How to use the Entity Framework

Objectives

Applied

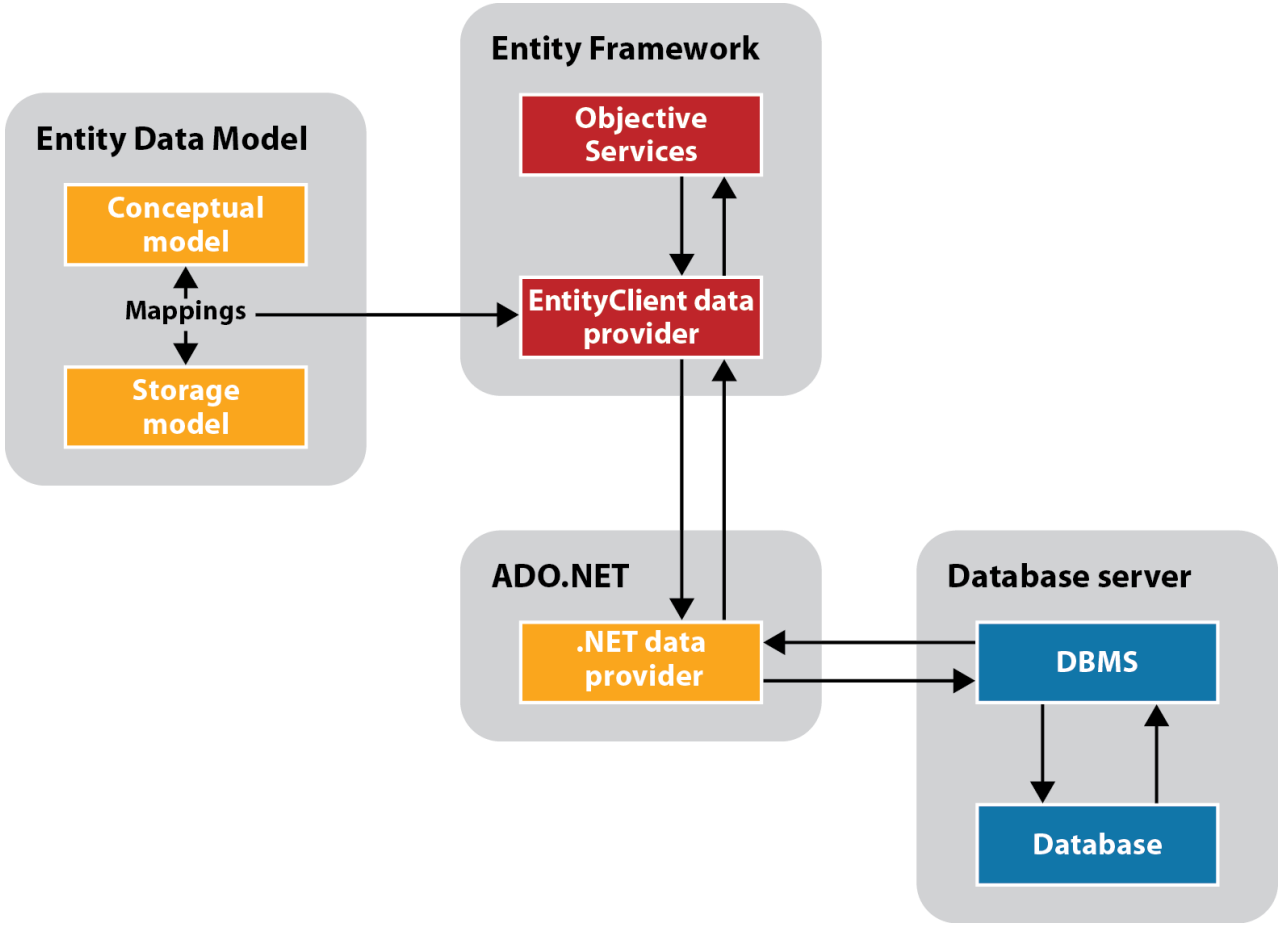
1. Create an Entity Data Model that contains one or more entity classes.
2. Use the Entity Data Model Designer to work with an Entity Data Model.
3. Use LINQ to Entities to retrieve data from one or more database tables.
4. Use the Entity Framework to add, modify, and delete rows in a database table.

Objectives (cont.)

Knowledge

1. In general terms, explain how the Entity Framework works.
2. Describe these terms as they relate to an Entity Data Model: conceptual model, storage model, mappings, object context, entity class, navigation property, and association.
3. Explain how you can provide for concurrency when using the Entity Framework.

How the Entity Framework works



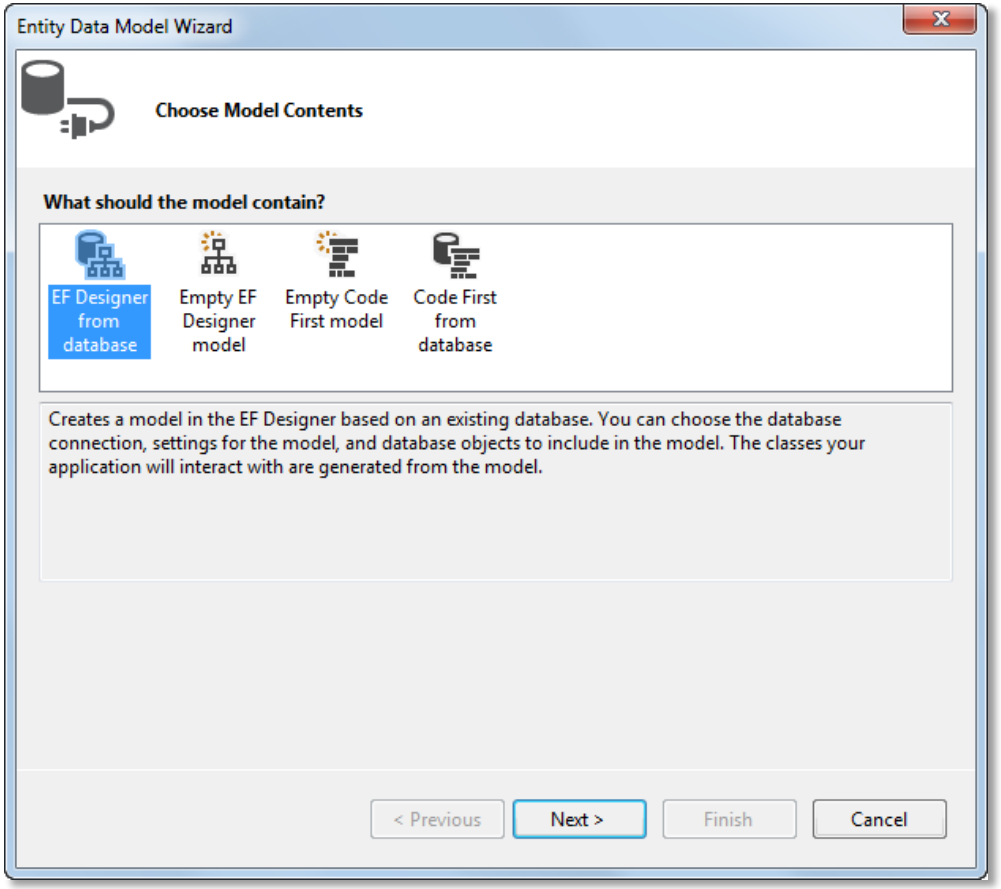
Entity Framework concepts

- The *Entity Framework* provides a layer between the database used by an application and the objects used by an application. This layer provides an interface that allows the data in the database to be presented to the application as objects.
- To provide the interface between objects and database, the Entity Framework uses an *Entity Data Model* that defines a *conceptual model*, a *storage model*, and *mappings* between the two models.
- When you execute a query against a conceptual model, *Object Services* works with the *EntityClient data provider* and the Entity Data Model to translate the query into one that can be executed by the database. When the results are returned from the database, Object Services translates them back to the objects defined by the conceptual model.
- The Entity Framework also provides for tracking changes and for submitting those changes to the database.

