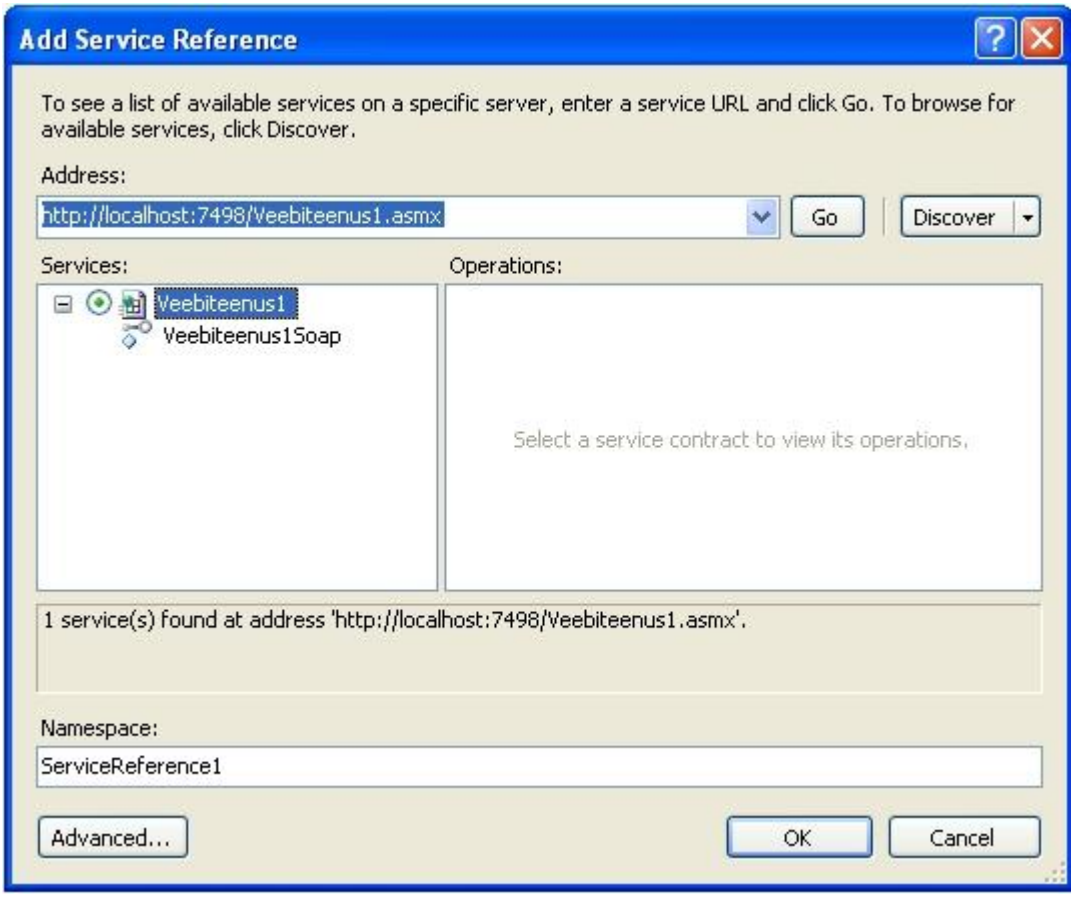
Lisa teenusele viit:

Paremklõpsa „Solution Explorer“-is „SilverlightApp“-le (viita saab lisada ka veebilehel, ent seda pole pragu vaja) ja vali rippmenüüst :“Add Service Reference“ (Joonis 11).



Joonis 11 Lisa teenuseviit

See avab akna kuhu kleebime varem loodud veebiteenuse aadressiriba sisu (Joonis 4 ja 12).



Joonis 12 Teenuseviida lisamise aken

Nüüd klõpsa OK. See lisab veebiteenus projekti ja genereerib proksi klassi, nii saame veebiteenuse meetodeid Silverlighti kaudu kutsuda.. Vahetevahel võib tekkida probleeme mille lahendamiseks on vaid tarvis veebiteenuse taas ehitamine, sel juhul valige veebiteenuse instants ja vajutage veelkord Ctrl-F5t.

„Service References“ kaustast, leiad „ServiceReference1" faili, seda paremklõpsates ja valides „View Object Browser“ võid leida SilverlightApp.ServiceReference1 alt Veebiteenus1SoapClient klassi ja selle meetodid. Nendele tugineb teenuse päring.

Nüüd loome oma rakenduse kujunduse.

