C# ja .NET Framework

ADO.NET ja ASP.NET peruskäyttö

Sisällys

Harjoitus 1: **Visual Studio ja SQL Express**

Tausta

SQL Express on maksuton, helppohallintainen tietokanta. Tätä tietokantaa käytetään erityisesti WEB-sivustojen yhteydessä.

SQL Express-kantaa hallittaan tyypillisesti suoraan Visual Studiosta käsin.

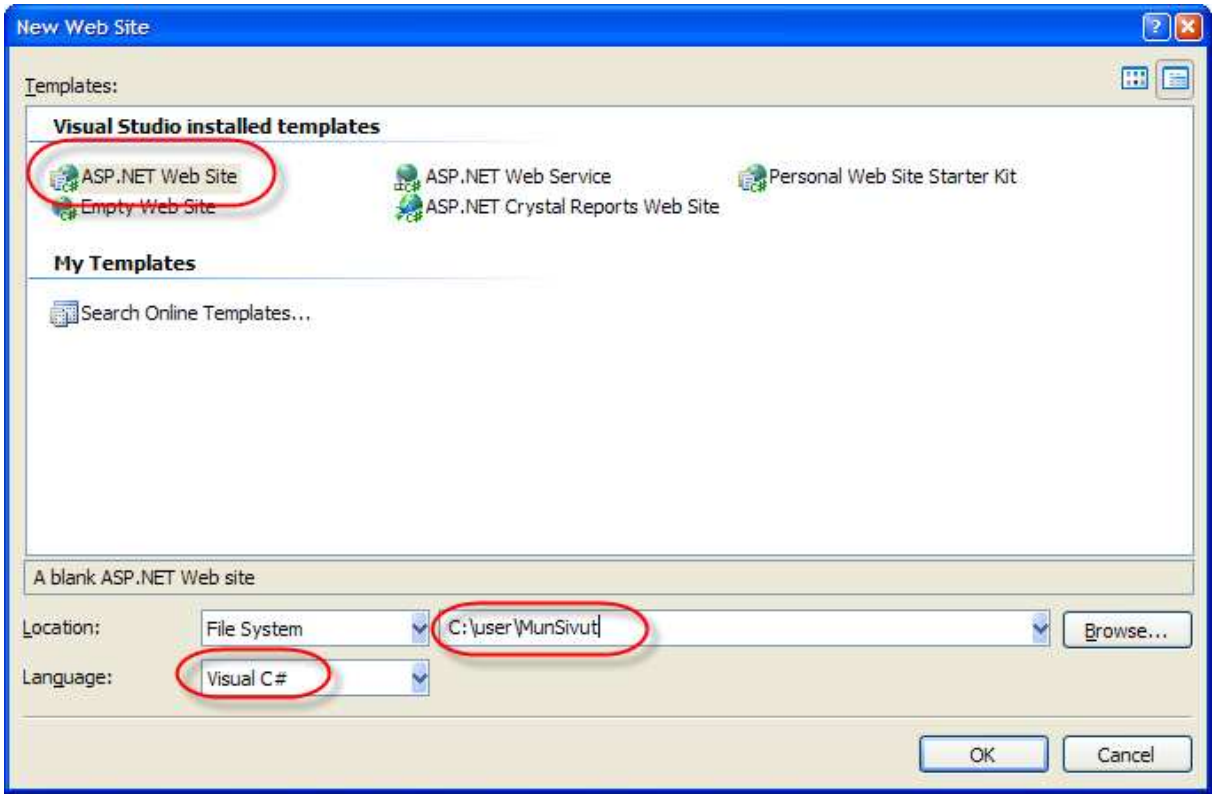
Tehtävä

Tee Web-sivusto, johon toteutetaan käyttäjähallinta: käyttäjien lisääminen Admin-sivulta sekä kirjautumis-sivu.

Ensin tee Web-projekti johon lisäät uuden tietokannan. Kantaan tehdään Asiakas-taulu.

Toimenpiteet

1. Tee Visual Studiossa uusi Web Site (**File | New | Web Site…**). Anna nimeksi MunSivut ja sijoita sivusto `C:\User` –hakemistoon.



2. Lisää sovellukseen SQL Express kanta nimeltään Database.mdf.

    2.1. Paina Solution Explorerissa hiiren oikeaa App_Data –folderin päällä **| Add New Item…**

    2.2. Valitse SQL Database

    2.3. Name: `Database.mdf` (on oletuksena)

3. Avaa kanta (hiiren oikea **| Open**). Tämä avaa tietokannan Server Explorer -ikkkunaan

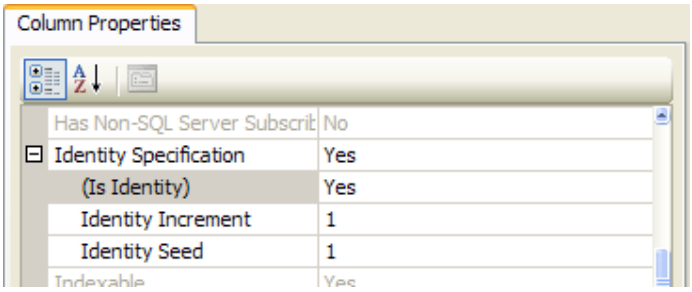
4. Tehdään tarvittava taulu. Server Eplorerissa paina hiiren oikeaa Tables-rivin kohdalla **| Add New Table**. Määrittele seuraavat sarakkeet (huom: kaikkien kenttien Allow Nulls on pois)

Column Name	Data Type	Allow Nulls
Id	int	☐
Sposti	nvarchar(50)	☐
Salasana	nvarchar(50)	☐

5. Aseta `Id`-sarake identity-kentäksi.

    5.1. Valitse Id-rivi

    5.2. Column Properties-osassa (alareunassa) avaa Identity Specification-kohta, jossa Is Identity: true)



6. Aseta Id-sarake taulun perusavaimeksi

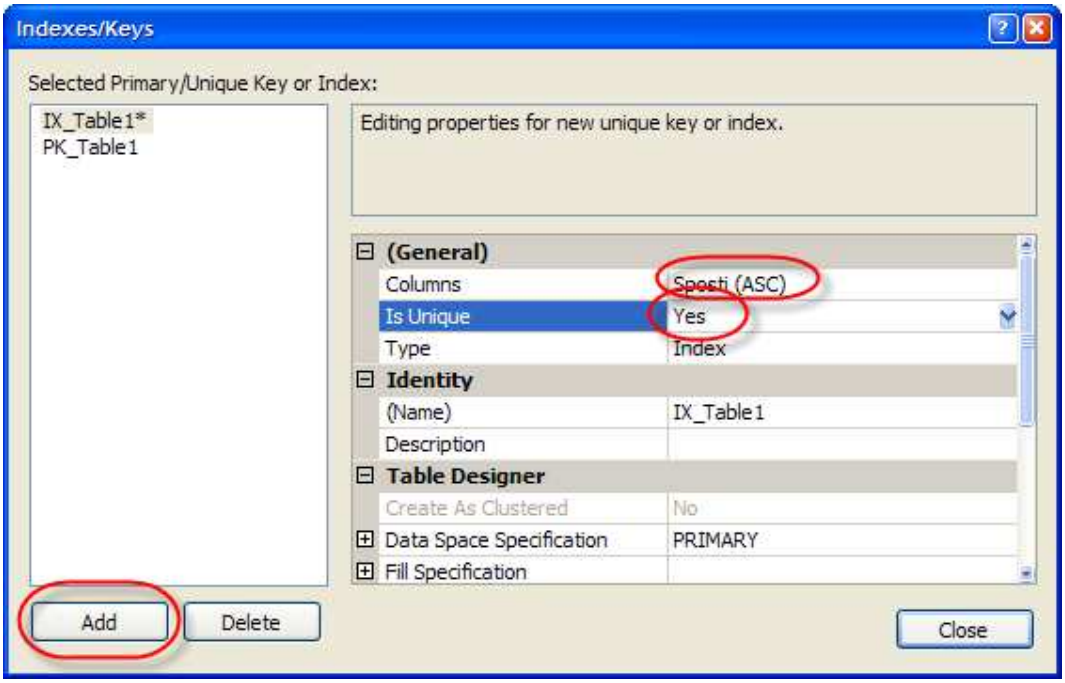
    6.1. Paina hiiren oikeaa Id-rivin päällä **| Set Primary Key**