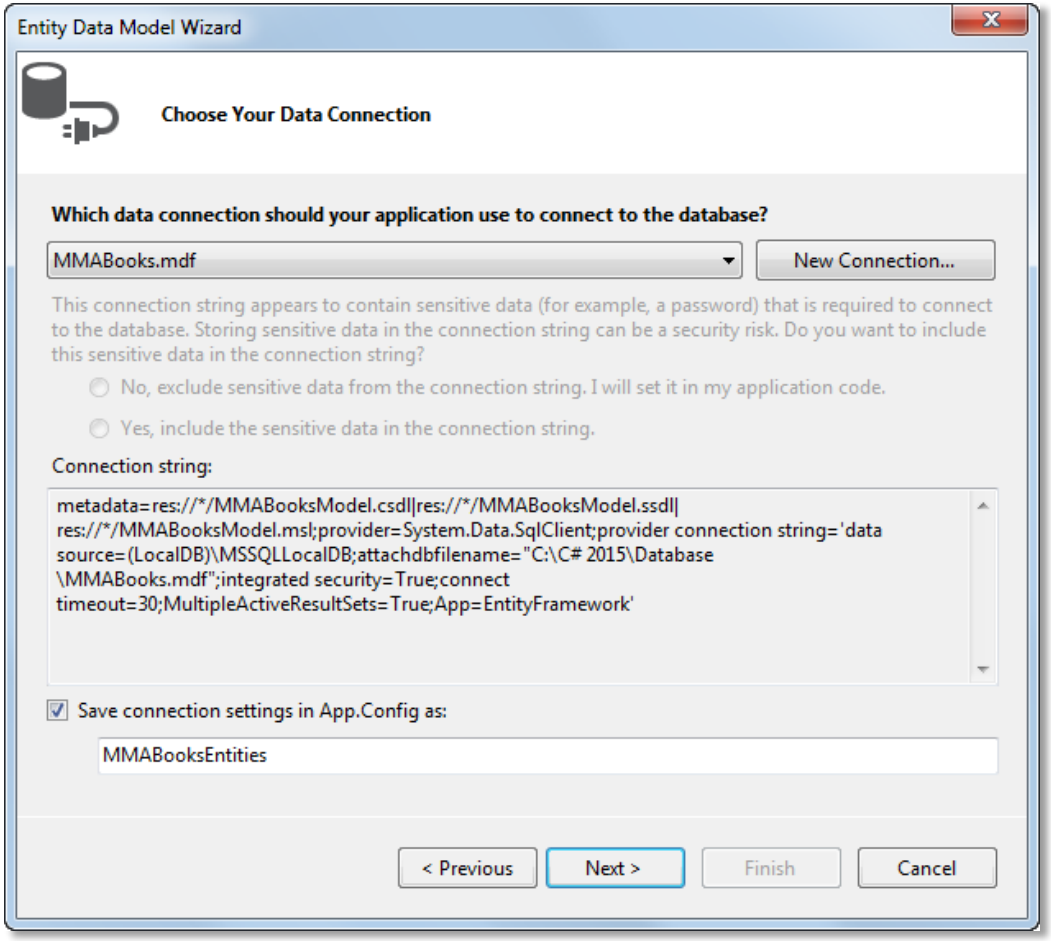
Three ways to query a conceptual model

- LINQ to Entities
- Entity SQL
- Query builder methods

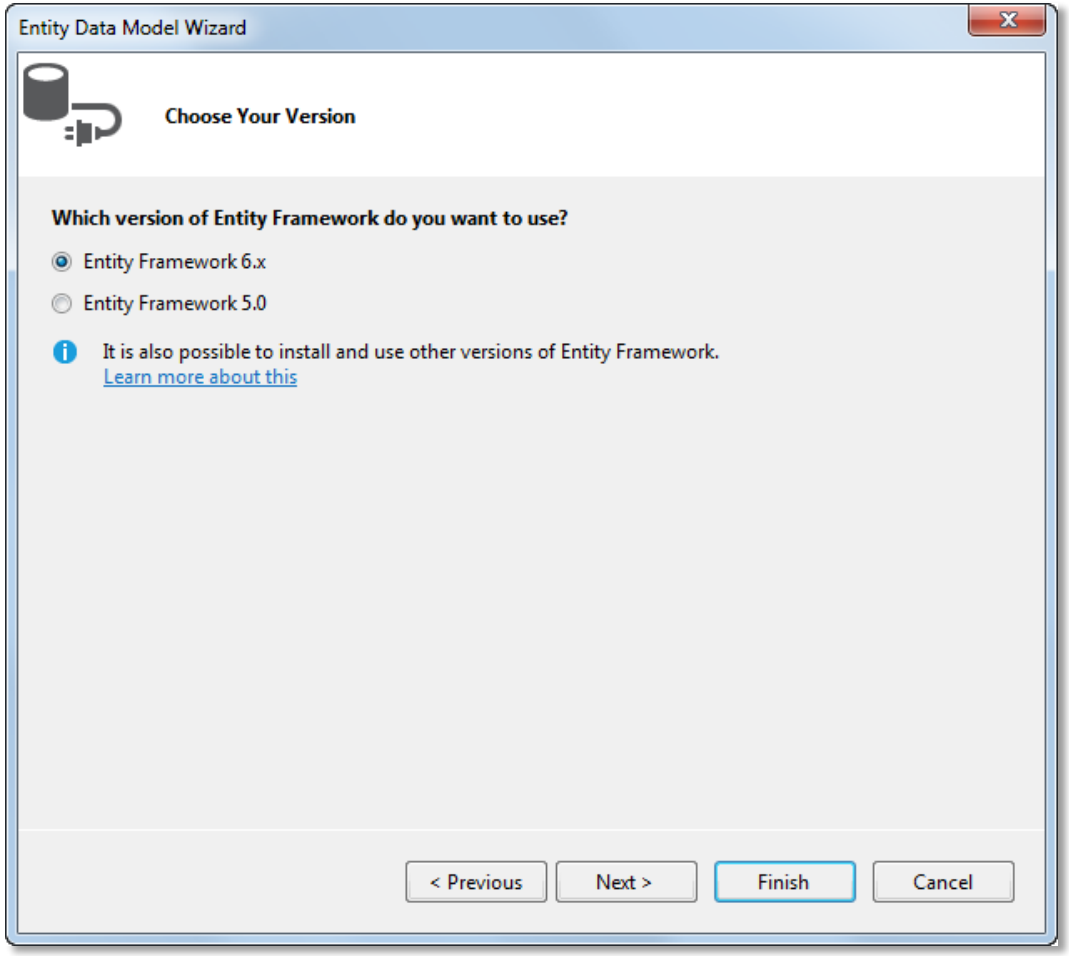
