Bài 5: Databinding và DataDisplay

- ✓ DataBinding
- ✓ Định dạng dữ liệu với Data Template
- ✓ DataSource và DataDisplay Controls

DataBinding

1. DataBinding

Là khái niệm khá quen thuộc trong NNLT như VB.net, C# hoặc trong các ứng dụng như Windows App và có thể được áp dụng trong các ứng dụng Web. Khái niệm này mô tả cách thức kết nối giá trị từ một dữ liệu nguồn nào đó, ví dụ như Recordset, với các điều khiển trong form. Trong lúc thực thi (run-time), dữ liệu nguồn (data source) sẽ được nạp cho điều khiển liên kết (data bound), ***đây gọi là mô hình liên kết dữ liệu động***.

Tuy nhiên, vì trạng thái tự nhiên của giao thức HTTP là ngắt kết nối, nên việc kết nối dữ liệu như trên là không thể thực hiện được. Khi IE phát triển, nó được tích hợp các COM components cho phép sử dụng kỹ thuật data binding trên giao thức HTTP, kỹ thuật này được gọi là **client-side data binding**. Nhược điểm của kỹ thuật này là phụ thuộc vào trình duyệt và không đảm bảo an toàn.

.Net Framework thiết kế với mô hình server-side databinding thực hiện trên giao thức HTTP tương tự như sử dụng trong Win Form

DataBinding

Các lớp dữ liệu cơ sở trong DataBinding (WebControl)

Để thực hiện được liên kết dữ liệu động các điều khiển dữ liệu phải kế thừa từ các lớp **cơ sở trừu tượng** (hỗ trợ DataBinding)

- **BaseDataBoundControl:** lớp cơ sở hỗ trợ DataBinding
- **DataBoundControl:** kế thừa từ BaseDBC, hiển thị dữ liệu dạng bảng, list (Chart, ListView)
- **HierarchicalDataBoundControl:** kế thừa từ BaseDBC, hiển thị dữ liệu dạng thứ bậc phân cấp (Menu, TreeView)

Tất cả các điều khiển có thuộc tính DataSource, DataSourceID, và DataBind() đều hỗ thực thi các lớp cơ sở trừu tượng về DataBinding

DataBinding

Cú pháp sử dung DataBinding

<%# tên dữ liệu nguồn %>

Liên kết dữ liệu nguồn với các điều khiển có 2 loại:

- Dữ liệu nguồn **có 1 giá trị** cần liên kết với điều khiển. Trường hợp này chỉ sử dụng với các loại điều khiển đơn (**single value databinding**)
- Dữ liệu nguồn gồm **nhiều giá trị lặp lại** (vd: list, collection, rowset) và ta cần liên kết để hiển thị nhiều giá trị. Trường hợp này ta phải sử dụng với các List Controls, Conposite Controls (**repeated value databinding**)

DataBinding

1.1. Single Value DataBinding

Tất cả các điều khiển ASP.NET Web Controls đều kế thừa phương thức DataBind() của lớp dữ liệu gốc Control, cho phép chúng có khả năng liên kết dữ liệu với các thuộc tính của điều khiển.

Dạng thức liên kết với dữ liệu nguồn:

- **<%# Tên Thuộc Tính %>**
- **<%# Tên Hàm(tham số) %>**
- **<%# Biểu thức %>**

SingleValue DataBinding

File: Default.aspx

```
<%@ Page Language="C#”
        AutoEventWireup="true"
        CodeFile="Default.aspx.cs".
        Inherits="SingleValueBinding" %>
<form id="form1" runat="server">
  <div>
    <asp:Image runat="server"
        ImageUrl='<%# FilePath %>’
        ID="Image1"/><br/>
    <asp:Label runat="server"
        Text='<%# FilePath %>’
        ID="Label1"/><br/>
    <asp:TextBox runat="server"
        Text='<%# GetFilePath() %>'
        ID="Textbox1"/><br/>
    <asp:HyperLink runat="server".
        NavigateUrl='<%# LogoPath.Value %>'
        Font-Bold="True" Text="Show logo"
        ID="Hyperlink1"/><br/>.
  </div>
</form>
```

File: Default.aspx.cs

```
using. System.Web.UI.WebControls;
using. System.Web.UI.HtmlControls;

public partial class. SingleValueBinding :
System.Web.UI.Page {

protected void Page_Load(object sender, EventArgs e) {
        this.DataBind();
}
protected. string GetFilePath() {
        return
        "http://www.mysite.com/style/logo.png";
}
protected string FilePath {
        get {
        return
        "http://www.mysite.com/style/logo.png";
        }
}
}
```