7. Muodosta `Sposti`-kenttään Unique-indeksi, ettei samalla spostiosoitteella voi olla kahta asiakasta.

    7.1. Paina hiiren oikeaa Sposti-rivin päällä **| Indexes/Keys**

7.2. Paina **Add**

7.3. Valitse kentäksi Sposti ja Unique: Yes



8. Tallenna Taulu (**File | Save Table1**), jolloin kysytään taulun nimi. Anna nimeksi `Asiakas`

Harjoitus 2: **Connection- ja Command- luokat, komennon välittäminen tietokantaan**

Tausta

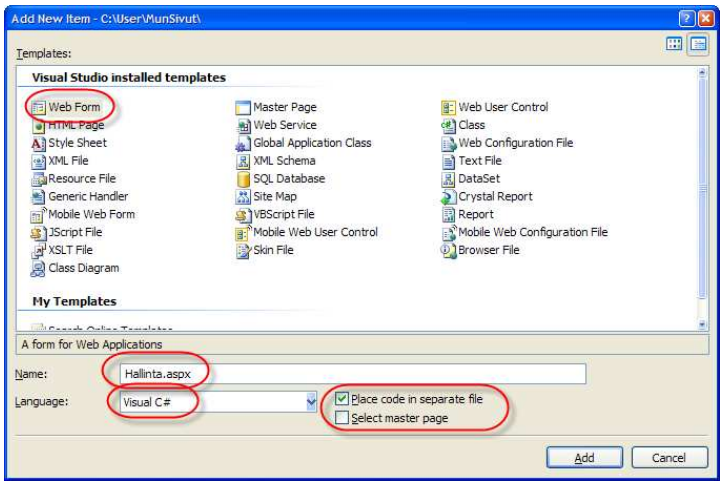
`Connection`- ja `Command`-olioiden avulla sovellus voi välittää SQL-komennon (merkkijono) tietokantaan. Näitä olioita tarvitaan aina, kun sovellus käyttää tietokantaa.

Tehtävä

Tee sivustoon `Hallinta.Aspx` –sivu, jolla voidaan lisätä uusi Asiakas tietokantaan.

Toimenpiteet

1. Hiiren oikeaa projektinimen päällä **| Add New Item….** Valitse seuraavaa:



2. Maalaa lomakkeelle sivun asettelua varten Table.

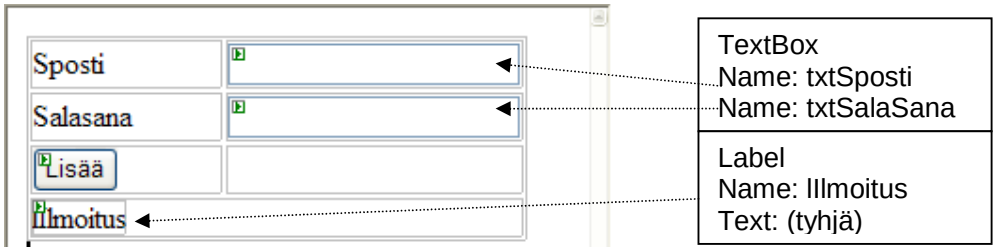
   2.1. Siirry ensin Design-tilaan (kanvaasin alareunan välilehdeltä)

   2.2. Lisää lomakkeelle taulukko **Layout | Insert Table**.

      2.2.1. Taulukon koko Rows: 4, Colums: 2

   2.3. Valitse taulukon alarivi (molemmat sarakkeet), ja hiiren oikeaa **| Merge Cells**

3. Maalaa taulukkoon kontrollit ja nimeä ne. Ohjekentät on suoraan taulukkoon kirjoitettua tekstiä.



4. Voit kokeilla ajaa sovellusta painamalla hiiren oikeaa kanvaasilla **| View In Browser**.

5. Tehdään Asiakkaan tietojen lisääminen kantaan. Generoi ensin tapahtumakäsittelijä *Lisää*-painonapille tuplaklikkaamalla kontrollia.

6. Ota käyttöön `SQLClient`-nimiavaruus. Lisää siis koodin alkuun

```
using System.Data.SqlClient;
```

7. Lisää Hallinta-luokkaan vakiomerkkijonokenttä yhteys, jossa on käytettävä yhteysmerkkijono. Tämä muutetaan konfiguroitavaksi tiedoksi seuraavassa harjoituksessa. Kenttä lisätään luokan alkuun.

```
 private const string yhteys =
@"server=(local)\SQLExpress;
AttachDbFileName=|DataDirectory|Database.mdf;
Integrated Security=true;User Instance=true";
```

8. Koodaa painonappiin Asiakkaan lisäys. Huomaa, että alla oleva ohjelmointitapa ON VAARALLINEN, tämä mahdollistaa ns. SQL Injection-hyökkäykset koska käyttäjän syöte lisätään SQL-merkkijonoon sellaisenaan. Tätä asiaa tutkitaan ja ennen kaikkea korjataan seuraavassa harjoituksessa.

```
using (SqlConnection con = new SqlConnection(yhteys)){
 SqlCommand cmd = con.CreateCommand();
```

```
  cmd.CommandText =
    string.Format("insert into Asiakas(Sposti, Salasana)
values('{0}', '{1}')",
                  txtSposti.Text, txtSalasana.Text);
  try {
    con.Open();
    cmd.ExecuteNonQuery();
    lIlmoitus.Text = " Asiakas lisäys OK";
  }
  catch (Exception ex) {
    lIlmoitus.Text =
      "<h3>Asiakkaan lisäyksessä tuli virhe</h3>" +
      "Virheilmoitus:" + ex.Message;
  }//catch
}//using
```

9. Testaa. Totea, ettei samaa sposti-nimeä voida lisätä useaan kertaa. Mieti, miten virhekäsittelyä pitäisi laajentaa, jos siitä tilanteessa pitäisi antaa käyttäjälle selkeämpi virheilmoitus.