Ava *MainPage.xaml* ja lisa *grid*i tekstiblokk, me täidame selle hiljem teenuse vastusega(Koodiäide 4).

```
<UserControl x:Class="SilverlightAppl.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="400">

    <Grid x:Name="LayoutRoot" Background="White">
    <TextBlock Height="20" HorizontalAlignment="Left"
        Name="tekstiblokk" Text="sisu" VerticalAlignment="Top"/>

    </Grid>
</UserControl>
```

Koodinäide 4.*Mainpage.xaml*

Liigume edasi programmi loogika juurde.

Ava *MainPage.xaml.cs* (vajutades väiksele ristikesele *Mainpage.xaml* kõrval) ja lisa sinna viit teenusele (Koodinäide 5).

```
using SilverlightApp.ServiceReference1;
```

Koodinäide 5. Viit teenusele

Lisa „*MainPage*“ meetodisse koodinäide 6.

```
//eelpool mainitud ühendus klassi konstruktor

VeebiteenusSoapClient proksi = new VeebiteenusSoapClient();

public MainPage()

        {
            InitializeComponent();
            //vastuse kuulaja mis saadab tagasi tulevad andmed
            meetodisse :"proksi_HelloWorldKuulaja"
            proksi.HelloWorldCompleted +=
            new
            EventHandler<HelloWorldCompletedEventArgs>(proksi_HelloWorl
            dKuulaja);
            //meetodi päring
            proksi.HelloWorldAsync();
        }
```

Koodinäide 6. MainPage()

Nagu näha on koodis mainitud ka HelloWorldCompleted sündmuse kuulajat. Loo see koodinäite 7 järgi.

```
        void proksi_HelloWorldKuulaja(object sender, HelloWorldCompletedEventArgs
        e)
         {
             tekstiblokk.Text = e.Result.ToString();

         }
```

Koodinäide 7. „*Hello World*“ kuulaja

Valmis Mainpage.xaml.cs peaks nüüd väljanägema nii (Koodinäide 8).

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using SilverlightApp.ServiceReference1;
namespace SilverlightApp
{
    public partial class MainPage : UserControl
    {
            //eelpool mainitud ühendus klassi konstruktor
        Veebiteenus1SoapClient proksi = new Veebiteenus1SoapClient();
        public MainPage()
        {
            InitializeComponent();
            //On vastuse kuulaja mis saadab tagasi tulevad andmed
meetodisse :“proxy_ HelloWorldCompleted“
            proksi.HelloWorldCompleted +=
            new
EventHandler<HelloWorldCompletedEventArgs>(proxy_HelloWorldKuulaja);
            //meetodi päring
            proksi.HelloWorldAsync();
        }
        void          proxy_HelloWorldCompleted(object          sender,
HelloWorldCompletedEventArgs e)
        {
            tekstiblokk.Text = e.Result.ToString();

        }
    }
}
```

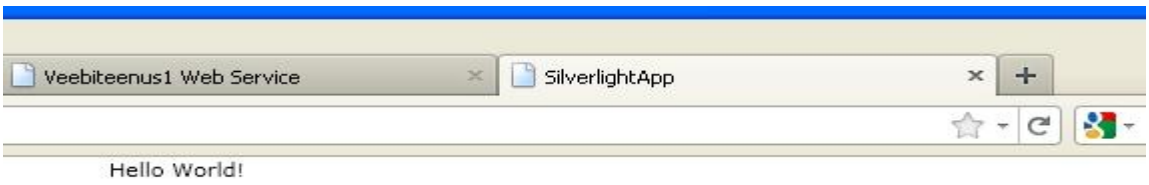
Koodinäide 8. MainPage.xaml.cs

Nagu näha loob „*MainPage*“ meetod seega ühendusklassi ja kutsub sealt asünkroonselt välja „*HelloWorld*“ meetodi, „proksi_HelloWorldKuulaja“ paneb seejärel veebiteenuselt tuleva vastuse tekstiblokki. Selline on üks tüüpiline veebiteenuse päring *Silverlightis*.

Proksi klassil on kaks liiget iga operatsiooni jaoks teenuses: asünkroonne meetod ja lõpetatud sündmus.

Visual Studio on lihtne kirjutada asünkroonseid meetodeid. Kirjuta „proksi.HelloWorldCompleted + =“ ja vajutades TABi kaks korda luuakse kuulaja automaatselt (Code Project, 2011).

Nüüd vajuta Ctrl + F5 ja ehita projekt (Joonis 13).



Joonis 13 Hello World rakendus

Seega oleme loonud veebiteenust tarvitava rakenduse.

3.4 Andmeid salvestav teenus

See teenus on väga lihtne ja ei salvesta mingeid andmeid, asi on veebiteenuste disainis nimelt eeldavad teenused olekuta (*stateless*) veebikeskkonda kus päringuid tegevatel klientidel ei tehta vahet ja igale päringule vastab omaette instants. See on mõeldud vältimaks igasugu keerukusi mis muul juhul interneti kaudu andmeid vahetades tekivad. Asünkroonsed päringud on märksa töökindlamad.

Muuhulgas tähendab see ka seda, et veebiteenused ei soovi kunagi saata päringuid kliendile (kuigi ka see on võimalik), märksa lihtsam on reaalaja aplikatsiooni vaatenurgast kui kõik päringud sealhulgas ka andmete värskendamine toimuksid kliendipoolses koodis ja teenust kasutada vaid andmesideme ülalpidamiseks (Webucator, 2011).

Vaatame veelkord meie veebiteenust ja lisame sellele koodinäite 9.

