MỤC LỤC

Bài 7

DYNAMIC DATA VỚI LINQ TO SQL

Bài này giới thiệu tổng quan về Dynamic Data, những tính năng của Dynamic Data. Cách xây dựng ứng dụng Dynamic Data với LINQ to SQL.

1. Dynamic Data là gì?

- ASP.NET Dynamic Data cung cấp một Framework cho phép chúng ta nhanh chóng xây dựng một chức năng ứng dụng driver-data, dựa trên LINQ to SQL hay Entity Framework .
- Dựa trên cấu trúc của CSDL mà Dynamic Data Framework (DDF) sẽ tạo nên các trang web cho phép người dùng xem/chèn/xóa/sửa dữ liệu.
- Nhiều tính linh hoạt cho các DetailsView, FormView, GridView, ListView trong kiểm tra tính hợp lệ của dữ liệu, hoặc chỉnh sửa lại các mẫu để thay đổi cách hiển thị dữ liệu.
- ASP.NET Dynamic Data mang đến cho chúng ta các tiện ích và RAD (Rapid Application Development) để thay đổi dữ liệu các các control ASP.NET.

2. Các tính năng của Dynamic Data

- Web Scaffolding để tạo ra một ứng dụng web dựa trên các lược đồ cơ bản của cơ sở dữ liệu. Dynamic Data scaffolding có thể tạo ra một chuẩn UI (User Interface – Giao diện người dùng) từ các mô hình dữ liệu.
- Đầy đủ các thao tác (tạo, cập nhật, xóa bỏ, hiển thị) cho việc truy cập dữ liệu truy cập dữ liệu, các thao tác về quan hệ giữa các bảng và kiểm tra tính hợp lệ của dữ liệu.
- Tự động hỗ quan các quan hệ khóa ngoài (foreign-key). Dynamic Data phát hiện ra các quan hệ giữa các bảng và từ đó tạo ra các giao diện người dùng trên các bảng quan hệ.
- Khả năng tùy chỉnh các UI.
- Khả năng tùy chỉnh tính hợp lệ cho các trường dữ liệu.

3. Xây dựng ứng dụng Dynamic Data với LINQ to SQL

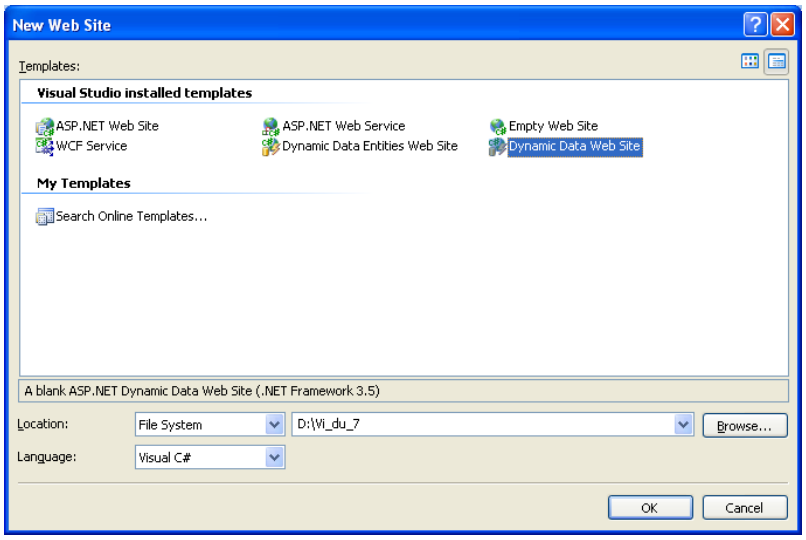
Để xây dựng một trang Web với Dynamic Data sẽ bắt đầu bằng cách sử dụng scaffolding. Dynamic Data hỗ trợ mô hình dữ liệu LINQ to SQL và mô hình dữ liệu ADO.NET Entity Framework. Trong một ứng dụng web của chúng ta có thể có nhiều loại mô hình dữ liệu nhưng để sử dụng Dynamic Data thì các mô hình dữ liệu phải cùng kiểu.

Chúng ta phải đăng ký mô hình dữ liệu sử dụng Dynamic Data với file "**Global.asax"**. Sau khi mô hình dữ liệu được đăng ký với Dynamic Data, dữ liệu mô hình có thể tự động thực hiện xác nhận các trường dữ liệu, và nó cho phép chúng ta kiểm soát sự xuất hiện và hành vi của dữ liệu ở cấp độ tầng dữ liệu.

Trong ví dụ sau đây sẽ xây dựng một Project sử dụng Dynamic Data với mô hình dữ liệu LINQ to SQL.

2.1 Tạo Project Dynamic Data

Để bắt đầu, **File⟶New Web Site** và chọn ***"Dynamic Data Entities Web Site"*** hay ***"Dynamic Data Web Site"***. Ở đây chúng ta dùng Dynamic Data với LINQ to SQL nên chọn **"Dynamic Data Web Site"** (Hình 1).

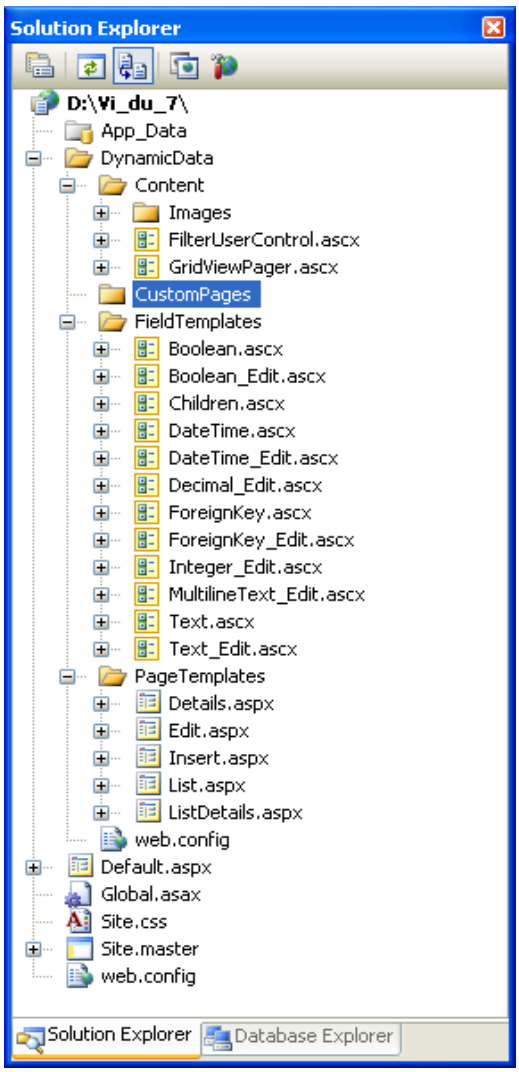


Hình 1: Tạo Project Dynamic Data với LINQ to SQL

Trong Project này chúng ta đặt tên là **Vi_du_7,** ngôn ngữ là Visual C# sau đó bâm "**OK".**

Khi đã tạo xong, bạn sẽ thấy một số **Folder/File** được đưa vào trong **Solution Explorer** (Hình 2).

