ASP.NET Web Services

Web Services tietokantaohjelmoinnin perusteet

Harjoitukset

Sisällys

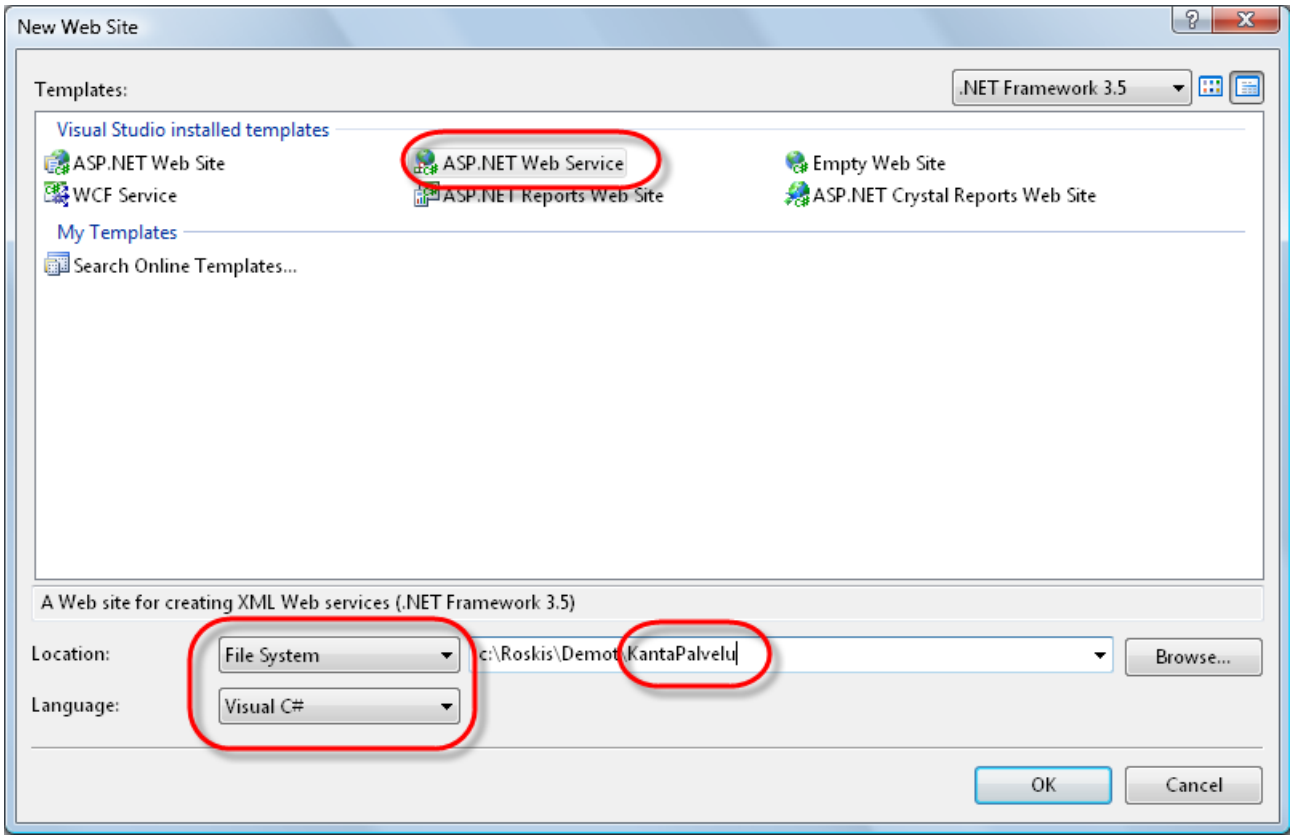
Harjoitus 1: **Tietokannat ja Web Services**

Tehtävä

Harjoituksessa tehdään Web Service, jonka avulla voidaan hakea tietoja tietokannasta. Tässä harjoituksessa käytetään VS:n generoimaa TableAdapteria ja palvelu palauttaa tyypitetyn DataTablen. Tämä paluutyyppi soveltuu vain .NET<-> .NET –tyyppisiin palveluihin.

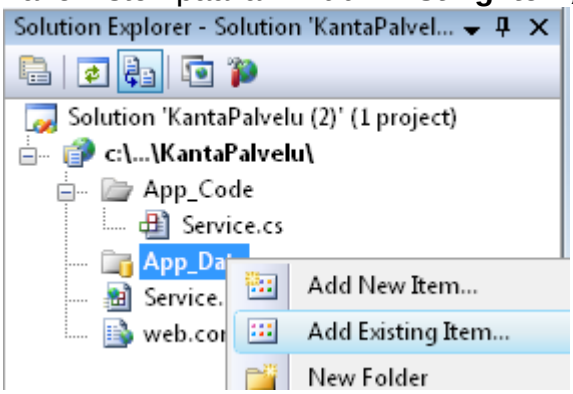
Toimenpiteet

1. Aloita uusi *ASP.NET Web Service* nimellä *KantaPalvelu* (**File > New Web Site > ASP.NET Web Service**). Käytä ASP.NET Developer Serveriä (eli valitse location: **File System**). Palvelu saa olla .**NET Framework 3.5** –sovellus.

