
# ASP.NET Data Binding

**Harjoitukset C# ja VB**

## Sisällys

# Harjoitus 1: **Kannan luonti**

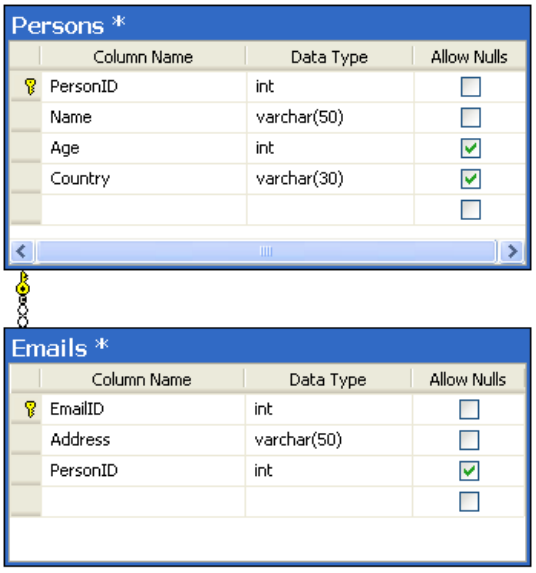
## Tehtävä

Harjoituksessa toteutetaan kanta, johon tallennetaan henkilöiden sähköpostitietoja. Kanta toteutetaan Visual Studiossa SQL Serveriin

## Toimenpiteet

1. Valitse **Server Explorer | Data Connections**. Voit luoda uuden kannan ponnahdusvalikosta **Create New Sql DataBase**. Kannan nimeksi tulee *Contacts*.

2. Lisää kantaa diagrammi ja toteuta oheiset taulut.




3. Muuta *PersonID* ja *EmailID* perusavaimien *Is Identity* muotoon *Yes*, jolloin kanta luo avaimen automaattisesti.

4. Sulje lopuksi diagrammi ja tallenna muutokset kantaan. Voit syöttää kannan tauluihin testiaineistoa valitsemalla taulun ponnahdusvalikosta **Show Table Data**. Esim:

| | PersonID | Name | Age | Country |
|---|---|---|---|---|
| | 1 | John | 12 | Finland |
| | 2 | Sara | 33 | USA |
| | 3 | Michael | 55 | USA |
| | 4 | Anna | 44 | Finland |
| ▶* | NULL | NULL | NULL | NULL |

| | EmailID | Address | PersonID |
|---|---|---|---|
| | 1 | john@gmail.com | 1 |
| | 2 | sara@hotmail.com | 2 |
| ▶ | 3 | Michael@hotmail.com | 3 |
| * | NULL | NULL | NULL |

5. Tarkista, että kannan viite-eheys toimii oikein. Yritä poistaa henkilö, jolla on sähköpostiosoite. Poiston ei pitäisi onnistua ennen kuin kyseisen henkilön sähköpostiosoitteet on poistettu.

# Harjoitus 2: **Tietokantahaku ja tiedon editointi**

*Tietokannasta suoraan raahattu taulu luo kaikki tarvittavat SQL-lauseet suoraan aspx-sivulle. Näihin kuuluu select, update, delete ja insert. C# tai VB.NET koodia ei synny lainkaan.*

## Tehtävä

| | PersonID | Name | Age | Country |
|---|---|---|---|---|
| Edit | 1 | John | 12 | Finland |
| Edit | 2 | Sara | 33 | USA |
| Edit | 3 | Michael | 55 | USA |
| Edit | 4 | Anna | 44 | Finland |

## Toimenpiteet

1. Aloita uusi *ASP.NET Web Site* - projekti.
2. Luo *Persons.aspx* –sivu ja rahaa tietokannan henkilöt taulu siihen. Sivulle syntyy *SqlDataSource* ja *GridView*.

| PersonID | Name | Age | Country |
|---|---|---|---|
| 0 | abc | 0 | abc |
| 1 | abc | 1 | abc |
| 2 | abc | 2 | abc |
| 3 | abc | 3 | abc |
| 4 | abc | 4 | abc |

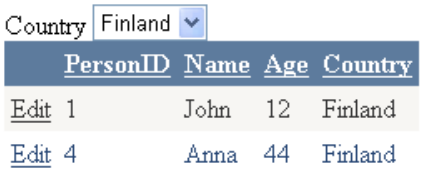
SqlDataSource - SqlDataSource1

3. Aja sovellus. Tietojen pitäisi näkyä selaimessa.
4. Tutki sivun html-koodia ja etsi sieltä *SqlDataSource*. Miten SQL-lauseet on toteutettu? Etsi lisäksi *ProviderName* ja siinä käytettävä *ConnectionString Web.Config* –tiedostosta.
5. Tutki sivun taustalla olevaa luokkaa *Persons.aspx.cs / vb*. Näkyykö tietokantakoodi luokassa millään tavalla?
6. Valitse **Smart Tag | Enable sorting** ja **Enable Editing**. Voit lisäksi valita uuden tyylin kohdasta **Auto Format…**
7. Testaa sovellus.

# Harjoitus 3: **Henkilöiden lajittelu maittain**

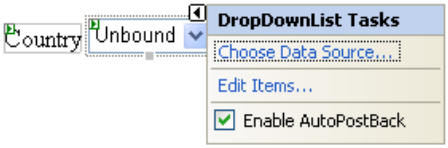
*Kaikkien henkilöiden listaaminen kerralla on epäkäytännöllistä. Henkilötiedot kannattaa jakaa pienempiin kokonaisuuksiin esim. maittain. Harjoituksessa maat tuodaan erilliseen listaan.*

## Tehtävä




## Toimenpiteet

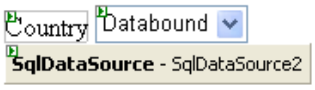
1. Lisää sivun ylälaitaan *Label* ja *DropDownList* –kontrollit. Muuta **Enable AutoPostBack** –päälle.




2. Valitse **Choose Data Source…** Ruudulle käynnistyy velho, jossa voit määrittää käytettävän tietolähteen. Tee seuraavat valinnat velhossa

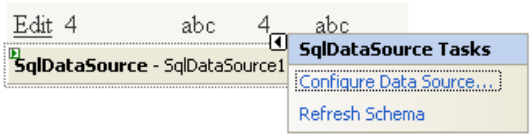
   - Select a datasource: **<New Data Source>**
   - Choose DataSource Type: **DataBase**
   - Choose Your Data Connection: **valitse listasta yhteys.**
   - Configure the Select Statement: **SELECT DISTINCT [Country] FROM [Persons]**”
   - Loput valinnat menevät oletusarvoilla.