The Entity Data Model Wizard: Step 1



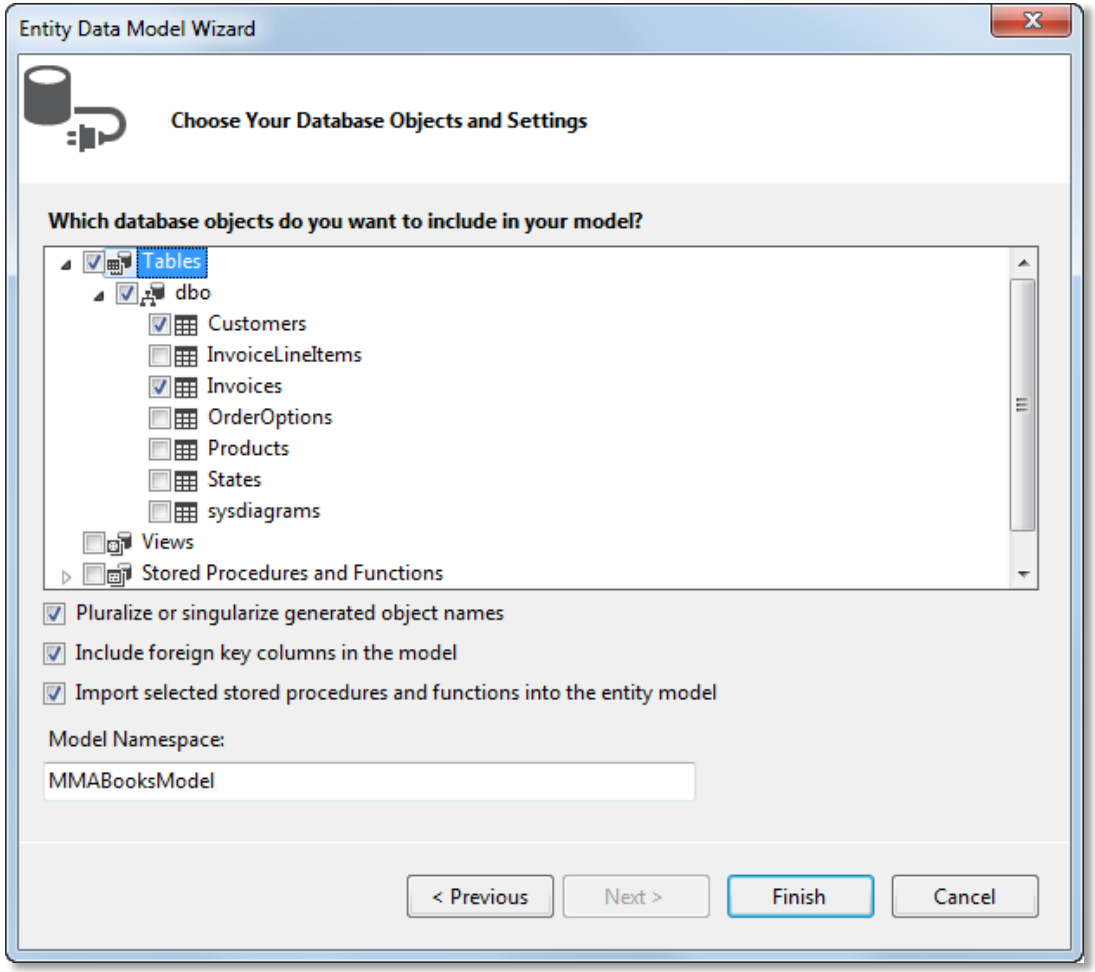
The Entity Data Model Wizard: Step 2



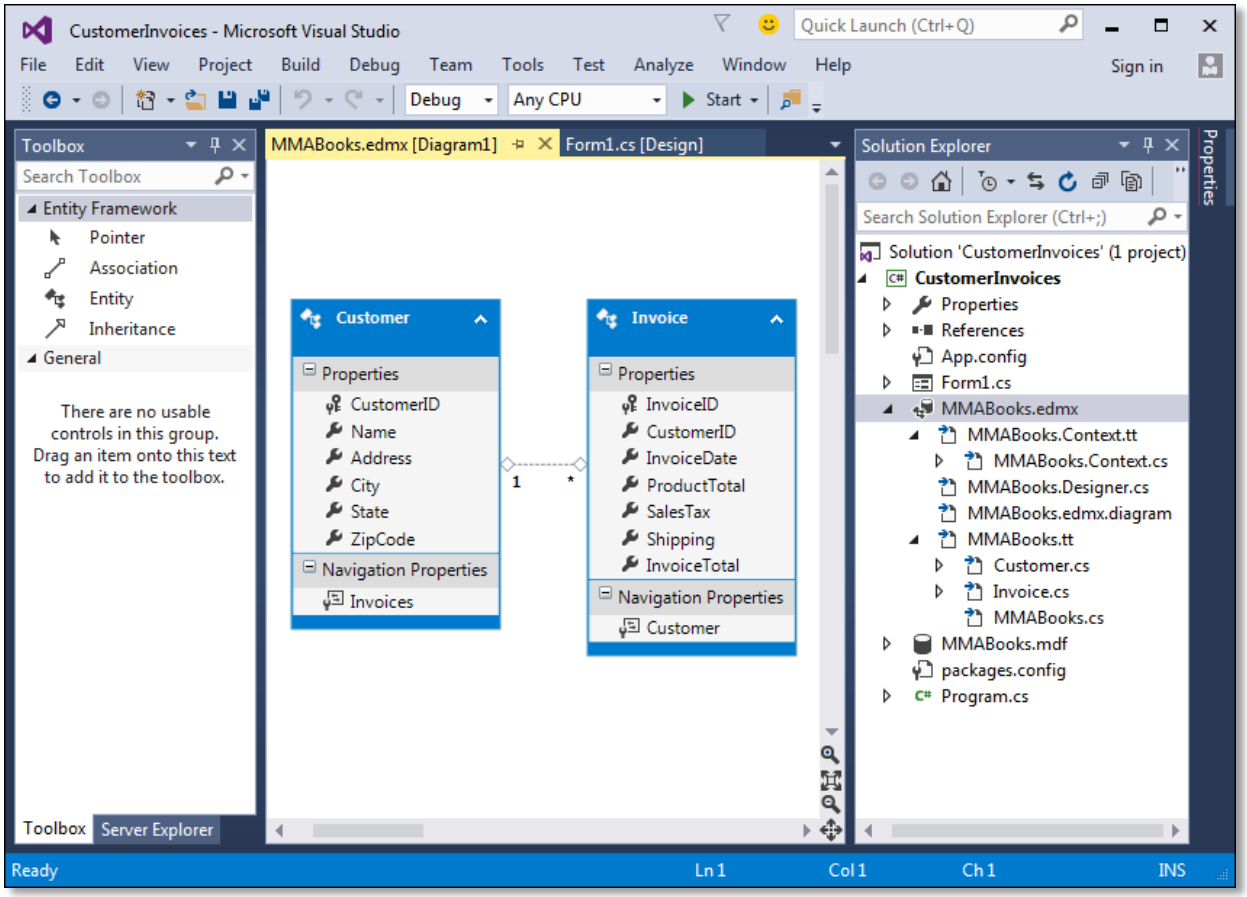
The Entity Data Model Wizard: Step 3



