Tuomas Liikala

**Palvelinohjelmointi 1**

oppilas13

Harjoitustyö  
Syksy 2011  
Tekniikan yksikkö  
Tietotekniikan koulutusohjelma

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Koulutusyksikkö: Tekniikka

Koulutusohjelma: Tietotekniikan

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Tekijä: Tuomas Liikala

Työn nimi: Palvelinohjelmointi 1

Ohjaaja: Lahti Markku

ADO, ASP, SQL, C#, tietokanta, DB, ohjelmointi

Vuosi: 2011 Sivumäärä: Liitteiden lukumäärä:

Harjoitustyö ADO.- ja ASP. NET: in käytöstä palvelinohjelmointi 1 -kurssia varten.

SISÄLTÖ

[Opinnäytetyön tiivistelmä 1](#_Toc308972805)

[SISÄLTÖ 2](#_Toc308972806)

[MASTER/CONTENT-rakenne 3](#_Toc308972807)

[ASP.NET 5](#_Toc308972808)

[1.1 RequiredFieldValidator 5](#_Toc308972809)

[1.2 RangeValidator 6](#_Toc308972810)

[1.3 RegularExpressionValidator 7](#_Toc308972811)

[1.4 CustomValidator 8](#_Toc308972812)

[1.5 CompareValidator 9](#_Toc308972813)

[1.6 Tiedon lähettäminen 10](#_Toc308972814)

[1.7 Tiedon vastaanottaminen 11](#_Toc308972815)

[2 ADO.NET 12](#_Toc308972816)

[2.1 Tietokannan taulut 12](#_Toc308972817)

[2.1.1 KCTITE09\_auto 12](#_Toc308972818)

[2.1.2 KCTITE09\_valmistaja 14](#_Toc308972819)

[2.1.3 KCTITE09\_perustaja 15](#_Toc308972820)

[2.2 Ohjelman toiminta 17](#_Toc308972821)

[2.2.1 Tietokannasta hakeminen 18](#_Toc308972822)

[2.2.2 Liitoskyselyn tekeminen 20](#_Toc308972823)

[2.2.3 Tietokantaan päivittäminen 21](#_Toc308972824)

[2.2.4 Tietokantaan lisääminen 22](#_Toc308972825)

[2.2.5 Tietokannasta poistaminen. 22](#_Toc308972826)

[2.2.6 Tietokannasta hakeminen ehdoilla 22](#_Toc308972827)

[LIITTEET 24](#_Toc308972828)

MASTER/CONTENT-rakenne

Sivu toimii Site.Master-tiedostosta, jossa on Sitemap ja kaikki muukin sivulla näkyvä informaatio sulautettuna. Rivit alkavat muotoa:

<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="Site.master.cs" Inherits="Harjoitustyo.Site" %>

Samoin myös esimerkiksi ASP.aspx-tiedoston alussa on hieman samanlainen rivi:

<%@ Page Language="C#" MasterPageFile="~/Site.Master" AutoEventWireup="true" CodeBehind="ASP.aspx.cs" Inherits="Harjoitustyo.ASP" %>

Erotuksena on vain, että jälkimmäinen on ensimmäisen sisällä.

Sivuilla näkyy aina sama otsikko ”Harjoitustyö” ja seamk-logo. Tämän lisäksi vasemmalle piirretään TreeView-valikko, joka luonnollisesti päivittyy sivuja selatessa.

<asp:TreeView ID="TreeView1" runat="server"

DataSourceID="SiteMapDataSource1">

</asp:TreeView>

Tärkein ja aina muuttuva sivu on sivu oikealla reunalla olevassa ContentPlaceHolderissa:

<asp:ContentPlaceHolder ID="MainContent" runat="server">

</asp:ContentPlaceHolder>

Lopuksi loppuun keskelle tulee copyright-teksti:

<p align=center>Copyright&#169 2011 Tuomas Liikala</p>

Web.sitemap sisältää seuraavanlaisen koodin:

<?xml version="1.0" encoding="utf-8" ?>

<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >

<siteMapNode url="default.aspx" title="Home" description="">

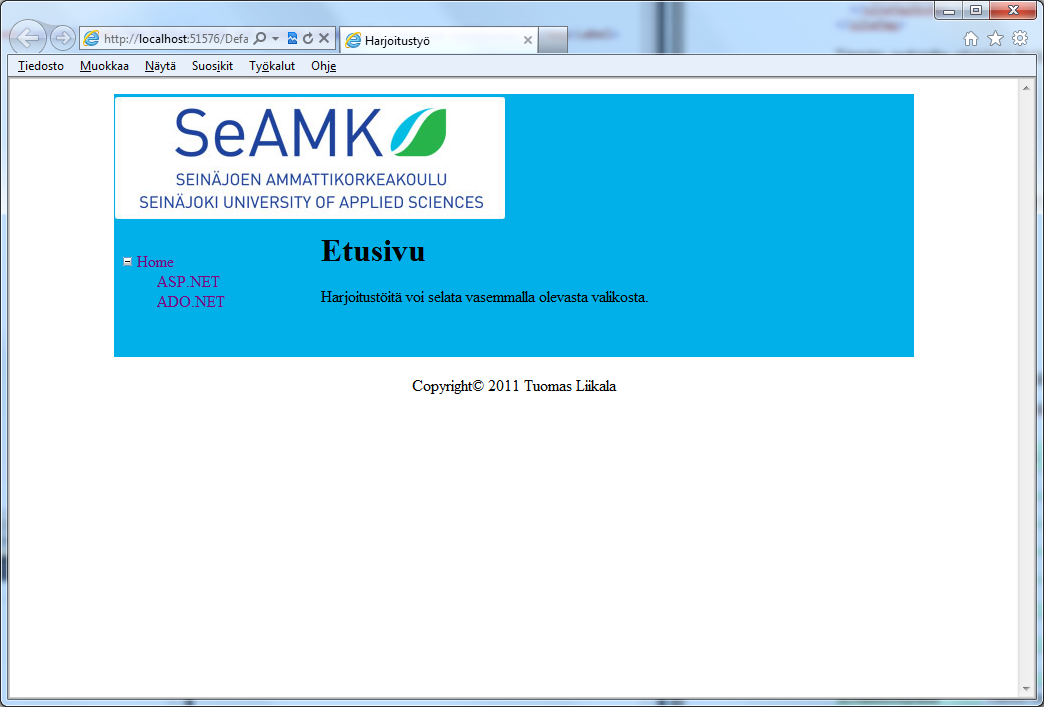
<siteMapNode url="ASP.aspx" title="ASP.NET" description="" />