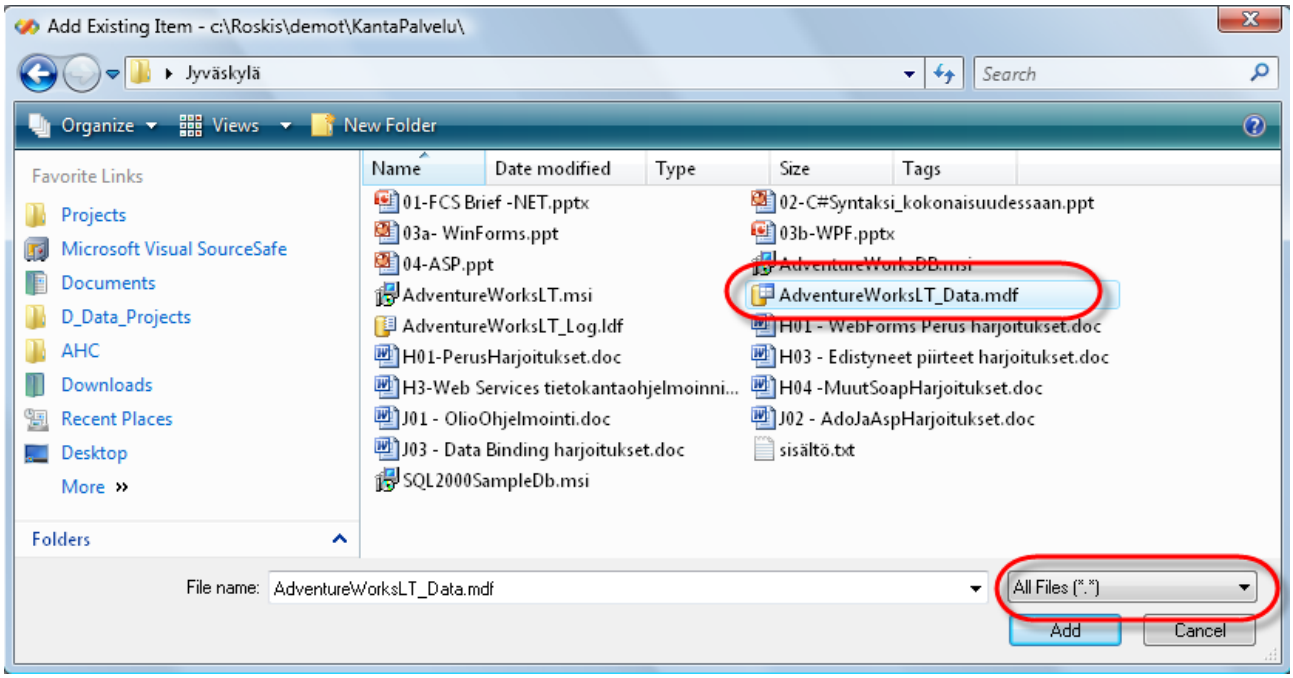


2. Lisää projektiin SQL Server tietokanta AdventureWorksLT

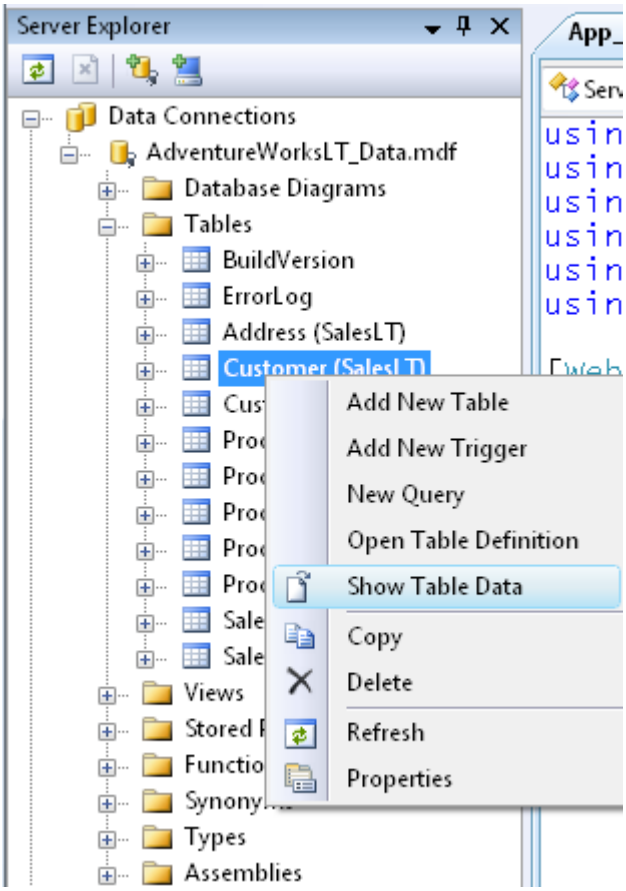
2.1. RClick (==Paina hiiren oikeaa) Solution Explorerissa App_Data –hakemiston päällä > **Add Existing Item**,



2.2. valitse All Files, ja AdventureWorksLT_Data.mdf

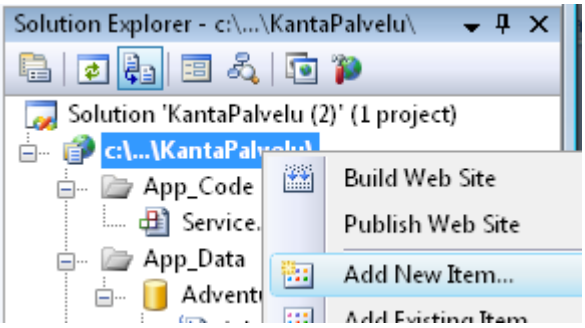


3. Tuplaklikkaa lisättyä tiedostoa (edelleen Solution Explorerissa), jolloin saat avattua tietokannan Server Exploreriin. Siellä näet esim. tietokannan rakenteen ja voit myös ylläpitää kannassa olevaa dataa.

