Create a new ASP.NET 4.5 Web Forms project.

2. Add a reference to System.ComponentModel.DataAnnotations assembly.

3. Create a new class called Person:

using System.ComponentModel.DataAnnotations;

public class Person

{

public int ID { get; set; }

[Required(ErrorMessage="First name is required")]

public string FirstName { get; set; }

public string LastName { get; set; }

public int Age { get; set; }

}

4. Create something to display the data, so open default.aspx and add the

following:

<asp:FormView ID="FormView1" ItemType="WebApplication7.Person" UpdateMethod="UpdatePeople"

AllowPaging="true" runat="server" SelectMethod="GetPeople" DataKeyNames="ID"

DefaultMode="Edit">

<EditItemTemplate>

<ol>

<li>

<%#Item.ID%>

</li>

<li>

<asp:TextBox ID="textFirstName" runat="server" Text='<%# BindItem.FirstName

%>' />

</li>

<li>

<asp:TextBox ID="txtLastName" runat="server" Text='<%# BindItem.LastName%>'/>

</li>

</ol>

<asp:linkbutton id="UpdateButton"

text="Update"

commandname="Update"

runat="server"/>

<asp:ValidationSummary ID="ValidationSummary1" runat="server"/>

</EditItemTemplate>

</asp:FormView>

5. Note the use of the new ItemType property. You might also have noticed we set

two new properties: SelectMethod and UpdateMethod. SelectMethod and

UpdateMethod are new methods that unsurprisingly allow us to specify the

methods to bind and update data. We will complete our binding example by

creating these select and update methods.

6. We will return an IQueryable list of our person class (you can think of

IQueryable as a query that hasn’t been run yet and can be modified on the

fly—this is needed to enable the FormView control’s paging functionality). To

do this, open default.aspx.cs and add the following code:

public IQueryable<Person> GetPeople()

{

IQueryable<Person> people = new List<Person>(){

new Person()

{

ID=0,

Age=33,

FirstName="Belinda",

LastName="Lord"

},

new Person()

{

ID=1,

Age=62,

FirstName="Rhonda",

LastName="Lord"

},

new Person()

{

ID=2,

Age=64,

FirstName="Gary",

LastName="Lord"

},

new Person()

{

ID=4,

Age=1,

FirstName="Darcy",

LastName="Lord"

},

}.AsQueryable();

return people;

}

public void UpdatePeople(Person model)

{

var success = TryUpdateModel<Person>(model);

if (success)

{

//TODOwritebackupdate

};

}

7. Run the application and you should see a highly exciting form view control

bound to the list of people we created.

BindItem

BindItem is similar to Item but should be used if you want to persist changes that the user has made. To

see the difference between Item and BindItem, follow these steps:

1. Modify the textboxes in the example above to the following:

<asp:TextBox ID="textFirstName" runat="server" Text='<%# Item.FirstName %>' /></asp:TextBox>

<asp:TextBox ID="txtLastName" runat="server" Text='<%# BindItem.LastName %>' /></asp:TextBox>

2. Put a breakpoint on UpdatePeople method and run the application.

3. Make a change in both textboxes.

4. Click the Update button and the breakpoint should be hit.

You will see only the LastName property in the submitted model has been populated with changes.