<siteMapNode url="ADO.aspx" title="ADO.NET" description="" />

</siteMapNode>

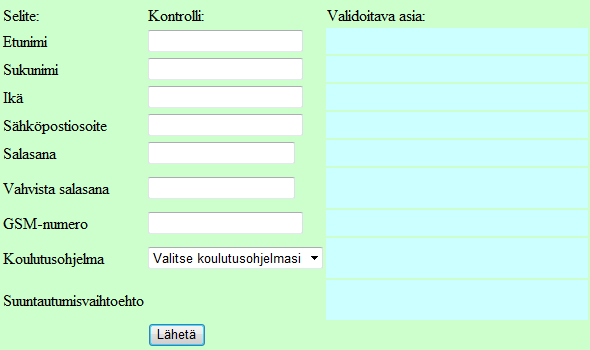
</siteMap>

Tämän pohjalta ohjelma tietää mitä sivuja tulee näyttää milloinkin. Oletussivuna toimii default.aspx.



ASP.NET

Ohjelma toimii taulukon sisällä, jossa on vasemmassa sarakkeessa selite, keksellä kontrolli ja oikealla validoitava asia.



Kun syötteessä huomataan validoitaessa virheen, niin ohjelma ilmoittaa siitä validoitavan asian Validationin ErrrMessageen ja myös taulun ensimmäisessä solussa olevassa ValidationSummaryssä .

* 1. RequiredFieldValidator

Ohjelmassa täytyy täyttää tekstikentät; tb\_name ja tb\_sukunimi. Tämä validointi toteutetaan seuraavanlaisesti:

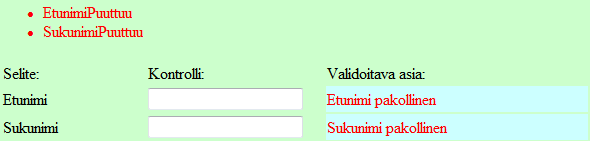
<asp:RequiredFieldValidator ID="rfv\_name" runat="server"

ControlToValidate="tb\_name" ErrorMessage="EtunimiPuuttuu">Etunimi pakollinen</asp:RequiredFieldValidator>

<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"

ControlToValidate="tb\_sukunimi" ErrorMessage="SukunimiPuuttuu">Sukunimi pakollinen</asp:RequiredFieldValidator>

Kuten koodistakin näkyy, niin ohjelma antaa nämä tiedot tyhjäksi jätettäessä virheilmoitukse ”Etunimi puuttuu” tai ”Sukunimi puuttuu” riippuen kumman kentän on jättänyt tyhjäksi.



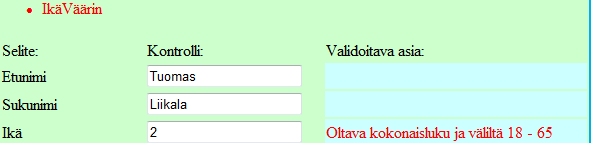
* 1. RangeValidator

Seuraavana on vuorossa validaattori, jolla voidaan vaatia täytetyn kentän olevan joltain lukuväliltä. Me haluamme iän olevan väliltä 18 – 65 vuotta. Tämä tapahtuu RangeValidattorilla seuraavasti:

<asp:RangeValidator ID="RangeValidator1" runat="server"

ControlToValidate="tb\_ika" ErrorMessage="IkäVäärin" Type="Integer" MinimumValue="18" MaximumValue="65">Oltava kokonaisluku ja väliltä 18 - 65</asp:RangeValidator>

Jos ilmoitettu tekstikenttä on validoitaessa pienempi kuin MinimumValue:n arvo tai suurempi kuin MaximumValue:n arvo, niin silloin ohjelma sanoo: ”Oltava kokonaisluku ja välilät 18 - 65”. Myöskin tulee virheilmoitus: ”IkäVäärin”.



* 1. RegularExpressionValidator

Ensimmäinen hieman mielenkiintoisempi Validaattori on RegularExpressionValidator, jolla voidaan tehdä hieman dynaamisempia tarkastuksia. Tässä ohjelmassa me haluamme tarkistaa annetun sähköpostiosoitteen olevan muotoa: [abc@xyz.fi](mailto:abc@xyz.fi), joka tapahtuu koodissa seuraavasti:

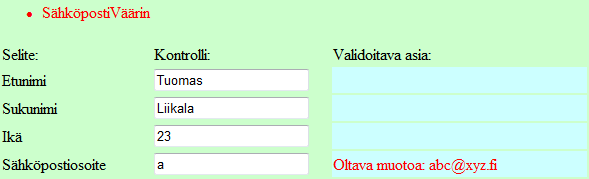
<asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"

ControlToValidate="tb\_sähköposti" ErrorMessage="SähköpostiVäärin"

ValidationExpression="\w+([-+.']\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*">Oltava muotoa: [abc@xyz.fi</asp:RegularExpressionValidator](mailto:abc@xyz.fi%3c/asp:RegularExpressionValidator)>

ValidationExpress on Visual Studion oma ehdotus tarkistaa sähköpostiosoitteen oikeellisuus. Tämän kryptisen merkkijonon merkitys jää ylempien tietoisuuksien salaisuudeksi.