   Mallivastaus on VSS projekti **MunSivut_01** Jos käytätä mallivastausta, muista Check Out:ata tietokanta, jotta se ei jää ReadOnly-tilaan.

Harjoitus 3: **Konfigurointitiedoston käsittely**

Tausta

Tietokantayhteys-merkkijono on tyypillinen asia, jota tulee voida muuttaa sovelluksen deployment-vaiheessa. Siksi se on syytä laittaa konfigurointitiedostoon.

Tehtävä

Muuta sovellusta siten, että yhteysmerkkijono luetaan konfigurointitiedostosta. Olkoon connectionsString:in nimi `OmaKanta`.

Toimenpiteet

1. Lisää projektiin `Web.Config`

   1.1. projektinimen päällä hiiren oikeaa **| Add New Item…**

   1.2. Valitse **Web Configuration File**, ja jätä nimeksi `Web.Config`

2. Lisää ConnectionsString-elementti appSettings- elementin alle. Huomaa, että voit käyttää intellisenseä (ja copy/pastea itse merkkijonon siirtoon kooditiedostosta tänne)

```
<appSettings/>
<connectionStrings>
  <add name="OmaKanta"
connectionString="server=(local)\SQLExpress;AttachDbFileName=|
DataDirectory|Database.mdf;Integrated Security=true;User
Instance=true" />
</connectionStrings>
```

3. Korjaa koodia siten, että käytetään tätä connection-stringiä. Tämä kutsu lisätään tapahtumakäsittelymetodin alkuun.

```
string yhteys =
ConfigurationManager.ConnectionStrings["OmaKanta"].ConnectionS
tring;
```

4. Poista `Hallinta.aspx.cs` –tiedostoon lisätty `private const yhteys`-kenttä.

5. Testaa sovellus. Mitään muutosta ei toiminnassa pitäisi olla.

   Mallivastaus on VSS projekti **MunSivut_03** Jos käytätä mallivastausta, muista Check Out:ata tietokanta, jotta se ei jää ReadOnly-tilaan.

Harjoitus 4: **Datan lukeminen, parametroitu SQL-komento**

Tausta

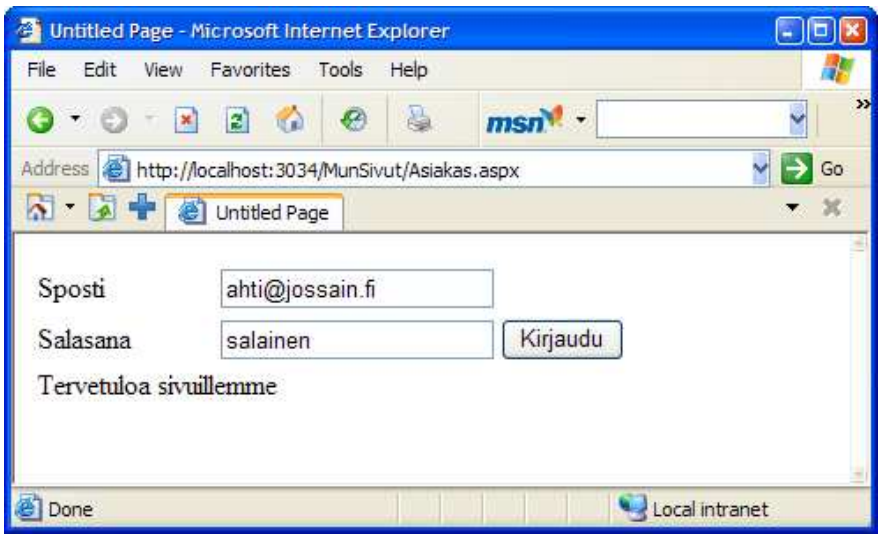
Seuraavaksi tutustutaan datan lukemiseen, ensin ExecuteScalar-metodiin. Samalla harjoitellaan parametroidun SQL-komennon käyttöä ja varautumista SQL Injection-tyyppisiin hyökkäyksiin.

Tehtävä

Toteuta `Asiakas.aspx`-sivu, jossa käyttäjältä kysytään sposti- ja salasanatiedot. Sivulla ilmoitetaan, tiesikö käyttäjä salasanansa.

Tee kantakysely käyttäen parametroitua kyselyä, jotta käyttäjä ei voisi antaa mitä tahansa tietokantakomentoa.

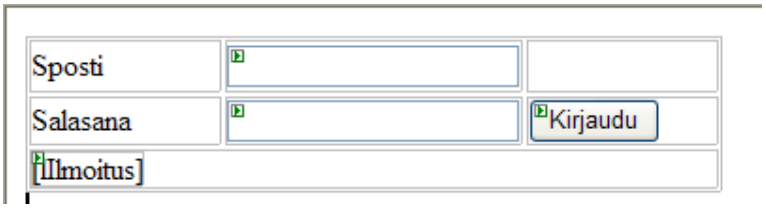
Sivun ulkoasu on seuraava:



Toimenpiteet

1. Lisää projektiin uusi Web Form: `Asiakas`. Tämä tehdään hiiren oikealla Solution Explorer-ikkunassa projektinimen päällä **| Add New Item…**valitse Web Form ja anna nimeksi: `Asiakas`. Kun olet lomakkeen luonut, aseta se käynnistyväksi sivuksi (hiiren oikeaa Solution Explorer-ikkunassa lomakenimen päällä **| Set as Start Page**)

2. Maalaa lomake alla olevan mukaiseksi. Käytä asettelussa Tablea



3. Generoi tapahtumakäsittelijä *Kirjaudu*-painonapille tuplaklikkaamalla kontrollia.

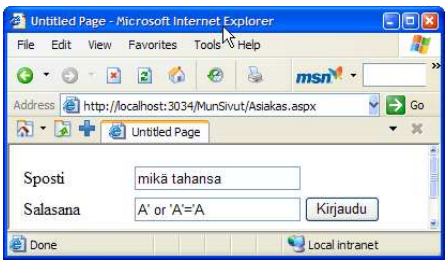
4. Ota käyttöön `SQLClient`-nimiavaruus. Lisää siis koodin alkuun

```
using System.Data.SqlClient;
```

5. Tehdään tapahtumakäsittelijä. Huomaa, että toistaiseksi ei ole parametroitu kysely.

```
string yhteys =
ConfigurationManager.ConnectionStrings["OmaKanta"].ConnectionS
tring;
using (SqlConnection con = new SqlConnection(yhteys)){
 SqlCommand cmd = con.CreateCommand();
 cmd.CommandText =
     string.Format(@"select count(*) from Asiakas WHERE
                  sposti = '{0}' and salasana = '{1}'",
               txtSposti.Text, txtSalasana.Text);
 int rivilkm;
 con.Open();
 rivilkm = (int)cmd.ExecuteScalar();
 if (rivilkm > 0) {
  lIlmoitus.Text = "Tervetuloa sivuillemme";
 }
 else {
  lIlmoitus.Text = "Älä hakkeroi!";
 }
}//using
```

6. Testaa. Sovellus on nyt "vaarallinen", koska käyttäjän syöte sellaisenaan lisätään SQL-komentoon. Mieti, mitä tapahtuu seuraavilla syötteillä, salasana on siis `A' or 'A'='A`