```
static int loend=0;
```

Koodinäide 9. loend

„*Static*“ integeri ees tähendab, et andmed jäävad samasse seisundisse iga päringu järel.

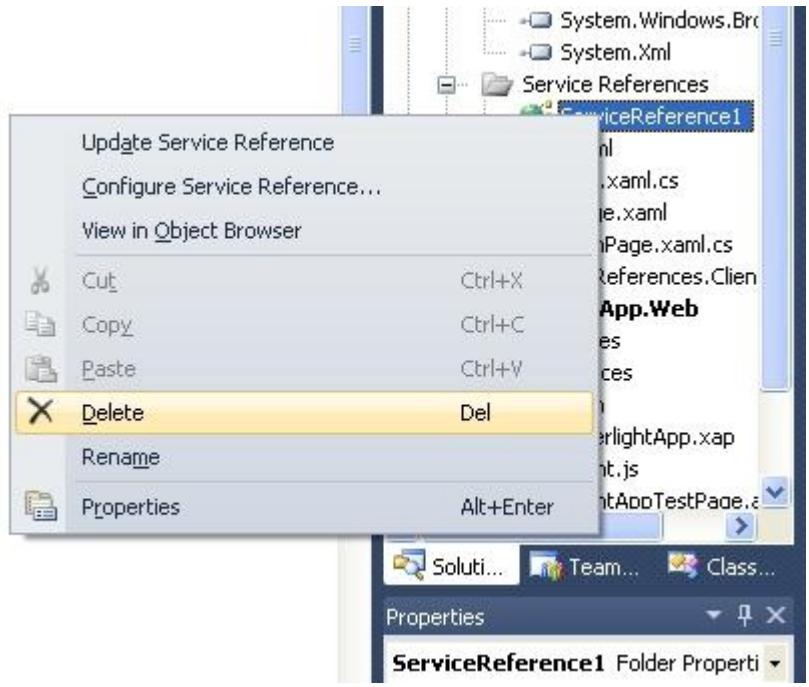
Loome uue *Web Methodi* (Koodinäide 10).

```
public class Veebiteenus1 : System.Web.Services.WebService
{
    static int loend = 0;
    [WebMethod]
    public string HelloWorld()
    {
        return "Hello World";
    }
    [WebMethod]
    public int loendaja()
    {
        loend++;
        return loend;
    }

}
```

Koodinäide 10. Loendaja

Uuendame nüüd teenuse viita.

Vajuta Ctrl-F5, kopeeri URL, Ava „*Service References*“ kaust *Silverlighti* rakenduses ja kustu seal asuv „*Service Reference1*“ paremklõpsuga valides menüüst *delete* ja seejärel loo uus viit valides menüüst „*Add Service Reference*“(Joonis 14).



Joonis 14 Teenuseviida kustutamine

Nüüd on meil andmeid salvestav veebiteenus.

3.5 Tuvastus

Ava „*SilverlightApp*“ ja lisa juhendi lühim koodinäide 11.

```
int id;
```

Koodinäide 11. Id

Seejärel asenda olemas olevad päringud loendaja meetodi päringuga (Koodinäide 12) ning lisa kuulajasse id-le vastust omistav käsk.

```
{

    public partial class MainPage : UserControl
    {
        int id;
        public MainPage()
        {
            InitializeComponent();
            //eelpool mainitud ühendus klassi konstruktor
            Veebiteenus1SoapClient         proksi         =         new
Veebiteenus1SoapClient();
            //On  vastuse  kuulaja  mis  saadab  tagasi  tulevad  andmed
      meetodisse :"proksi_loendajaKuulaja "
            proksi.loendajaCompleted +=
            new
EventHandler<loendajaCompletedEventArgs>(proksi_loendajaKuulaja);
            //meetodi päring
            proksi.loendajaAsync();
        }
        void           proksi_loendajaKuulaja(object           sender,
loendajaCompletedEventArgs e)
        {
            id = e.Result;
            tekstiblokk.Text = e.Result.ToString();