Myös tästä väärin annetusta sähköpostiosoitteesta tulee viesti ”Oltava muotoa: [abc@xyz.fi](mailto:abc@xyz.fi)”, kuten muistakin vääristä validaatioista. Samoin tulee virheilmoitus: ”SähköpostiVäärin”.



Samoin myös tarkastamme, että annettu GSM-numero vastaa muotoa 999-1234567:

<asp:RegularExpressionValidator ID="RegularExpressionValidator2" runat="server" ErrorMessage="GSMväärin" ControlToValidate="tb\_GSM" ValidationExpression="([0123456789][0123456789][0123456789]){1}(\-)?[^0\D]{1}\d{6}$" Text="Oltava muotoa: 999-1234567"></asp:RegularExpressionValidator>

Käyttäjälle se näyttää seuraavalta:



* 1. CustomValidator

Tämä on mitä luultavimmin kaikista dynaamisin validaattori. Validaattorilla voidaan suorittaa oma funktionsa, joka palauttaa tiedon, onko annettu tieto tai jokin muu asia ollut oikea.

Näitä käytetään validoimaan salasana, alavetovalikko koulutusohjelmaa valinta ja suuntautumisvaihtoehto.

Näin itse validoinnin on määritelty. tässä ensimmäisenä salasana:

<asp:CustomValidator ID="cvalSalasana" ErrorMessage="SalasanaVäärin" runat="server" ControlToValidate="tb\_salasana" OnServerValidate="valSalasana\_serverValidate" Text="Oltava vähintään 8 merkkiä pitkä"/>

Seuraavaksi alavetovalikko:

<asp:CustomValidator ID="CustomValidator2" runat="server" ControlToValidate="DropDownList3" OnServerValidate="valddl\_serverValidate" ErrorMessage="KoulutusohjelmaVäärin" Text="Tarkistettava, että tehdään valinta (valinta pakollinen)"/>

Viimeisenä RadioButtonList:

<asp:CustomValidator ID="CustomValidator3" runat="server" ControlToValidate="RadioButtonList1" OnServerValidate="valddl\_serverValidate2" Text="Tarkistettava, että tehdään valinta (valinta pakollinen)" ErrorMessage="SuuntautuminenVäärin" />

Näissäkin ohjelma antaa virheilmoituksen ja virhetekstin, kuten muissa aiemmissa validaattoreissa. Suurimpana erona on OnServerValidate, jolla kutsutaan scriptiä, jolla taas suoritetaan validointi sen mukaan, että palauttaako koodi ”true”- vai ”false”-arvon. Itse scripti näyttää seuraavalta:

<script runat="server">

void valSalasana\_serverValidate(Object source, ServerValidateEventArgs args)

{

if (args.Value.Length >= 8)

args.IsValid = true;

else args.IsValid = false;

}

void valddl\_serverValidate(Object source, ServerValidateEventArgs args)

{

if (DropDownList3.SelectedIndex > 0 && DropDownList3.SelectedIndex <= DropDownList3.Items.Count)

args.IsValid = true;

else args.IsValid = false;

}

void valddl\_serverValidate2(Object source, ServerValidateEventArgs args)

{

if (RadioButtonList1.SelectedIndex >= 0 && RadioButtonList1.SelectedIndex < RadioButtonList1.Items.Count)

args.IsValid = true;

else args.IsValid = false;

}

</script>

Mielenkiintoisinta on huomata, että CustomValidate toimii vasta viimeisenä. Jos jättää GSM-numeron ja salasanan vahvistuksen pois ja täyttää vain muut tiedot, niin silloin ohitetaan validointi.

* 1. CompareValidator

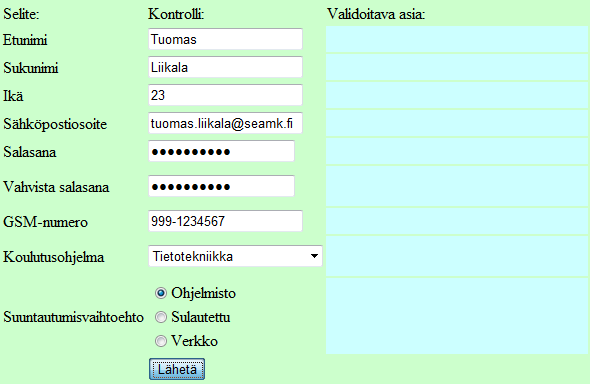
Tämä on taas yksi hyvin simppeli validaattori, joka vain testaa, että annettu kenttä vastaa toista kenttää. Koodina se käy näin:

<asp:CompareValidator ID="CompareValidator1" runat="server" ErrorMessage="VahvistusVäärin" ControlToValidate="tb\_vahvistus" ControlToCompare="tb\_salasana">Syötetyn tiedon on vastattava salasana-kentän arvoa</asp:CompareValidator>

tb\_Vahvista on validoitava kontrolli, jonka takia se on annettu ControlToValidate:n sisälle. Sen on vastattava kenttää tb\_salasana, jonka takia se taas on ControlToCompare:n sisällä. Jos nämä eivät täsmää toisiinsa, niin silloin tulee teksti: ”Syötetyn tiedon on vastattava salasana-kentän arvoa. Virheviestinä tulee: ”VahvistusVäärin”.