   Lopputuloksen tulisi näyttää seuraavalta.

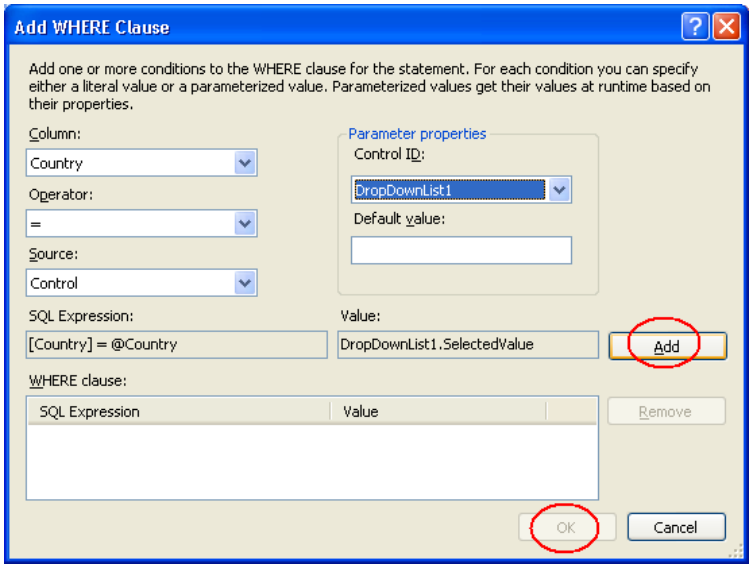



…


3. Valitse *GridView* -kontrollissa käytetyn *SqlDataSource* –kontrollin **Configure Data Source….**




Ruudulle käynnistyy velho, jonka kohdassa **Configure the Select Statement** paina **Where** nappia. Tee seuraavat muutokset kyseisessä ikkunassa.

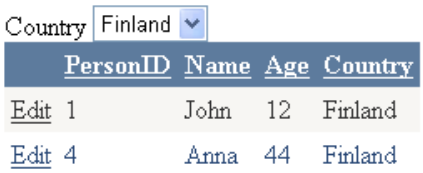



Muutokset heijastuvat SQL-lauseeseen seuraavasti:

```
SELECT [PersonID], [Name], [Age], [Country] FROM [Persons]
WHERE ([Country] = @Country)
```

Lauseeseen lisättiin parametri, joka haetaan maalistasta. Tutki sivun html-lähdekoodia ja etsi kohta *ControlParameter*. Miten parametri sidotaan kontrolliin?

4. Testaa sovellus.




5. Muuta maalistan *AutoPostBack=false*. Toimiiko sovellus oikein?

# Harjoitus 4: **Henkilöiden sähköpostiosoitteet**

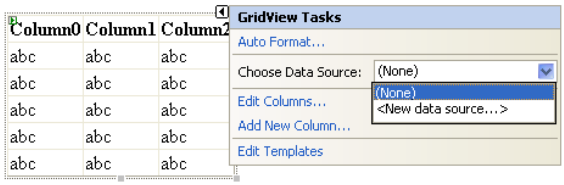
*Aikaisemmassa harjoituksessa tietokannan taulu raahattiin suoraan aspx-sivulle. Periaatteessa samaa tapaa voitaisiin käyttää myös tässä harjoituksessa. Toteuta haku kuitenkin oheisella tavalla, jossa vastaava rakenne tehdään hieman pidemmällä kaavalla.*

## Tehtävä

Harjoituksessa luodaan master – details – tyyppinen näkymä henkilöiden sähköpostiosoitteisiin.

## Toimenpiteet

1. Lisää sovellukseen uusi sivu nimellä *Emails.aspx*.
2. Sivulle tulee *GridView* - kontrolli, jolla voit muodostaa yhteyden **<New data source…>** kohdan avulla.




3. Luo kysely, joka hakee kaikki sähköpostiosoitteet. Testaa sovellus

| EmailID | Address | PersonID |
|---|---|---|
| 1 | john@gmail.com | 1 |
| 2 | sara@hotmail.com | 2 |
| 3 | Michael@hotmail.com | 3 |

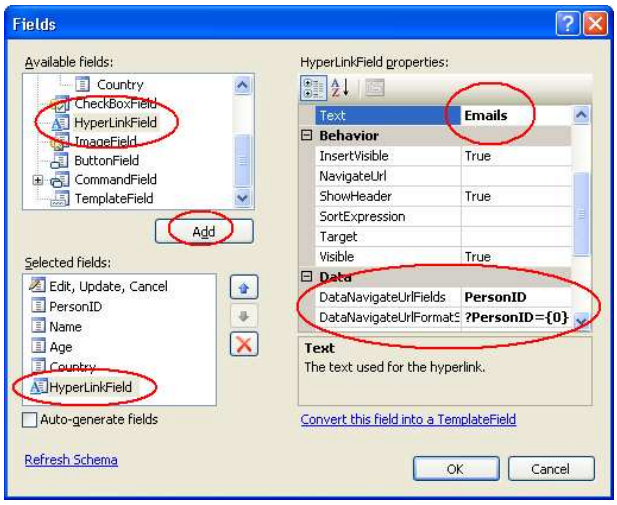
4. Tutki aspx-sivulle syntyviä SQL-lauseita. Syntyikö *select, insert, update* ja *delete*?

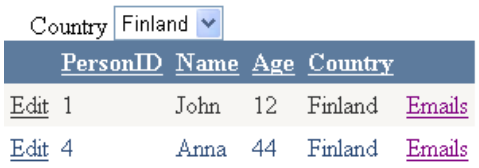
   *Velho luo oletusarvoisesti vain select –lauseen. Muut lauseet voidaan generoida tarvittaessa kohdassa **Configure the Select Statement | Avanced | Generate INSERT, UPDATE, and DELETE statements**. Suoraan kannasta raahattu taulu luo aina kaikki lauseet huolimatta siitä tarvitaanko niitä.*

   Sivun rakennetta halutaan muuttaa siten, että vain valitun henkilön sähköpostiosoitteet tuodaan listaan. Tämä edellyttää muutoksia *Persons.aspx* - sivulle.