        }
    }
}
```

Koodinäide 12. Loendaja päring ja kuulaja

3.6 Mänguteenus

Nüüd suudab veebiteenus klientide vahel vahet teha. Selle abil suudame luua rakendusi mis on võimelised veebiteenuste kaudu nii omavahel kui ka muu tarkvaraga suhtlema. Loome niisiis kahe mängijaga nupuvajutus mängu mis saadab teenusele teate kasutaja käitumise kohta ja saab tagasi info mängu seisu kohta.

Selle komponendid on veebiteenuses meetodid:

- Mang
- loendaja
- Skoor 1 ja 2
- Lopp

Mang meetod tegeleb sellega mis juhtub kui teenus saab teate kliendi nupuvajutuse kohta, skoor 1 ja 2 lihtsalt tagastavad kliendile vastavad arvud ja loendaja peab arvet selle kohta kui mitu ühendust on veebiteenus vahepeal loonud. Lopp seab kõik staatilised variaablid null ja alustab protsessi algusest peale.

Töökäik peaks olema rakendusel järgmine, kui üks klient ja teine klient on ühendust võtnud ja id saanud ignoreeritakse järgnevaid päringuid mäng algab ja ootab nupule vajutamise sündmust mis tõstab skoori seni kuni tuleb lõpu teade ja skoor ning id jagaja sätitakse nulli. Selle sündides saavad ka kliendid teate oma id nulli seada.

Loo *Web Method* Mang ja kaks *static integer*i skoor1 ja skoor2(Koodinäide 13)

```
static int skoor1 = 0;
static int skoor2 = 0;
[WebMethod]
public int Mang(int id)
{
    if (id == 1)
    {
        skoor1++;
    }
    else if (id == 2)
    {
        skoor2++;
    }
    return id;
}
```

Koodinäide 13. Täpitähti tuleks vältida

See võimaldab kahel esimesel saabujal võidu päringuid saata.

Loo ka meetodid skoor_1 ja skoor_2 nende võidusõidu jälgimiseks (Koodinäide 14).

```
[WebMethod]

public int skoor_1()
{
    return skoor1;
}
[WebMethod]

public int skoor_2()
{
    return skoor2;
}
```

Koodinäide 14.Skoorid

Ilmselt tekib siin olukord kus üks mängijatest lõpetab varem kui teine, lisa meetod lõpp (Koodinäide 15).

```
[WebMethod]
public void lopp()
{
    loend = 0;
    skoor1 = 0;
    skoor2 = 0;
}
```

Koodinäide 15.Lopp

See viib teenuse taas null seisu, see aga tekitab probleemi.

Mängija kes ei katkesta ühendust võib mängu jätkata ja teenus ei tee vahet uuel ja vanal id omanikul. Lihtne lahendus on lisada koodinäide 16.

```
static Boolean manglabi = true;
```

Koodinäide 16.*Boolean*

Mis muutub loendaja meetodis (Koodinäide 17).

```
[WebMethod]
public int loendaja()
{
    loend++;
    if (loend >= 3)
    {
        Boolean manglabi = false;
    }
    return loend;
}
```

Koodinäide 17.loendaja 2

Ja Mang meetodis (Koodinäide 18).

```
[WebMethod]
public int Mang(int id)
{
    if (manglabi == false)
    {
        if (id == 1)
        {
            skoor1++;
        }
        else if (id == 2)
        {
            skoor2++;
        }
    }
    else if (id == 1)
    {
    }
    else { id = 0; }

    return id;
}
```

Koodinäide 18.Mang 2

Mis väldib muuhulgas ka seda, et ootav esimene mängija oma eelist väärkasutaks.

Ning lõpuks lõpp meetodis (Koodinäide 19).

```
[WebMethod]