* 1. Tiedon lähettäminen



Kun kaikki kentät on oikein täytettynä ja painetaan ”Lähetä”-nappia, niin ohjelma siirtyy btnValitaClick-funktion kautta lähettämään tietoa Server.Transferilla. Ohjelmassa on luotuna julkiset funktiot tiedon lukemiseen, jotta toisessa päässä tiedon lukeminen tiedot omistavasta päästä olisi ylipäätänsä mahdollista:

public TextBox pp\_parametri

{

get { return tb\_name; }

}

Kaikki muutkin tiedot välitetään samanlaisilla funktioilla.

* 1. Tiedon vastaanottaminen

Tietoja vastaanottaessa vain kopioidaan tiedot edelliseltä sivulta ja pistetään ne näkyville:

Label1.Text = PreviousPage.pp\_parametri.Text;

Label2.Text = PreviousPage.pp\_parametri2.Text;

Label3.Text = PreviousPage.pp\_parametri3.Text;

Label4.Text = PreviousPage.pp\_parametri4.Text;

Label5.Text = PreviousPage.pp\_parametri5.Text;

Label6.Text = PreviousPage.pp\_parametri6.Text;

Label7.Text = PreviousPage.pp\_parametri7.SelectedItem.Text;

Label8.Text = PreviousPage.pp\_parametri8.SelectedValue;

Tämä näyttää käyttäjän antamista tiedoista riippuen hieman tällaiselta:



1. ADO.NET
   1. Tietokannan taulut

Harjoitustyön tietokannassa on 2-3 taulua. Taulut on nimetty seuraavasti: KCTITE09\_auto, KCTITE09\_valmistaja, KCTITE09\_perustaja. Tietokanta on toteutettu **Tekniikan-yksikön Lukki-palvelimelle** SQLServer – tietokantaan. Tauluihin kohdistuvat komennot (luonti, testiaineiston lisäys-komennot, proseduurit jne. tallennetaan txt–tiedostona)

* + 1. KCTITE09\_auto

KCTITE09\_auto - taulun luonti:

CREATE TABLE KCTITE09\_auto

(

auto\_id smallint

IDENTITY(1,1)

PRIMARY KEY CLUSTERED,

merkki varchar(50) NOT NULL,

malli varchar(50) NOT NULL,

vuosi int NOT NULL

CHECK (vuosi <= 2011 and vuosi >= 1767),

moottori varchar(50) NOT NULL,

teho int NOT NULL,

kiihtyvyys float NOT NULL,

kulutus float NOT NULL,

nopeus float NOT NULL,

kuva varchar(255)

)

’auto\_id’ on auton identifioiva muuttuja.

’merkki’ on merkkijono, joka viittaa auton valmistaneeseen yritykseen. ’KCTITE09\_valmistaja’-taulusta voidaan myös hakea viittaamalla ’vl\_nimi’-muuttujaan, jos tietokannasta haetaa valmistajan tietoja. Ohjelmassa tehdään liitoskysely, jossa nimenomaan käytetään tätä mahdollisuutta hyväksi.

’malli’ käsittää koko valmistajan iloittamaan mallia, johon kuuluu koko mallikohtainen nimi, eikä muuta autoa kuvaavaa merkkijonoa ole.

’vuosi’ on se vuosi jolloin auto mallivuosi, valmistusvuosi tai käyttöönottovuosi. Harjoitustyössä sitä ei ole sen tarkemmin määritelty, elle käyttäjä itse halua.

’moottori’ on koko auton moottoria kuvastava merkkijono.

’kiihtyvyys’, ’kulutus’ ja ’nopeus’ ovat kaikki samanlaisia liukulukuja, joilla kuvastetaan auton ominaisuuksia, joille voidaan tehdä ohjelmassa hakuja.

’teho’ on samanlainen kuin edelliset muuttujat, mutta se on vain tasalukumuuttuja. Voisi olla samoin myös liukuluku, mutta käytännössä tasaluku on riittävän tarkka kuvastamaan auton tehoa.

’kuva’ on merkkijono, joka sisältää Wikipedian kuvapankista auton kuvaan viittaavan urlin loppuosan. Alkuosa on muotoa: <http://upload.wikimedia.org/wikipedia/commons/>

Esiluodaan tauluun malliksi muutamia autoja:

insert into KCTITE09\_auto(merkki, malli, vuosi, moottori, teho, kiihtyvyys, kulutus, nopeus, kuva)

values ('Toyota', 'Corolla Levin', 1996, '1.6l 4A-GE (20v) "Black-Top"', 165, 7.5, 7.8, 210, 'f/ff/1997-2000\_Toyota\_Corolla\_Levin\_(AE111)\_BZ-R\_coupe\_01.jpg')

insert into KCTITE09\_auto(merkki, malli, vuosi, moottori, teho, kiihtyvyys, kulutus, nopeus, kuva)

values ('Nissan', 'Skyline R33 GT-R LM', 1993, '2.6l RB26DETT', 280, 6.0, 8.9, 240, '7/7d/Nissan\_Skyline\_R33\_GT-R\_001.jpg')

insert into KCTITE09\_auto(merkki, malli, vuosi, moottori, teho, kiihtyvyys, kulutus, nopeus, kuva)

values ('GM', 'Buick Electra', 1970, '7.5l V8', 370, 7.0, 13.0, 210, 'a/a1/Buick\_Electra\_225\_1965.jpg')

insert into KCTITE09\_auto(merkki, malli, vuosi, moottori, teho, kiihtyvyys, kulutus, nopeus, kuva)

values ('Ford', 'Focus', 1998, '1.6l Zetec-SE DOCH', 100, 10.9, 6.3, 185, '8/84/Ford\_Focus.jpg')

Autoissa on pyritty valitsemaan erilaisia autoja, jotta myöhemmin ohjelmassa voidaan suorittaa mielekkäämpiä hakuja tietokannasta. Käyttäjä voi myös luoda itse uusia autoja, joka on yksi ohjelman keskeisinpiä ominaisuuksia. Samoin autoja voidaan myös poistaa ja päivittää.

