**ASP.NET 4.0 与 Entity Framework 4-第二篇-使用Entity Framework 进行CRUD操作**

本文将向你介绍如何使用Entiity Framework对SQL Sever中的数据进行增删改查(CRUD)操作。文中使用的数据库是上一篇文章《采用Model-First开发方式创建数据库》创建的。当然你还需要下载Visual Studio 2010来打开上一篇文章的项目文件。项目文件的下载链接：点击下载。

**步骤1：修改第一篇文章中的数据库**

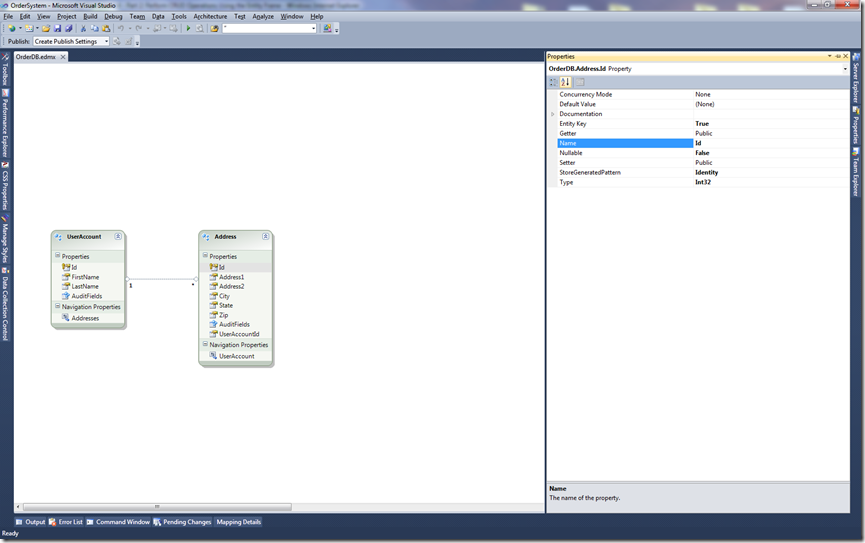
在第一篇里，我们使用EF4创建了2个实体。它们分别是UserAccount和Address，每一个实体都有一个名为Id的主键和多个字段。但是在那篇文章中，没有告诉大家如何将主键设置为Identity字段，设置为Identity字段的好处是主键可以自增。

1打开第一篇文章使用的项目文件。

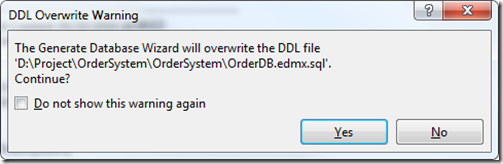
2.双击Solution Explorer的OrderDB.edmx文件，将会看到Entity Framework设计器。

3.点击UserAccount的Id字段，查看它的属性(Alt+Enter)。

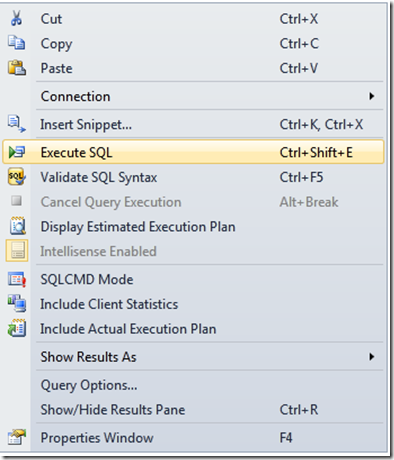
4.将StoreGeneratedPattern的设置为Identity，Address实体做同样的操作。



5.实体更新了，数据库也要同步更新。右击视图设计器选择"Generate Database From Model"，这个时候VS会提示你是否覆盖OrderDB.edmx.sql文件 ，点击"Yes"。



6.覆盖后，VS会自动打开OrderDB.edmx.sql文件，右键弹出菜单中选择 “Execute SQL”。



这时VS会提示你输入数据数连接信息，输入后点击“connect”就可以了。

**步骤2：创建Web表单**

下面将在程序中创建一个表单，用来管理UserAccount数据。

1.在项目文件上右键，选择Add->New Item..