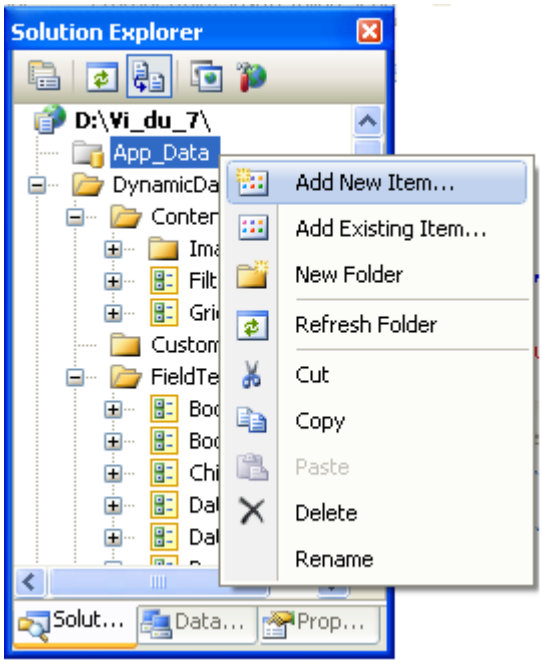
Trong đó sẽ có một Folder có tên là DynamicData, bên trong chứa một số các Folder khác, và trong mỗi Folder con này sẽ chứa các UserControl và các trang ASP.NET.



Hình 2: Các Folder và File trong Dynamic

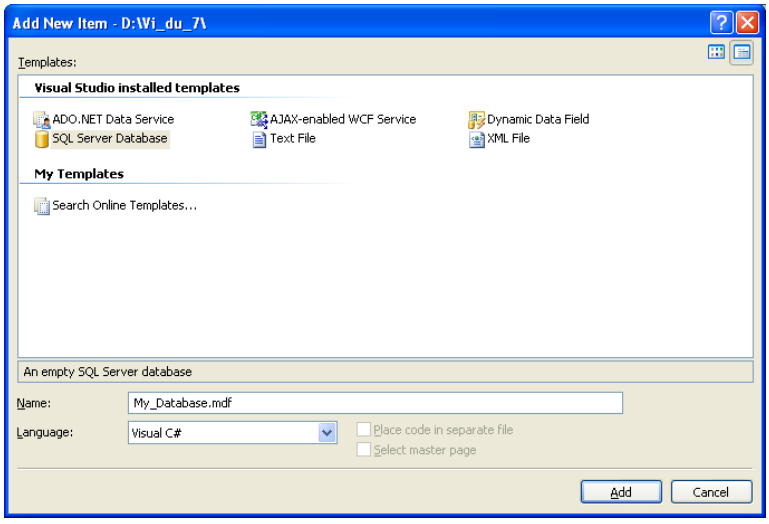
2.2 Xây dựng Cơ sở dữ liệu cho Project

Từ Folder **"App_Data"** trong **"Solution Explore"** click phải chuột chọn **"Add New Item"** (Hình 3).

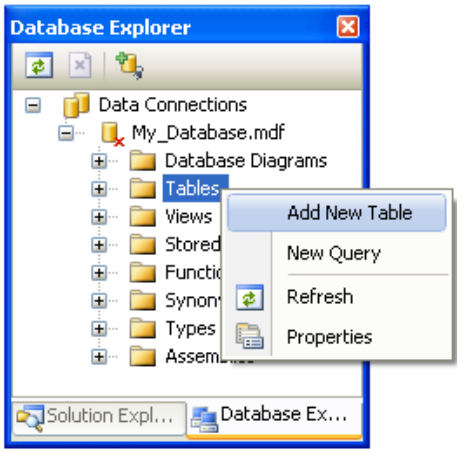


Hình 3: Tạo Cơ sở dữ liệu

Chọn **"SQL Server Database"**, trong ví dụ này đặt tên cho CSDL là **"My_Database.mdf"**, chọn ngôn ngữ là **"Visual C#"**, sau đó bấm "**Add**" (Hình 4).



Hình 4: SQL Server Database



Hình 5: Tạo bảng cho My_Database.mdf

Để tạo Table cho **"My_Database"**, trong **"Database Explore"** chúng click chuột phải vào Folder **"Table"** chọn "**Add New Table**".

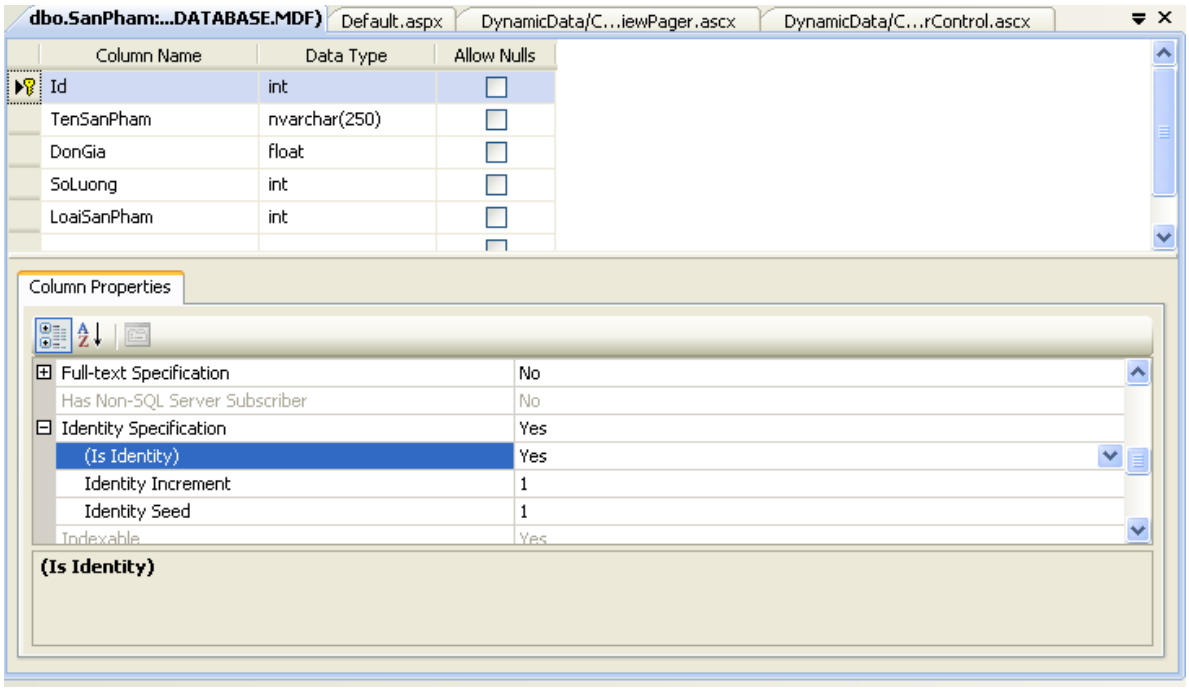
Trong ví dụ này chúng ta sẽ tạo 2 bảng:

SanPham gồm các trường:

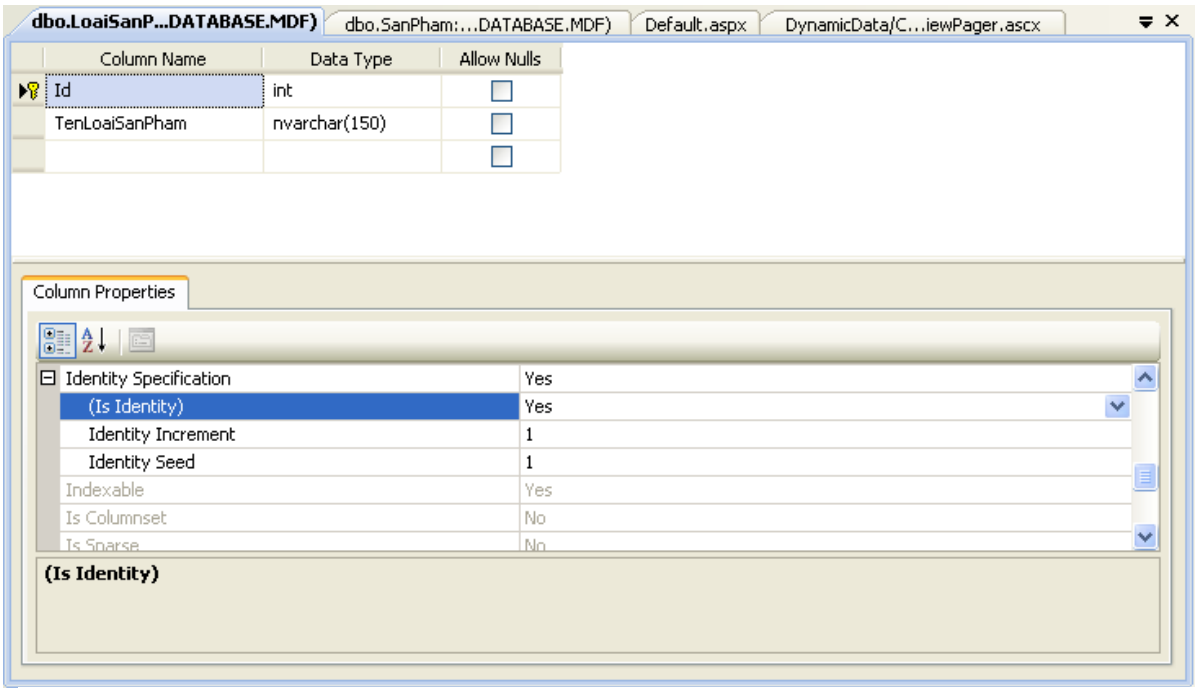
Id, TenSanPham, DonGia, SoLuong, LoaiSanPham (Hình 6).

LoaiSanPham gồm các trường:

Id, TenLoaiSanPham (Hình 7).



Hình 6: Bảng SanPham trong CSDL My_Database.mdf

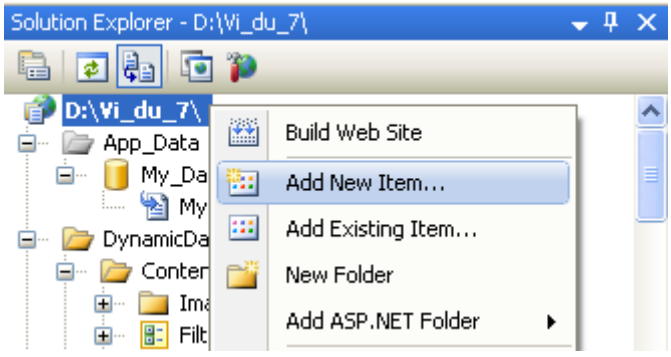


Hình 7: Bảng LoaiSanPham trong CSDL My_Database.mdf

2.3 Tạo DataContext với LINQ to SQL

Trong ví dụ này chúng ta dùng LINQ to SQL để truy cập vào CSDL My_Database.mdf.

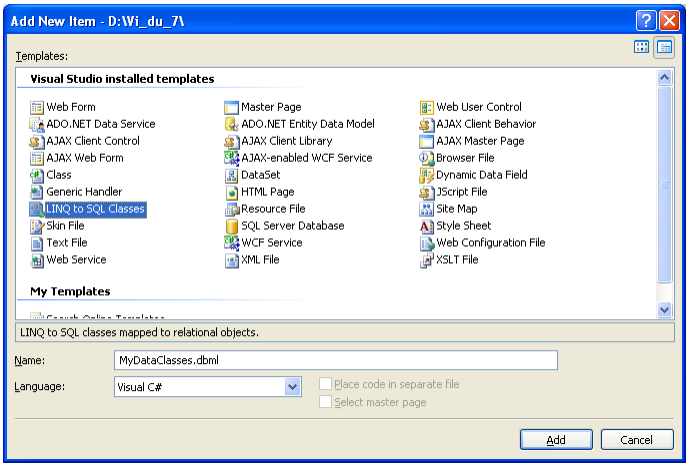
Để tạo các lớp cho cho LINQ to SQL, trong **"Solution Explore"** click chuột phải chọn **"Add New Item"** (Hình 8).