* + 1. KCTITE09\_valmistaja

KCTITE09\_valmistaja – taulun luonti:

CREATE TABLE KCTITE09\_valmistaja

(

vl\_id smallint

IDENTITY(1,1)

PRIMARY KEY CLUSTERED,

vl\_nimi varchar(50) NOT NULL,

vl\_perustettu int NOT NULL

CHECK (vl\_perustettu <= 2011 and vl\_perustettu >= 1767),

vl\_perustaja varchar(50) NOT NULL,

vl\_maa varchar(50) NOT NULL,

vl\_liikevaihto decimal NOT NULL,

vl\_tyovoima int NOT NULL,

vl\_kotisivu varchar(50) NOT NULL

)

’vl\_id’ on valmistajan identifioiva muuttuja.

’vl\_nimi’-merkkijono on valmistajan yritysnimi.

’vl\_perustettu’ vuosi, jolloin yritys on perustettu.

’vl\_perustaja’ merkkijono on kertoo yrtityksen perustaneen henkilön nimen. Tällä voidaan viitata ’KCTITE09\_perustaja’-taulun ’pr\_nimi’ muuttujaan, jotta saadaan haetuksi perustajan henkilökohtaiset tiedot.

’vl\_maa’ on maa, jossa yritys on perustettu tai pääkonttorin sijaintimaa.

’vl\_liikevahto’ on desimal-muuttuja, joka kertoo yrityksen liikevaihdon.

’vl\_tyovoima’ on luku, joka ilmoittaa työvoiman määrän.

’vl\_kotisivu’ on yrityksen kotisivu muodossa: [www.yritys.com](http://www.yritys.com)

Esiluodaan tauluun malliksi muutamia valmistajia:

insert into KCTITE09\_valmistaja(vl\_nimi, vl\_perustettu, vl\_perustaja, vl\_maa, vl\_liikevaihto, vl\_tyovoima, vl\_kotisivu)

values ('Toyota', 1937, 'Sakichi Toyoda', 'Japani', 202.86, 299394, 'www.toyota.fi')

insert into KCTITE09\_valmistaja(vl\_nimi, vl\_perustettu, vl\_perustaja, vl\_maa, vl\_liikevaihto, vl\_tyovoima, vl\_kotisivu)

values ('Porsche', 1931, 'Ferdinand Porsche', 'Saksa, Stuttgart', 7.273, 11910, 'www.porsche.com')

insert into KCTITE09\_valmistaja(vl\_nimi, vl\_perustettu, vl\_perustaja, vl\_maa, vl\_liikevaihto, vl\_tyovoima, vl\_kotisivu)

values ('Ferrari', 1947, 'Enzo Ferrari', 'Italia, Maranello', 1.919, 2721, 'www.ferrari.com')

insert into KCTITE09\_valmistaja(vl\_nimi, vl\_perustettu, vl\_perustaja, vl\_maa, vl\_liikevaihto, vl\_tyovoima, vl\_kotisivu)

values ('Ford', 1903, 'Henry Ford', 'USA, Michigan', 128.954, 164000, 'www.ford.fi')

insert into KCTITE09\_valmistaja(vl\_nimi, vl\_perustettu, vl\_perustaja, vl\_maa, vl\_liikevaihto, vl\_tyovoima, vl\_kotisivu)

values ('Mercedes-Benz', 1886, 'Karl Benz', 'Saksa, Stuttgart', 0, 164000, 'www.mercedes-benz.fi')

insert into KCTITE09\_valmistaja(vl\_nimi, vl\_perustettu, vl\_perustaja, vl\_maa, vl\_liikevaihto, vl\_tyovoima, vl\_kotisivu)

values ('Opel', 1862, 'Adam Opel', 'Saksa, Rüsselsheim', 9.994, 40458, 'www.opel.fi')

insert into KCTITE09\_valmistaja(vl\_nimi, vl\_perustettu, vl\_perustaja, vl\_maa, vl\_liikevaihto, vl\_tyovoima, vl\_kotisivu)

values ('BMW', 1916, 'Franz Josef Popp', 'Saksa, Munich', 60.48, 95450, 'www.bmw.fi')

insert into KCTITE09\_valmistaja(vl\_nimi, vl\_perustettu, vl\_perustaja, vl\_maa, vl\_liikevaihto, vl\_tyovoima, vl\_kotisivu)

values ('Citroen', 1919, 'Andre Citroen', 'Ranska, Saint-Ouen', 0, 0, 'www.citroen.fi')

insert into KCTITE09\_valmistaja(vl\_nimi, vl\_perustettu, vl\_perustaja, vl\_maa, vl\_liikevaihto, vl\_tyovoima, vl\_kotisivu)

values ('GM', 1908, 'William C. Durant', 'USA, Michigan', 135.592, 209000, 'www.gm.com')