5. Valitse henkilö listan **Smart Tag | Edit Columns**. Lisää uusi *HyperLinkField* ja tee siihen seuraavat muutokset:

   *Huom. DataNavigateUrlFormat = Emails.aspx?PersonID={0}*



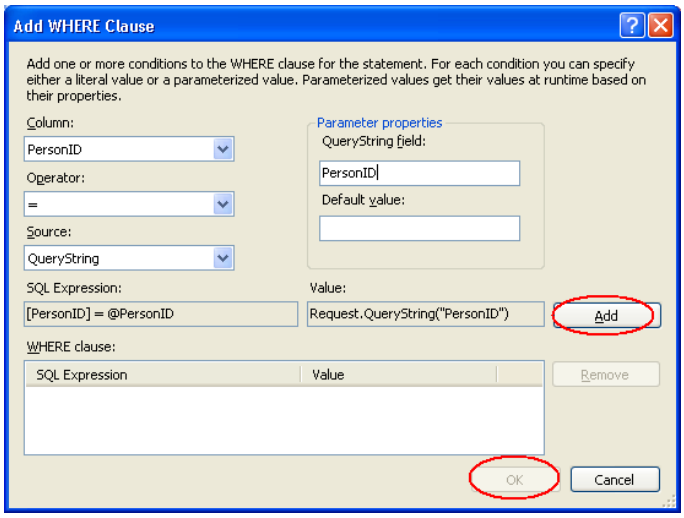




6. Aja sovellus ja kokeile listan *Emails* - linkkiä. Sivun pitäisi vaihtua siten, että Url-kentässä näkyy henkilön id.

```
http://localhost:1034/WebSite1/Emails.aspx?PersonID=1
```

Nykytilanteessa *Emails.aspx* – sivulle listautuu kaikki sähköpostiosoitteet. Sovelluksessa halutaan kuitenkin visualisoida vain valitun henkilön osoitteet.

7. Valitset sähköposti – listasta **Smart tag | Configure Data Source**. Muuta SQL-lauseen *where* ehtoa seuraavasti.




```
Muutoksen heijastuvat SQL-lauseeseen seuraavasti: SELECT *
FROM [Emails] WHERE ([PersonID] = @PersonID)
```

8. Testaa sovellus. *Emails.aspx* - sivulla pitäisi näkyä vain valitun henkilön sähköpostiosoite.

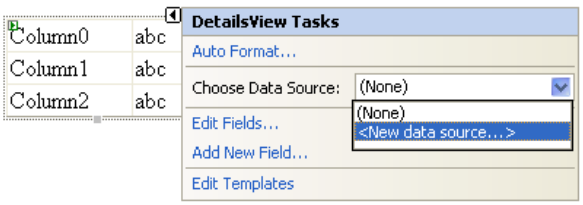
# Harjoitus 5: **Tiedon lisäys**

*DataGridView –kontrolli mahdollistaa tiedon editoinnin. Tiedon lisäystä ei sillä voida suoraan tehdä vaan siihen tarvitaan muita kontrolleja.*

## Tehtävä

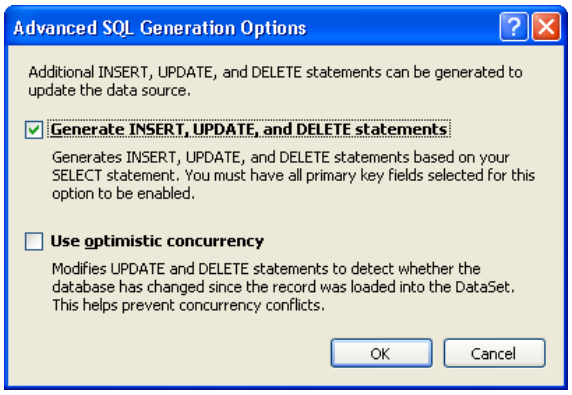

## Toimenpiteet

1. Lisää sovellukseen *AddPerson.aspx* –sivu. Sivulle tulee *DetailsView* -kontrolli. Valitse sille tietolähde.

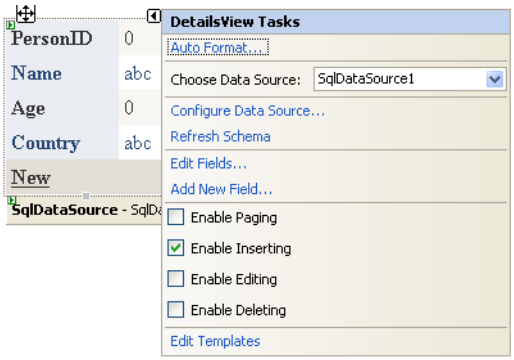


2. Toteuta seuraava SQL-lause: SELECT * FROM [Persons]

3. Lisäksi tarvitaan vastaava *Insert* - lause, joka voidaan generoida **Advanced…** –napilla.




4. Aja velho loppuun ja muuta listan **Smart Tag | Enable Inserting** päälle.




5. Muuta *DefaultMode=Insert*. Varmista, että *AutoGenerateInsertButton=true*.

6. Lisää *Persons.aspx* –sivun loppuun *HyperLink* –kontrolli.

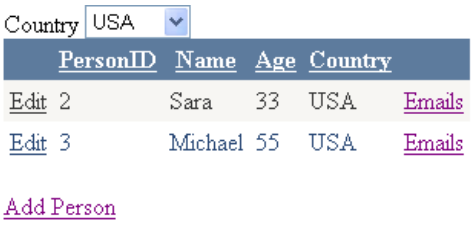
   - NavigateUrl = AddPerson.aspx
   - Text = Add Person

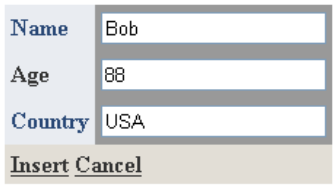
7. Tee vastaavasti linkki myös *AddPersons.aspx* – sivulta takaisin.

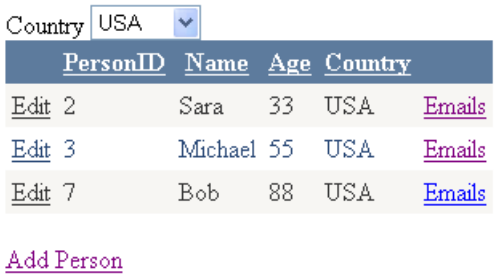
   - NavigateUrl = Persons.aspx

- Text = Back to persons

8. Aja sovellus. Pystytkö lisäämään henkilön listaan?




9. Peruutusnappi (Cancel) ei vielä toimi *AddPerson.aspx* -sivulla. Lisää seuraavaa koodi *ItemCommand* –tapahtumaan.

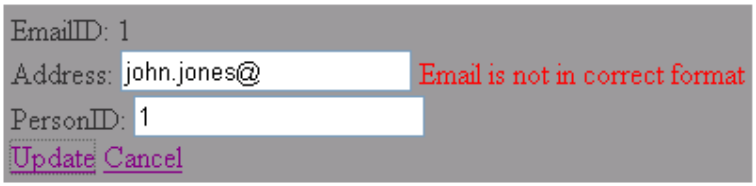
**C#**
```
  protected void DetailsView1_ItemCommand(object sender,
DetailsViewCommandEventArgs e)
  {
    if(e.CommandName.Equals("Cancel")){
      Server.Transfer("Persons.aspx");
    }
  }
```

```
VB
    Protected Sub DetailsView1_ItemCommand(…) Handles
DetailsView1.ItemCommand
        If e.CommandName.Equals("Cancel") Then
            Server.Transfer ("Persons.aspx")
        End If
    End Sub
```