Hình 8

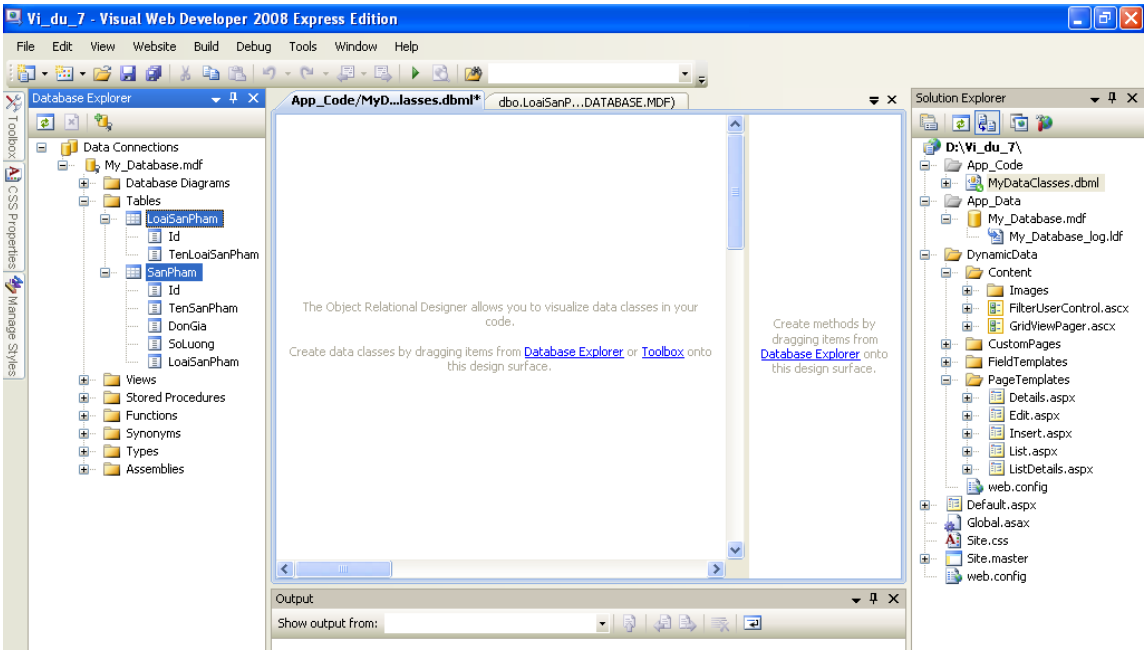
Trong "**Add New Item**" (Hình 9) chọn **"LINQ to SQL Classes",** đặt tên tệp là "**MyDataClassess.dbml",** chọn ngôn ngữ là **"Visual C#",** sau đó bấm vào **"Add"** .

Sau khi bấm **"Add"** sẽ xuất hiện một thông báo khuyên chúng ta nên đặt các file kiểu (LINQ to SQL Class) vào trong thư mục **"App_Code"** của Project. Chúng ta sẽ bấm "Yes" để các file **"MyDataClasses.dbml"** sẽ được chứa trong thư mục **"App_Code"**.



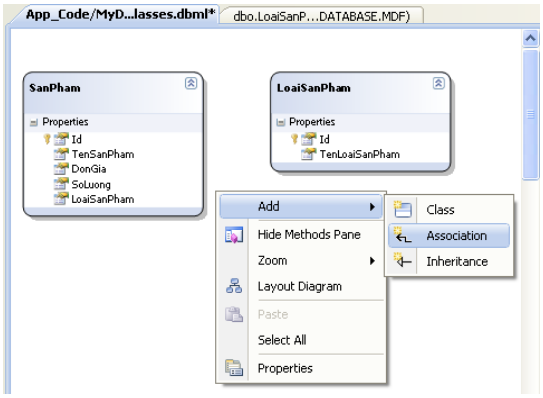
Hình 9: Tạo LINQ to SQL Classes

Mở file **"MyDataClasses.dbml"** trong thư mục **"App_Code",** đánh dấu vào 2 bảng **"LoaiSanPham"** và **"SanPham"**, kéo thả 2 bảng này vào file **"MyDataClasses.dbml"** (Hình 10).

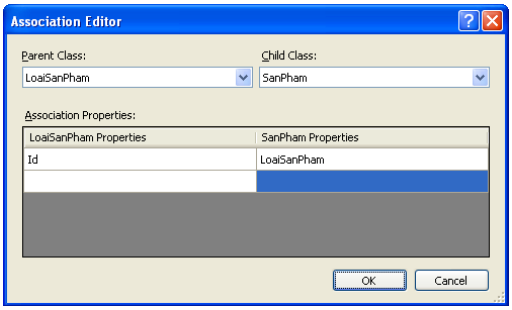


Hình 10: Tạo file MyDataClasses với 2 bảng SanPham và LoaiSanPham

Sau kéo thả 2 bảng này vào trong **"MyDataClasses.dbml"** chúng ta sẽ tạo liên kết cho 2 bảng. Trong file **"MyDataClasses.dbml"** click phải chuột chọn **Add→Association** để tạo liên kết giữa 2 bảng (Hình 11).

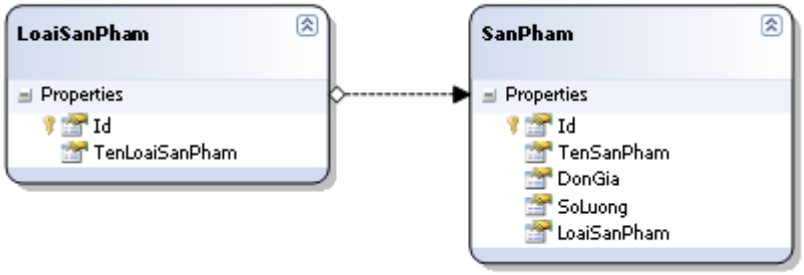


Hình 11:



Hình 12: Tạo liên kết cho bảng LoaiSanPham và SanPham

Chúng ta sẽ tạo liên kết giữa trường ***Id*** của bảng **LoaiSanPham** với trường ***LoaiSanPham*** của bảng **SanPham** (Hình 12). Sau đó chúng ta bấm **"OK"** tạo được kết quả như Hình 13.



Hình 13

2.4 Đăng ký DataContext

Sau khi tạo ra DataContext, chúng ta mở tệp **"MyDataClasses.designer.cs"**, chương trình đã tạo cho chúng ta một lớp có tên là **MyDataClassesDataContext.**

```
public partial class MyDataClassesDataContext : System.Data.Linq.DataContext
```

Chúng ta phải đăng ký MyDataClassesDataContext với hệ thống DynamicData. Mở file Global.asax,

Sửa:

```
//model.RegisterContext(typeof(YourDataContextType), new ContextConfiguration() { ScaffoldAllTables = false });
```

Thành:

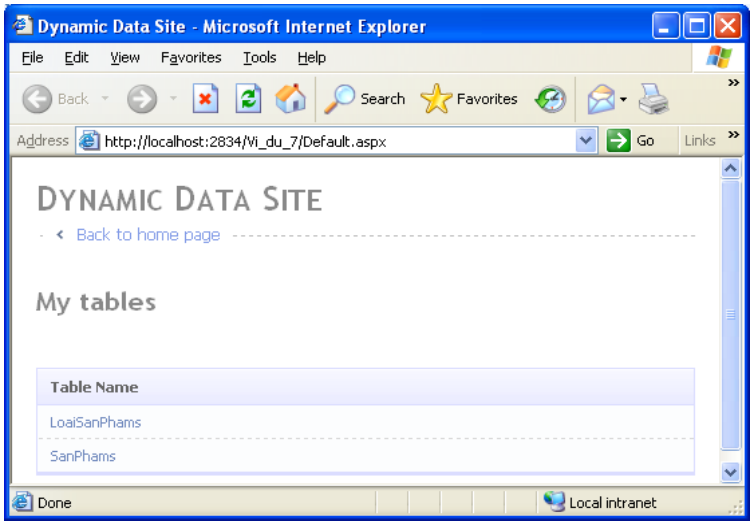
```
model.RegisterContext(typeof (MyDataClassesDataContext), new ContextConfiguration() { ScaffoldAllTables = true });
```

Khi đó chúng ta được file Golobal.asax như sau:

```
<%@ Application Language="C#" %>
<%@ Import Namespace="System.Web.Routing" %>
<%@ Import Namespace="System.Web.DynamicData" %>

<script RunAt="server">
   public static void RegisterRoutes(RouteCollection routes) {
      MetaModel model = new MetaModel();

      //                IMPORTANT: DATA MODEL REGISTRATION
      // Uncomment this line to register LINQ to SQL classes or an ADO.NET Entity Data
      // model for ASP.NET Dynamic Data. Set ScaffoldAllTables = true only if you are sure
      // that you want all tables in the data model to support a scaffold (i.e. templates)
      // view. To control scaffolding for individual tables, create a partial class for
      // the table and apply the [Scaffold(true)] attribute to the partial class.
      // Note: Make sure that you change "YourDataContextType" to the name of the data context
      // class in your application.
      model.RegisterContext(typeof(MyDataClassesDataContext), new ContextConfiguration() { ScaffoldAllTables = true
});

      // The following statement supports separate-page mode, where the List, Detail, Insert, and
      // Update tasks are performed by using separate pages. To enable this mode, uncomment the following
      // route definition, and comment out the route definitions in the combined-page mode section that follows.
      routes.Add(new DynamicDataRoute("{table}/{action}.aspx") {
         Constraints = new RouteValueDictionary(new { action = "List|Details|Edit|Insert" }),
         Model = model
      });

      // The following statements support combined-page mode, where the List, Detail, Insert, and
      // Update tasks are performed by using the same page. To enable this mode, uncomment the
      // following routes and comment out the route definition in the separate-page mode section above.
      //routes.Add(new DynamicDataRoute("{table}/ListDetails.aspx") {
```

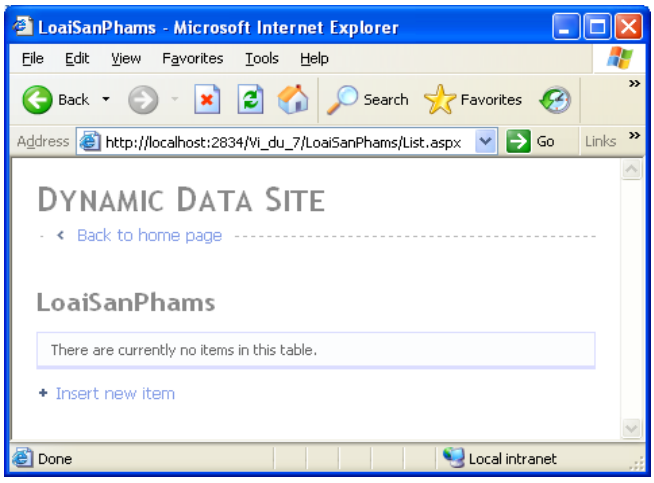
```
    //    Action = PageAction.List,
    //    ViewName = "ListDetails",
    //    Model = model
    //});

    //routes.Add(new DynamicDataRoute("{table}/ListDetails.aspx") {
    //    Action = PageAction.Details,
    //    ViewName = "ListDetails",
    //    Model = model
    //});
  }

  void Application_Start(object sender, EventArgs e) {
    RegisterRoutes(RouteTable.Routes);
  }

</script>
```

Chúng ta vào **Deburg→StartDebugging** (hoặc **F5**) để chạy thử chương trình. Kết quả khi chạy chương trình (Hình 14).

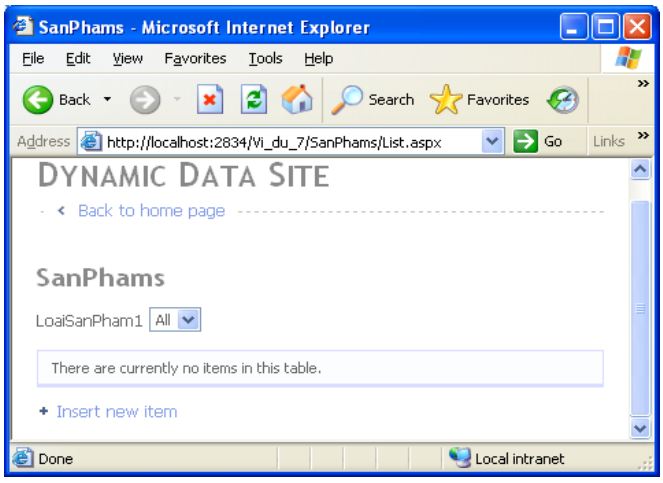


Hình 14

Khi click vào **LoaiSanPhams** kết quả như hình 15**.** Khi click vào **SanPhams** kết quả như hình 16.



Hình 15



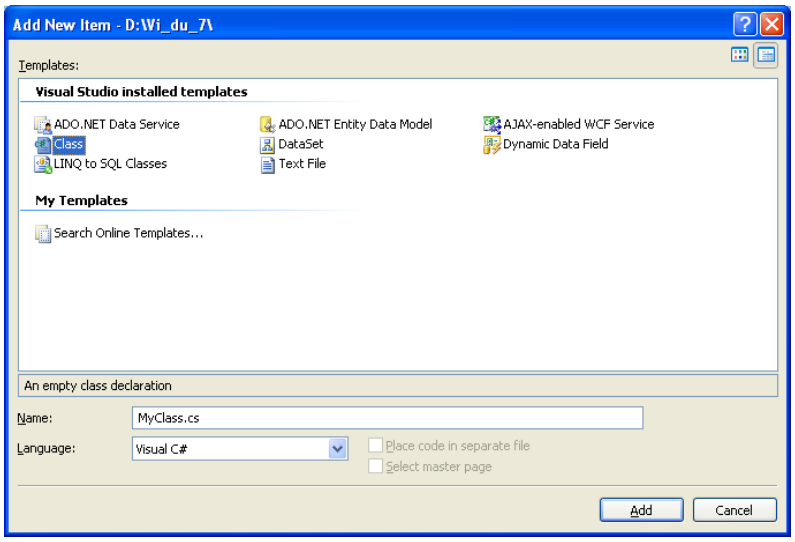
Hình 16

2.5 Thêm Custom Metadata vào Model

Để thêm tùy biến các mục, chúng ta cần tạo lớp **Metadata** để nó cung cấp cho hệ thống Dynamic Data thông tin về các thực thể. Đầu tiên bạn cần tạo thêm một lớp partial với cùng tên của lớp entity trong mô hình dữ liệu, sau đó muốn thay đổi thuộc tính lên lớp này phải chỉ ra lớp **Metadata** cho lớp này.