DataBinding

1.1. Repeated Value DataBinding

Trường hợp dữ liệu nguồn là tập hợp gồm nhiều giá trị hoặc tập hợp (collections) thì ta phải sử dung các điều khiển có nhiều thuộc tính hoặc các điều khiển tập hợp để liên kết các giá trị dữ liệu tương ứng. Các phương pháp này là:

- **Simple Data Binding** hoặc **Inline Data Binding**: Dữ liệu nguồn không cần cấu trúc, không hỗ trợ sắp xếp điều khiển (ListControl)
- **Declarative Data Binding:** Dữ liệu nguồn là loại có cấu trúc, có hỗ trợ sắp xếp điều khiển (CompositeDataBoundControl)

DataBinding

Simple Data Binding

Cho phép tập hợp giá trị của Data Source gán kết với các thuộc tính của Data Bound. Các lớp dữ liệu hỗ trợ Simple Data Binding (ListControl)

- **BulletList**
- **CheckBoxList**
- **DropDownList**
- **ListBox**
- **RadioButtonList**

Cú pháp:

Đối với các loại dữ liệu nguồn chỉ có 2 trường Key và Value (HashTable, ArrayList, Dictionary…).

- **<%# Container.DataItem.Key %>**

và

- **<%# Container.DataItem.Value %>**

Simple DataBinding

Ví dụ: SimpleDataBind.aspx

```
<script runat="server">
  void Page_Load (object s, EventArgs e) {
  if (!Page.IsPostBack) {
    Hashtable mycountries = new Hashtable();
    mycountries.Add("N","Norway");
    mycountries.Add("S","Sweden");
    mycountries.Add("F","France");
    mycountries.Add("I","Italy");
    rb.DataSource=mycountries;
    rb.DataValueField="Key";
    rb.DataTextField="Value";
    rb.DataBind();
  }
}
</script>
```

```
<html>
<body>
    <form runat="server">
        <asp:RadioButtonList id="rb"
            runat="server" AutoPostBack="True" />
    </form>
</body>
</html>
```

DataBinding

Declarative Data Binding

Cho phép liên kết với các Data Source có cấu trúc dữ liệu phức tạp và có khả năng hỗ trợ định dạng (sort, paging, editing) tại các điều khiển Data Bound. Các lớp dữ liệu hỗ trợ Declarative Data Binding (CompositeDataBoundControl)

- **DetailsView**
- **FormView**
- **GridView**
- **RecordList**

Cú pháp:

Để tham chiếu tới bản ghi hoặc DataItem trong điều khiển ta sử dụng tên tham chiếu lúc runtime là **Container.DataItem**

<%# Container.DataItem[“FieldName”] %>

hoặc

<%# DataBinder.Eval (Container.DataItem, “FieldName”) %>

<%# DataBinder.Eval (Container.DataItem, “FieldName”, “{Format}”) %>

Chú ý: Khi sử dụng **Container.DataItem** với C#, bắt buộc phải ép kiểu thích hợp

Declarative DataBinding

Vídụ:

```
<script runat="server">
private void Page_Load (object s, EventArgs e) {
  if (!Page.IsPostBack) {
    Hashtable mycountries = new Hashtable();
    mycountries.Add("MS", 49.56);
    mycountries.Add(“IBM, 55);
    mycountries.Add(“Oracle”, 41.1);
    dgr.DataSource=mycountries;
    dgr.DataValueField="Key";
    dgr.DataTextField="Value";
    rpt.DataSource=mycountries;
    Page.DataBind();
  }
}
</script>

<form runat="server">
<asp:DataGrid id="dgr" runat="server" A
                utoGenerateColumns="false">
    <Columns>
      <asp:BoundColum HeaderText=”Key”
                DataField=”Key” />
      <asp:BoundColum HeaderText=”Value”
                DataField=”Value” />
    </Columns>
</asp:DataGrid>
 <asp:Repeater id=”rpt” runat=”server”>
    <ItemTemplate>
    <li>
        ‘<%# Container.DataItem.Key %>’ giá trị:
    </li>
    </ItemTemplate>
    <AlternatingItemTemplate>
    <li>
        <%# DataBinder.Eval(Container.DataItem),
                “Value”, “{0:F}”) %>
    </li>
</ AlternatingItemTemplate >
</asp:Repeater>
</form>
```

Data Template

2. Định dạng dữ liệu với Data Template

Templates là một tập hợp các thẻ HTML và các điều khiển được sử dụng để thiết lập định dạng cho tưng bộ phận cụ thể của điều khiển nào đó. Ví dụ ta có thể thiết lập giao diện và định dạng cho từng Row trong một DataList.