  public void lopp()
  {
      loend = 0;
      skoor1 = 0;
      skoor2 = 0;
      manglabi = true;
  }
```

Koodinäide 19.Lopp 2

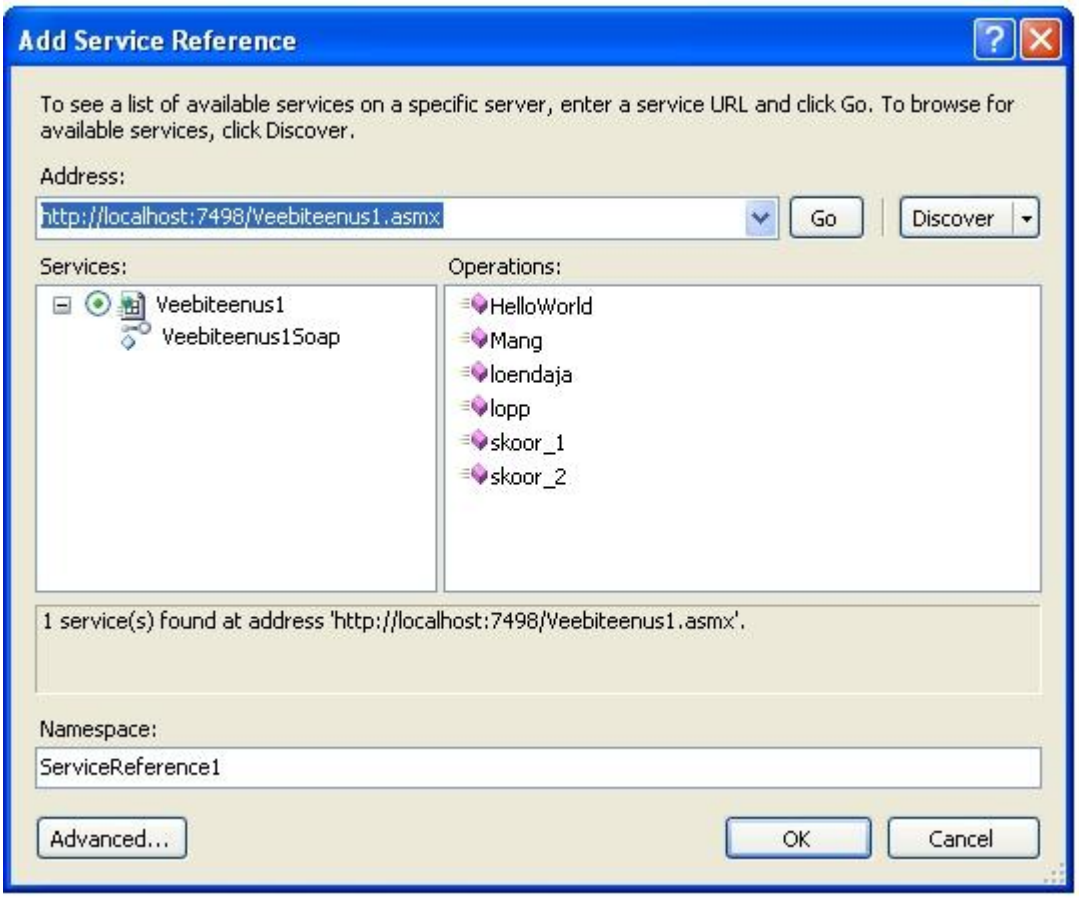
See on siis meie võidupäringu mäng, mis loob andmevahetuse kahe kliendi vahel. Ehitame selle Ctrl-F5ga (Koodinäide 20).

```
static int loend=0;
      static Boolean manglabi = true;
      [WebMethod]
      public int HelloWorld()
      {
          return loend;
      }
      [WebMethod]
      public int loendaja()
      {
          loend++;
          if (loend >= 3)
          {
              manglabi = false;
          }
          return loend;
      }
      static int skoor1 = 0;
      static int skoor2 = 0;
      [WebMethod]
      public int Mang(int id)
      {
          if (manglabi == false)
          {
              if (id == 1)
              {
                  skoor1++;
              }
              else if (id == 2)
              {
                  skoor2++;
              }
          }
          else if (id == 1)
          {
          }
          else { id = 0; }

          return id;
      }
      [WebMethod]
      public int skoor_1()
      {
             return skoor1;
        }
      [WebMethod]

      public int skoor_2()
      {
             return skoor2;
        }
      [WebMethod]
      public void lopp()
      {
          loend = 0;
          skoor1 = 0;
          skoor2 = 0;
          manglabi = true;
      }}}
```

Koodinäide 20.Veebiteenus

3.7 Mängu rakendus

Nüüd on aeg luua sellele kasutaja liidesed. Mine *Silverlighti* rakendusse ja uuenda „*Service Reference1*“ sarnaselt varasemaga (Joonis 15).



Joonis 15 Teenuseviida aken 2

Ilmselt ei ole kasutaja enda poolt tuleva sisendita asjal erilist mõtet seega paneme *MainPage.xaml*-il e nupu ja lisame 2 tekstblokki (Koodinäide 21).

```
    <Grid x:Name="LayoutRoot" Background="White">
        <TextBlock Height="20" HorizontalAlignment="Left"
        Name="tekstiblokk" Text="sisu" VerticalAlignment="Top"/>
        <TextBlock Height="20" HorizontalAlignment="Center"
        Name="tekstiblokk2" Text="sisu" VerticalAlignment="Top"
Margin="39,0,339,0"/>
        <TextBlock Height="20" HorizontalAlignment="Right"
        Name="tekstiblokk3" Text="sisu" VerticalAlignment="Top"
Margin="0,0,300,0"/>
        <Button Name="b1" Click="b1_Click" Margin="0,26,300,224">Vajuta
Mind!</Button>
    </Grid>
```

Koodinäide 21.*MainPage.xaml 2*

Lisame *MainPage.xaml.cs* uued kuulajad (Koodinäide 22).