# Harjoitus 6: **Syötteen tarkistaminen**

*Harjoitus tehdään Emails.aspx – sivulle.*

## Tehtävä




## Toimenpiteet

1. Lisää *GridView* – kontrolliin uusi *HyperLinkField*, jonka nimeksi tulee *Edit*.
2. Siirrä *EmailID* ja *PersonID* tieto URL-kentässä sivulle *EditEmail.aspx* (siirto tehdään samalla tavalla kuin harjoituksessa ”Henkilön sähköpostiosoitteet”).


*http://localhost:1041/VBDataBindingHarj/EditEmail.aspx?EmailID=1&PersonID=1*

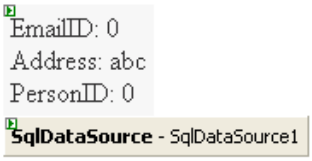
DataNavigateUrlFields = EmailID,PersonID

DataNavigaterUrlFormatString = EditEmail.aspx?EmailID={0}&PersonID={1}

3. Lisää *EditEmail.aspx* – sivu ja lisää sivulle *FormView*-kontrolli. Kyseinen kontrolli sidotaan *SqlDataSourceen*, jossa haetaan valittu sähköposti.

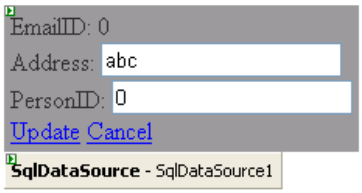
```
SELECT * FROM [Emails] WHERE ([EmailID] = @EmailID)
```

@EmailID = QueryString EmailID



Huom. Muista generoida **Advanced…** –napilla update, delete ja insert –lauseet.

Muuta DefaultMode=Edit.




4. Toteuta tapahtumankäsittelijät *Update* ja *Cancel* – linkeille, joilla siirrytään takaisin *Emails.aspx* –sivulle. Kyseiselle sivulle täytyy välittää *PersonID* –tunniste.

**C#**

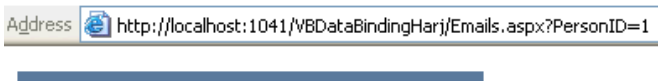
```
  protected void FormView1_ItemCommand(object sender,
FormViewCommandEventArgs e)
  {
    if(e.CommandName.Equals("Cancel"))
    {
      Server.Transfer("Emails.aspx?PersonID=" +
Request.QueryString["PersonID"]);
    }
  }
  protected void FormView1_ItemUpdated(object sender,
FormViewUpdatedEventArgs e)
  {
    if(Page.IsValid)
    {
      Server.Transfer("Emails.aspx?PersonID=" +
Request.QueryString["PersonID"]);
    }
  }
```

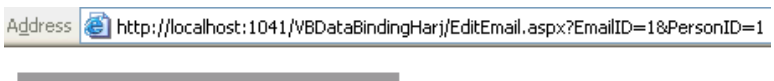
**VB**

```
    Protected Sub FormView1_ItemUpdated(…) Handles
FormView1.ItemUpdated
        If Page.IsValid Then
           Server.Transfer("Emails.aspx?PersonId=" &
Request.QueryString("PersonID"))
        End If
    End Sub

    Protected Sub FormView1_ItemCommand(…) Handles
FormView1.ItemCommand
        If e.CommandName.Equals("Cancel") Then
```

```
            Server.Transfer ("Emails.aspx?PersonId=" &
Request.QueryString("PersonID"))
        End If
    End Sub
```

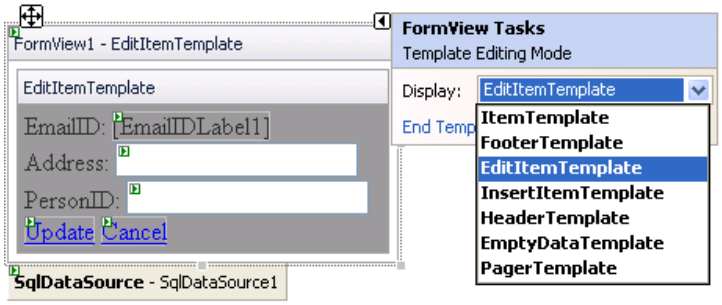
5. Testaa sovellus.



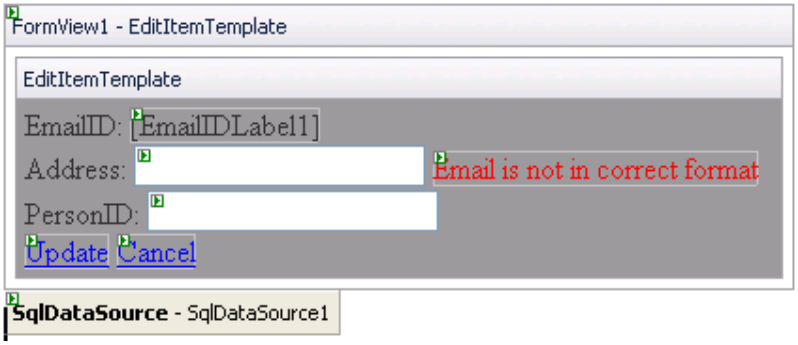


6. Sähköpostin muokkauksesta puuttuu syötteen tarkistaminen. Valitse **Smart Tag | Edit Template**



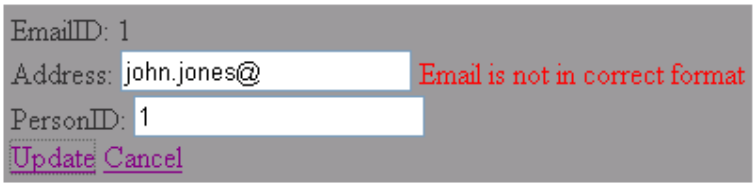
7. Lisää RegularExpressionValidator- kontrolli osoitteen alle.

