1. **데이터베이스 들어가기**

**[ ASP.NET에서의 데이터 접근 구조 ]**

- 데이터 접근 구조

웹 응용 프로그램 데이터 계층

공급자 데이터 소스

==> 웹 응용프로그램에서는 데이터 계층의 ADO.NET과 공급자를 통해 해당 데이터 소스에 접근함

웹 응용 프로그램

( ASP.NET)

데이터 계층

(ADO.NET)

공급자

공급자

공급자

데이터 스소

(XML)

(

데이터 스소

(Oracle)

(

데이터 스소

(SQL Server)

(

\*) 데이터 소스(Data Source)

데이터의 원천( 디스크 파일, 데이터 베이스, XML 파일, XML웹 서비스 등)

데이터 소스중 가장 많이 사용되는 것은 데이터베이스임

\*) 공급자(Provider)

데이터 소스에 연결하고 명령을 실행하며 데이터를 검색하는 데 사용

- 데이터 공급자를 구성하는 4가지 핵심요소

Connect : 특정 데이터 소스에 연결 기본 클래스는 DbConnection

Command : 데이터 소스에 명령을 실행 기본 클래스는 DbCommand

DataReader: 데이터 소스에서 정반향 읽기 전용으로 데이터 조회 기본 클래스는 DbDataReader

DataAdapter : 웹 서버 메모리에 가상의 DataSet을 생성하고 업데이트 내용을 데이터 소스에 적

용( 비연결기반) 기본 클래스는 DbDataAdapter

\*) 데이터 계층

ADO.NET을 통해 개발자는 데이터 소스의 종류에 상관없이 일관성있는 방법으로 데이터 검색 조작

가능 이러한 처리를 데이터 계층에서 함

\*) 웹 응용 프로그램

ASP.NET 서버 컨트롤 및 개발자 코드를 사용하여 데이터 소스로부터 가져온 결과 집합을 표현하는

계층

**[ ADO.NET ]**

연결기반 DB 연동 & 비연결기반 DB 연동

\*) 접속자 수 , 하드웨어 사양, 시스템의 업무 프로세스 등등 성능과 관련된 여러가지 요인들을 분석해

서 연결기반 연동 방법을 사용할 것인지, 아니면 비연결기반 연동 방법을 사용할 것인지를 판단

**[ 데이터 소스 컨트롤 ] ( New in 2.0)**

- 데이터 소스에 연결하는 작업 및 데이터 조회, 입력, 수정, 삭제 등의 작업을 관리하는 서버 컨트롤임

- 총 5개의 컨트롤 존재

SqlDataSource : SQL Server를 비롯하여 OLE DB, ODBC, Oracle데이터베이스에 엑세스 할 수

있게 도와줌

AccessDataSource : Microsoft Access 데이터 페이스에 액세스 할 수 있도록 도와줌

ObjectDataSource : 특정 클래스를 사용하여 데이터베이스에 액세스 할수 있도록 해줌

XmlDataSource : 데이터 소스가 XML 파일일 경우 사용하는 데이터 소스 컨트롤

SiteMapDataSource : ASP.NET사이트 탐색에 사용되는 데이터 소스 컨트롤임

- 해당 컨트롤을 사용하면 데이터 접근이 간소화됨

웹 응용 프로그램(ASP.NET)

XmlDataSource

AccessDataSource

SqlDataSource

데이터 스소

(XML)

(

데이터 스소

(Access)

(

데이터 스소

(SQL Server)