- Các điều khiển có thuộc tính giao diện và định dạng
- Điều chỉnh giao diện bằng CSS styles
- Một số điều khiển hỗ trợ templates

Tuy nhiên, mỗi điều khiển sử dụng tập hợp templates khác nhau

Data Template

2.1. Các loại điều khiển và templates

Control	Templates
ChangePassword	•ChangePasswordTemplate •SuccessTemplate
CompleteWizardStep	•ContentTemplate •CustomNavigationTemplate
CreateUserWizard	•HeaderTemplate •SideBarTemplate •StartNavigationTemplate •StepNavigation •FinishNavigation
CreateUserWizardStep	•ContentTemplate •CustomNavigationTemplate

Data Template

2.1. Các loại điều khiển và templates

DataList	•HeaderTemplate •FooterTemplate •ItemTemplate •AlternatingItemTemplate •SeparatorTemplate •SelectedItemTemplate •EditItemTemplate
DetailsView	•HeaderTemplate •FooterTemplate •PagerTemplate •EmptyDataTemplate
FormView	•ItemTemplate •EditItemTemplate •InsertItemTemplate •HeaderTemplate •FooterTemplate •PagerTemplate •EmptyDataTemplate
GridView	•PagerTemplate •EmptyDataTemplate

ListView	•LayoutTemplate •ItemTemplate •ItemSeparatorTemplate •GroupTemplate •GroupSeparatorTemplate •EmptyItemTemplate •EmptyDataTemplate •SelectedItemTemplate •AlternatingItemTemplate •EditItemTemplate •InsertItemTemplate
Login	•LayoutTemplate
LoginView	•AnonymousTemplate •LoggedInTemplate

Data Template

2.1. Các loại điều khiển và templates

Menu	•DynamicTemplate •StaticTemplate
PasswordRecovery	•QuestionTemplate •SuccessTemplate •UserNameTemplate
Repeater	•HeaderTemplate •FooterTemplate •ItemTemplate •AlternatingItemTemplate •SeparatorTemplate

SiteMapPath	•CurrentNodeTemplate •RootNodeTemplate •NodeTemplate •PathSeparatorTemplate
TemplatePagerField	•PageTemplate
UpdatePanel	•ContentTemplate
UpdateProgress	•ProgressTemplate
Wizard	•FinishNavigationTemplate •HeaderTemplate •StartNavigationTemplate •StepNavigationTemplate •SideBarTemplate

Data Template

2.2. Sử dụng templates

- Templates được sử dung trực tiếp trong các tệp giao diện *.aspx, chúng được tạo với các khai báo dạng XML.
- Các điều khiển templates chỉ được tạo theo yêu cầu phía Client

Cú pháp:

```
<asp:ComplexControl id="ID" runat="server">
<template>
        <asp:Control id="ID" runat="server">
                <%# DataBiding %>
        </asp:Control>
</template>
</asp:ComplexControl>
```

Chú ý: Master files và template lồng nhau (nested) không hiển thị trong design VS.NET 2005 và Visual Web Developer, nhưng vẫn hiển thị trên Web Browser

Data Template

2.2. Sử dụng templates

Ví dụ:

```
<%@ Page Language=”C#”%>
<%@ Import Namespace=”System.Web.UI.WebControls” %>
<%@ Import Namespace=”System.Data” %>
<%@ Import Namespace=”System.Data.SqlClient” %>

<form runat=”server”>
<asp:GridView id=”grvData” runat=”server”
                autogeneratecolumns=”false”>
   <columns>
      <asp:TemplateField HeaderText=”Authors”>
      <ItemTemplate>
         <asp:Image id=”imgPhotoLeft”
            ImageUrl=’ <%# Eval(“zip”)%> ’
            AlternateText=”Author Photo Left Align”
            runat=”server” />
          <asp:Label id=”lblFName” Text=’ <%#
            Eval(“au_fname”)%> ’
            runat=”server” />
          <asp:Label id=”lblLName” Text=’ <%#
            Eval(“au_lname”)%> ’
            runat=”server” />
      </ItemTemplate>
      <AlternatingItemTemplate>
         <asp:Label id=”lblFName”
            Text=’ <%# Eval(“au_fname”)%> ’
            runat=”server” />
          <asp:Label id=”lblLName” Text=’ <%#
            Eval(“au_lname”)%> ’
            runat=”server” />
          <asp:Image id=”imgPhotoRight”
            ImageUrl=’<%# Eval(“zip”)%> ’
            AlternateText=”Author Photo Right Align”.
            runat=”server” />
      </AlternatingItemTemplate>
      </asp:TemplateField HeaderText=”Authors”>
   </columns>
</asp:GridView>
<form>

<script runat=”server”>
   public void Page_Load(object s, EventArgs e) {
      SqlConnection Cnn = new SqlConnection (“server=(local);
            Database=pubs; Integrated Security=SSPI”);
      SqlDataAdapter da = new SqlDataAdapter(“select * from.
            authors”, Cnn);
      DataSet ds = new DataSet();
      da.Fill(ds, “authors”);
      grvData.DataSource = ds.Tables[“authors”];
      grvData.DataBind();
   }
</script>
```