7. Kokeile vieläkin "pahempaa" (huom:. '--' on TSQL-kielen rivikommenttimerkki):
   SPosti: `jotain`
   Salasana: `A' or 'A'='A'; delete from asiakas –`

Paremetroitu kysely

8. Korjataan SQL Injection-virhe. Muuta kysely käyttämään parametroitua komentoa. Muutetut rivit on vahvennettu.

```
string yhteys =
ConfigurationManager.ConnectionStrings["OmaKanta"].ConnectionS
tring;
using (SqlConnection con = new SqlConnection(yhteys)){
 SqlCommand cmd = con.CreateCommand();

 cmd.CommandText = @"select count(*) from Asiakas WHERE
             sposti = @sposti and salasana = @salasana";
 cmd.Parameters.AddWithValue("@sposti", txtSposti.Text);
 cmd.Parameters.AddWithValue("@salasana", txtSalasana.Text);
 int rivilkm;
 con.Open();
 rivilkm = (int)cmd.ExecuteScalar();
 if (rivilkm > 0) {
  lIlmoitus.Text = "Tervetuloa sivuillemme";
 }
 else {
  lIlmoitus.Text = "Älä hakkeroi!";
 }
}//using
```

9. Testaa. Sovelluksen toiminnassa tapahtui se muutos, että nyt noilla ”häkkäys”-salasanoilla ei pääse kirjautumaan.

Mallivastaus on VSS projekti **MunSivut_04** Jos käytätä mallivastausta, muista Check Out:ata tietokanta, jotta se ei jää ReadOnly-tilaan.

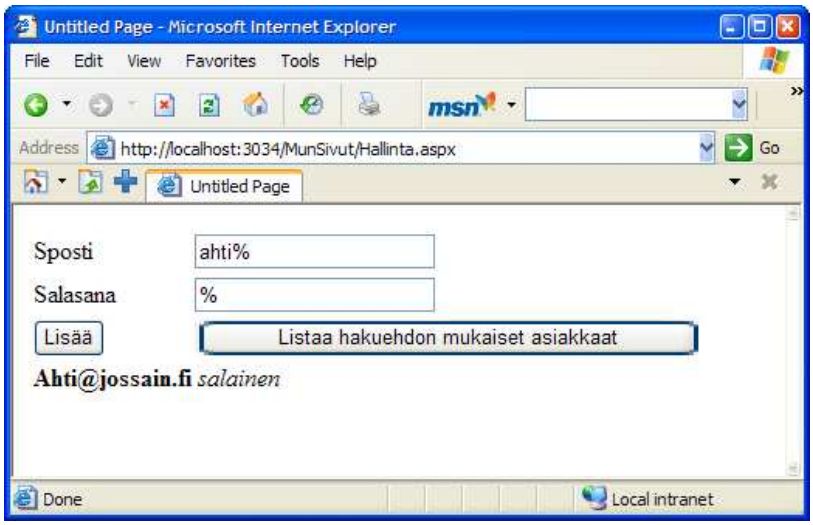
Harjoitus 5: **Datan lukeminen, datareader**

Tausta

DataReader-objektilla voidaan tulosjoukko lukea, ja vain lukea. Lukeminen tehdään rivi kerrallaan, ja kursoria voidaan siirtää vain eteenpäin.

Tehtävä

Lisää Hallinta-sivulle painonappi, jota painamalla listataan nykyiset käyttäjätiedot Ilmoitus-kenttään.



Toimenpiteet

1. Maalaa Hallinta-lomakkeelle painonappi. Aseta Text: "*Listaa hakuehdon mukaiset asiakkaat*".

```
string yhteys =
ConfigurationManager.ConnectionStrings["OmaKanta"].ConnectionS
tring;

using (SqlConnection con = new SqlConnection(yhteys)) {
  SqlCommand cmd = con.CreateCommand();
  SqlDataReader reader;
System.Text.StringBuilder teksti =
            new System.Text.StringBuilder(1024);

if (txtSposti.Text == "" ) txtSposti.Text = "%";
if (txtSalasana.Text == "" ) txtSalasana.Text = "%";
cmd.CommandText = @"SELECT * FROM asiakas
        WHERE sposti LIKE @sposti AND salasana LIKE @salasana";
cmd.Parameters.AddWithValue("@sposti", txtSposti.Text);
cmd.Parameters.AddWithValue("@salasana", txtSalasana.Text);
```

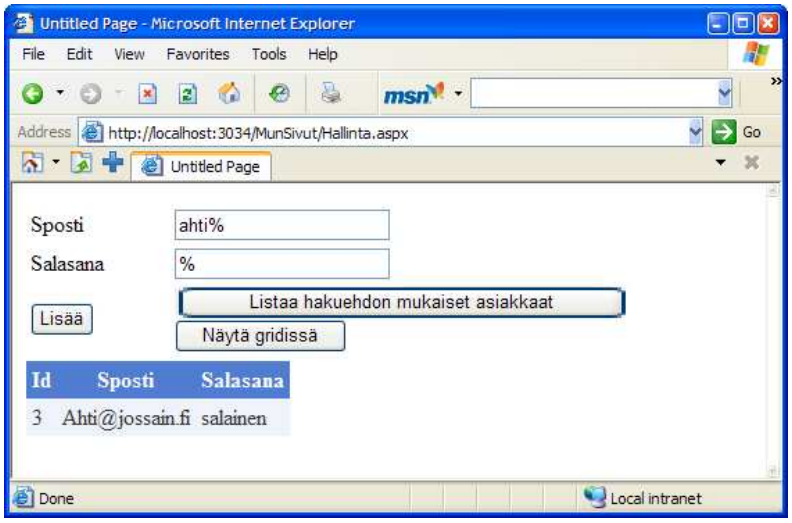
```
try {
 con.Open();
 reader = cmd.ExecuteReader();
 while (reader.Read()) {
  teksti.Append("<b>");
  teksti.Append(reader["sposti"].ToString());
  teksti.Append("</b> <i>");
  teksti.Append(reader["salasana"].ToString());
  teksti.Append("</i><br>");
 }
 lIlmoitus.Text = teksti.ToString();
}
catch (Exception ex) {
 lIlmoitus.Text =
  "<h3>Virhe</h3>" +
  "Virheilmoitus:" + ex.Message;
}//catch
}//using
```

2. Testaa. Voit laittaa Hallinta-sivun käynnistyväksi sivuksi Solution Explorerista.

3. Mieti, miksi `StringBuilder`-objektin käyttö string-muuttujan asemesta (muuttuja `teksti`) on tässä tarkoituksenmukaista.

Lisätehtävä

DataReader voidaan sitoa ASP.NET sovelluksissa suoraan kontrolleihin. Tehdään tulostus käyttäen sidontaa. Lisää lomakkeelle painonappi (samaan taulukon soluun, jossa on jo Listaa-nappi) sekä DataGridView taulukon alle.