2.选择Web Form模板，将名字改为Users.aspx，点“Add”。

3.在User.aspx的div之间添加如下代码：

<table>

<tr>

<td>Select A User:</td>

<td><asp:DropDownList runat=server ID="ddlUsers" AutoPostBack="True">

</asp:DropDownList> </td>

</tr>

<tr>

<td>First Name:</td>

<td><asp:TextBox runat="server" ID="txtFirstName"></asp:TextBox></td>

</tr>

<tr>

<td>Last Name:</td>

<td><asp:TextBox runat="server" ID="txtLastName"></asp:TextBox></td>

</tr>

<tr>

<td>Inserted:</td>

<td><asp:Label runat="server" ID="lblInserted"></asp:Label> </td>

</tr>

<tr>

<td>Updated:</td>

<td><asp:Label runat="server" ID="lblUpdated"></asp:Label> </td>

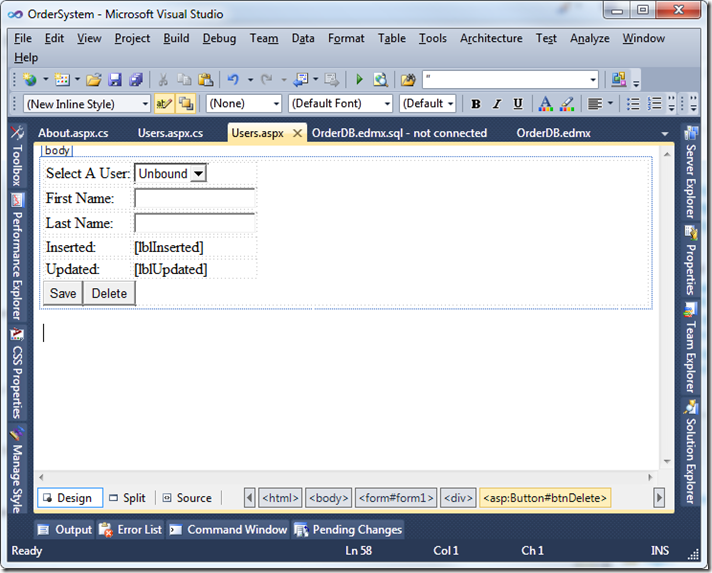
</tr>

</table>

<asp:Button runat=server ID="btnSave" Text="Save" />

<asp:Button ID="btnDelete" runat="server" Text="Delete" />

这里采用的table的对form进行了简单的布局。转到Design视图你会看到表单的样子如下：



**步骤3：将数据加载到Drop Down List中**

要完成的是在页面加载时，将UserAccount的Name和Id数据加载到Drop Down List中。当选择特定项时，加载更加详细的信息。

1.双击Degsin视图（F7），在后台代码中添加Page\_Load 事件。

2.处理Page\_Load 事件的代码如下：

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

LoadUserDropDownList();

}

}

3.Page\_Load方法中的LoadUserDropDownList方法代码如下：

private void LoadUserDropDownList()

{

using (OrderDBContainer db = new OrderDBContainer())

{

ddlUsers.DataSource = from u in db.UserAccounts

orderby u.LastName

select new { Name = u.LastName + ", " + u.FirstName, Id = u.Id };

ddlUsers.DataTextField = "Name";

ddlUsers.DataValueField = "Id";

ddlUsers.DataBind();

ddlUsers.Items.Insert(0, new ListItem("Create New User", ""));

}

}

该方法创建了一个OrderDBContainer类的实例，这个类在创建OrderDB.edmx自动生成了。它的功能和ADO.NET中的connction类似，我们使用OrderDBContainer连接数据库，操作数据库中的对象。这里的Drop Down List的数据源是一个Linq查询。当然，EF将会把它翻译为SQL 语句。Linq就像是SQL倒过来写，它From语句在前，而SELECT语句在后。本例中，我将所有的UserAccount数据查询了出来，并且按Last Name排序。同时选择了Name和Id列，Name是First Name和LastName用逗号拼接而成。

DataTextField属性设置为Name，DataValueField 设置为Id，这些都是在Linq查询中创建的。设置完成后，就是绑定了。绑定时，才真正开始调用数据库查询的操作。最后给Drop Down List添加一项“Crate New User.”，这项是用来区分更新和添加操作的。

现在数据库中还没有任何数据，Drop Down List中只有"Crete New User"一项。如果你打开了SQL Server Profiler，会看到EF4生成的SQL语句如下：

SELECT

[Project1].[Id] AS [Id],

[Project1].[C1] AS [C1]

FROM ( SELECT

[Extent1].[Id] AS [Id],

[Extent1].[LastName] AS [LastName],

[Extent1].[LastName] + N', ' + [Extent1].[FirstName] AS [C1]

FROM [dbo].[UserAccounts] AS [Extent1]

) AS [Project1]