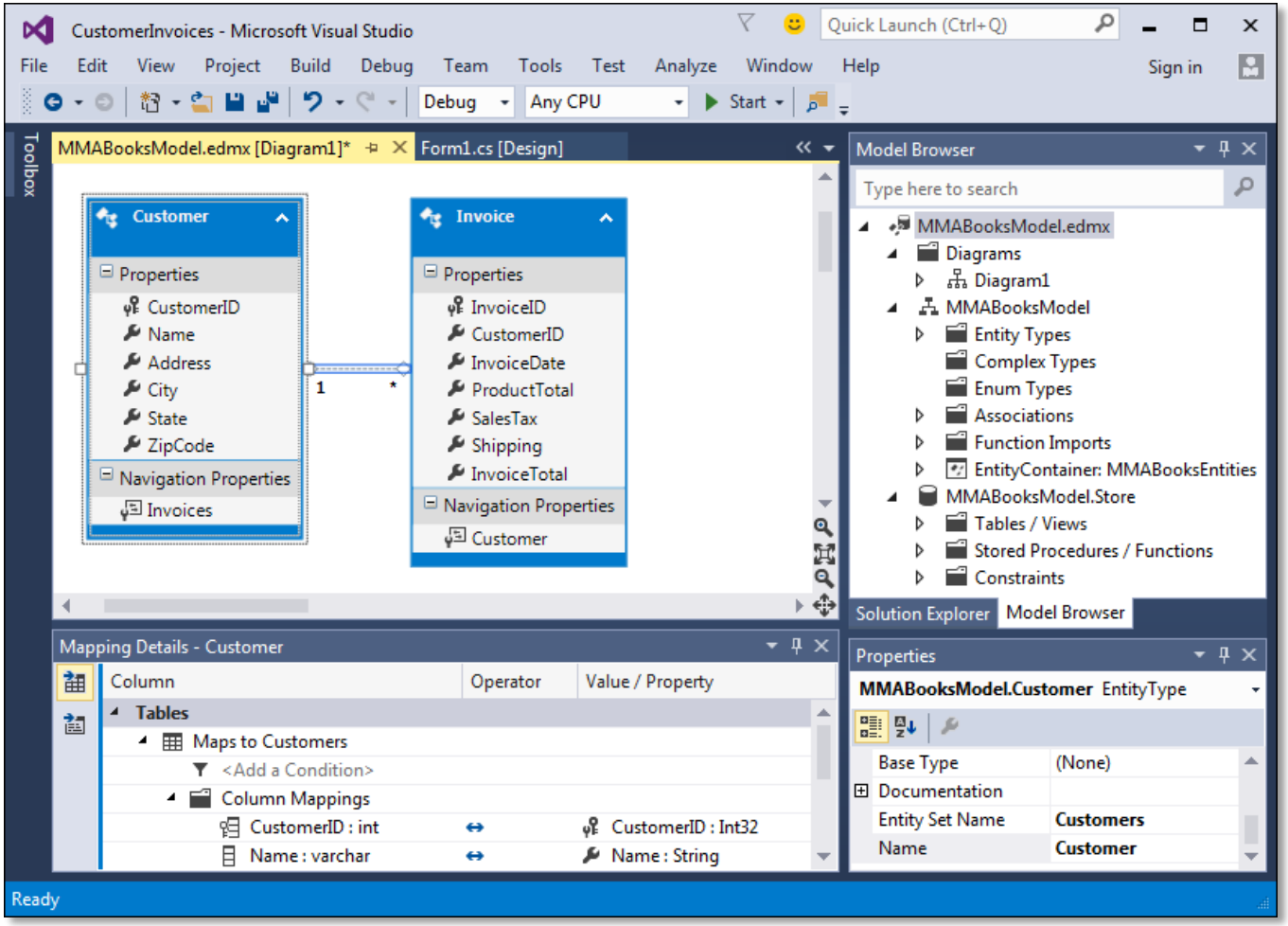
The Entity Data Model Wizard: Step 4



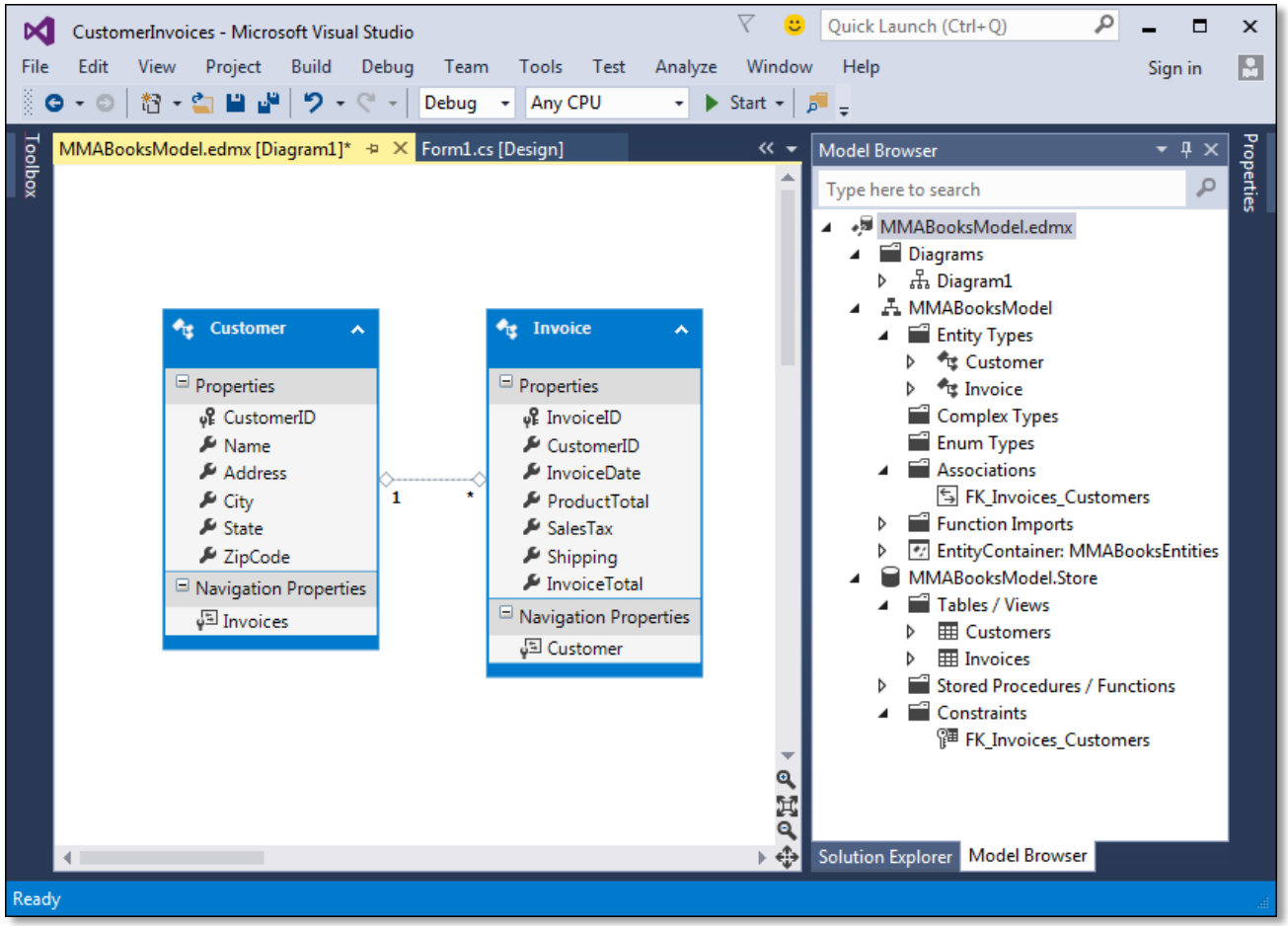
The Entity Data Model in the Entity Data Model Designer