4. Tee tapahtumakäsittely. Koodi on valtaosin samaa kuin edellisessä tehtävässä (joten Copy/Paste), mutta nyt `datareader` sidotaan suoraan kontrolliin. Muuttuneet rivit vahvennettu ja `teksti`-muuttuja poistettu.

```
string yhteys =
ConfigurationManager.ConnectionStrings["OmaKanta"].ConnectionS
tring;

using (SqlConnection con = new SqlConnection(yhteys)) {
 SqlCommand cmd = con.CreateCommand();
 SqlDataReader reader;
 if (txtSposti.Text == "") txtSposti.Text = "%";
 if (txtSalasana.Text == "") txtSalasana.Text = "%";
 cmd.CommandText = @"SELECT * FROM asiakas
       WHERE sposti LIKE @sposti AND salasana LIKE @salasana";
 cmd.Parameters.AddWithValue("@sposti", txtSposti.Text);
 cmd.Parameters.AddWithValue("@salasana", txtSalasana.Text);
 try {
  con.Open();
  reader = cmd.ExecuteReader();
  GridView1.DataSource = reader;
  GridView1.DataBind();
 }
 catch (Exception ex) {
  lIlmoitus.Text =
   "<h3>Virhe</h3>" +
   "Virheilmoitus:" + ex.Message;
 }//catch
}//using
```

5. Testaa. Tämä ASP.NET-käyttöliittymää käyttävä harjoitus päättyy tähän.

   Mallivastaus on VSS projekti **MunSivut_05** Jos käytätä mallivastausta, muista Check Out:ata tietokanta, jotta se ei jää ReadOnly-tilaan.

Harjoitus 6: **Dataset, ohjelmallinen käsittely**

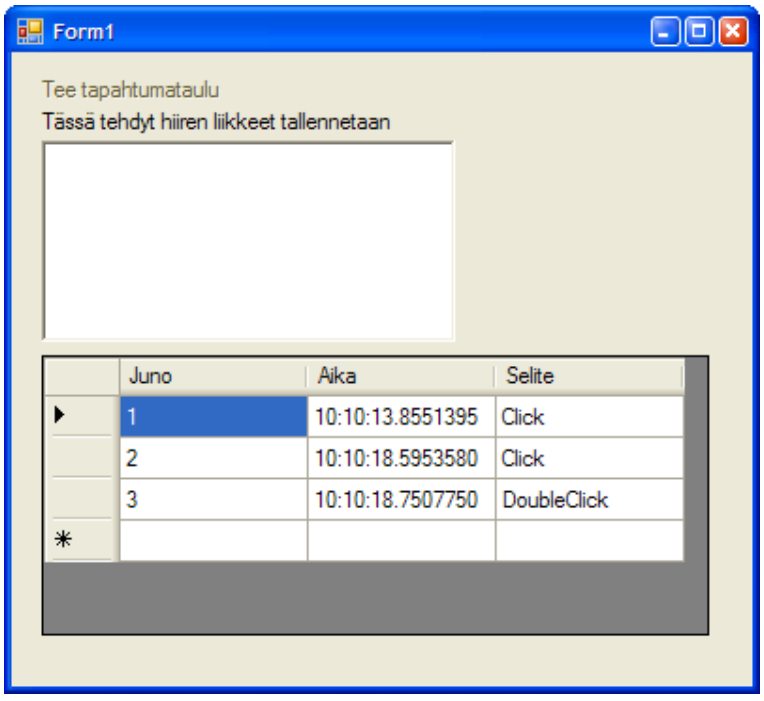
Tausta

DataSet on .NETin tietotyyppi, joka on "kokoelmien kokoelma". Se muistuttaa paljon InMemoryDataBase'a (IMDB). Datan käsittely on kuitenkin kokoelmamaista, esim. rivi lisätään Add-metodilla eikä "INSERT INTO ..."-merkkijonokomennolla. Riviä voidaan kuitenkin seuloa WHERE-ehtoa muistuttavalla komennolla.

Tehtävä

Tee Windows-sovellus, jossa on `PictureBox`-kontrolli. Kaikki hiiren liikkeet tämän `PictureBox`-kontrollin päällä on tarkoitus laittaa talteen. Tiedot kirjoitetaan `DataSet`:iin, johon tehdään taulu ”Tapahtumat”. Taulussa on JuoksevaNumero, Kellonaika ja Selite-kentät.

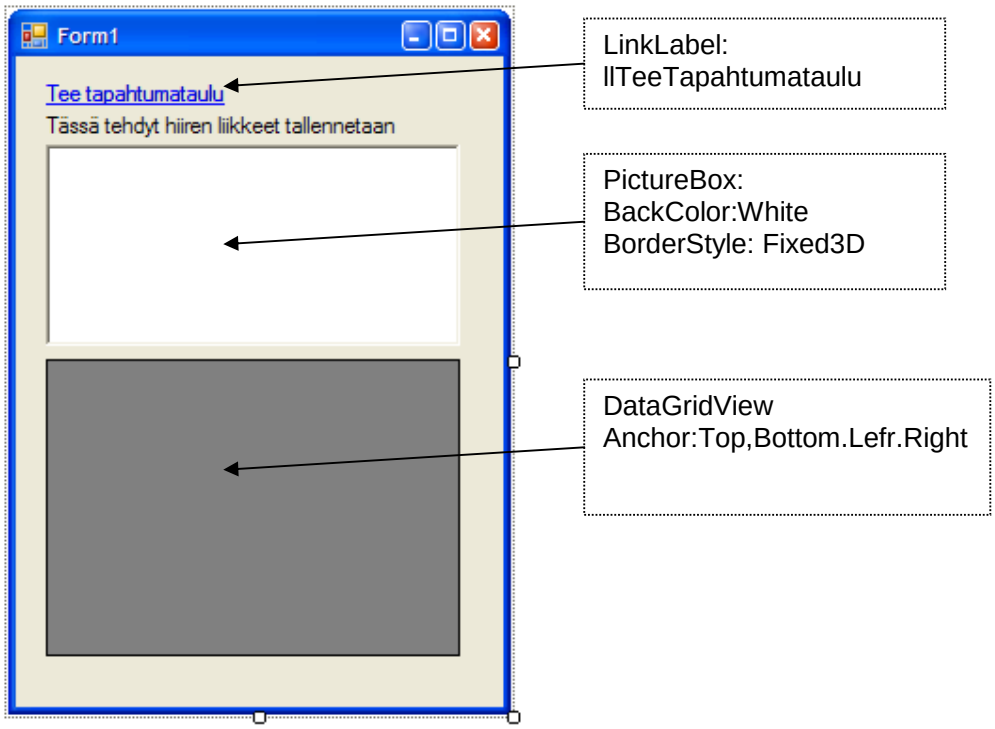
Dataset:n tiedot näytetään lomakkeella olevassa `DataGrid`-kontrollissa.