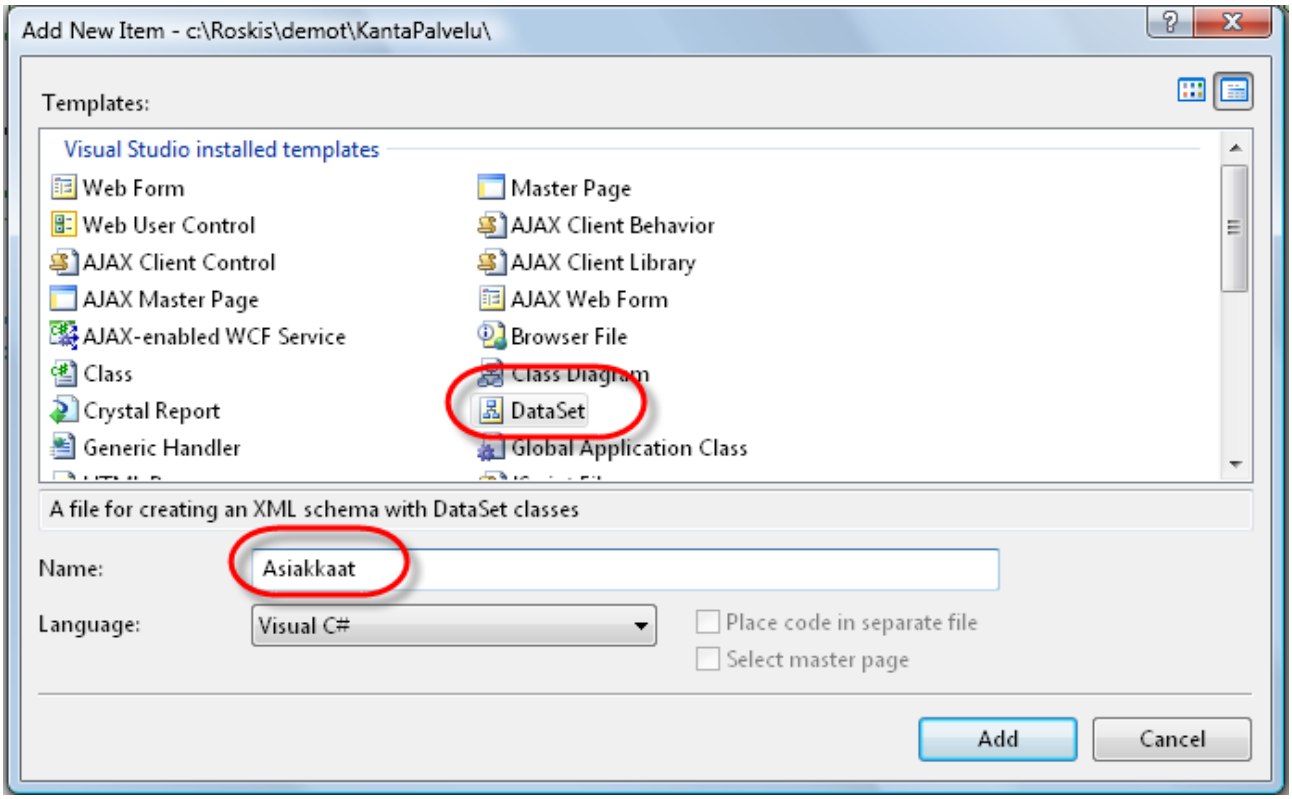


4. Lisää projektiin uusi *DataSet* nimellä Asiakkaat.xsd

4.1. RClick Solution Explorerissa Web Siten päällä > **Add new Item..**

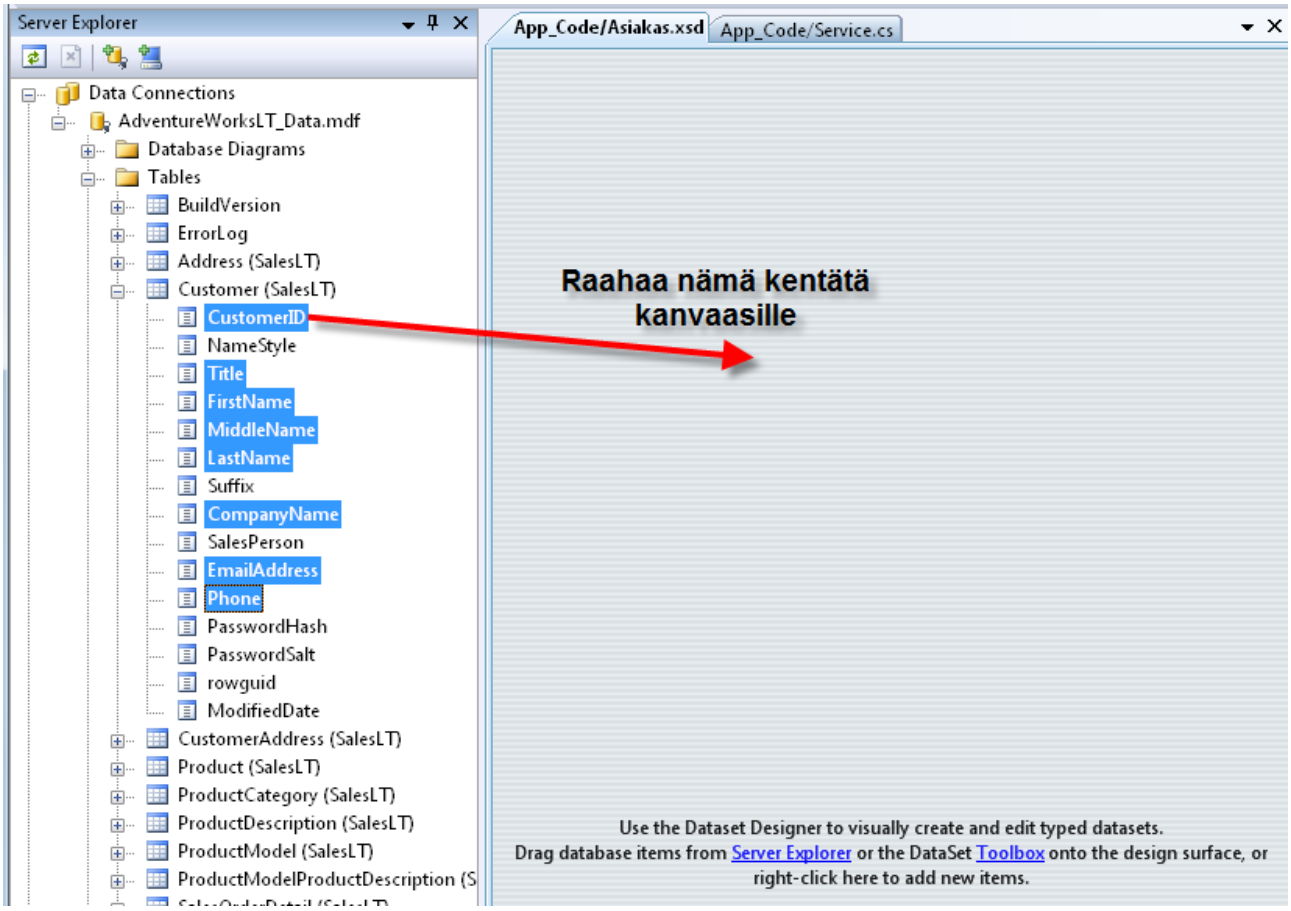


4.2. Valitse Dataset ja anna nimeksi *Asiakkaat.xsd*

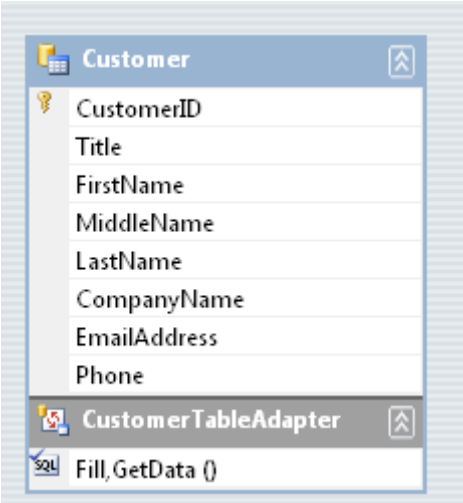


4.1. Kun VS huomauttaa, että dataset kannattaisi lisätä App_Code –hakemistoon, hyväksy tämä (==paina Yes)

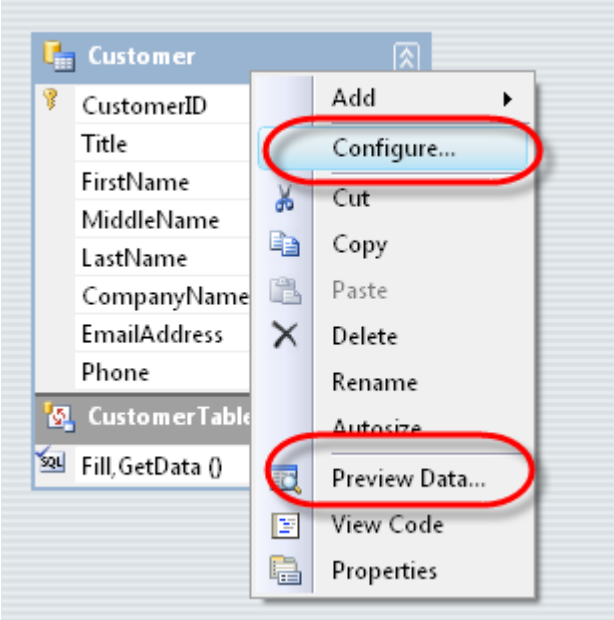
5. Valitse Server Explorerissa Data **Connections>AdventureWorksLT_Data.mdf>Tables>Customer(SalesLT)** –taulusta alla olevat sarakkeet ja raahaa ne Asiakas.XSD –designerin kanvaasille. Tämä muodostaa DataSet-määrittelyn ja tarvittavat SQL-lauseet.