   Validation Expression = Internet e-mail address
   ErrorMessage = Email is not in correct format
   ControlToValidate = AdressTextBox




Valitse lopuksi **FormView | Smart Tag | End Template Editing**.

Testaa sovellus siten, että syötät virheellisen sähköpostiosoitteen.



# Harjoitus 7: **Optimistinen lukitus**

*Optimistisessa lukituksessa tiedot lukitaan vain päivityksen ajaksi. Yhteentörmäysten havaitseminen tehdään esim. aikaleimalla. Pessimistisellä lukituksella ei voida saavuttaa vastaavaa skaalautuvuutta, koska siinä lukitus on käytössä koko toimenpiteen ajan.*

## Tehtävä

Tutkitaan eri asetusten vaikutusta samanaikaisesti kantaan vietyyn tietoon.

## Toimenpiteet

1. Käynnistä sovellus kahteen eri selaimeen. Editoi samaa henkilöä samanaikaisesti. Kumpi tieto viedään kantaa?

2. Lisää Label –kontrolli nimellä lblAffectedRows.

3. Lisää koodi, joka näyttää rivien lukumäärän, johon päivitys vaikutti.

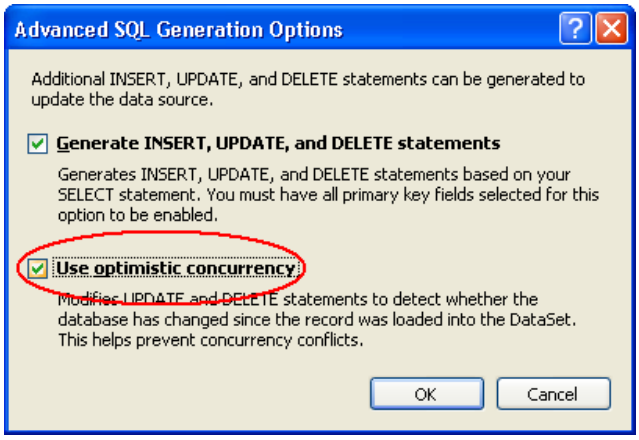
**C#**
```
  protected void SqlDataSource1_Updated(object sender,
SqlDataSourceStatusEventArgs e)
  {
    lblAffectedRows.Text = "Affected rows: "
+e.AffectedRows.ToString();
  }
```

**VB**
```
Protected Sub SqlDataSource1_Updated(ByVal sender As Object,
ByVal e As
System.Web.UI.WebControls.SqlDataSourceStatusEventArgs) …
        lblAffectedRows.Text = "Affected rows: " +
e.AffectedRows.ToString()
End Sub
```

4. Tutki sivun lähdekoodista käytettävää *Update* - lausetta. Sen pitäisi olla muodossa:

```
 UPDATE [Persons] SET [Name] = @Name, [Age] = @Age, [Country]
= @Country WHERE [PersonID] = @PersonID
```

5. Muuta SQL-lauseen generointia siten, että käytössä on optimistinen lukitus.




6. Tutki *Update* sql-lauseen rakennetta. Miten se on muuttunut? Mikä arvo menee kantaan?

   *Use optimistic concurrency vaikuttaa SQL-lauseiden lisäksi mm. seuraaviin ominaisuuksiin*

   *ConflictDetection = CompareAllValues*

   *OldValuesParameterFormatString = original_{0}*

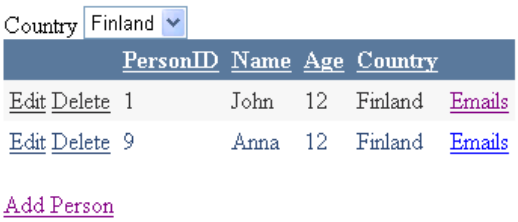
# Harjoitus 8: **Virheiden käsittely ja tietojen poisto**

## Tehtävä

Tietokannan viite-eheys voi aiheuttaa virheen henkilöä poistettaessa. Nykytilanteessa käyttäjä saa liian matalatasoisen virheilmoituksen, joka korvataan harjoituksessa omalla virhekäsittelyllä.

## Toimenpiteet

1. Lisää henkilöiden poisto-ominaisuus henkilö listaan. Voit tehdä sen **Smart Tag | Enable Deleting**.




2. Kokeile poistaa henkilö, jolla ei ole sähköpostiosoitetta. Onnistuuko?

3. Kokeile poistaa henkilö, jolla on sähköpostiosoite. Kannan eheystarkistuksen pitäisi laukaista seuraava virhe: *The DELETE statement conflicted with the REFERENCE constraint "FK_Emails_Persons". The conflict occurred in database "Contacts", table "dbo.Emails", column 'PersonID'. The statement has been terminated.*

   Kyseisen muotoista virhettä ei pitäisi näyttää loppukäyttäjälle. Nykytilanteessa virhettä ei käsitellä lainkaan vaan virhe valuu oletuskäsittelijään.

4. Lisää toteutukseen virheidenkäsittelijä SqlDataSource1 -kontrolliin. Koodissa virhe poistetaan ja tieto siitä tallennetaan sessioon.

**C#**
```
  protected void SqlDataSource1_Deleted(object sender,
SqlDataSourceStatusEventArgs e){
    if(e.Exception != null)    {
        Context.Items["PersonDBError"] = e.Exception.Message;

        Server.ClearError();
        Server.Transfer("Persons.aspx");
    }
}
```

```
VB
Protected Sub SqlDataSource1_Deleted(ByVal sender As Object,
ByVal e As
System.Web.UI.WebControls.SqlDataSourceStatusEventArgs) …
    If e.Exception IsNot Nothing Then
           Context.Items("PersonDBError") = _
                  e.Exception.Message
            Server.ClearError()
            Server.Transfer("Persons.aspx")
    End If
End Sub
```

Voit ajaa sovelluksen. Sovellus ei ns. kaadu henkilön poistoon. Toisaalta käyttäjä ei vielä saa mitään ilmoitusta tilanteesta.

5. Lisää *Persons.aspx* –sivulle *Label* nimellä *lblError*, joka ei ole näkyvissä (*Visible = false*).

6. Lisäksi tarvitsemme koodin *Page_Load* – tapahtumaan, jossa tarkistetaan sessioon mahdollisesti tallennettu virhe ja tuodaan se esille *lblError* –kontrolliin.

```
C#
    protected void Page_Load(object sender, EventArgs e){
      if (Context.Items["PersonDBError"] != null){
        lblError.Text =
             Context.Items["PersonDBError"].ToString();
        lblError.Visible = true;
        Context.Items["PersonDBError"] = null;
      } else {
        lblError.Visible = false;
      }
    }
```

```
VB
Protected Sub Page_Load(…) Handles Me.Load
        If Context.Items("PersonDBError") IsNot Nothing Then
            lblError.Text = _
               Context.Items("PersonDBError").ToString()
            lblError.Visible = True
            Context.Items("PersonDBError") = Nothing
        Else
            lblError.Visible = False
        End If
End Sub
```

# Harjoitus 9: **Erillinen tietokantakerros**

*Tietokantakerroksen eriyttäminen mahdollistaa sen uudelleenkäytön.*

## Tehtävä

Harjoituksessa toteutetaan erillinen tietokantakerros, joka on erotettu aspx-sivulta luokkaan.