(

\*) 데이터 소스 컨트롤은 데이터 소스와의 연결 잡업을 비롯하여 데이터 조회, 입력, 수정, 삭제 등의 작

업을 자동으로 처리해 줌

**[ Data 바인딩 컨트롤 ]**

- 데이터 소스 컨트롤과 바인딩하여 데이터를 쉽게 표시 및 수정할 수 있도록 함

- 템플릿을 사용하여 컨트롤 레이아웃을 사용자가 지정할 수 있도록 해주는 서버 컨트롤

- 종류

Repeater : 데이터 소스에서 반환된 레코드 집합을 단순히 반복적으로 표현하는 컨트롤

기본 레이아웃 제공이 안되므로 템플릿을 사용하여 레이아웃을 구성해야 함

DataList : 데이터 소스에서 반환된 레코드 집합을 테이블 형식으로 랜더링하며

항목 선택 및 편집 템플릿을 제공함

DataGrid : 데이터 소스에서 반환된 레코드 집합을 테이블 형식으로 랜더링

래코드 선택, 편집, 삭제, 페이징 및 정렬 기능등을 제공하는 서버 컨트롤

2.0으로 넘어오면서 GridView컨트롤로 대체되어 더 이상 사용이 권장되지 않음

GridView : 2.0버전에서 추가된 컨트롤

데이터를 테이블 형식으로 랜더링

열, 정열, 데이터 페이징, 단일 레코드 편집 또는 삭제 기능을 제공

DetailsView : 2.0 버전에서 추가된 컨트롤

한번에 하나의 레코드를 테이블 형식으로 랜더링하고 삽입, 수정, 삭제

및 페이징 기능을 제공하는 컨트롤

주로 마스터 컨트롤(GridView)에서 선택한 레코드를 DetailsView 컨트롤에

표시하는 마스터-세부 시나리오에 사용됨

FormView : 2.0 버전에서 추가된 컨트롤

DetailsView와 비슷한 컨트롤

한번에 하나의 레코드를 랜더링

레코드 삽입, 수정, 삭제 및 페이징 기능 제공

레코드를 표시할때 미리 정의된 레이아웃을 제공하지 않으므로

개발자가 직접 템플릿을 이용하여 레코드의 개별 열을 표시해야 함

**[ 간단한 SQL문 ]**

SELECT pub\_id, pub\_name, city From publishers

INSERT INTO member (user\_id, password, name, phone) VALUES(;admin’, ‘1111’, ‘홍길동’, ‘011-111-1111’)

UPDATE Member SET password = ‘2222’ WHERE user\_id ‘admin’

DELETE FROM Member WHERE user\_id = ‘admin’

1. **데이터 소스 컨트롤을 사용한 DB연동**

**[ SqlDataSource ] New in 2.0**

SqlServer, Oracle, OLEDB, ODBC에 액세스 가능

- 기본 형태

<asp:SqlDataSource ID="SqlDataSource1" runat="server">

</asp:SqlDataSource>

=> 기본형태의 컨트롤은 어떠한 기능도 하지 못함

- 기능을 수행할 수 있는 형태

<asp:SqlDataSource ID="SqlDataSource1" runat="server"

ConnectionString="<$ ConnectionStrings:pubsConnectionString %>"

SelectCommand="SELECT \*FROM publishers"></asp:SqlDataSource>

\*\*) ConnectionString: 연결 속성

SelectCommand : 조회 명령

실습) 특정 테이블에 접근해 ID와 Name 가져오기

1. 웹 폼에서 SqlDataSource 컨트롤 생성
2. 연결 테이블에 접근

* 연결 문자열 확인
* Select 문 구성 : 여기서 전체를 선택할 수도 있고, 부분을 선택할 수 도 있슴
* 쿼리 테스트 확인

1. 생성된 연결 문자열을 구성파일(Web.Comfig)에서 확인가능
2. 생성된 SqlDataSource 컨트롤을 웹 폼의 소스 뷰에서 확인 가능
3. SqlDataSource 컨트롤과 바인딩 가능한 서버 컨트롤을 사용하여 데이터를 표현(리스트 컨트롤 사용 : BulletedList, RadioButtonList, DropDownList 등등)

* CheckBoxList, RadioButttonList, DorpDownList, ListBox, BulleteList
* <b>CheckBoxList</b> 태그를 사용하여 문자열 출력

<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">

protected void DropDownList1\_SelectedIndexChanged(object sender, EventArgs e)

{

Label1.Text = "Text : " + DropDownList1.SelectedItem.Text + "<br>" + "Value : " + DropDownList1.SelectedItem.Value;

}

</script>

<html xmlns="http://www.w3.org/1999/xhtml" >

<head id="Head1" runat="server">

<title>제목 없음</title>

</head>

<body>

<form id="form1" runat="server">

<div>

<h3>SqlDataSource 컨트롤과 리스트 컨트롤을 사용한 예제</h3>

<table border="1">

<tr valign="top">

<td>

<b>CheckBoxList</b><br />

<asp:CheckBoxList ID="CheckBoxList1" runat="server" DataSourceID="SqlDataSource1" DataTextField="pub\_name" DataValueField="pub\_id">

</asp:CheckBoxList>

</td>

<td>

<b>RadioButtonList</b><br />

<asp:RadioButtonList ID="RadioButtonList1" runat="server" DataSourceID="SqlDataSource1" DataTextField="pub\_name" DataValueField="pub\_id">

</asp:RadioButtonList>

</td>

<td>

<b>DropDonwList</b><br />

<asp:DropDownList ID="DropDownList1" runat="server" DataSourceID="SqlDataSource1" DataTextField="pub\_name" DataValueField="pub\_id" AutoPostBack="True" OnSelectedIndexChanged="DropDownList1\_SelectedIndexChanged">