6. Raahauksen jälkeen syntyy tämä DataSet ja sen TableAdapter :it



7. Voit tutkia TableAdapterin ominaisuuksia RClickaamalla Datasettiä ja valitsemalla **Configure**. Voit myös tutkia TableAdapterin palauttamaa dataa **Preview Data** -komennolla.

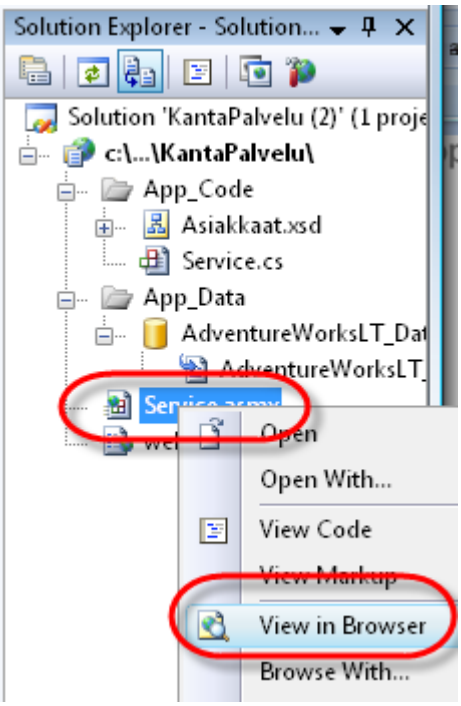


8. Sulje DataSet - editori ja käännä sovellus **(Build | Build Solution)**

9. Lisää *GetScrapReason* – metodi Service.cs – tiedostoon. Metodi hakee tiedot kannasta. Huom: jos teet alustariippumattoman palvelun (==palvelua käytetään myös muista kuin .NET-sovelluksista käsin niin ei ole hyvä idea palauttaa datasetiä tai dataTablea. Nyt teemme palvelun, jota on tarkoitus käyttää vain .NET-clienteista käsin. Siksi DataTablen palauttama metodi)

```
C#
[WebMethod]
public Asiakkaat.CustomerDataTable HaeAsiakkaat() {
  AsiakkaatTableAdapters.CustomerTableAdapter ta
    = new AsiakkaatTableAdapters.CustomerTableAdapter();

  return ta.GetData();
}
```

10. Käännä ja testaa sovellus selain-käyttöliittymällä



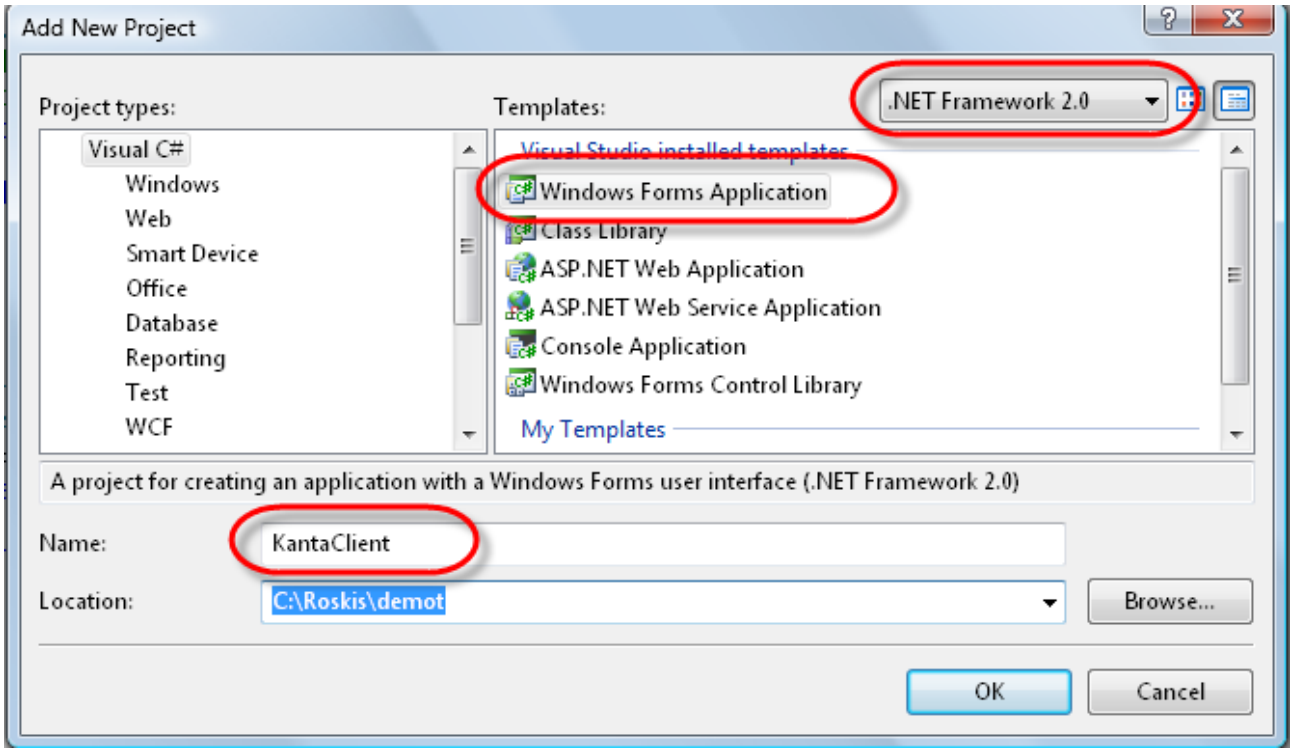
Harjoitus 2: **Windows Client**

Tehtävä

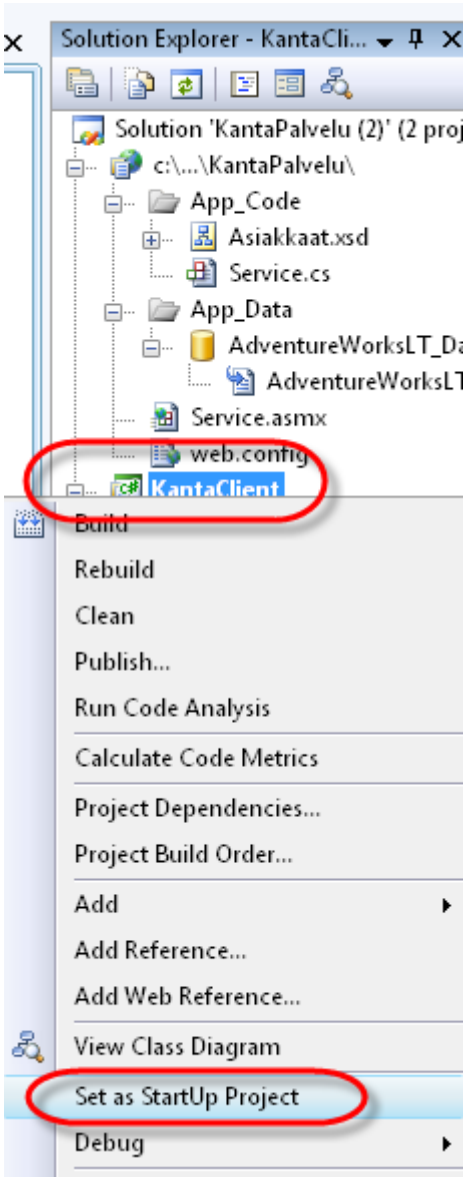
Tee palvelua käyttävä Windows-testeri, joka näyttää asiakastiedot DataGridView-kontrollissa.