Trong ví dụ này chúng ta sẽ 2 lớp partial có tên là: LoaiSanPham và SanPham. Để thêm 2 lớp này chúng ta làm như sau:

Click chuột phải vào **"Add_Code"** chọn **"Add New Item",** trong hộp thoại **"Add New Item",** chọn **"Class"**, đặt tên cho file là **"MyClass.cs"**, chọn ngôn ngữ là **"Visual C#"**, sau đó bấm "**Add**" (Hình 17).



Hình 17: Thêm file MyClass.cs

Mở file "MyClass.cs" khai báo thêm các **namespace** sau:

```
using System.Web.DynamicData;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel;
```

Thêm 2 lớp partial có tên là: LoaiSanPham, SanPham và tên lớp của MetadataType.

```
[MetadataType(typeof(LoaiSanPham_Metadata))]
public partial class LoaiSanPham
{
        public class LoaiSanPham_Metadata
    {
        }
}
[MetadataType(typeof(SanPham_Metadata))]
public partial class SanPham
{
        public class SanPham_Metadata
        {
        }
}
```

Các bảng muốn thay đổi tên hiển thị sử dụng thuộc tính TableName. Khi sử dụng thuộc tính này phải được viết trong thân của lớp partial.

Các trường muốn thay đổi tên hiện thị sử dụng thuộc tính DisplayName

Các trường muốn có định dạng tùy biến sử dụng thuộc tính DisplayFormat

Các trường muốn ẩn khỏi giao diện sửa dụng thuộc tính ScaffoldColumn

Các trường muốn kiểu tra tính hợp lệ khi nhập sử dụng thuộc tính Required

Khi sử dụng các thuộc tính DisplayName,DisplayFormat,ScaffoldColumn,Required được viết trong thân của lớp MetadataType.

Trong ví dụ này sử dụng các thuộc tính TableName, DisplayName để thay đổi giao diện hiện thị cho 2 bảng **LoaiSanPham** và **SanPham**. Sử dụng Required để kiểu tra việc nhập dữ liệu cho trường ***"SoLuong"*** và ***"DonGia"*** của bảng **SanPham**.

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Web;
using System.Web.DynamicData;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel;
/// <summary>
/// Thay đổi cho các thuộc tính cho 2 bảng SanPham và LoaiSanPham
/// </summary>
[MetadataType(typeof(LoaiSanPham_Metadata))]
public partial class LoaiSanPham
{
    partial void OnTenLoaiSanPhamChanging(string value)
    {
        if (Char.IsLower(value[0]))//Kiểm tự đầu tiên phải có phải là chữ hoa
        {
            throw new ValidationException("Tên loại sản phẩm ký tự đầu tiên phải là chữ hoa!");
        }
    }

    [TableName("Loại sản phẩm")]//Sửa lại tên bảng "LoaiSanPham" thành "Loại sản phẩm"
    public class LoaiSanPham_Metadata
    {
        [DisplayName("Tên loại sản phẩm")]//Sửa lại tên hiển thị cho trường TenLoaiSanPham
        public object TenLoaiSanPham { get; set; }

        [DisplayName("Sản phẩm")]
        public object SanPhams { get; set; }
    }
}

[MetadataType(typeof(SanPham_Metadata))]
public partial class SanPham
{
    partial void OnTenSanPhamChanging(string value)
    {
        if (Char.IsLower(value[0]))//Kiểm tự đầu tiên phải có phải là chữ hoa
        {
            throw new ValidationException("Tên sản phẩm ký tự đầu tiên phải là chữ hoa!");
        }
    }

    [TableName("Sản phẩm")]//Sửa lại tên bảng "SanPham" thành "Sản phẩm"
    public class SanPham_Metadata
    {
        [DisplayName("Tên sản phẩm")]//Sửa lại tên hiển thị cho trường TenLoaiSanPham
        public object TenSanPham { get; set; }

        [DisplayName("Số lượng")]
        [Required]
        [Range(0, 300)]//Nhập số lượng trong khoảng từ 0 đến 300
        public object SoLuong { get; set; }

        [DisplayName("Đơn giá")]
        [Required]
        [Range(100000, 3000000000)]//Nhập đơn giá trong khoảng từ 100.000 đến 3.000.000.000
        public object DonGia { get; set; }

        [DisplayName("Loại sản phẩm")]
        public object LoaiSanPham1 { get; set; }
    }
}
```

Để sửa lại đổi giao diện của trang Master chúng ta mở file **"Site.master"**. Ví dụ chúng ta một số nội dung do chương trình tạo ra bằng tiếng Anh và thay bằng tiếng Việt.

Sửa:

```
<h1><span class="allcaps">Dynamic Data Site</span></h1>
```

Thành:

```
<h1 style="font-family:Times New Roman"><span class="allcaps">Dynamic Data VỚI LINQ to SQL</span></h1>
```

Sửa:

```
    <a runat="server" href="~/"><img alt="Back to home page" runat="server"
src="DynamicData/Content/Images/back.gif" />Back to home page</a>
  </div>
```

Thành:

```
    <a runat="server" href="~/"><img alt="Trở về trang chủ" runat="server" src="DynamicData/Content/Images/back.gif"
/>Trở về trang chủ</a>
```

Trong file **"Details.aspx"** sửa một số giao diện tiếng Anh và thay bằng tiếng Việt.