ORDER BY [Project1].[LastName] ASC

**步骤4：添加和更新数据**

下面将向你介绍如何添加和更新表中的数据。

1.转到设计视图，双击Save按钮，创建该按钮的点击事件。

2.处理点击事件的代码如下：

using (OrderDBContainer db = new OrderDBContainer())

{

UserAccount userAccount = new UserAccount();

userAccount.FirstName = txtFirstName.Text;

userAccount.LastName = txtLastName.Text;

userAccount.AuditFields.UpdateDate = DateTime.Now;

if (ddlUsers.SelectedItem.Value == "")

{

//Adding

userAccount.AuditFields.InsertDate = DateTime.Now;

db.UserAccounts.AddObject(userAccount);

}

else

{

//Updating

userAccount.Id = Convert.ToInt32(ddlUsers.SelectedValue);

userAccount.AuditFields.InsertDate = Convert.ToDateTime(lblInserted.Text);

db.UserAccounts.Attach(userAccount);

db.ObjectStateManager.ChangeObjectState(userAccount, System.Data.EntityState.Modified);

}

db.SaveChanges();

lblInserted.Text = userAccount.AuditFields.InsertDate.ToString();

lblUpdated.Text = userAccount.AuditFields.UpdateDate.ToString();

//Reload the drop down list

LoadUserDropDownList();

//Select the one the user just saved.

ddlUsers.Items.FindByValue(userAccount.Id.ToString()).Selected = true;

}

}

代码首先创建OrderDBContainer对象，再创建UserAccount对象，用输入的值填充UserAccount对象属性。更新日期用系统当前时间，接着判断是更新操作还是添加操作了。最后就是更新Drop Down List的值并选中刚刚操作的UserAccout。这里的db.SaveChanges()最后实际上是在数据库中执行添加或更新语句。

**步骤5.查询数据**

下面是实现当用户选择某一个Drop Down List项时，显示详细信息的功能。

1.双击视图设计器中的Drop Down List，这时会创建Drop Down List的SelectedIndexChanged方法。

2.编写SelectedIndexChanged方法的代码如下：

if (ddlUsers.SelectedValue == "")

{

txtFirstName.Text = "";

txtLastName.Text = "";

lblInserted.Text = "";

lblUpdated.Text = "";

}

else

{

//Get the user from the DB

using (OrderDBContainer db = new OrderDBContainer())

{

int userAccountId = Convert.ToInt32(ddlUsers.SelectedValue);

List<UserAccount> userAccounts = (from u in db.UserAccounts

where u.Id == userAccountId

select u).ToList();

if (userAccounts.Count() > 0)

{

UserAccount userAccount = userAccounts[0];

txtFirstName.Text = userAccount.FirstName;

txtLastName.Text = userAccount.LastName;

lblInserted.Text = userAccount.AuditFields.InsertDate.ToString();

lblUpdated.Text = userAccount.AuditFields.UpdateDate.ToString();

}

else

{

//Error: didn't find user.

txtFirstName.Text = "";

txtLastName.Text = "";

lblInserted.Text = "";

lblUpdated.Text = "";

}

}

}

代码根据Drop Down List选择的Id，用Linq从数据库中查询一条数据并且显示出来。

**步骤6：删除数据**

最后就是删除数据的功能了。 1.转到视图设计器，双击“Delete”按钮。 2.添加如下代码：

if (ddlUsers.SelectedItem.Value != "")

{

using (OrderDBContainer db = new OrderDBContainer())

{

UserAccount userAccount = new UserAccount();

userAccount.Id = Convert.ToInt32(ddlUsers.SelectedValue);

db.UserAccounts.Attach(userAccount);

db.ObjectStateManager.ChangeObjectState(userAccount, System.Data.EntityState.Deleted);

db.SaveChanges();

LoadUserDropDownList();

txtFirstName.Text = "";

txtLastName.Text = "";

lblInserted.Text = "";

lblUpdated.Text = "";

}

}

代码首先创建了一个UserAccount对象，将它的Id设置为选中项的Id.然后将UserAccount附加到UserAccount集合中，设置它的状态为删除。调用SaveChanges操作，将该条数据删除，刷新Drop Down List的数据源，搞定！

**小结**

可以发现，使用EF4 轻松的就搞定了一般的数据增删改查操作。

下一篇文章，我将向你介绍如何采用存储过程进行增删改查操作，而不使用EF生成的SQL语句，希望你能喜欢。

本文相关项目文件：点击下载

本文独立博客链接：http://lloydsheng.com/2010/04/aspnet40-entityframework4-how-to-crud-using