Toimenpiteet

1. Tee uusi Windows Application, nimeltään `DataSetKäsittely` (et tarvitset Solutionia, pelkkä projekti riittää).

2. Maalaa lomake.



3. Tuplaklikkaa painonappia, jotta generoituu tapahtumakäsittelijä, ja siirrytään koodi-ikkunaan.

4. Lisätään Form1-luokkaan seuraavat kentät

```
private DataSet ds = new DataSet("kurssi");// lomakkeen
Dataset
private DataTable tapahtumataulu;
```

5. LinkLabelin clikissä muodostetaan taulu seuraavasti

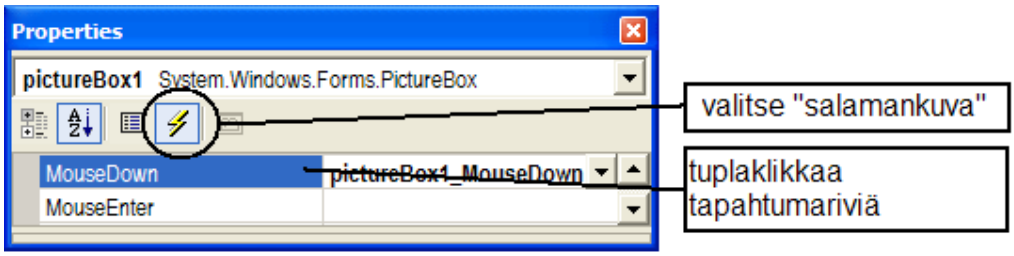
```
tapahtumataulu = new DataTable("Tapahtumat");
DataColumn dc = tapahtumataulu.Columns.Add("Juno",
typeof(int));
dc.AutoIncrement = true;
dc.AutoIncrementSeed = 1;
dc.ReadOnly = true;
dc.Unique = true;
tapahtumataulu.Columns.Add("Aika", typeof(TimeSpan));
tapahtumataulu.Columns.Add("Selite", typeof(string));
ds.Tables.Add(tapahtumataulu);

llTeeTapahtumataulu.Enabled = false;
```

```
//sidotaan taulu näkyviin
dataGridView1.DataSource = tapahtumataulu;
```

6. Tehdään lomakeluokkaan metodi, jolla voidaan lisätä rivi "helposti" tapahtumatauluun.

```
private void LisääRiviTapahtumatauluun(string selite) {
  DataRow rivi;
  if (tapahtumataulu != null) {
    rivi = tapahtumataulu.NewRow();
    rivi["Aika"] = DateTime.Now.TimeOfDay;
    rivi["Selite"] = selite;
    tapahtumataulu.Rows.Add(rivi);
  }
}
```

7. Kutsutaan metodia PictureBox'in eri hiiri-tapahtumissa, esim. (lisää tapahtumakäsittely ainakin MouseDown-, Click- ja MouseUp-tapahtumiin). Tapahtumakäsittelijät lisätään lomakkeen Designer-ikkunassa valitsemalla Properties-työkaluikkunassa "salamankuva"-ikoni, ja Event-listasta tuplaklikkaamalla kyseistä tapahtumaa.



.

```
private void pictureBox1_Click(object sender, EventArgs e) {
  LisääRiviTapahtumatauluun("Click");
}
```

8. Testaa. LinkLabeliä painamalla pitäisi datasettiin tulla Tapahtumataulu. PictureBox1:n päällä tehdyt hiiren liikkeet lisäävät rivin kyseiseen tauluun. Huomaa, että juno-kenttä on ReadOnly, ja että sen arvo tulee automaagisesti.

Tämän koko harjoituksen mallivastaus on VSS projekti **DatasetKäsittely**.

Harjoitus 7: **XML käsittely Datasetin avulla**

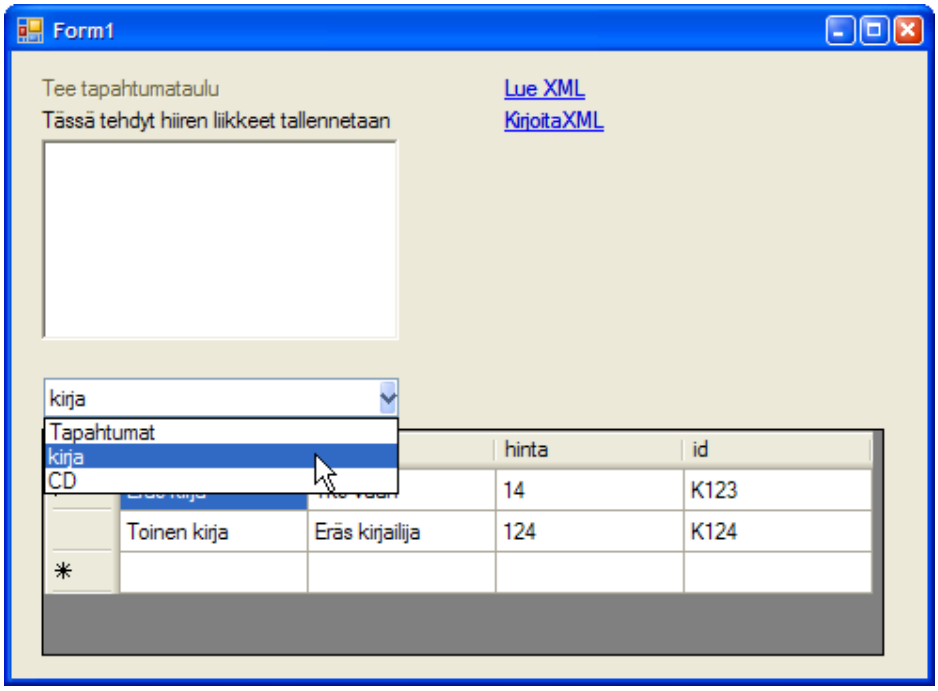
Tausta

XML-dokumentteja voidaan käsitellä ja tuottaa helposti DataSetin avulla.

Tehtävä

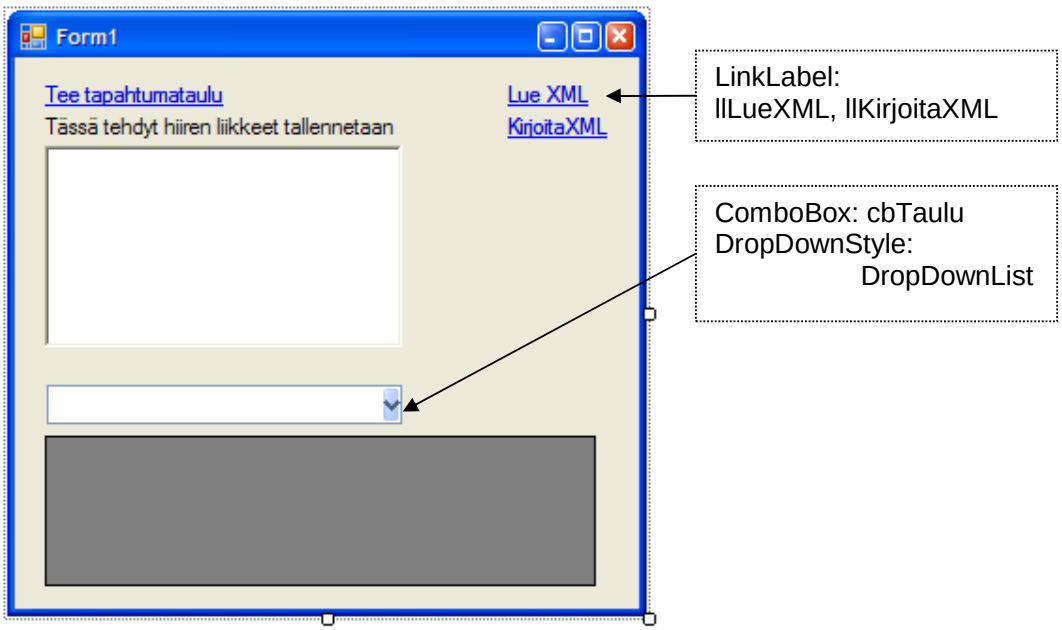
Lue `Kirjasto.xml` –dokumentti datasettiin ja näytä sen sisältö gridissä. Lisää käyttöliittymään mahdollisuus tulostaa koko dataset XML-dokumentina.