```
<h2 style="font-family:Times New Roman">Chi tiết một bản ghi của bảng <%= table.DisplayName %></h2>
```

```
        <asp:DetailsView ID="DetailsView1" runat="server" DataSourceID="DetailsDataSource"
OnItemDeleted="DetailsView1_ItemDeleted"
            CssClass="detailstable" FieldHeaderStyle-CssClass="bold" >
            <Fields>
                <asp:TemplateField>
                    <ItemTemplate>
                        <asp:HyperLink ID="EditHyperLink" runat="server"
                            NavigateUrl='<%# table.GetActionPath(PageAction.Edit, GetDataItem()) %>'
                            Text="Sửa" />
                        <asp:LinkButton ID="DeleteLinkButton" runat="server" CommandName="Delete"
CausesValidation="false"
                            OnClientClick='return confirm("Bạn có chắc chắn xóa bản ghi này không?");'
                            Text="Xóa" />
                    </ItemTemplate>
                </asp:TemplateField>
            </Fields>
        </asp:DetailsView>
```

Trong file **"Edit.aspx" sửa lại** <asp:DetailsView ID="DetailsView1" như sau:

```
        <asp:DetailsView ID="DetailsView1" runat="server" DataSourceID="DetailsDataSource" DefaultMode="Edit"
            AutoGenerateEditButton="False" OnItemCommand="DetailsView1_ItemCommand"
OnItemUpdated="DetailsView1_ItemUpdated"
            CssClass="detailstable" FieldHeaderStyle-CssClass="bold">
            <Fields>
                <asp:TemplateField>
                    <ItemTemplate>
                        <asp:Button ID="UpdateLinkButton" runat="server" CommandName="Update"
CausesValidation="false"
                            Text="Cập nhật" />
                        <asp:Button ID="Button1" runat="server" CommandName="Cancel" CausesValidation="false"
                            Text="Hủy bỏ" />
                    </ItemTemplate>
                </asp:TemplateField>
            </Fields>
        </asp:DetailsView>
```

Trong file **"Insert.aspx" sửa lại** <asp:DetailsView ID="DetailsView1" như sau:

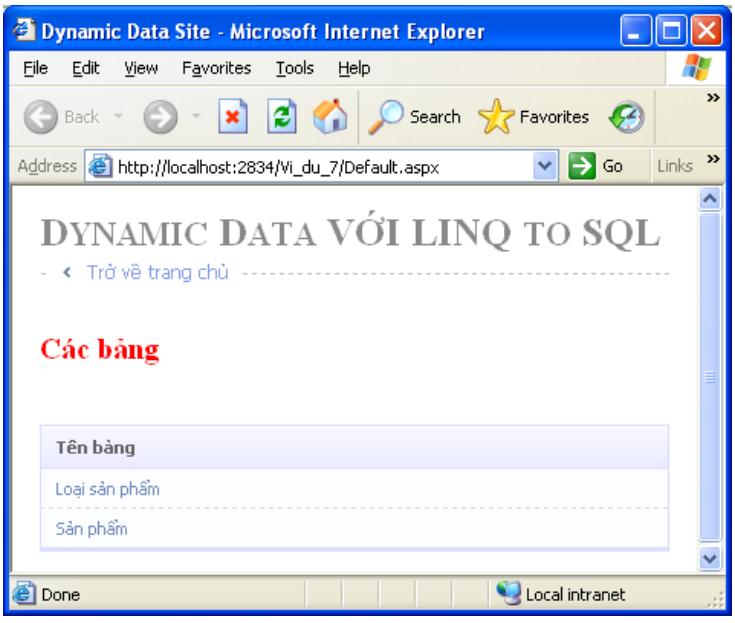
```
        <asp:DetailsView ID="DetailsView1" runat="server" DataSourceID="DetailsDataSource" DefaultMode="Insert"
            AutoGenerateInsertButton="False" OnItemCommand="DetailsView1_ItemCommand"
OnItemInserted="DetailsView1_ItemInserted"
            CssClass="detailstable" FieldHeaderStyle-CssClass="bold">
            <FieldHeaderStyle CssClass="bold" />
            <Fields>
                <asp:TemplateField>
```

```
                <ItemTemplate>
                    <asp:Button ID="InsertLinkButton" runat="server" CommandName="Insert" Text="Thêm mới" />
                    <asp:Button ID="Button1" runat="server" CommandName="Cancel" CausesValidation="false"
Text="Hủy bỏ" />
                </ItemTemplate>
            </asp:TemplateField>
        </Fields>
    </asp:DetailsView>
```

Trong file **"List.aspx" sửa lại** <asp:GridView ID="GridView1" như sau:

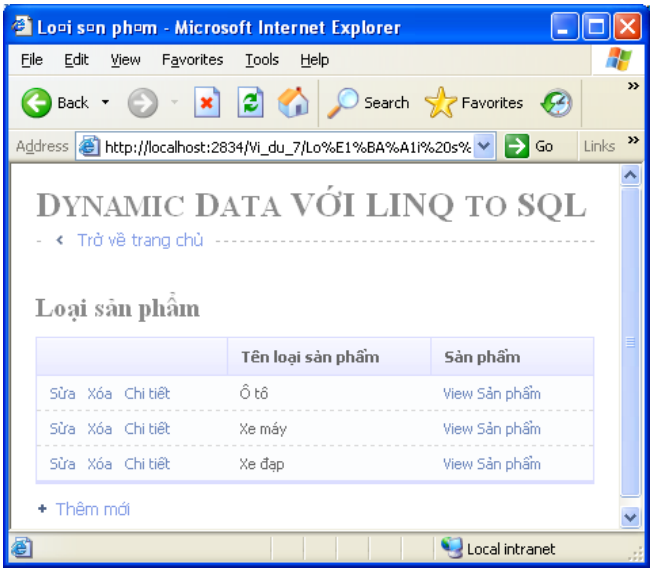
```
    <asp:GridView ID="GridView1" runat="server" DataSourceID="GridDataSource"
      AllowPaging="True" AllowSorting="True" CssClass="gridview">
      <Columns>
        <asp:TemplateField>
          <ItemTemplate>
            <asp:HyperLink ID="EditHyperLink" runat="server"
              NavigateUrl='<%# table.GetActionPath(PageAction.Edit, GetDataItem()) %>'
            Text="Sửa" /> <asp:LinkButton ID="DeleteLinkButton" runat="server" CommandName="Delete"
              CausesValidation="false" Text="Xóa"
              OnClientClick='return confirm("Bạn có chắc chắn xóa bản nghi này không?");'
            /> <asp:HyperLink ID="DetailsHyperLink" runat="server"
              NavigateUrl='<%# table.GetActionPath(PageAction.Details, GetDataItem()) %>'
              Text="Chi tiết" />
          </ItemTemplate>
        </asp:TemplateField>
      </Columns>