## Toimenpiteet

1. Lisää *App_code* – hakemistoon uusi luokka nimellä *DBLayer*.

2. Luokkaan tehdään *GetEmails* – metodi, jonka avulla voidaan hakea kaikki sähköpostiosoitteet. Toteutus tehdään käyttämällä ADO.NET 2.0:n *Factory* rakennetta, jota voidaan käyttää lähes kaikkia kantoja vasten (koodin oletus on kuitenkin SQL Server).

**C#**
```
using System.Data.Common;

public class DBLayer
{
  public DataSet GetEmails()
  {
     string constr = "Data Source=localhost;Initial
Catalog=Contacts;Integrated Security=True";
     string provider="System.Data.SqlClient";
     string sql = "select * from emails";

    DataSet ds = new DataSet();
    DbProviderFactory factory =
DbProviderFactories.GetFactory(provider);
    DbConnection con = factory.CreateConnection();
    DbCommand cmd = con.CreateCommand();
    DbDataAdapter adap = factory.CreateDataAdapter();
    adap.SelectCommand = cmd;

    con.ConnectionString = constr;
     cmd.CommandText = sql;

     adap.Fill(ds, "Emails");
     return ds;
  }
}
```

```
VB
Imports System.Data
Imports System.Data.Common

Public Class DBLayer
    Public Function GetEmails() As DataSet
        Dim constr As String
        Dim provider As String
        Dim sql As String

        constr = "Data Source=localhost;Initial
Catalog=Contacts;Integrated Security=True"
        provider = "System.Data.SqlClient"
        sql = "select * from emails"

        Dim ds As New DataSet
        Dim factory As DbProviderFactory
        factory = DbProviderFactories.GetFactory(provider)

        Dim con As DbConnection = factory.CreateConnection()
        Dim cmd As DbCommand = con.CreateCommand()
        Dim adap As DbDataAdapter =
factory.CreateDataAdapter()

        adap.SelectCommand = cmd

        con.ConnectionString = constr
        cmd.CommandText = Sql

        adap.Fill(ds, "Emails")
        Return ds

    End Function
End Class
```

3. Käännä sovellus.

4. Lisää harjoitukseen uusi aspx-sivu nimellä *AllEmails.aspx*.

5. Lisää sivulle GridView - kontrolli.

6. Valitse **Smart Tag | Choose Data Source…**. Ruudulle käynnistyy velho, jossa voit määrittää käytettävän tietolähteen. Tee seuraavat valinnat velhossa

   - Select a datasource: **<New Data Source>**
   - Choose DataSource Type: **Object**
   - Choose a Business Object: **DBLayer**
   - Define Data Methods: **Select tab | Choose a method | GetEmails**

7. Testaa sovellus.

8. Korjaa *GetEmails* –metodia siten, että käytettävät tietokantaluokat ja yhteystiedot haetaan *Web.Config* –tiedostosta. Tietojen pitäisi olla siellä valmiina edellisten harjoitusten jäljiltä. Tarkista ensimmäiseksi, että *Web.config* –tiedostosta löytyy tarvittavat rivit.

```
<connectionStrings>
<add name="ContactsConnectionString1" connectionString="Data
Source=localhost;Initial Catalog=Contacts;Integrated
Security=True" providerName="System.Data.SqlClient"/>
</connectionStrings>
```

9. Toteuta muutokset *GetEmails* –metodiin.

**C#**
```
string constr =
ConfigurationManager.ConnectionStrings["ContactsConnectionStri
ng1"].ConnectionString;

string provider =
ConfigurationManager.ConnectionStrings["ContactsConnectionStri
ng1"].ProviderName;
```

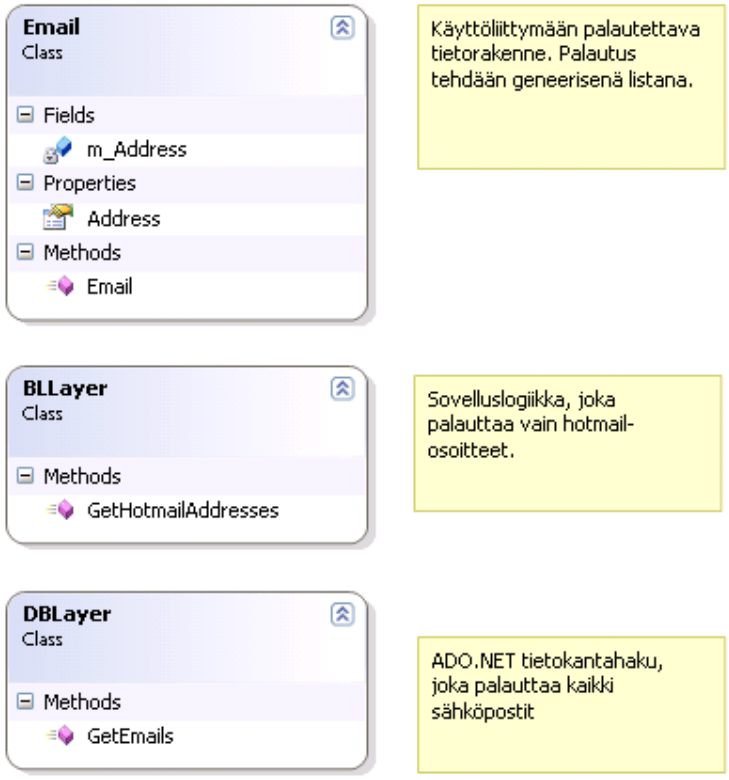
**VB**
```
constr = _

ConfigurationManager.ConnectionStrings("ContactsConnectionStri
ng1").ConnectionString

provider = _

ConfigurationManager.ConnectionStrings("ContactsConnectionStri
ng1").ProviderName
```

# Harjoitus 10: **Erillinen sovelluslogiikka kerros**

*Eriytetty sovelluslogiikka peittää käyttöliittymältä täysin käytettävän tietolähteen. Tietoa voidaan koostaa esim. useammasta eri tietolähteestä ilman, että se näkyy käyttöliittymään.*

## Tehtävä

Kuvasta nähdään sovelluksen yleinen rakenne.