DataSource và DataDisplay

3. DataSource và DataDisplay

Data Source Control là những điều khiển dữ liệu nguồn tương tác với các Data Bound Control để che quá trình DataBiding phức tạp. Chúng cung cấp các công cụ với khả năng thực thi trên dữ liệu như insert, delete, update, sort …

Mỗi Data Source Control được đóng gói với trình điều khiển dữ liệu nguồn cụ thể như RelationalDB, tệp XML, hoặc những Custom Control và hỗ trợ khả năng:

- **Quản lý liên kết:** thiết lập chuỗi liên kết, mở và đóng connection
- **Lấy dữ liệu từ nguồn:** Select dữ liệu, mở tệp XML
- **Quản lý sự hiển thị:** Định dạng hiển thị, phân trang, cache, …
- **Thao tác với dữ liệu:** Thêm, sửa, xoá, …

DataSource và DataDisplay

3.1 DataSource Controls

ASP.NET bao gồm nhiều nhiều loại Data Source Control cho các loại dữ liệu khác nhau như SQL Server, Oracle, OLE-DB servers, tệp XML và các loại dữ liệu, phân làm 2 loại:

- **Hierarchical Data Source Controls:**
  - XmlDataSource: tệp XML
  - SiteMapDataSource: SiteMap Control
- **Table-based Data Source Controls:**
  - SqlDataSource: dư liệu SQL, Oracle, OLE-BD, ODBC
  - ObjectDataSource: Data Object (DataSet, DataTable…)
  - LinqDataSource: Linq từ phiên bản 3.5
  - AccessDataSource: Microsoft Access Database

DataSource và DataDisplay

3.1. DataSource Controls

SqlDataSource Control đại diện một kết nối tới CSDL quan hệ như SQL, Oracle, …. Connection được tạo dựa vào thông tin liên kết ConnectionString và ProviderName. Điều khiển dữ lieu này cung cấp các thuộc tính có thể thiết lập để thực hiện thao tác trên dữ liệu

```
<asp:SqlDataSource runat="server" ID= "MySqlSource"
    ProviderName='<%$ ConnectionStrings:LocalWindows.ProviderName %>'
            ConnectionString=' <%$ ConnectionStrings:LocalWindows %>'
    SelectCommand= "SELECT * FROM EMPLOYEES"
    UpdateCommand= "UPDATE EMPLOYEES SET LASTNAME=@lame"     DeleteCommand=
    "DELETE FROM EMPLOYEES WHERE EMPLOYEEID=@eid"    FilterExpression=
    "EMPLOYEEID > 10"> ..... .....
</asp:SqlDataSource>
```

Chú ý: DataSource Controls thao tác trên dữ liệu dựa vào khả năng DataSourceView

DataSource và DataDisplay

3.2. DataDisplay Controls

Là các điều khiển hiển thị dữ liệu (DataBoundControl) thường được sử dung trong các liên kết động (DataBinding) với các điều khiển dư liệu nguồn.

Các điều khiển hiển thị dữ liệu có khả năng thực hiện các tác vụ trên dữ liệu nguồn tuỳ thuộc vào khả năng View thiết lập bởi các DataSourceView của các điều khiển dữ liệu nguồn đó.

Các DataDisplay Controls:

- GridView: Hiển thị dữ liệu dạng bảng
- DetailsView: Hiển thị dữ liệu dạng bản ghi
- FormView: Hiển thị dữ liệu theo template
- TreeView: Hiển thị dữ liệu dạng cây với các nhãn
- Menu: Hiển thị chế độ thực đơn

DataSource và DataDisplay

3.3. Kỹ thuật DataControls

Chi tiết Controls tham khảo DataControls

- Các điều khiển Danh sách
- Điều khiển GridView
- Các điều khiển DataSource
- Điều khiển DataList
- DetailsView và FormView