      <PagerStyle CssClass="footer"/>
      <PagerTemplate>
        <asp:GridViewPager runat="server" />
      </PagerTemplate>
      <EmptyDataTemplate>
        Không có dữ liệu trong bảng!
      </EmptyDataTemplate>
    </asp:GridView>
```

Bây giờ chúng ta chạy thử chương trình. Các giao diện đã được sửa thành tiếng Việt. Chúng ta vào **Deburg→StartDebugging** (hoặc **F5**) để chạy thử chương trình. Kết quả khi chạy chương trình file **"Defautl.axpx"** (Hình 18).

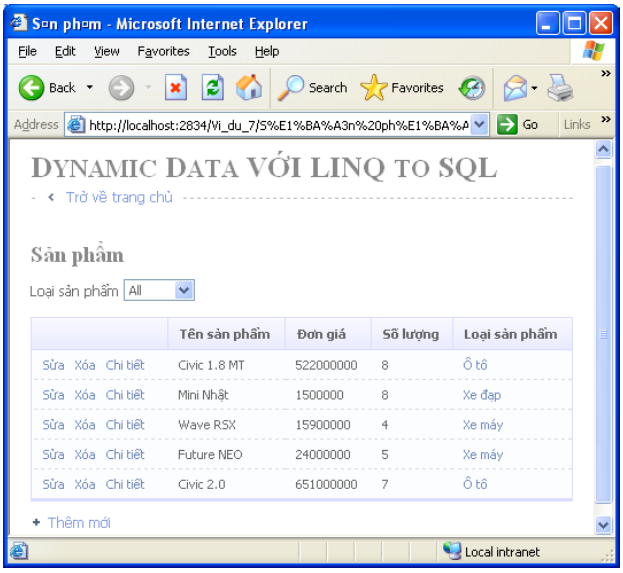


Hình 18: Trang chủ của Vi_du_7

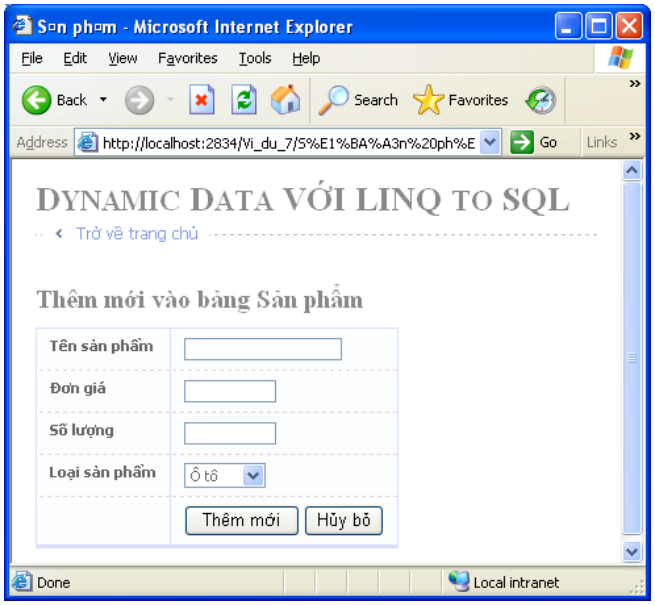
Khi click vào **"Loại sản phẩm"** kết quả như hình 19 hoặc click vào **"Sản phẩm"** kết quả như hình 20.



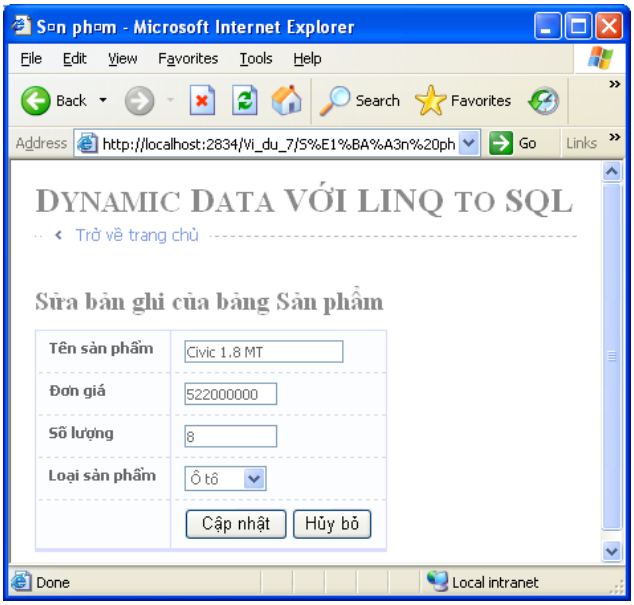
Hình 19: Giao diện của bảng LoaiSanPham



Hình 20: Giao diện của bảng SanPham



Hình 21: Thêm mới vào bảng SanPham



Hình 22: Sửa bản ghi của bản SanPham

4. Câu hỏi ôn tập

1. **Các tính năng của Dynamic Data?**

   Trả lời:

   - Web Scaffolding để tạo ra một ứng dụng web dựa trên các lược đồ cơ bản của cơ sở dữ liệu. Dynamic Data scaffolding có thể tạo ra một chuẩn UI (User Interface – Giao diện người dùng) từ các mô hình dữ liệu.
   - Đầy đủ các thao tác (tạo, cập nhật, xóa bỏ, hiển thị) cho việc truy cập dữ liệu truy cập dữ liệu, các thao tác về quan hệ giữa các bảng và kiểm tra tính hợp lệ của dữ liệu.
   - Tự động hỗ quan các quan hệ khóa ngoài (foreign-key). Dynamic Data phát hiện ra các quan hệ giữa các bảng và từ đó tạo ra các giao diện người dùng trên các bảng quan hệ.
   - Khả năng tùy chỉnh các UI.
   - Khả năng tùy chỉnh tính hợp lệ cho các trường dữ liệu.

2. **Dynamic Data hỗ trợ các các mô hình dữ liệu nào?**

Trả lời:

Dynamic Data hỗ trợ 2 mô hình dữ liệu LINQ to SQL và ADO.NET Entity Framework.

3. **Các bước cơ bản xây dựng Dynamic Data với LINQ to SQL**

Trả lời:

- Tạo project Dynamic Data Web Site
- Xây dựng Cơ sở dữ liệu
- Tạo DataContext với LINQ to SQL
- Đăng ký DataContext với file Global.asax
- Thêm Custom Metadata vào Model

5. Tài liệu tham khảo

1. Using ASP.NET Dynamic Data, URL: http://msdn.microsoft.com/en-us/library/cc488545.aspx
2. Microsoft ASP.NET, URL: http://www.asp.net/DynamicData/
3. ScottGu's Blog, URL: http://weblogs.asp.net/scottgu/archive/2008/12/02/dec-2nd-links-asp-net-asp-net-dynamic-data-asp-net-ajax-asp-net-mvc-visual-studio-silverlight-wpf.aspx