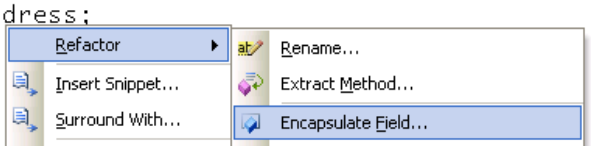

## Toimenpiteet

1. Toteutetaan ensin tietorakenne ja refaktoroidaan se.

   *Huom. VB.NET ei oletuksena sisällä refaktorointia. Se voidaan asentaa kehitysympäristöön jälkikäteen.*

**C#**
```
public class Email
{
  public string m_Address;
}
```

```
public class Email
{
  public string m_Address;
}
```




2. Toteuta lisäksi konstruktori, jossa sähköpostiosoite voidaan antaa. Lopputuloksen tulisi näyttää seuraavalta

**C#**
```
public class Email
{
  private string m_Address;

  public string Address
  {
    get { return m_Address; }
    set { m_Address = value; }
  }
  public Email(string Address)
  {
    this.Address = Address;
  }
}
```

**VB**
```
Public Class Email
    Private m_Address As String

    Public Property Address() As String
        Get
            Return m_Address
        End Get
        Set(ByVal value As String)
            m_Address = value
        End Set
    End Property

    Public Sub New(ByVal Address As String)
        Me.Address = Address
    End Sub
End Class
```

3. Lisää sovellukseen “BLLayer” –luokka.

4. Luokkaan toteutetaan metodi, joka palauttaa vain hotmail-osoitteet. Tietokantakoodi pidetään erillään kyseisestä sovellusloogisesta luokasta.

**C#**
```
using System.Collections.Generic;

public class BLLayer
{
  public List<Email> GetHotmailAddresses(){
    DBLayer dbl = new DBLayer();
    DataSet ds = dbl.GetEmails();
    List<Email> ea = new List<Email>();
    string s;

    foreach(DataRow r in ds.Tables["Emails"].Rows){
      s = r["Address"].ToString();
      if(s.Contains("hotmail")){
        ea.Add(new Email(s));
      }
    }

    return ea;
   }
}
```

**VB**
```
Public Class BLLayer
    Public Function GetHotmailAddresses() As List(Of Email)
        Dim ea As New List(Of Email)
        Dim ds As DataSet
        Dim db As New DBLayer
        Dim s As String

        ds = db.GetEmails()

        Dim r As DataRow
        For Each r In ds.Tables("Emails").Rows
            s = r("Address").ToString()
            If s.Contains("hotmail") Then
                ea.Add(New Email(s))

            End If
        Next

        Return ea
    End Function
End Class
```

5. Lisää sovellukseen aspx –sivu nimellä *Hotmails.aspx*.

6. Toteuta sivulle *GridView* ja *ObjectDataSource* –kontrollien avulla rakenne, joka käyttää *BLLayer* –luokan *GetHotmailAddresses* –metodia.

7. Käännä ja testaa sovellus.

# Harjoitus 11: **Tyypitetty DataSet ja DataList**

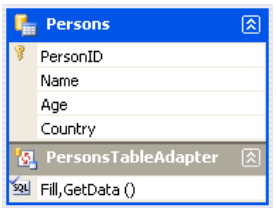
*Tyypitetyssä DataSetissa on taustalla aina valmis XML-skeema. Tyypittämättömässä DataSetissa skeema luodaan lennosta.*

## Tehtävä

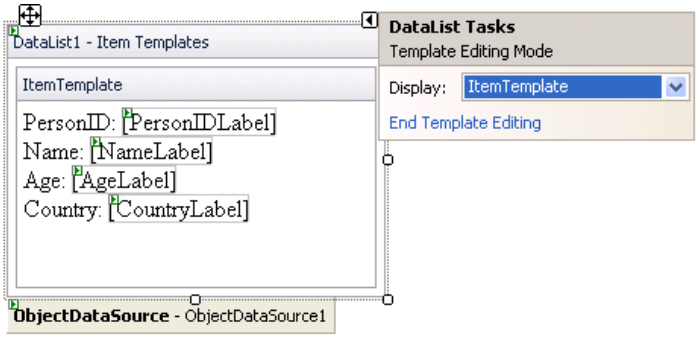
Toteutetaan tyypitetty *DataSet*, johon haetaan henkilötiedot. Tiedot visualisoidaan *DataList* - kontrolliin.

## Toimenpiteet

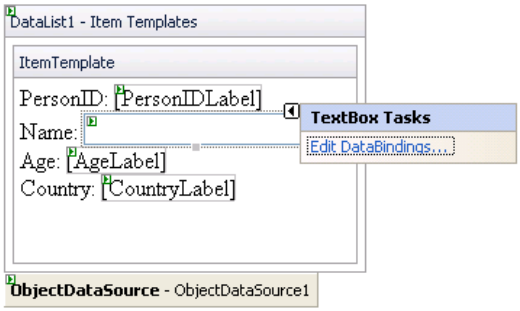
1. Lisää sovellukseen *DataSet* nimellä *MyDataSet.* Ota *MyDataSet* esille tupla klikillä. Ruudulle tulee *TableAdapter Configuration Wizard*, jonka voi sulkea *Cancel* – napista.
2. raahaa datasettiin *Persons* –taulu. Lopputuloksen tulisi näyttää seuravalta:




3. Valitse ponnahdusvalikosta **View Code** ja tutki tiedoston rakennetta. Tiedoston alussa on SQL-lauseet ja lopussa käytettävä xml skeema. Käännä sovellus.
4. Lisää sovellukseen uusi aspx-sivu nimellä *PersonList.aspx*.
5. Sivulle tulee *DataList* -kontrolli, jonka tietolähteeksi tulee uusi *ObjectDataSource*.
   - Select a datasource: **<New Data Source>**
   - Choose DataSource Type: **Object**
   - Choose a Business Object: **MyDataSetTableAdapters.PersonsTableAdapter**
   - Define Data Methods: **Select tab | Choose a method | GetData**
6. Testaa sovellus.
7. Valitse **Smart Tag | Edit Templates**. Tee muutokset **ItemTemplate** -pohjalle.