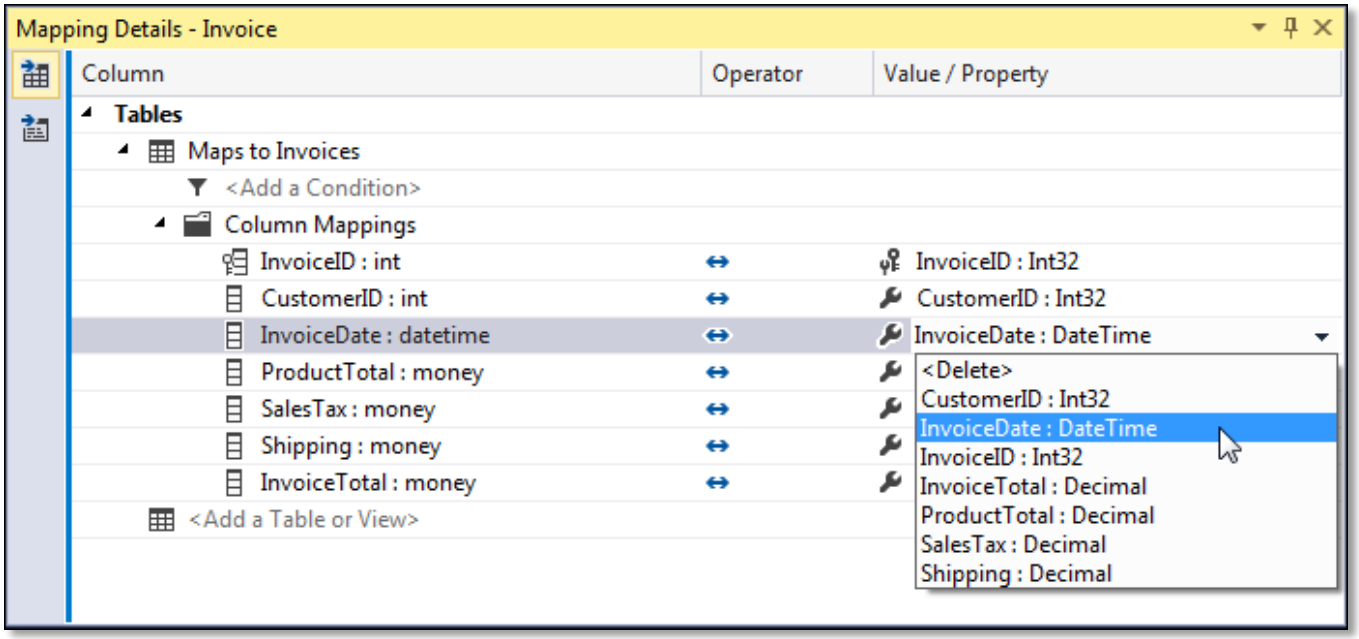
The Entity Data Model Designer



The Model Browser window with some of its nodes expanded



A Mapping Details window that displays the mappings for the Invoice entity



A LINQ query that gets data from the Invoices table

A statement that creates an instance of the object context

```
MMABooksEntities mmaBooks = new MMABooksEntities();
```

A query expression that retrieves invoices over $200

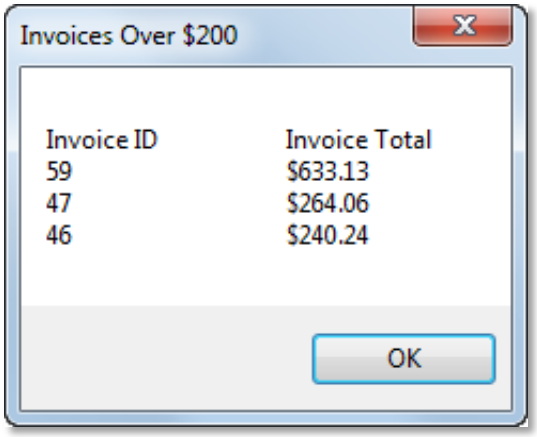
```
var highInvoices = from invoice in mmaBooks.Invoices
                   where invoice.InvoiceTotal > 200
                   orderby invoice.InvoiceTotal descending
                   select new { invoice.InvoiceID,
                                invoice.InvoiceTotal };
```

A LINQ query that gets data from the Invoices table (cont.)

Code that executes the query

```
string displayResult = "Invoice ID\t\tInvoice Total\n";
foreach (var invoice in highInvoices)
{
    displayResult += invoice.InvoiceID + "\t\t" +
        invoice.InvoiceTotal.ToString("c") + "\n";
}
MessageBox.Show(displayResult, "Invoices Over $200");
```

The resulting dialog box



A LINQ query that gets invoice data through the Customer object

```
var invoicesList = from customer in mmaBooks.Customers
                   from invoice in customer.Invoices
                   orderby customer.Name, invoice.InvoiceDate
                   select new { customer.Name, invoice.InvoiceID,
                                invoice.InvoiceTotal };
```

A query expression that uses a navigation property to load related objects

```
var customerInvoices =
    (from customer in mmaBooks.Customers
     where customer.CustomerID == customerID
     select new { customer.Name, customer.Invoices }).Single();
```

A query expression that uses the Include method to load related objects

```
var selectedCustomer =
    (from customer in mmaBooks.Customers.Include("Invoices")
     where customer.CustomerID == customerID
     select customer).Single();
```

Code that explicitly loads the objects on the many side of a relationship

```
var selectedCustomer = (from customer in mmaBooks.Customers
                        where customer.CustomerID == customerID
                        select customer).Single();

if (! mmaBooks.Entry(selectedCustomer).Collection(
        "Invoices").IsLoaded)
    mmaBooks.Entry(selectedCustomer).Collection(
        "Invoices").Load();
```

Code that explicitly loads the object on the one side of a relationship

```
var selectedInvoice = (from invoice in mmaBooks.Invoices
                       where invoice.InvoiceID == invoiceID
                       select invoice).Single();

if (! mmaBooks.Entry(selectedInvoice).Reference(
        "Customer").IsLoaded)
    mmaBooks.Entry(selectedInvoice).Reference("Customer").Load();
```

Code that retrieves a customer row

```
var selectedCustomer =
    (from customer in mmaBooks.Customers
     where customer.CustomerID == CustomerID
     select customer).Single();
```

Code that modifies the data in the customer row

```
selectedCustomer.Name = txtName.Text;
selectedCustomer.City = txtCity.Text;
```

A statement that saves the changes to the database

```
mmaBooks.SaveChanges();
```

Code that assigns an invoice to a different customer

```
int invoiceID = Convert.ToInt32(txtInvoiceID.Text);
var selectedInvoice =
    (from invoice in mmaBooks.Invoices
     where invoice.InvoiceID == invoiceID
     select invoice).Single();
selectedInvoice.Customer = selectedCustomer;
```

Code that retrieves an invoice row and its related line item rows

```
var selectedInvoice =
    (from invoice in mmaBooks.Invoices.Include(
        "InvoiceLineItems")
     where invoice.InvoiceID == Convert.ToInt32(
         txtInvoiceID.Text)
     select invoice).Single();
```

A statement that marks the Invoice object for deletion

```
mmaBooks.Invoices.Remove(selectedInvoice);
```

A statement that deletes the invoice and line items from the database

```
mmaBooks.SaveChanges();
```

Code that creates a new Invoice object

```
Invoice newInvoice = new Invoice {
    CustomerID = 14,
    InvoiceDate = new DateTime(2016, 01, 04),
    ProductTotal = 156.00m,
    SalesTax = 11.70m,
    Shipping = 6.25m,
    InvoiceTotal = 173.95m };
```

Code that adds the object to the Invoices collection and updates the database

```
mmabooks.Invoices.Add(newInvoice);
mmabooks.SaveChanges();
```

Code that creates a new InvoiceLineItem object

```
InvoiceLineItem newLineItem = new InvoiceLineItem {
    InvoiceID = newInvoice.InvoiceID,
    ProductCode = "A46V",
    UnitPrice = 57.50m,
    Quantity = 1,
    ItemTotal = 57.50m };
```

Code that adds the object to the InvoiceLineItems collection and updates the database

```
mmabooks.InvoiceLineItems.Add(newLineItem);
mmabooks.SaveChanges();
```

Another way to add the object to the InvoiceLineItems collection

```
newInvoice.InvoiceLineItems.Add(newLineItem);
```