</asp:DropDownList>

</td>

<td>

<b>ListBox</b><br />

<asp:ListBox ID="ListBox1" runat="server" DataSourceID="SqlDataSource1" DataTextField="pub\_name" DataValueField="pub\_id"></asp:ListBox>

</td>

<td>

<b>BulletedList</b><br />

<asp:BulletedList ID="BulletedList1" runat="server" DataSourceID="SqlDataSource1" DataTextField="pub\_name" DataValueField="pub\_id">

</asp:BulletedList>

</td>

</tr>

</table>

<asp:Label ID="Label1" runat="server" Text=""></asp:Label>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:pubsConnectionString %>" SelectCommand="SELECT [pub\_id], [pub\_name] FROM [publishers]"></asp:SqlDataSource>

</div>

</form>

</body>

</html>

**[ Repeater]**

데이터소스에서 반환된 레코드 집합을 템플릿을 이용해서 반복적으로 표현하는 컨트롤

예제 확인1) SqlDataSource와 Repeator를 사용한 예제

<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">

</script>

<html xmlns="http://www.w3.org/1999/xhtml" >

<head id="Head1" runat="server">

<title>제목 없음</title>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:TESTConnectionString %>"

SelectCommand="SELECT \* FROM [Weather]"></asp:SqlDataSource>

</div>

<asp:Repeater ID="Repeater1" runat="server" DataSourceID="SqlDataSource1">

<ItemTemplate>

지역 : <%# Eval("AREA") %><br />

날짜 : <%# Eval("DATE") %><br />

<hr />

</ItemTemplate>

</asp:Repeater>

</form>

</body>

</html>

Repeater의 사용

* Repeater와 SqlDataSource를 바인딩함
* Repeater 컨트롤의 탬플릿 중 가장 기본적인 템플릿인 ItemTemplate만을 사용하여 데이터 소스에서 가져온 레코드 집합을 레코드 수만큼 ITemTemplate에서 지정한 레이아웃대로 표현하게 됨
* Eval메서드 : 데이터 소스열(컬럼)의 값을 가져와 문자열로 반환하는 메서드임

예제 확인2) 다른 템플릿을 활용한 출력

<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">

</script>

<html xmlns="http://www.w3.org/1999/xhtml" >

<head id="Head1" runat="server">

<title>제목 없음</title>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:TESTConnectionString %>"

SelectCommand="SELECT \* FROM [Weather]"></asp:SqlDataSource>

</div>

<asp:Repeater ID="Repeater1" runat="server" DataSourceID="SqlDataSource1">

<HeaderTemplate>

<b>등록된 지역별 날씨의 정보</b>

<table border="1">

<tr align="center">

<td><b>지역</b></td>

<td><b>날씨</b></td>

</tr>

</HeaderTemplate>

<ItemTemplate>

<tr>

<td><%# Eval("AREA")%></td>

<td><%# Eval("DATE")%></td>

</tr>

</ItemTemplate>

<AlternatingItemTemplate>

<tr style="background-color: ButtonFace;">

<td><%# Eval("AREA")%></td>

<td><%# Eval("DATE")%></td>

</tr>

</AlternatingItemTemplate>

<SeparatorTemplate>

<tr>

<td colspan="5" style="height: 5px; background-color: Blue;"></td>

</tr>

</SeparatorTemplate>

<FooterTemplate></table></FooterTemplate>

</asp:Repeater>

</form>

</body>

</html>

\*) ItemTemplate, AlternatinItemTemplate, SeparatorTemplate : 레코드 수만큼 반복적 레코딩 수행

HeaderTemplate, FooterTemplate는 단 한번만 템플릿 수행함

**[DataList ]**

데이터 소스에서 반환된 레코드 집합을 테이블 형태로 랜더링하며 선택 및 편집 항목에 사용되는 템플릿 제공

<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">

</script>

<html xmlns="http://www.w3.org/1999/xhtml" >

<head id="Head1" runat="server">

<title>제목 없음</title>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:TESTConnectionString %>"

SelectCommand="SELECT \* FROM [Weather]"></asp:SqlDataSource>

<asp:DataList ID="DataList1" runat="server" DataKeyField="AREA" DataSourceID="SqlDataSource1" BorderColor="Blue" BorderWidth="1px" GridLines="Horizontal">

<HeaderTemplate>

지역별 날씨 정보

</HeaderTemplate>

<ItemTemplate>

AREA:

<asp:Label ID="AREALabel" runat="server" Text='<%# Eval("AREA") %>'></asp:Label><br />

DATE:

<asp:Label ID="DATELabel" runat="server" Text='<%# Eval("DATE") %>'></asp:Label><br />

MIN\_TEM:

<asp:Label ID="MIN\_TEMLabel" runat="server" Text='<%# Eval("MIN\_TEM") %>'></asp:Label><br />

MAX\_TEM:

<asp:Label ID="MAX\_TEMLabel" runat="server" Text='<%# Eval("MAX\_TEM") %>'></asp:Label><br />

WEA:

<asp:Label ID="WEALabel" runat="server" Text='<%# Eval("WEA") %>'></asp:Label><br />

<br />

</ItemTemplate>

<FooterTemplate>

이상 입니다.