insert into KCTITE09\_valmistaja(vl\_nimi, vl\_perustettu, vl\_perustaja, vl\_maa, vl\_liikevaihto, vl\_tyovoima, vl\_kotisivu)

values ('Volkswagen', 1937, 'Ferdinand Porsche', 'Saksa, Wolfsburg', 80.251, 368500, 'www.volkswagen.fi')

insert into KCTITE09\_valmistaja(vl\_nimi, vl\_perustettu, vl\_perustaja, vl\_maa, vl\_liikevaihto, vl\_tyovoima, vl\_kotisivu)

values ('Audi', 1909, 'August Horch', 'Saksa, Ingolstadt', 35.441, 46372, 'www.audi.fi')

insert into KCTITE09\_valmistaja(vl\_nimi, vl\_perustettu, vl\_perustaja, vl\_maa, vl\_liikevaihto, vl\_tyovoima, vl\_kotisivu)

values ('Skoda', 1895, 'Vaclav Laurin & Klement', 'Tsekki', 13.5, 24714, 'www.skoda.fi')

Valmistajia valittaessa on pyritty saamaan yleisinpien autonvalmistajien tiedot. Koska käyttäjä ei voi ohjelmassa itse luoda uusia valmistajia, niin omatoiminen riittävän kattavien tietojen kerääminen oli hyvin tärkeää.

* + 1. KCTITE09\_perustaja

KCTITE09\_perustaja - taulun luonti:

CREATE TABLE KCTITE09\_perustaja

(

pr\_id smallint

IDENTITY(1,1)

PRIMARY KEY CLUSTERED,

pr\_nimi varchar(50) NOT NULL,

pr\_syntynyt int NOT NULL,

pr\_kuollut int,

pr\_omaisuus float,

pr\_kuva varchar(247) NOT NULL,

pr\_maa varchar(50) NOT NULL,

pr\_kuvaus varchar(7666) NOT NULL

)

’pr\_id’ on perustajan identifioiva muuttuja.

’pr\_nimi’-merkkijono kertoo yrityksen perustaneen henkilön nimen. Join-liitoskyselyä tehdessä käytetään tätä hyödyksi.

’pr\_syntynyt’ on perustajan syntymävuosi

’pr\_kuollut’ vuosi jolloin kyseinen henkilö on kuollut, jos niin on jo käynyt, kuten lähes kaikkie merkittävien autonvalmistajien perustajille on käynyt.

’pr\_omaisuus’ henkilön omaisuus. Tietoa ei tarvita ohjelmassa mihinkään.

’pr\_kuva’ on merkkijono, joka sisältää Wikipedian kuvapankista henkilön kuvaan viittaavan urlin loppuosan. Alkuosa on muotoa: <http://upload.wikimedia.org/wikipedia/commons/>

’pr\_kuvaus’ on lyhyt – korkeintaan 7666 merkkiä pitkä kuvaus henkilöstä.

Esiluodaan tauluun malliksi muutamia perustajia:

insert into KCTITE09\_perustaja(pr\_nimi, pr\_syntynyt, pr\_kuollut, pr\_omaisuus, pr\_kuva, pr\_maa, pr\_kuvaus)

values ('Henry Ford', 1863, 1947, 188.1, '1/18/Henry\_ford\_1919.jpg', 'USA', 'Was an American industrialist.')

insert into KCTITE09\_perustaja(pr\_nimi, pr\_syntynyt, pr\_kuollut, pr\_omaisuus, pr\_kuva, pr\_maa, pr\_kuvaus)

values ('Sakichi Toyoda', 1867, 1930, 0, '5/55/Sakichi\_Toyoda\_new.png', 'Japani', 'Was a Japanese inventor and industrialist.')

insert into KCTITE09\_perustaja(pr\_nimi, pr\_syntynyt, pr\_kuollut, pr\_omaisuus, pr\_kuva, pr\_maa, pr\_kuvaus)

values ('Ferdinand Porsche', 1875, 1951, 0, 'c/cf/Ferdinand\_Porsche.jpg', 'Itävalta', 'Was an Austrian automotive engineer and honorary Doctor of Engineering.')

insert into KCTITE09\_perustaja(pr\_nimi, pr\_syntynyt, pr\_kuollut, pr\_omaisuus, pr\_kuva, pr\_maa, pr\_kuvaus)

values ('Enzo Ferrari', 1898, 1988, 0, '2/2d/Ilario\_Bandini\_ed\_Enzo\_Ferrari.jpg', 'Italia', 'Was an Italian race car driver and entrepreneur, the founder of the Scuderia Ferrari Grand Prix motor racing team, and subsequently of the Ferrari car manufacturer.')

insert into KCTITE09\_perustaja(pr\_nimi, pr\_syntynyt, pr\_kuollut, pr\_omaisuus, pr\_kuva, pr\_maa, pr\_kuvaus)

values ('Karl Benz', 1844, 1929, 0, 'f/fd/Carl-Benz\_coloriert.jpg', 'Saksa', 'Was a German engine designer and car engineer, generally regarded as the inventor of the gasoline-powered car, and together with Bertha Benz pioneering founder of the automobile manufacturer Mercedes-Benz.')

insert into KCTITE09\_perustaja(pr\_nimi, pr\_syntynyt, pr\_kuollut, pr\_omaisuus, pr\_kuva, pr\_maa, pr\_kuvaus)

values ('Adam Opel', 1837, 1895, 0, 'd/df/Adam\_Opel.jpg', 'Saksa', 'Was the founder of the German automobile company Adam Opel AG.')