Toimenpiteet

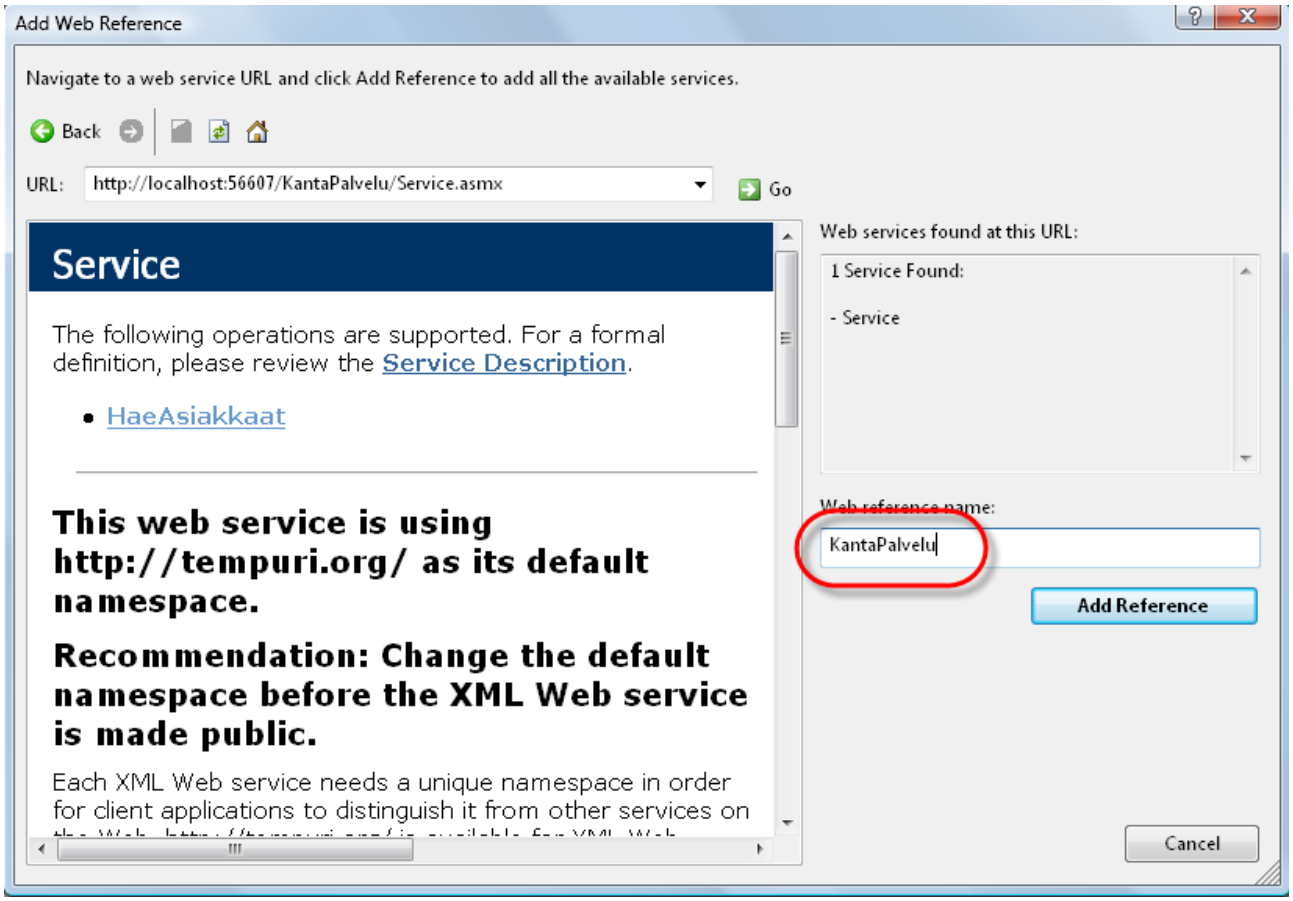
1. Lisää (**File > Add New Project…)** uusi *Windows Application* - projekti nimellä *KantaClient.* Huom.: valitse .NET Framework 2



2. Aseta tämä projekti oletusprojektiksi (RClick projektinimen päällä > **Set as Startup Project**)



3. Lisää Windows sovellukseen referenssi Web Service – palveluun **(Project | Add Web Reference… | WebServices in this solution | Service)** ja anna nimeksi *KantaPalvelu*.