A try-catch statement that uses store wins for concurrency exceptions

```
try
{
    mmaBooks.SaveChanges();
}
catch (DbUpdateConcurrencyException ex)
{
    ex.Entries.Single().Reload();
    ...
}
```

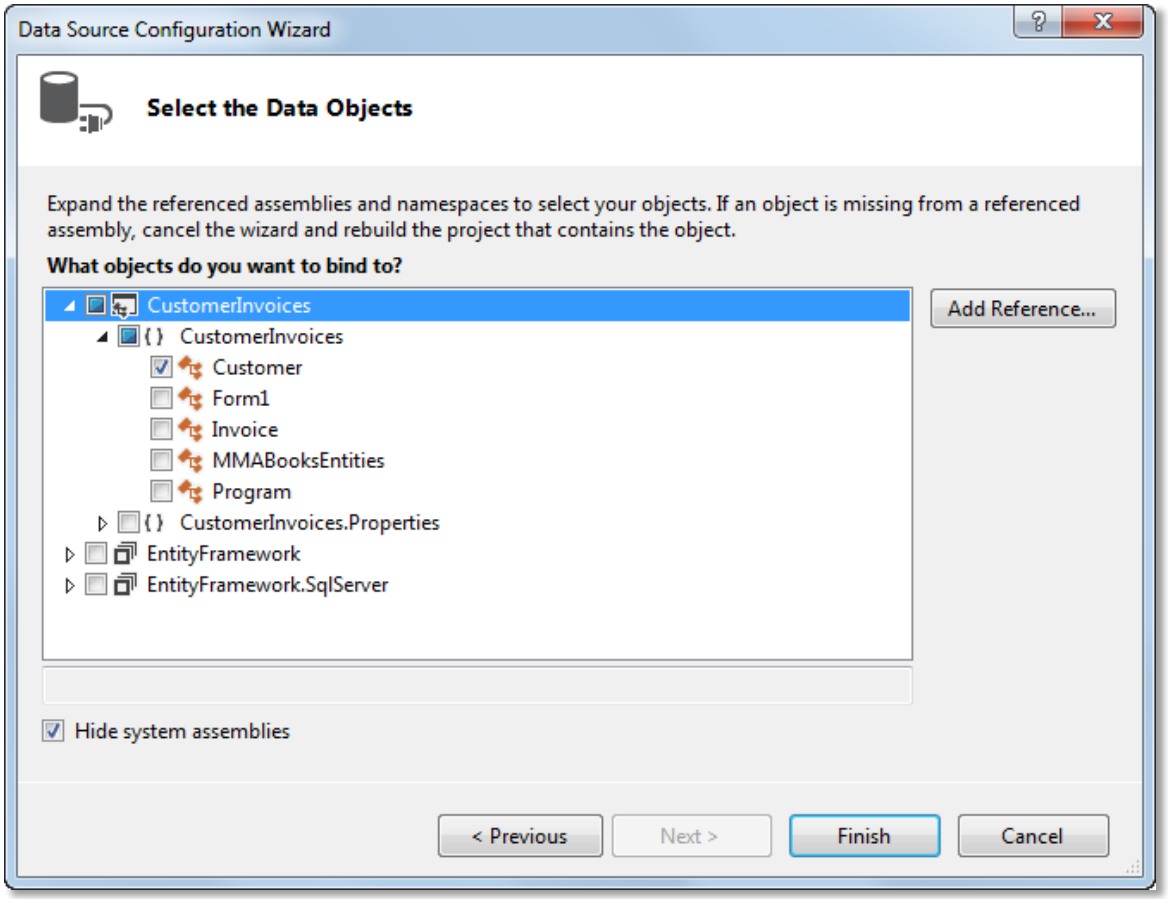
Code in a catch block that uses client wins for concurrency exceptions

```
catch (DbUpdateConcurrencyException ex)
{
    var entry = ex.Entries.Single();
    entry.OriginalValues.SetValues(
        entry.GetDatabaseValues());
    ...
}
```

Checking if a currency exception occurred due to the row being deleted

```
if (mmaBooks.Entry(customer).State ==
    EntityState.Detached) ...
```

The Data Source Configuration Wizard: Step 2



Code that binds a combo box to an entity collection

```
cboCustomers.DataSource = mmaBooks.Customers.ToList();
cboCustomers.DisplayMember = "Name";
cboCustomers.ValueMember = "CustomerID";
```

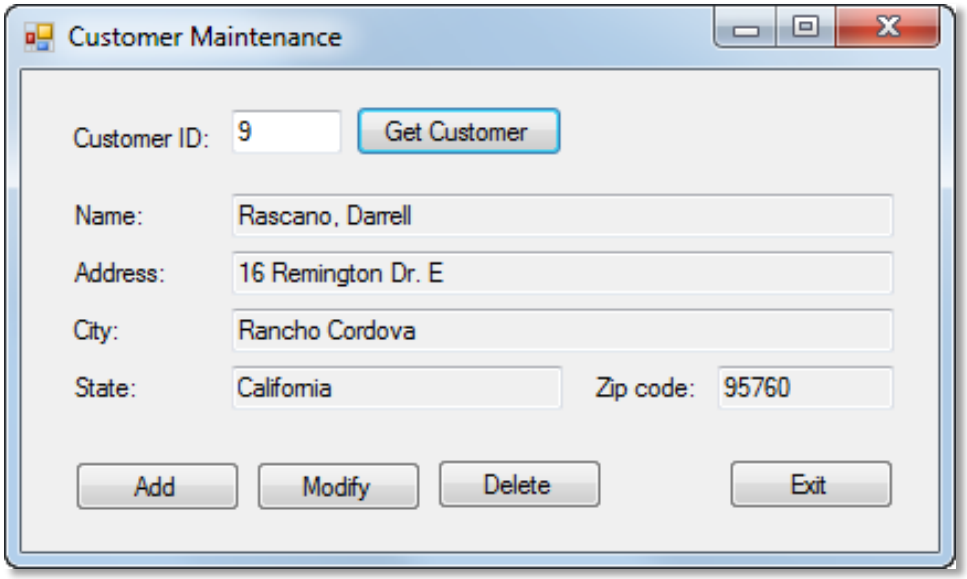
Code that binds a combo box to the results of a query

```
var customers =
    from customer in mmaBooks.Customers
    orderby customer.Name
    select new { customer.CustomerID, customer.Name };

cboCustomers.DataSource = customers.ToList();
cboCustomers.DisplayMember = "Name";
cboCuatomers.ValueMember = "CustomerID";
```

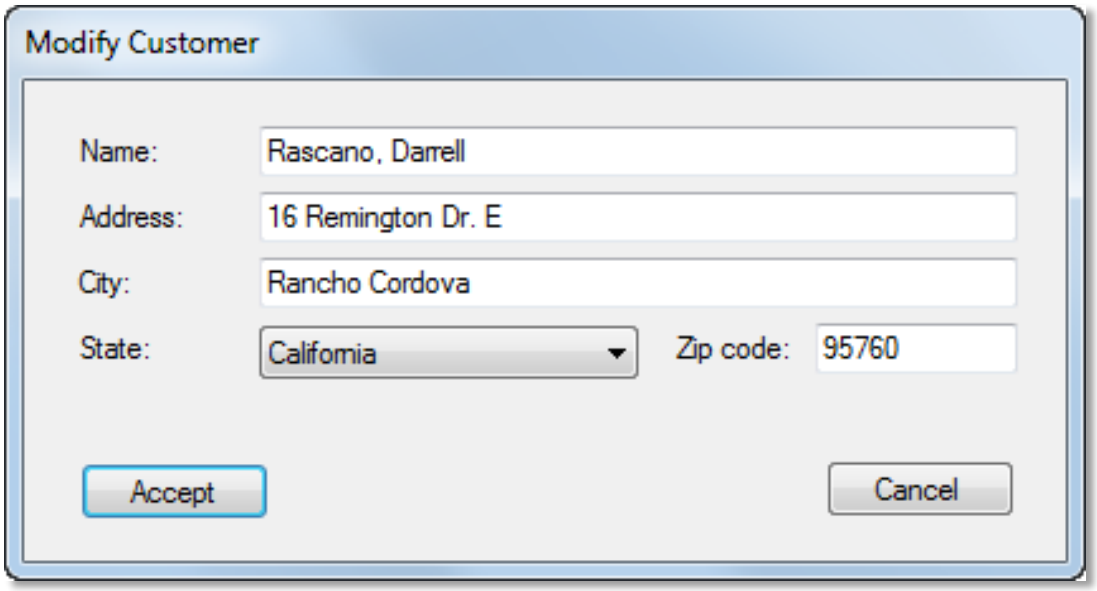
A statement that binds a combo box using its binding source

```
customerBindingSource.DataSource = customers.ToList();
```