insert into KCTITE09\_perustaja(pr\_nimi, pr\_syntynyt, pr\_kuollut, pr\_omaisuus, pr\_kuva, pr\_maa, pr\_kuvaus)

values ('Franz Josef Popp', 1886, 1954, 0, '', 'Itävalta', 'Was one of three men responsible for the founding of BMW AG and the First General Director of BMW AG from 1922 to 1942.')

insert into KCTITE09\_perustaja(pr\_nimi, pr\_syntynyt, pr\_kuollut, pr\_omaisuus, pr\_kuva, pr\_maa, pr\_kuvaus)

values ('Andre Citroen', 1878, 1935, 0, 'b/b5/AndreCitroen.jpg', 'Ranska', 'Was a French industrialist. He is remembered chiefly for the make of car named after him, but also for his application of double helical gears.')

insert into KCTITE09\_perustaja(pr\_nimi, pr\_syntynyt, pr\_kuollut, pr\_omaisuus, pr\_kuva, pr\_maa, pr\_kuvaus)

values ('William C. Durant', 1861, 1947, 0, 'd/d6/WilliamCDurant.jpg', 'USA', 'Was a leading pioneer of the United States automobile industry, the founder of General Motors and Chevrolet who created the system of multi-brand holding companies with different lines of cars.')

insert into KCTITE09\_perustaja(pr\_nimi, pr\_syntynyt, pr\_kuollut, pr\_omaisuus, pr\_kuva, pr\_maa, pr\_kuvaus)

values ('August Horch', 1868, 1951, 0, '0/0e/Horch.gif', 'Saksa', 'Was a German engineer and automobile pioneer, the founder of the manufacturing giant which would eventually become Audi.')

insert into KCTITE09\_perustaja(pr\_nimi, pr\_syntynyt, pr\_kuollut, pr\_omaisuus, pr\_kuva, pr\_maa, pr\_kuvaus)

values ('Vaclav Laurin', 1865, 1930, 0, '', 'Tsekki', 'Was a Czech engineer, entrepreneur and industrialist.')

Jokaiselle valmiiksi luodulle yritykselle on yritetty luoda oma perustaja. Kaikista ei löytynyt kuvaa Wikipediasta. Koska käyttäjä ei voi ohjelmassa itse luoda uusia perustajia, niin omatoiminen riittävän kattavien tietojen kerääminen oli hyvin tärkeää.

* 1. Ohjelman toiminta

Ohjelma ottaa aluksi yheyttä tietokantaan Tekniikan-yksikön Lukki-palvelimelle. Web.config-tiedostossa on tätä varten seuraavanlainen koodipätkä:

<connectionStrings>

<!-- yhteysmerkkijono lukki-palvelimen tietokantaan -->

<add name="KCTITE09\_auto" connectionString="Data Source=lukki;

Database=sqldb13; uid=oppilas13; pwd=oppi13Las"/>

</connectionStrings>

ADO.aspx-tiedostossa on itse ohjelman perimmäinen toiminta.

* + 1. Tietokannasta hakeminen



Ohjelman käytön aloittaessa ohjelma ottaa yhteyttä ADO.spx.cs-tiedostossa KCTITE09\_auto-tauluun.

private string conStr =

WebConfigurationManager.ConnectionStrings["KCTITE09\_auto"].ConnectionString.ToString();

Kun yhteys on luotu, niin seuraavaksi ohjelma siirtyy page\_load-funktioon, jossa ohjelma siirtyy FillAutoList-funktioon:

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

FillAutoList();

}

}

FillAutoList-funktiossa ohjelma täyttää DropDownList1-listan sql-hausta saaduilla autoilla, joista se näyttää text-ominaisuudessa riveittään ’merkki’-, ’malli’- ja ’vuosi’-tiedot autoittain käyttäen apuna readeria.

string selectSql = @"select merkki, malli, vuosi, auto\_id from KCTITE09\_auto";

…

while (reader.Read())

{

ListItem newitem = new ListItem();

newitem.Text = reader["merkki"] + " " + reader["malli"] + " " + reader["vuosi"];

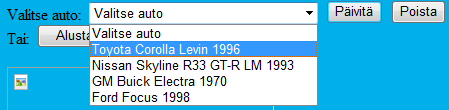
// tallennetaan auto\_id itemin value

newitem.Value = reader["auto\_id"].ToString();

DropDownList1.Items.Add(newitem);

}

Alusta-nappia painettaessa ohjelma tyhentää auton-tietoja ilmoittavat textboxin tekstit tyhjiksi.



Kun ohjelmassa vaihdetaan DropDownList1:n indeksiä, niin ohjelma tekee tietokannasta haun auto\_id-muuttujan perusteella, jonka se löytää DropDownList1:n valitun value-muuttujan perusteella.

string selectSql = @"

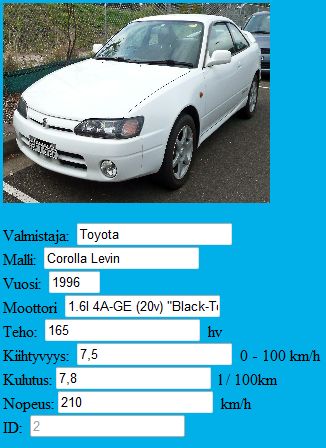
select \*

from KCTITE09\_auto where auto\_id = " + DropDownList1.SelectedValue;

Tämän jälkeen ohjelma täyttää readerin ja syöttää tekstboksiin tiedot merkki, malli, vuosi, moottori, teho, kiihtyvyys, kulutus, nopeus ja auto\_id. Lisäksi näytetään kuva autosta.