Lisää lomakkeelle myös ComboBox, jossa listataan ds:ssä olevat taulut ja josta voi vaihtaa näytettävän taulun.



Toimenpiteet

1. Maalaa lomakkeelle uudet kontrollit alla olevan ohjeen mukaisesti



2. Tehdään ensin ComboBox-käsittely. Valitettavasti DataSetin Tables-kokokoelmaa ei voi suoraan sitoa ComboBoxin DataSourceksi, vaan se on tehtävä käsin. Ensiksi kiinnitetään kuuntelija Dataset.Tables –kokoelman muutoksiin. Joten lisää seuraava koodi lomakkeen Load-tapahtumaan (Huom: seuraa wizardia, älä kirjoita kaikkea käsin!)

```
ds.Tables.CollectionChanged += new
CollectionChangeEventHandler(Tables_CollectionChanged);
```

3. Toteuta taulu-kokoelman tapahtumakäsittely (wizardi generoi metodin rungon!)

```
cbTaulu.Items.Clear();
foreach (DataTable t in ds.Tables)
  cbTaulu.Items.Add(t.TableName);
```

4. Tee cbTaulu-kontrollin valintaan (oletustapahtuma) käsittelijä, jossa asetetaan DataGridView:ssä näytettävä taulu

```
private void cbTaulu_SelectedIndexChanged(object sender,
EventArgs e) {
  dataGridView1.DataSource = ds.Tables[cbTaulu.Text];
}
```

5. Voit testata sovelluksen tässä vaiheessa.

6. Lisää *LueXML*-tapahtumakäsittelijä

```
private void llLueXml_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e) {
 ds.ReadXml(@"..\..\kirjasto.xml", XmlReadMode.InferSchema);
}
```

7. Lisää *KirjoitaXML*-tapahtumakäsittelijä

```
private void llKirjoita_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e) {
   ds.WriteXml(@"..\..\Data.XML");
}
```

8. Testaa. Tarkista myös kirjoitettu XML-tiedosto (`Data.xml`)

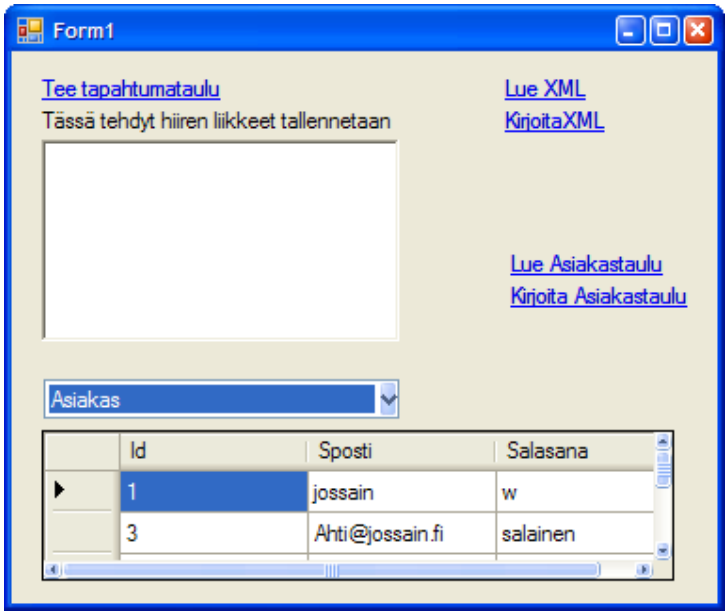
Harjoitus 8: **Tietokantadatan lukeminen ja päivittäminen DataSetin avulla**

Tausta

Toki myös tietokannassa olevaa dataa voidaan lukea DataSettiin ja datasettiin dataan tehtyjen muutosten perusteella voidaan generoida SQL-lauseet, jolla päivitetään tietokanta. `TableAdapter` *on suunniteltu juuri tähän tarkoitukseen.*

Tehtävä

Tee sovellukseen toiminnot, jolla edellisessä harjoituksessa tehtyä Asiakas-taulua voidaan ylläpitää (tai jos em. taulua ei ole, mitä tahansa valittua tietokantataulua).



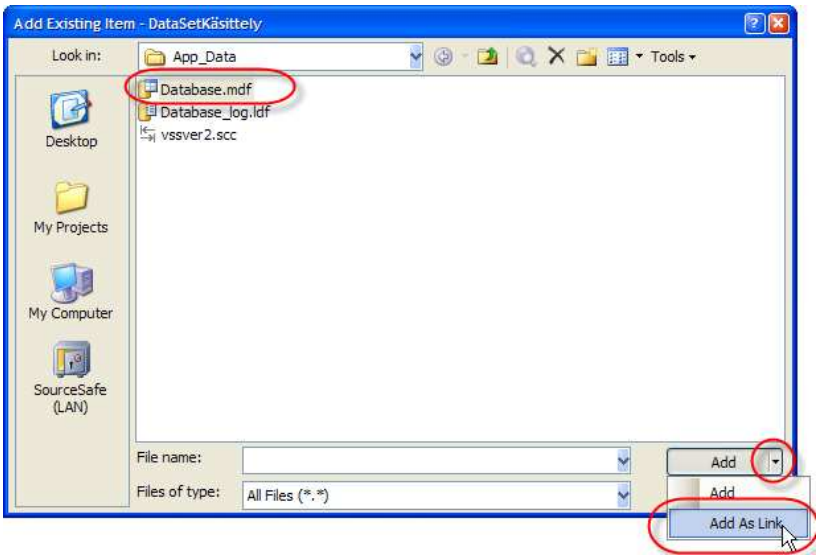
Toimenpiteet

1. Lisää projektiin linkki käytettyyn tietokantaan (jos käytetään SQL Express-kantaa) tai lisää dataconnection Visual Studioon (jos käytetään varsinaista SQL Server-kantaa)