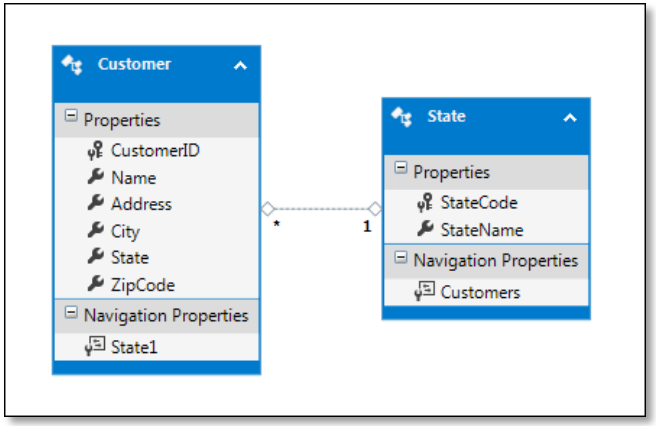
The Customer Maintenance Form



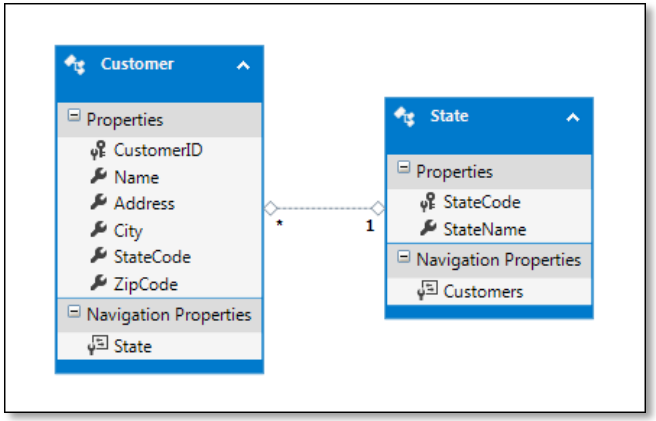
The Add/Modify Customer form



The default Entity Data Model



The modified Entity Data Model



The code for the MMABooksEntity class

```
public static class MMABooksEntity
{
    public static MMABooksEntities mmaBooks =
        new MMABooksEntities();
}
```

The Customer Maintenance form

```
public partial class frmCustomerMaintenance : Form
{
    public frmCustomerMaintenance()
    {
        InitializeComponent();
    }

    private Customer selectedCustomer;

    private void btnGetCustomer_Click(object sender, EventArgs e)
    {
        if (Validator.IsPresent(txtCustomerID) &&
            Validator.IsInt32(txtCustomerID))
        {
            int customerID = Convert.ToInt32(txtCustomerID.Text);
            this.GetCustomer(customerID);
        }
    }

    private void GetCustomer(int CustomerID)
    {
        try
        {
            selectedCustomer =
                (from customer in MMABooksEntity.mmaBooks.Customers
                 where customer.CustomerID == CustomerID
                 select customer).SingleOrDefault();
```

The Customer Maintenance form (cont.)

```
            if (selectedCustomer == null)
            {
                MessageBox.Show("No customer found with this ID. " +
                    "Please try again.", "Customer Not Found");
                this.ClearControls();
                txtCustomerID.Focus();
            }
            else
            {
                if (!MMABooksEntity.mmaBooks.Entry(
                        selectedCustomer).Reference("State").IsLoaded)
                    MMABooksEntity.mmaBooks.Entry(
                        selectedCustomer).Reference("State").Load();
                this.DisplayCustomer();
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, ex.GetType().ToString());
        }
    }
```

The Customer Maintenance form (cont.)

```
private void DisplayCustomer()
{
    txtName.Text = selectedCustomer.Name;
    txtAddress.Text = selectedCustomer.Address;
    txtCity.Text = selectedCustomer.City;
    txtState.Text = selectedCustomer.State.StateName;
    txtZipCode.Text = selectedCustomer.ZipCode;
    btnModify.Enabled = true;
    btnDelete.Enabled = true;
}

private void ClearControls()
{
    txtName.Text = "";
    txtAddress.Text = "";
    txtCity.Text = "";
    txtState.Text = "";
    txtZipCode.Text = "";
    btnModify.Enabled = false;
    btnDelete.Enabled = false;
}

private void btnAdd_Click(object sender, EventArgs e)
{
    frmAddModifyCustomer addModifyCustomerForm =
        new frmAddModifyCustomer();
    addModifyCustomerForm.addCustomer = true;
    DialogResult result = addModifyCustomerForm.ShowDialog();
```

The Customer Maintenance form (cont.)

```
        if (result == DialogResult.OK)
        {
            selectedCustomer = addModifyCustomerForm.customer;
            txtCustomerID.Text = selectedCustomer.CustomerID.ToString();
            this.DisplayCustomer();
        }
    }

    private void btnModify_Click(object sender, EventArgs e)
    {
        frmAddModifyCustomer addModifyCustomerForm =
            new frmAddModifyCustomer();
        addModifyCustomerForm.addCustomer = false;
        addModifyCustomerForm.customer = selectedCustomer;
        DialogResult result = addModifyCustomerForm.ShowDialog();
        if (result == DialogResult.OK || result == DialogResult.Retry)
        {
            selectedCustomer = addModifyCustomerForm.customer;
            this.DisplayCustomer();
        }
        else
        {
            txtCustomerID.Text = "";
            this.ClearControls();
        }
    }
```

The Customer Maintenance form (cont.)

```
private void btnDelete_Click(object sender, EventArgs e)
{
    DialogResult result =
        MessageBox.Show("Delete " + selectedCustomer.Name + "?",
        "Confirm Delete", MessageBoxButtons.YesNo,
        MessageBoxIcon.Question);
    if (result == DialogResult.Yes)
    {
        try
        {
            MMABooksEntity.mmaBooks.Customers.Remove(selectedCustomer);
            MMABooksEntity.mmaBooks.SaveChanges();
            txtCustomerID.Text = "";
            this.ClearControls();
        }
        catch (DbUpdateConcurrencyException ex)
        {
            ex.Entries.Single().Reload();
            if (MMABooksEntity.mmaBooks.Entry(selectedCustomer).State ==
                    EntityState.Detached)
            {
                MessageBox.Show("Another user has deleted " +
                    "that customer.", "Concurrency Error");
                txtCustomerID.Text = "";
                this.ClearControls();
            }
```

The Customer Maintenance form (cont.)

```
                else
                {
                    MessageBox.Show("Another user has updated " +
                        "that customer.", "Concurrency Error");
                    this.DisplayCustomer();
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, ex.GetType().ToString());
            }
        }
    }

    private void btnExit_Click(object sender, EventArgs e)
    {
        this.Close();
    }
}
```