4. Maalaa oheinen käyttöliittymä *DataGridView* ja *Button* - kontrollein

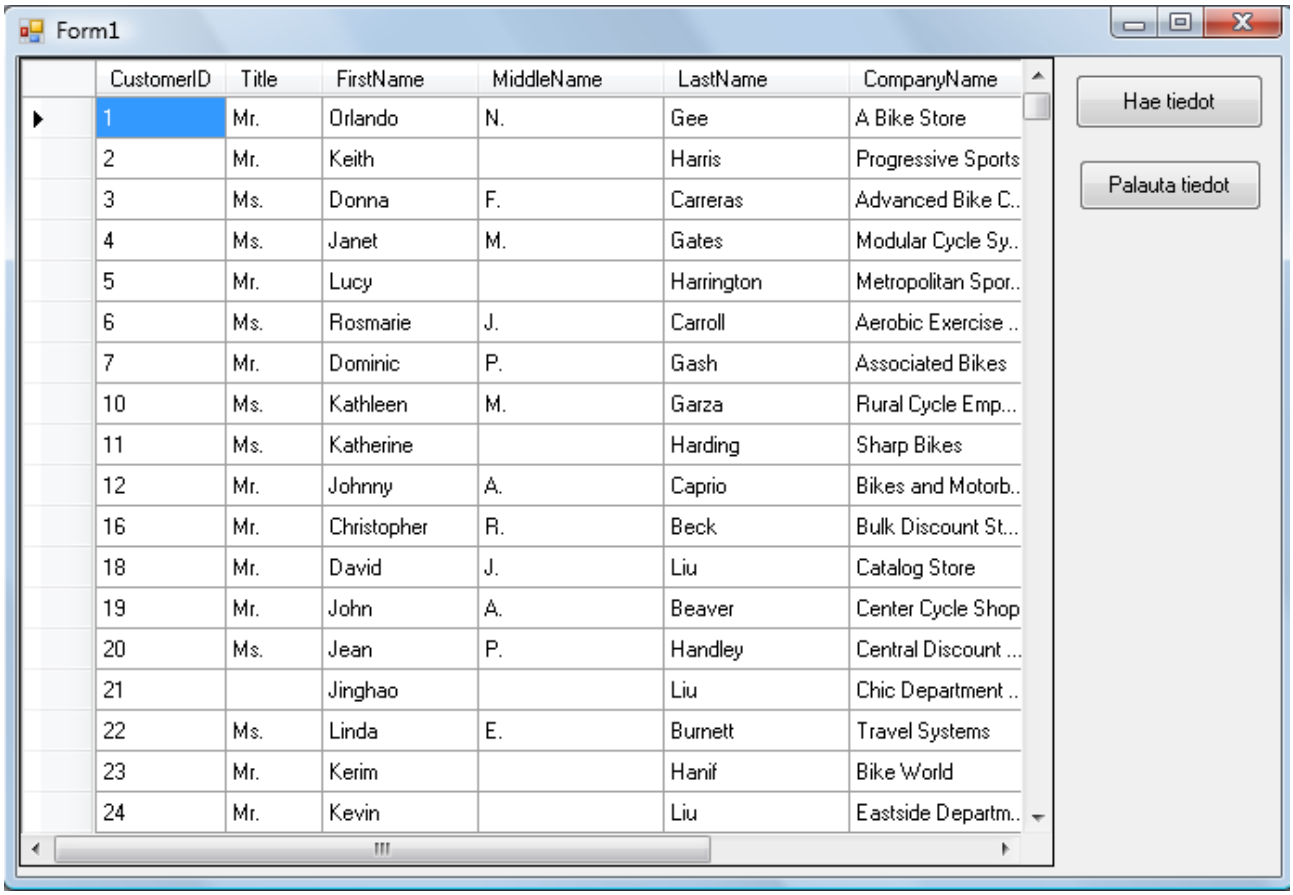
5. Lisää ikkuna-luokan alkuun *KantaPalvelu*–proksiolion luonti. Määrittele myös dtAsiakas-kenttä.

```
C#
public partial class Form1 : Form {
  KantaPalvelu.Service palvelu =
      new KantaClient.KantaPalvelu.Service();

  KantaPalvelu.Asiakkaat.CustomerDataTable dtAsiakas;

  public Form1() {
```

6. Toteuta *Hae Tiedot* – painonappi, jossa tehdään kutsutaan palvelua ja sidotaan tulos-datatable Grid-kontrolliin.

```
C#
private void bHae_Click(object sender, EventArgs e) {
  dtAsiakas = palvelu.HaeAsiakkaat();
  dataGridView1.DataSource = dtAsiakas;
}
```

7. Testaa *Hae Tiedot* – painonapin toiminta.



Harjoitus 3: **Datan päivitys TableAdapterin avulla**

Tehtävä

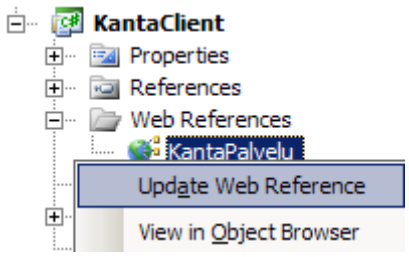
Lisää KantaPalvelut –luokkaan WebMethod, jolla muutetut tiedot voidaan päivittää takaisin tietokantaan. Metodi palauttaa päivitetyn dataTablen.

Toimenpiteet

1. Lisää *KantaPalvelut* projektin Service-luokkaan luokkaan uusi metodi, jolla tiedot viedään takaisin kantaan.

```
C#
[WebMethod]
public Asiakkaat.CustomerDataTable
  PäivitäAsiakkaat
   (Asiakkaat.CustomerDataTable muutetutAsiakkaat)
{
  AsiakkaatTableAdapters.CustomerTableAdapter ta
    = new AsiakkaatTableAdapters.CustomerTableAdapter();

  ta.Update(muutetutAsiakkaat);
  return HaeAsiakkaat();
}
```

2. Käännä sovellus **(Build | Build Solution)** ja päivitä *Web Reference*.



3. Kirjoita *Palauta Tiedot* – nappiin koodi, joka vie muuttuneet tiedot Web Services – palveluun.

```
C#
private void bPalauta_Click(object sender, EventArgs e)
{
  KantaPalvelu.Asiakkaat.CustomerDataTable muutetut;
  muutetut = dtAsiakas.GetChanges()
    as KantaPalvelu.Asiakkaat.CustomerDataTable;

  dtAsiakas = palvelu.PäivitäAsiakkaat(muutetut);
  dataGridView1.DataSource = dtAsiakas;

}
```