```
Veebiteenus1SoapClient proksi = new Veebiteenus1SoapClient();

public MainPage()
{
       InitializeComponent();
       proksi.loendajaCompleted                          +=                          new
       EventHandler<loendajaCompletedEventArgs>(proksi_loendajaKuulaja);
       proksi.MangCompleted                             +=                          new
       EventHandler<MangCompletedEventArgs>(proksi_MangKuulaja);
       proksi.skoor_1Completed                           +=                          new
       EventHandler<skoor_1CompletedEventArgs>(proksi_skoor_1Kuulaja);
       proksi.skoor_2Completed                           +=                          new
       EventHandler<skoor_2CompletedEventArgs>(proksi_skoor_2Kuulaja);
       proksi.loppCompleted                             +=                          new
       EventHandler<System.ComponentModel.AsyncCompletedEventArgs>(proksi_loppKuu
       laja);
       proksi.loendajaAsync();
 }
```

Koodinäide 22.deklareerimine

Ning nende meetodid: Alguse meie nupu kuulaja (Koodinäide 23).

```
     private void b1_Click(object sender, RoutedEventArgs e)
        {
            proksi.MangAsync(id);
        }
```

Koodinäide 23.Nupusündmus

Mis annab vastuse MangKuulajas(Koodinäide 24).

```
         void proksi_MangKuulaja(object sender, MangCompletedEventArgs e)
        {
            proksi.skoor_1Async();
            proksi.skoor_1Async();

        }
```

Koodinäide 24.MangKuulaja

Mis kutsub skoori meetodi mille kuulaja on koodinäites 25:

```
        void proksi_skoor_1Kuulaja(object sender, skoor_1CompletedEventArgs e)
        {
            tekstiblokk2.Text ="Mangja 1 Skoor:"+ e.Result.ToString();
        }
        void proksi_skoor_2Kuulaja(object sender, skoor_2CompletedEventArgs e)
        {
            tekstiblokk3.Text = "Mangja 2 Skoor:" + e.Result.ToString();
        }
```

Koodinäide 25.Skoorikuulajad

Lõpuks võiks veel parandada vana loendaja kuulajat (Koodinäide 26).

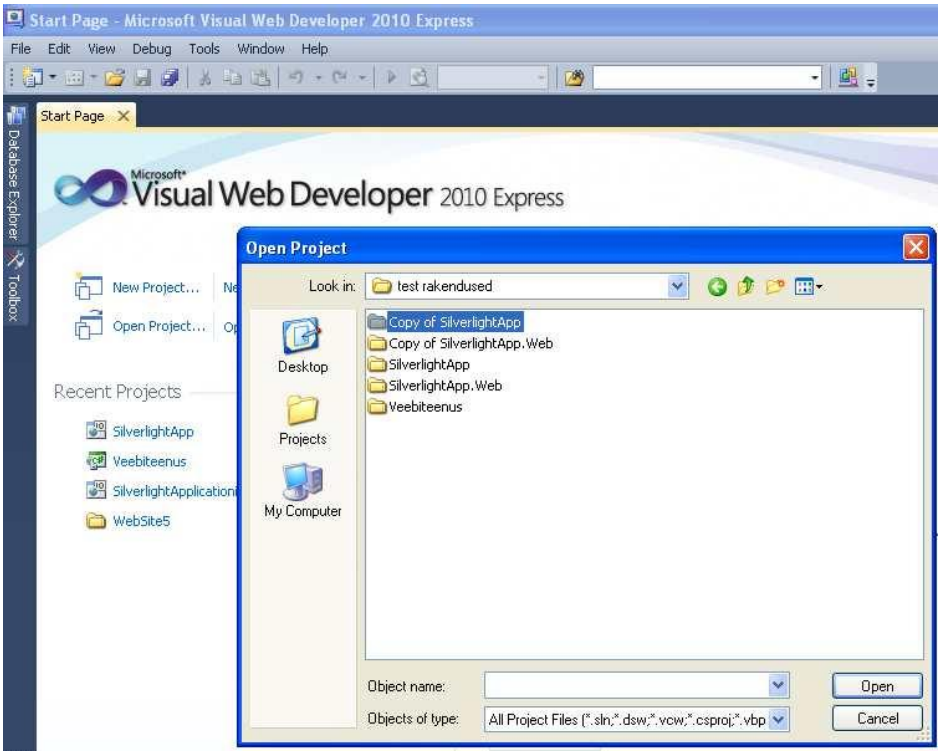
```
 void proksi_loendajaKuulaja(object sender,loendajaCompletedEventArgs e)