8. Poista NameLabel ja korvaa se *TextBox* –kontrollilla.




9. Editoi sidontaa seuraavasti:

   - Sidottava ominaisuus: **Text**
   - Field binding / Bound to: **Name**
   - Format: **(none)**

10. Testaa sovellus. Nimen pitäisi näkyä tekstikenttänä.

11. Muokkaa lisäksi **Edit Templates | Header Template**. Muuta listan otsikoksi *All Persons*.

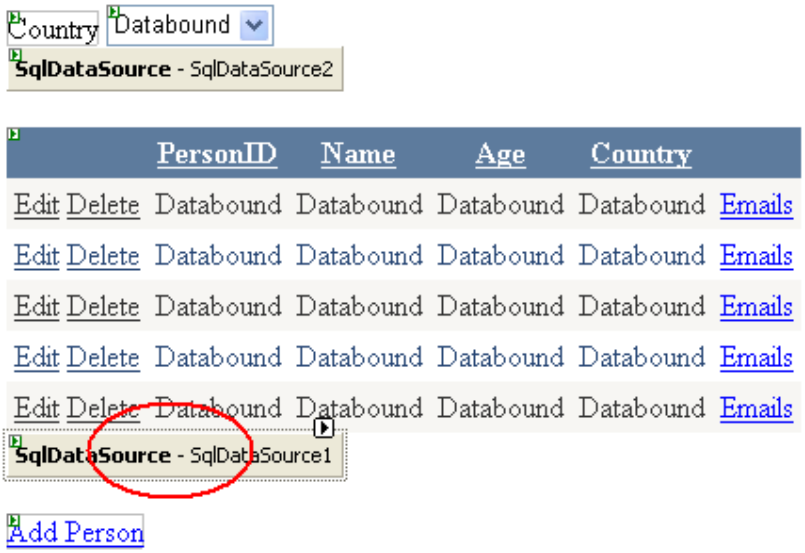
# Harjoitus 12: **Cache**

## Tehtävä

Henkilöiden hakuun liitetään cache-toiminto, joka vähentää tietokantahakujen määrää.

## Toimenpiteet

1. Avaa *Person.aspx* –sivu ja valitse *SqlDataSource* –kontrolli

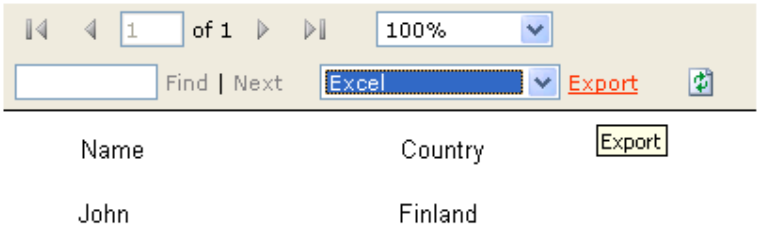



2. Muuta *EnableCaching=true* ja *CacheDuration=60* (aika on sekunteina).
3. Käynnistä sovellus ja lisää henkilö *Add Person* – linkillä.
4. Palaa takaisin *Person.aspx* – sivulle
5. Näkyykö lisätty henkilö välittömästi sivulla? Entä minuutin kuluttua jos päivität sivua?

# Harjoitus 13: **Raportointi**

## Tehtävä

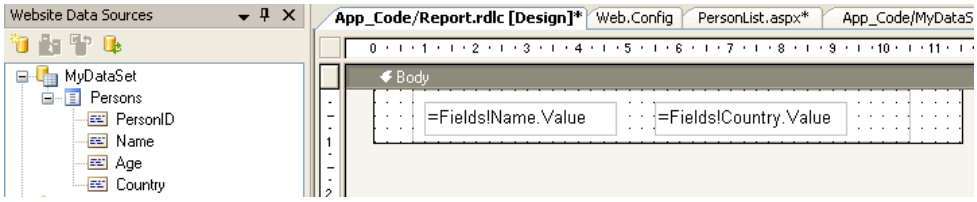
Harjoituksessa toteutetaan raportti, johon tuodaan henkilötiedot kannasta.



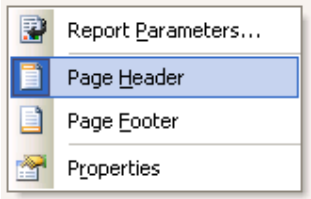

## Toimenpiteet

1. Lisää sovellukseen uusi raportti *App_code* hakemistoon nimellä *Report.rdlc*. Avaa raportti ja raahaa siihen **ToolBox** –ikkunasta *List* - kontrolli.

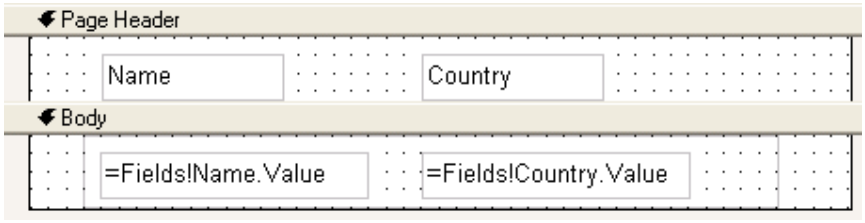
2. Vaihda näkymäksi **Website Data Sources** ikkuna. Rahaa listan sisään sekä *Name* että *Country*, jotka ovat *MyDataSet* kohdan alla.



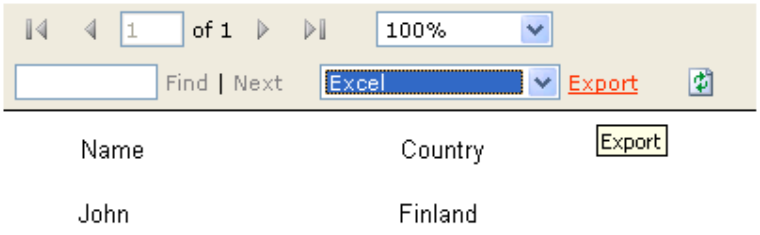

3. Seuraavaksi lisätään raporttiin otsikko. Sen voi tehdä ponnahdusvalikosta.




4. Lisää *Name* ja *Country* otsikot.




5. Lisää sovellukseen *PersonReport.aspx* – sivu.
6. Lisää sivulle *ReportViewe*r –kontrolli.
7. Valitse **Smart Tag | Choose Report | App_code\Report.rdlc**.
8. Käynnistä ja testaa sovellus. Raportin pitäisi sisältää henkilöt ja maat.
9. Kokeile vaihtaa raportin tyyppiä Exceliksi ja kokeile **Export** - toimintoa.




10. Avaa *Report.rdlc* –tiedosto XML editoriin. Voit tehdä sen **Solution Explorerissa | Open with** toiminnolla. Missä muodossa raportti on tallennettu?