Koodina kuva näyttäminen tapahtuu seuraavalla tavalla:

AutoKuva.ImageUrl = "http://upload.wikimedia.org/wikipedia/commons/" + reader["kuva"];



Auton tietoja täyttäessä ohjelma siirtyy lopulta haeLisaTiedot-funktioon, jossa ohjelma kerää myös tiedot auton valmistajasta KCTITE09\_valmistaja-taulusta.

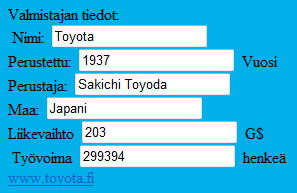
* + 1. Liitoskyselyn tekeminen

Ohjelma tekee liitoskyselyn seuraavasti:

string selectSql = @"Select \* from KCTITE09\_perustaja join KCTITE09\_valmistaja on KCTITE09\_perustaja.pr\_nimi = KCTITE09\_valmistaja.vl\_perustaja and KCTITE09\_valmistaja.vl\_nimi = '" + tbMerkki.Text + "'";

Valmistajan tiedot täytetään textbokseihin samoin kuin autonkin tiedot tehtiin, mutta tämän lisäksi ilmoitetaan myös kotisivu, joka koodin puolella näyttää seuraavalta:

vlLink.NavigateUrl ="http://" + reader["vl\_kotisivu"].ToString();



Lisäksi käytetään apuna monirivistä TextBoxia, jossa näytetään liitoskyselyssä tulleet perustajan tiedot. Myös perustajan kuva näytetään TextBoxin rinnalla.



* + 1. Tietokantaan päivittäminen

Kun painetaan päivitä-nappia, niin ohjelma aluksi muuttaa tekstikenttien pilkut pisteiksi omassa funtkiossaa, jotta tietokantaan päivittäminen ylipäätänsä onnistuisi.

protected void pilkkuPisteeksiAuto()

{

tbNopeus.Text = tbTeho.Text.Replace(",", ".");

tbKiihtyvyys.Text = tbKiihtyvyys.Text.Replace(",", ".");

tbKulutus.Text = tbKulutus.Text.Replace(",", ".");

}

Tämän jälkeen otetaan yhteyttä tietokantaan ja päivitetään auton tiedot tietokantaan.

string updateSql = @"

update KCTITE09\_auto

set merkki ='" + tbMerkki.Text +

"', malli ='" + tbMalli.Text +

"', vuosi =" + Convert.ToInt32(tbVuosi.Text) +

", moottori ='" + tbMoottori.Text +

"', teho =" + Convert.ToInt32(tbTeho.Text) +

", kiihtyvyys ='" + tbKiihtyvyys.Text +

"', kulutus ='" + tbKulutus.Text +

"', nopeus ='" + tbNopeus.Text +

"' where auto\_id = " + tbID.Text;

* + 1. Tietokantaan lisääminen

Tietokantaan lisääminen tapahtuu melkolailla samoin kuin tietokantaan päivittäminenkin. Kun käyttäjä on antanut auton tiedot ja painaa lisää uusi – nappia, niin aluksi ohjelma muuttaa tekstikenttien pilkut pisteiksi ja tämän jälkeen ottaa tietokantaan yhteyttä ja lisää tiedot palvelimelle.

insertSql = @"insert into KCTITE09\_auto(merkki, malli, vuosi, moottori, teho, kiihtyvyys, kulutus, nopeus) values (@merkki, @malli, @vuosi, @moottori, @teho, @kiihtyvyys, @kulutus, @nopeus)";

* + 1. Tietokannasta poistaminen.

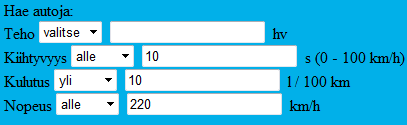
Kun käyttäjä painaa poista-nappia, niin ohjelma ottaa yhteyttä Lukki-palvelimeen ja poistaa kyseiset tiedot auto\_id:n perusteella.

deleteSql = @"delete from KCTITE09\_auto where auto\_id = @auto\_id";

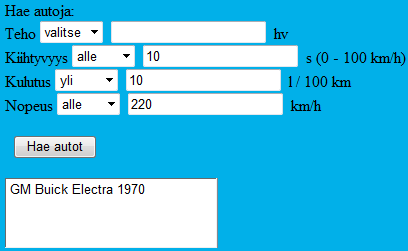
Lopuksi autoon liittyvät tekstikentät tyhennetään.

* + 1. Tietokannasta hakeminen ehdoilla

Tietokannasta voidaan hakea KCTITE09\_auto-taulun autoja tehon, kiihtyvyyden, kulutuksen ja nopeuden perusteella. DropDownList -objekteissa on valittavina joko ”tyhjä”-, ”yli”- tai ”alle”-vaihtoehto.



Kun käyttäjä painaa Hae autot-nappia, niin ohjelmaa hakee käyttäjän syöttämien arvojen perusteella ehdot täyttäviä autoja.



Koodina edellinen haku tapahtuu näin:

string selectSql = @"select \* from KCTITE09\_auto where teho " + ddlTeho.SelectedItem.Value + " '" + tbHaeTeho.Text + "' and kiihtyvyys " +

ddlKiihtyvyys.SelectedItem.Value + "'" + tbHaeKiihtyvyys.Text + "' and kulutus " + ddlKulutus.SelectedItem.Value + " '" + tbHaeKulutus.Text +

"' and nopeus " + ddlNopeus.SelectedItem.Value + " '" + tbHaeNopeus.Text + "'";

LIITTEET

* Tauluihin kohdistuvat komennot – Harjoitustyö.txt