{
    if (e.Result == 1 || e.Result == 2) { id = e.Result; }
    else { id = 0; }

    tekstiblokk.Text = e.Result.ToString();
}
```

Koodinäide 26.Id Kuulaja

Et see ühenduseta kliendid ei tekitaks liigset segadust.

Siis on veel järgi jäänud lopp kuulaja.

Kahjuks ei saa Silverlight väljalülitamisel veebiteenust kutsuda, seega tuleb probleemi lahenduseks javaskriptiga. See on keeruline ettevõtmine ja igaks juhuks tasuks rakendust siinkohal juba katsetada, kui kõik tundub töökorras võid teha senisest tööst koopia ja avada kolmanda *Visual Studio* akna, et rakendust ilma lõpukuulajata katsetada (Joonis 16). Ole ettevaatlik, et veebiteenuse loendaja sind välja ei sulgeks, loendaja läheb nulli testserveri taaskäivitamisel Ctrl-F5-ga. Samuti ära muuda koopia projekti koodi kuna see võib *Visual Studio* segadusse ajada, ennem kui koodi kallale tagasi lähed sulgeaken kindlasti.



Joonis 16 Open project: copy of SilverlightApp

3.8 Lõpukuulaja

Esiteks tuleb tuvastada *Silverlight* rakendus oma veebilehe keskkonnas selle jaoks ava *App.xaml.cs.*

Ja lisa loetellu koodinäide 27.

```
using System.Windows.Browser;
```

Koodinäide 27. Viide

ja „*Application_Startup*“ meetodisse koodinäide 28.

```
        private void Application_Startup(object sender, StartupEventArgs
e)
        {
            MainPage mainpage = new MainPage();
            HtmlPage.RegisterScriptableObject("mApp", mainpage);

            this.RootVisual = mainpage;
      }
```

Koodinäide 28. Application_Startup

mis teavitab *javaskripti* „*mainpage*“-il asuvast (veel mitte olemasolevast) meetodist.

Lähme loome selle nüüd, „MainPage.xaml.cs“-is lisa koodinäide 29 ja 30.

```
using System.Windows.Browser;
```

Koodinäide 29. Viide 2

```
       [ScriptableMember]
        public void Lopputoo()
        {
            if (id != 1 || id != 2)
            {
                proksi.loppAsync();
            }
            else { HtmlPage.Window.CreateInstance("lubasulgeda"); }

        }
```

Koodinäide 30. Lopppäring

Seejärel ava *SilverlightAppTest.asmx* rakendust ülalpidaval veebilehel lisa sinna koodinäide 31.

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

window.onbeforeunload = enneExit;
        var blocking = true;
        function enneExit() {
            var Appi = document.getElementById("Appi")
            Appi.Content.mApp.Lopputoo();
            while (blocking)
                alert('Katkestan ühenduse');
        }
        function lubasulgeda() {
            blocking = false;
        }
```

Koodinäide 31.Jaavaskript

Ja allpool lisa id=“Appi“(Koodinäide 32)

```
<div id="silverlightControlHost">
<object          id="Appi"          data="data:application/x-silverlight-2,"
type="application/x-silverlight-2" width="100%" height="100%">
```

Koodinäide 32.*SilverlightControllHost*

Mine veelkord *Mainpage.xaml.cs*-i ja lisa lopp kuulajale

```
void proksi_loppKuulaja(object sender,
System.ComponentModel.AsyncCompletedEventArgs e)
{
      HtmlPage.Window.CreateInstance("lubasulgeda");
}
```

Koodinäide 33.Lõpukuulaja

Ja valmis! Vajuta Ctrl-F5t ja naudi põnevat nupu vajutust !

Mine *Visual Studio* projekti kausta ja tee sealsest SilverlightAppist koopia ja ava see eraldi instantsis, senikaua kuni sa koodi ei muudeta suudavad mõlemad koos mängida.

Ülesanded

1. Loo kliendis meetod mida kutsutakse iga viie sekundi tagant ja mis omakorda kutsub skoor_1 ja skoor_2 meetodeid.
2. Loo teenuses võidutingimus, kes suudab esimesena kakskümmend klikki saada saab teateks kiituse, kes jääb teisele kohale kaastundeavalduse.
3. Kasutades *static ArrayList*-i loo tahvli teenus kuhu saab kirjeid lisada ja kuvada.
4. Asenda id *integer* kasutaja poolt sisestatud stringiga mida kasutatakse rakenduse veebiteenusele tuvastamiseks.