    1.1. Hiiren oikeaa projektinimen päällä **| Add Existing Item….**

    1.2. Valitse Database.mdf –kanta (`C:\user\MunSivu\App_Data` –hakemistossa)

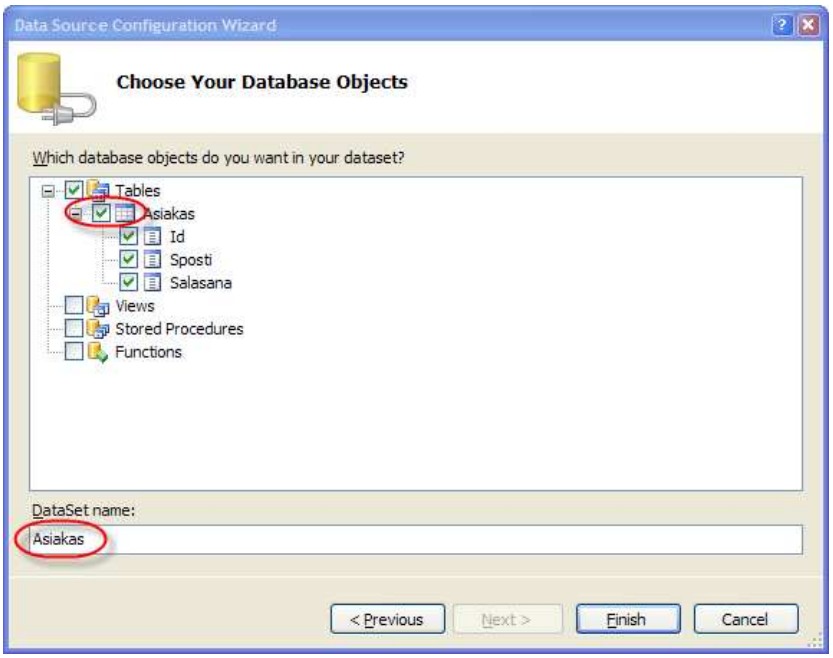
    1.3. Paina Add-napin alasvetonappia, ja valitse **Add As Link**



1.4. Configuration Wizardi mahdollistaa tehdä välittömästi TableAdapterin.

1.4.1. Valitse Asiakas-taulu

1.4.2. DataSet name: `Asiakas`



2. Maalaa lomekkeelle kaksi linkLabeliä (Lue Asiakas, ja Kirjoita Asiakas).

3. Lue tapahtumakäsittelijä:

```
AsiakasTableAdapters.AsiakasTableAdapter asiakasTA = new
AsiakasTableAdapters.AsiakasTableAdapter();
Asiakas.AsiakasDataTable taulu = new
Asiakas.AsiakasDataTable();

asiakasTA.Fill(taulu);

//jos ds:ssä on jo asiakastaulu, se poistetaan
if (ds.Tables.Contains(taulu.TableName))
 ds.Tables.Remove(taulu.TableName);

//ja lisätään asiakastaulu
ds.Tables.Add(taulu);
```

4. Kirjoita tapahtumakäsittelijä

```
AsiakasTableAdapters.AsiakasTableAdapter asiakasTA = new
AsiakasTableAdapters.AsiakasTableAdapter();
asiakasTA.Update(ds.Tables["Asiakas"] as
Asiakas.AsiakasDataTable);
```

5. Testaa. Kiinnitä huomiota siihen, että DataSetissä oleva kaikki data on ”tasa-arvoista”, sillä ei ole mitään tietoa, mistä data on peräisin. Tietokantayhteyttä ei ole lainkaan.

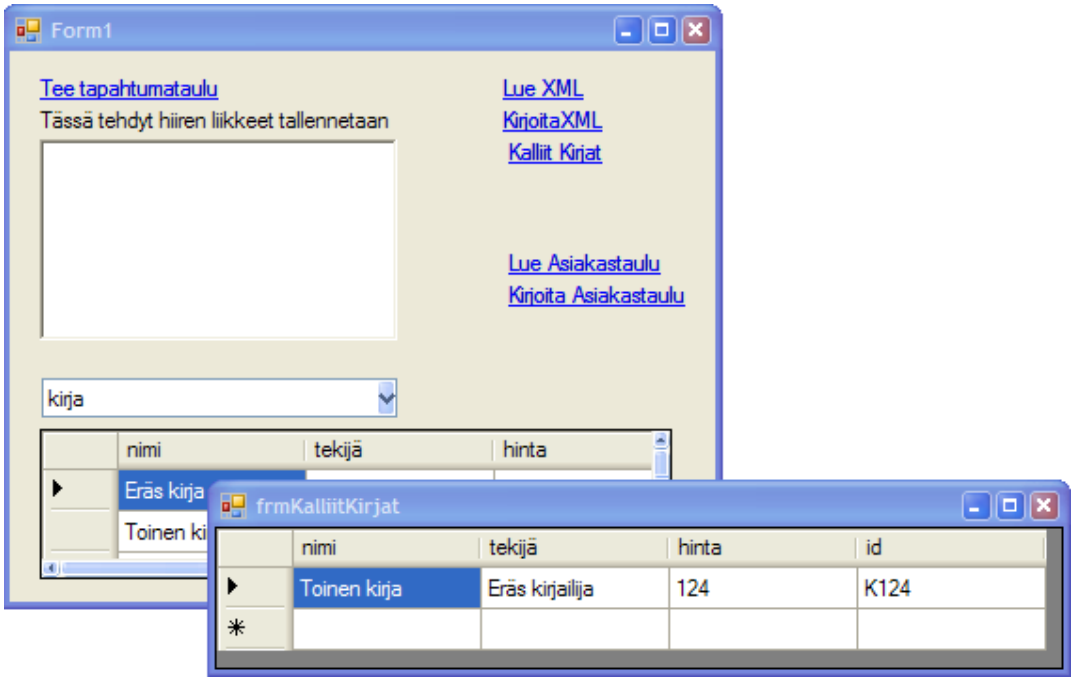
Harjoitus 9: **DataView**

Tausta

DataSet'illä on paljon tietokantatyyppisiä ominaisuuksia. Sen tauluihin voidaan kohdistaa Select-hakuja ja tauluista voidaan muodostaa näkymiä. View-käsittely tekee DataSetistä yhä enemmän "tietokannan tuntuisen".

Tehtävä

Tee KalliitKirjat-näkymä, johon selektoidaan Kirja-taulusta ne rivit, joissa hinta on yli satasen. KalliitKirjat näytetään toisella, uudella lomakkeella.



Toimenpiteet

1. Lisää projektiin uusi Windows Form, anna nimeksi `frmKalliitKirjat`.

2. Maalaa tälle uudelle lomakkeelle DataGridView, jonka Dock: fill.

3. Tee `frmKalliitKirjat`-luokkaan metodi, jolla annetaan näytettävä DataView. Koodi on

```
public void Näytä(DataView view) {
  this.dataGridView1.DataSource = view;
  this.Show();
}
```

4. Tee Form1-luokan "Kalliit Kirjat" –linkLabeliin tapahtumakäsittelijä, jossa muodostetaan uusi View, ja laitetaan se näkyviin `frmKalliitKirjat` -lomakkeelle. Koodi on

   ```
   DataView kalliitKirjat = new DataView(ds.Tables["Kirja"]);
   kalliitKirjat.RowFilter = "hinta > 100";
   frmKalliitKirjat f = new frmKalliitKirjat();
   f.Näytä(kalliitKirjat);
   ```

5. Testaa. Huomaa, että ensin täytyy lukea XML-dokumentti. Totea myös, että view on dynaaminen, ts. Kirja-tauluun tekemät muutokset näkyvät View-näkymässä välittömästi.

Tämän koko harjoituksen mallivastaus on VSS projekti **DatasetKäsittely**.