4. Käynnistä ja testaa sovellus. Hae kannasta tietoja ja kokeile palauttaa muutettuja tietoja takaisin tietokantaan.

Harjoitus 4: **Yhteentörmäyksen havaitseminen**

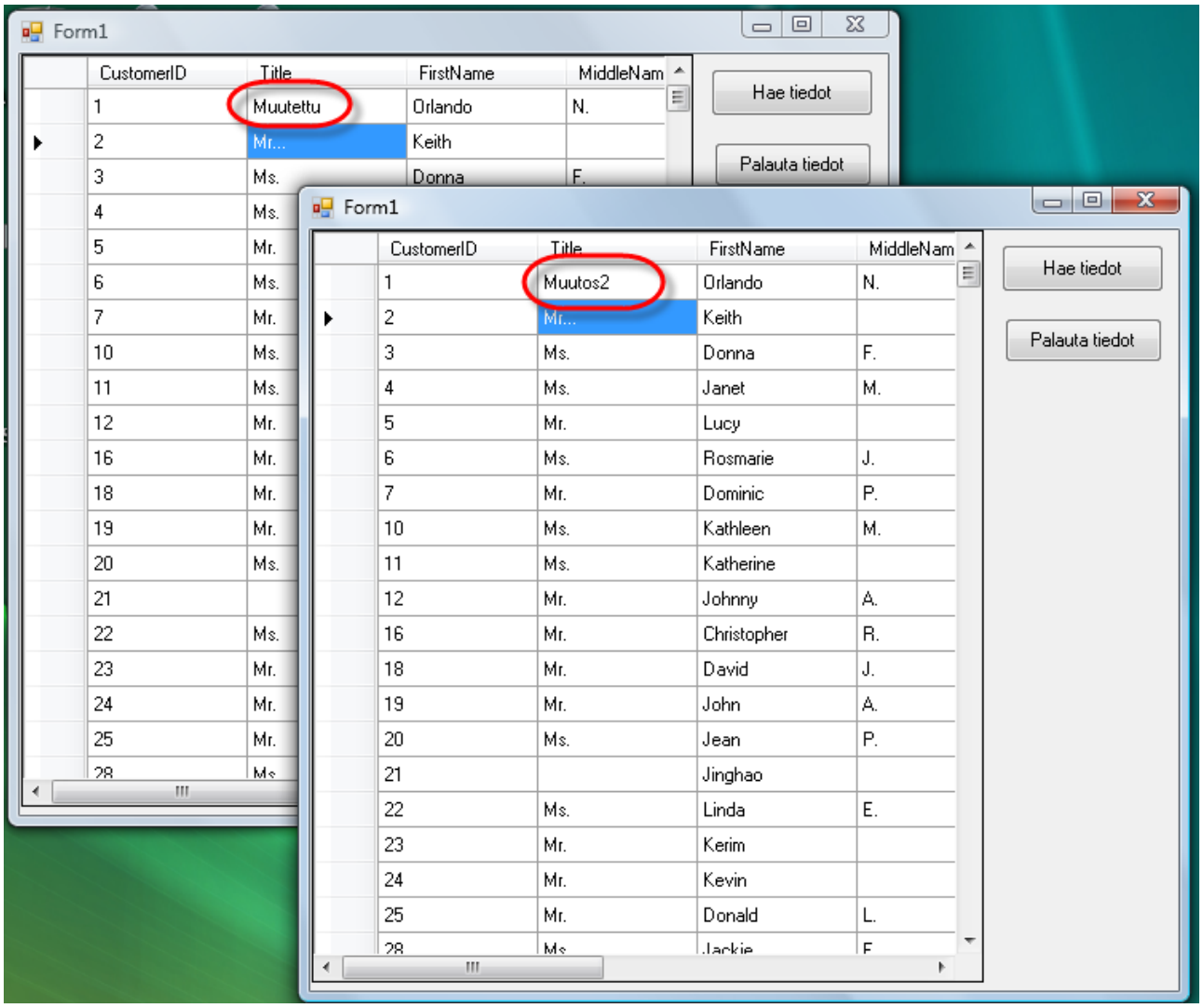
Tehtävä

DataSet – käsittely on aina yhteydetöntä ja yhteentörmäyksien havaitseminen perustuu ns. optimistiseen malliin. Harjoituksessa aiheutetaan tahallinen yhteentörmäys editoimalla samaa riviä. Lisäksi tehdään rakenteet, joilla tämä yhteentörmäys havaitaan.

Oletuksen optimistinen lukitus ei ole käytössä, mutta tässä harjoituksessa se asetetaan käyttöön.

Toimenpiteet

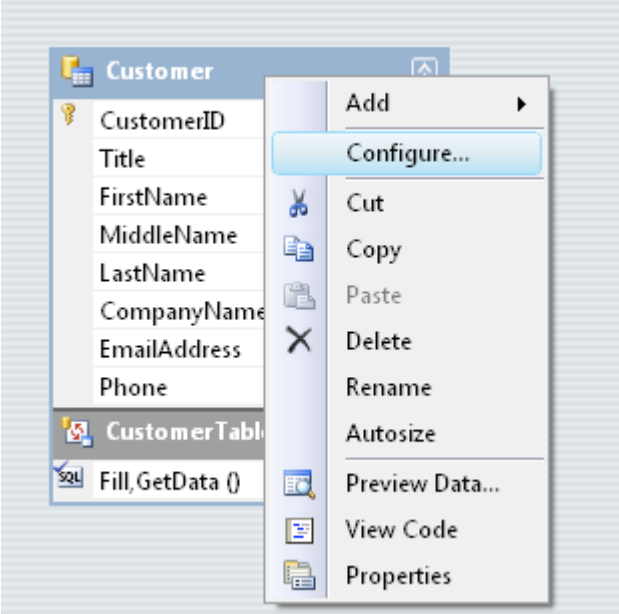
1. Käynnistä kaksi *KantaClient* sovellusta tiedostojärjestelmästä. Hae tiedot molempiin käyttöliittymiin ja muuta samaa riviä. Vie lopuksi tiedot takaisin kantaa *Palauta Tiedot* – painonapilla.



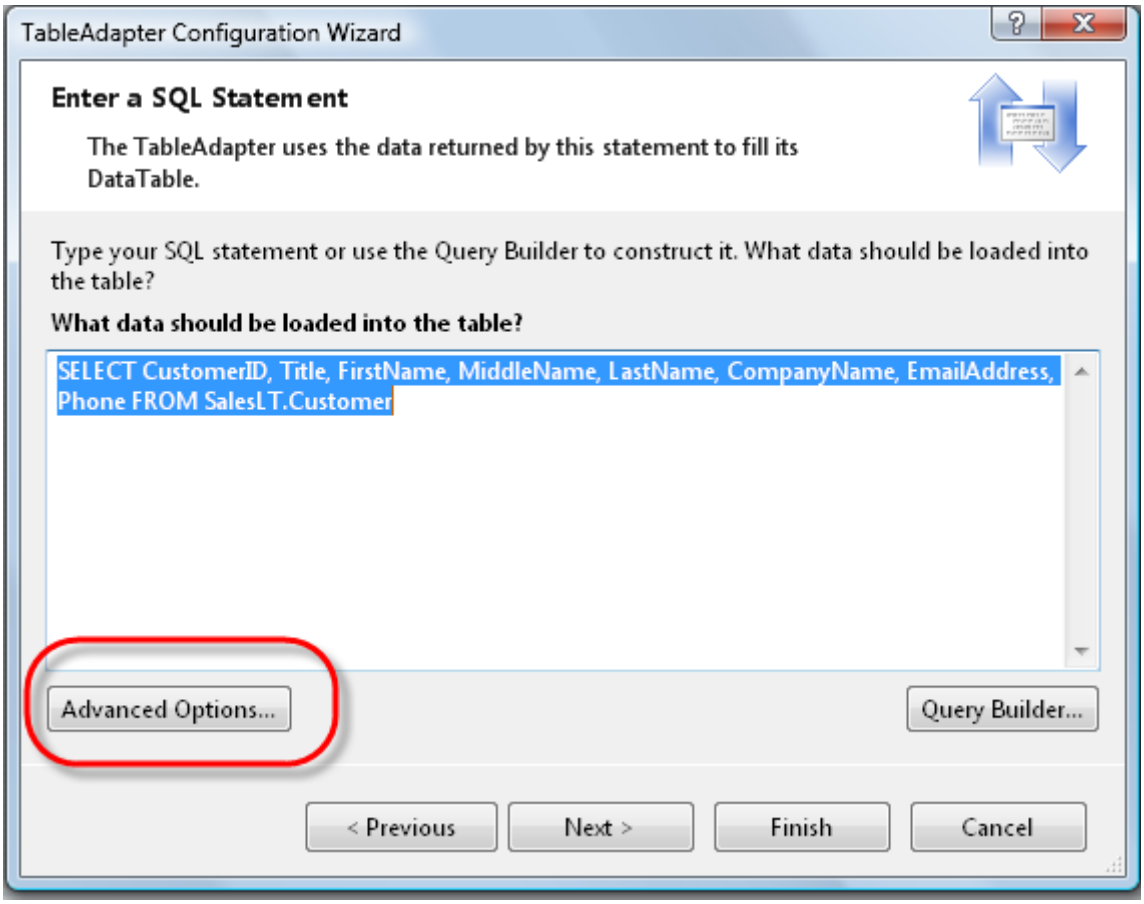
2. Totea, että viimeisin päivitys jää voimaan, optimistinen lukitus ei ole käytössä.