The Add/Modify Customer form

```
public partial class frmAddModifyVendor : Form
{
    public bool addCustomer;
    public Customer customer;

    private void frmAddModifyCustomer_Load(object sender, EventArgs e)
    {
        this.LoadComboBox();
        if (addCustomer)
        {
            this.Text = "Add Customer";
            cboStates.SelectedIndex = -1;
        }
        else
        {
            this.Text = "Modify Customer";
            this.DisplayCustomerData();
        }
    }
```

The Add/Modify Customer form (cont.)

```
private void LoadComboBox()
{
    try
    {
        var states = (from state in MMABooksEntity.mmaBooks.States
                      orderby state.StateName
                      select state).ToList();

        cboStates.DataSource = states;
        cboStates.DisplayMember = "StateName";
        cboStates.ValueMember = "StateCode";
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, ex.GetType().ToString());
    }
}

private void DisplayCustomerData()
{
    txtName.Text = customer.Name;
    txtAddress.Text = customer.Address;
    txtCity.Text = customer.City;
    cboStates.SelectedValue = customer.StateCode;
    txtZipCode.Text = customer.ZipCode;
}
```

The Add/Modify Customer form (cont.)

```
private void btnAccept_Click(object sender, EventArgs e)
{
    if (IsValidData())
    {
        if (addCustomer)
        {
            customer = new Customer();
            this.PutCustomerData(customer);
            MMABooksEntity.mmaBooks.Customers.Add(customer);
            try
            {
                MMABooksEntity.mmaBooks.SaveChanges();
                this.DialogResult = DialogResult.OK;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, ex.GetType().ToString());
            }
        }
        else
        {
            this.PutCustomerData(customer);
            try
            {
                MMABooksEntity.mmaBooks.SaveChanges();
                this.DialogResult = DialogResult.OK;
            }
```

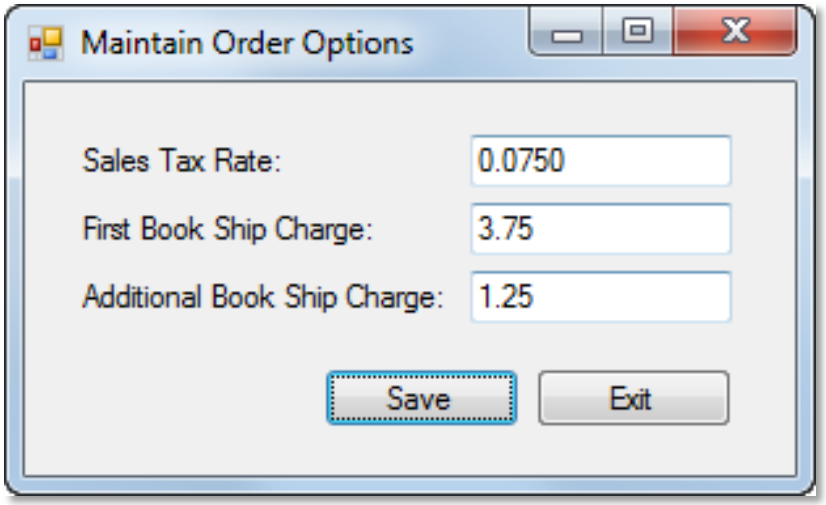
The Add/Modify Customer form (cont.)

```
                catch (DbUpdateConcurrencyException ex)
                {
                    ex.Entries.Single().Reload();
                    if (MMABooksEntity.mmaBooks.Entry(customer).State ==
                            EntityState.Detached)
                    {
                        MessageBox.Show("Another user has deleted " +
                            "that customer.", "Concurrency Error");
                        this.DialogResult = DialogResult.Abort;
                    }
                    else
                    {
                        MessageBox.Show("Another user has updated " +
                            "that customer.", "Concurrency Error");
                        this.DialogResult = DialogResult.Retry;
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message, ex.GetType().ToString());
                }
            }
        }
    }
```

The Add/Modify Customer form (cont.)

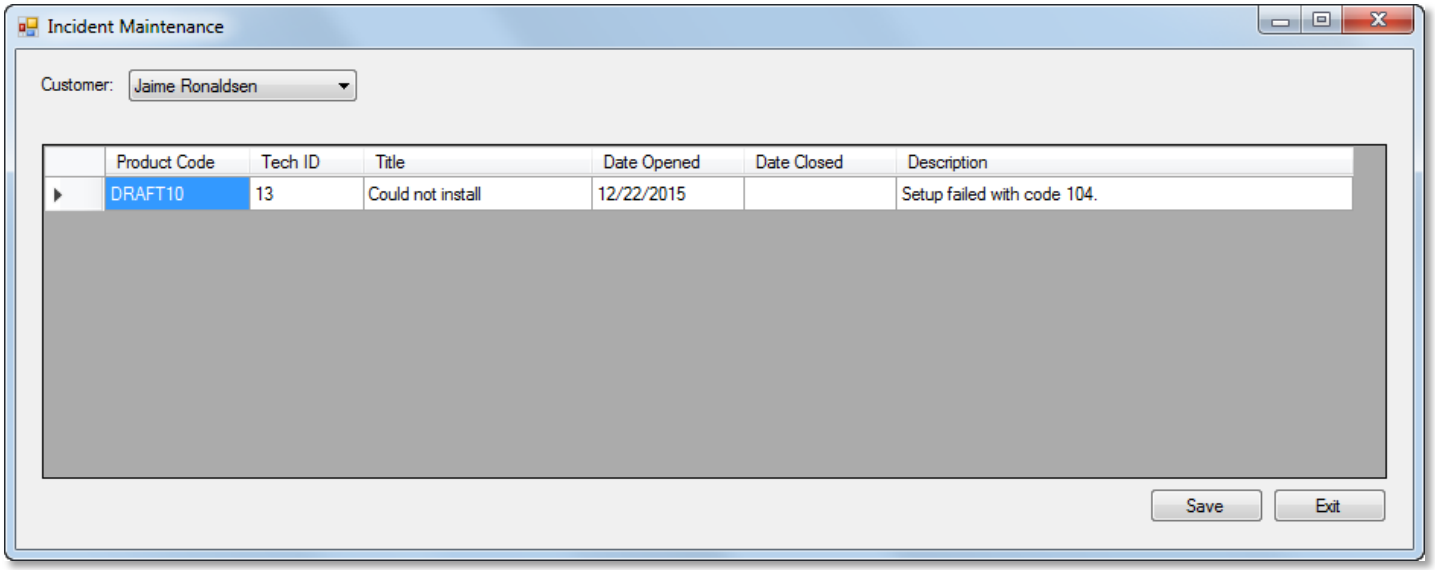
```
    private bool IsValidData()
    {
         return Validator.IsPresent(txtName) &&
                Validator.IsPresent(txtAddress) &&
                Validator.IsPresent(txtCity) &&
                Validator.IsPresent(cboStates) &&
                Validator.IsPresent(txtZipCode) &&
                Validator.IsInt32(txtZipCode);
    }

    private void PutCustomerData(Customer customer)
    {
        customer.Name = txtName.Text;
        customer.Address = txtAddress.Text;
        customer.City = txtCity.Text;
        customer.StateCode = cboStates.SelectedValue.ToString();
        customer.ZipCode = txtZipCode.Text;
    }
}
```

Extra 24-1 Use the Entity Framework to create an Order Options Maintenance application



Use the Entity Framework to create an application that lets the user update the data in a table of order options.

Project 5-3 Maintain incidents



Develop an application that lets the user assign a date closed and change the description for open incidents.