4.Tulemused ja järeldused

Rakenduse koostamise jooksul õppisin kuidas *Silverlight* klient saab veebi kaudu andmeid pärida ja talletada ja kuidas luua seeläbi ühendusi teiste rakendustega. Selline teadmine võiks olla veebi rakenduse programmeerimisel väga kasulik, kusjuures arusaamine selle kohta, kuidas veebiteenused rakendada, ei puutu üksnes *Silverlighti*.

Kahjuks tekkisid antud juhendiga probleemid mis puutus arusaadavusse, intervjueerides erapooletut kasutajat selgus, et: Seletus tarkvara paigalduse kohta oli ebapiisav. Sellele oleks võinud samuti pildid lisada, koodinäited ei olnud piisavalt hästi ülejäänud tekstist eraldatud töökäigu seletus oli raskesti järgitav ja õpitu kohta üldist kirjeldust oli vähe.

Need vead said töö käigus mõnevõrra leevendatud, kuid järgmisel katseproovil ilmnesid ka vead rakenduse koodinäidetes mis tegi nende väärtuslikkuse määramise kaheldavaks. Antud töövariandis on need vead parandatud ja lisaks veel õppejõu soovitusel teemad veel laiemalt jaotatud ja kirjeldatud, loodetavasti viimaks vabanedes eelpool mainitud vigadest.

Järeldada võib sellest loomulikult seda, et õppematerjali koostamisel tuleks pöörata vähemalt samapalju tähelepanu selle korrapärasusele ja arusaadavusele kui selle ka sisu valdamisele.

Kokkuvõte

Töö jooksul õppisin looma *Silverlighti* rakendust mis tõmbavad andmeid võrgust reaalajas ja mis suudavad andmeid talletada veebiteenuses ning see kaudu teiste rakendustega suhelda.

Küsimusel kas on võimalik luua *Silverlightis* reaalajas toimiv andmevahetus serveriga on vastus seega jaatav. Uurimata jäid teised standardid selle saavutamiseks, kuid eesmärgi saavutamiseks olid veebiteenused igati piisavad. Kindlasti on nad üks lihtsaimaid viise andmevahetuse saavutamiseks võrgu kaudu. Tõhusamaks võib ehk pidada *Silverlighti socket*eid , kuid nendega tegelemiseks ei jäänud aega. Antud teemal leidub üllatavalt vähe õppematerjali ja eesti keelset ilmselt ainult ENETA näidised.

Selle kohta, kuidas toimub ühendus *Silverlighti* ja veebiteenuste vahel õppisin, et praktiliselt on ühendus olematu. *Silverlighti* klient saadab ja pärib kõiki andmeid veebiteenuselt asünkroonselt. Igasugune tuvastus tuleb arendajal luua.

Õnnestus arendada lihtne viis kuidas luua sidet kahe *Silverlight* rakenduse vahel veebiteenuse vahendusel.

Halvast küljest ei õnnestunud seda infot eriti hästi juhendis talletada ja antud variant jäi katserühmaga järelproovimata.

Tulevase töö suunaks võiks olla teised võimalikud andmesideme loomise meetodid, võiks ka lähemalt uurida veebiteenuse võimalusi.

Kasutatud kirjandus

Code Project. vaadatud 7.november, 2011,aadressil:

http://www.codeproject.com/KB/silverlight/AccessWebService.aspx

E-teatmik, vaadatud 4.jaanuar, 2012, aadressil:

http://vallaste.ee/

Microsoft Development Network, vaadatud 2.jaanuar, 2012,aadressil:

http://msdn.microsoft.com/en-us/library/cc189047%28v=vs.95%29.aspx,

http://msdn.microsoft.com/en-us/library/ms731082.aspx ja

http://msdn.microsoft.com/en-us/library/ms752059.aspx

Definition: Rich Internet Application (RIA) , vaadatud 2,jaanuar, 2012, aadressil:

http://searchsoa.techtarget.com/definition/Rich-Internet-Application-RIA

Silverlight FAQ, vaadatud 2.jaanuar , 2012,aadressil:

http://forums.silverlight.net/p/95440/218611.aspx/1?Silverlight+General+FAQ

Visual Web Developer 2010 Express: System Requierments, vaadatud 3.jaanuar, 2012, aadressil:

http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-web-developer-express

Webucator, vaadatud 10. september,2011, aadressil:

http://www.learn-silverlight-tutorial.com/NetworkingInSilverlight.cfm

Web Service tutorial, vaadatud 7. november,2011, aadressil:

http://www.wstutorial.com/what-is-web-services/