3. Sulje molemmat client-sovellukset ja siirry Visual Studiossa KantaPalvelu-projektiin.

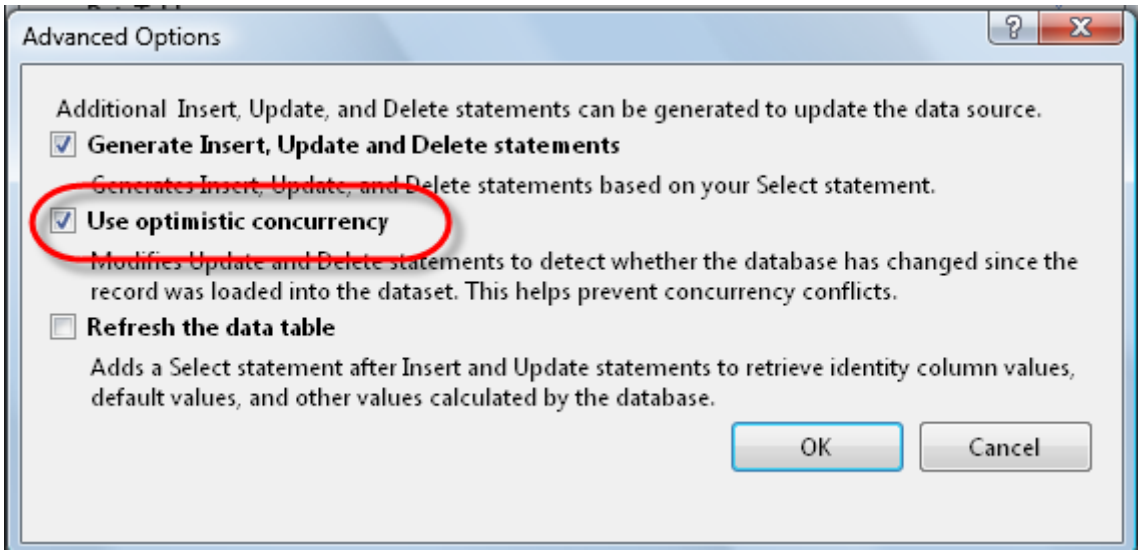
4. RClick Asiakkaat.xsd –designerissa Customer-datatable:n päällä > Configure



5. Paina TableAdapter Configuration Wizard-lomakkeen ”*Advanced Options*…” –nappia



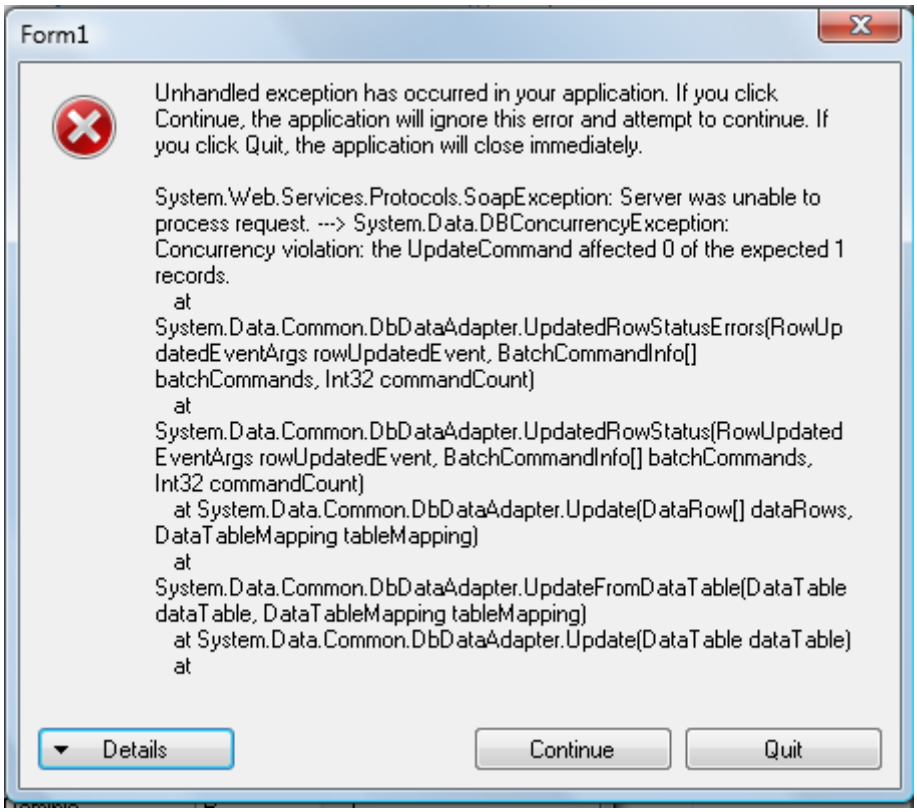
6. Valitse ”*Use optimistic concurrency*”.



7. Sulje Wizardi (Finnish –painonappi)

8. Käännä solution.

9. Testaa palvelua jälleen kahdella clientilla. Totea, että nyt ruudulle tulee *DBConcurrencyException* – poikkeus, jota ei käsitellä.



10. Tee koodiin seuraavat korjaukset. Ensin otetaan poikkeus kiinni *KantaPalvelussa.*.

```
[WebMethod]
public Asiakkaat.CustomerDataTable
PäivitäAsiakkaat(Asiakkaat.CustomerDataTable
muutetutAsiakkaat) {
  AsiakkaatTableAdapters.CustomerTableAdapter ta
    = new AsiakkaatTableAdapters.CustomerTableAdapter();

  try {
    ta.Update(muutetutAsiakkaat);
  }
  catch (System.Data.DBConcurrencyException ex) {
    return muutetutAsiakkaat;
  }

  return HaeAsiakkaat();

}
```

11. Palvelu palauttaa *DataTablen*, jossa on tieto yhteentörmäyksestä. Tee seuraavat muutokset käyttöliittymään, jotka tuovat tiedot näkyviin *DataGridView* – kontrolliin. Mikäli palautetussa datatablessa (muutetut) on virheitä, se ei korvaakaan nyt näytettävää datatablea, vaan se merga:taan siihen.

```
C#
private void bPalauta_Click(object sender, EventArgs e)
{
  KantaPalvelu.Asiakkaat.CustomerDataTable muutetut;
  muutetut = dtAsiakas.GetChanges()
      as KantaPalvelu.Asiakkaat.CustomerDataTable;

  muutetut = palvelu.PäivitäAsiakkaat(muutetut);

  if (!muutetut.HasErrors) {
    dtAsiakas = muutetut;
    dataGridView1.DataSource = dtAsiakas;
  }
  else {
    dtAsiakas.Merge(muutetut);
  }

}
```

12. Käynnistä kaksi *KantaClient* sovellusta tiedostojärjestelmästä. Hae tiedot molempiin ja muuta samaa riviä. Vie lopuksi tiedot takaisin kantaa *Palauta Tiedot* – painonapilla.

13. Tarkista, että yhteentörmäys näkyy *DataGridView* - kontrollissa.