</FooterTemplate>

<FooterStyle BackColor="Gray" Font-Bold="False" Font-Italic="False" Font-Overline="False"

Font-Strikeout="False" Font-Underline="False" ForeColor="Yellow" />

<ItemStyle Font-Bold="True" Font-Italic="False" Font-Overline="False" Font-Strikeout="False"

Font-Underline="False" />

<HeaderStyle BackColor="Fuchsia" Font-Bold="True" Font-Italic="False" Font-Overline="False"

Font-Strikeout="False" Font-Underline="False" ForeColor="Aqua" Font-Names="Arial" Font-Size="Larger" />

</asp:DataList></div>

</form>

</body>

</html>

\*) 지원되는 속성창을 통해 글자 변경 가능

**[ GridView ] New in 2.0**

데이터를 테이블 형식으로 랜더링하고 열정렬, 데이터 페이징, 단일 레코드 편집 기능 또는 삭제 기능을 제공하는 등 다른 데이터 바인딩 컨트롤보다 더 많은 기능을 가지고 있는 매우 유용한 컨트롤

예제 확인1) SqlDataSource와 GridViw를 사용한 예제

<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head id="Head1" runat="server">

<title>제목 없음</title>

</head>

<body>

<form id="form1" runat="server">

<div>

<h3>SqlDataSource와 GridView를 사용한 예제1</h3>

<asp:GridView ID="GridView1" runat="server" AllowPaging="True" AllowSorting="True" AutoGenerateColumns="False" DataKeyNames="emp\_id" DataSourceID="SqlDataSource1" Caption="직원(employee) 리스트" CaptionAlign="Left">

<Columns>

<asp:BoundField DataField="emp\_id" HeaderText="emp\_id" ReadOnly="True" SortExpression="emp\_id" />

<asp:BoundField DataField="fname" HeaderText="fname" SortExpression="fname" />

<asp:BoundField DataField="lname" HeaderText="lname" SortExpression="lname" />

<asp:BoundField DataField="job\_lvl" HeaderText="job\_lvl" SortExpression="job\_lvl" />

</Columns>

</asp:GridView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:pubsConnectionString %>" SelectCommand="SELECT [emp\_id], [fname], [lname], [job\_lvl] FROM [employee]"></asp:SqlDataSource>

</div>

</form>

</body>

</html>

예제 확인2) 항목 편집 기능과 관련된 예제

* sqlDataSource 컨트롤의 데이터 소스 구성 대화상자를 사용해서 업데이트 관련 속성과 SQL문을 작성함
* 우측에 있는 고급버튼 클릭후 Insert, Update및 Delte 문 생성 옵션 체크
* 디자이너에서 GridView의 AutoGenerateEditButton 속성을 true로 지정
* 디자이너에서 GridView의 DataKeyName 속성을 정의하여 편집 모드에서 해당 항목을 찾을 수 있게 끔 설정함

**[ DetailsView ] New in 2.0**

한번에 하나의 레코드를 테이블 형식으로 랜더링 하고 삽입, 수정,삭제및 페이징 기능을 제공하는 컨트롤

주로 마스터 컨트롤(GridView)에서 선택한 레코드를 DetailsView 컨트롤에 표시하는 마스터-세부 시나리오에 사용됨

예제 확인1) 데이터의 삽입, 수정, 삭제가 가능하며 페이징 기능을 포함(SqlDataSource& DetailsView)

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="\_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

<title>제목 없음</title>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:TESTConnectionString %>"

SelectCommand="SELECT \* FROM [Weather]" DeleteCommand="DELETE FROM [Weather] WHERE [AREA] = @AREA" InsertCommand="INSERT INTO [Weather] ([AREA], [DATE], [MIN\_TEM], [MAX\_TEM], [WEA]) VALUES (@AREA, @DATE, @MIN\_TEM, @MAX\_TEM, @WEA)" UpdateCommand="UPDATE [Weather] SET [DATE] = @DATE, [MIN\_TEM] = @MIN\_TEM, [MAX\_TEM] = @MAX\_TEM, [WEA] = @WEA WHERE [AREA] = @AREA">

<DeleteParameters>

<asp:Parameter Name="AREA" Type="String" />

</DeleteParameters>

<UpdateParameters>

<asp:Parameter Name="DATE" Type="DateTime" />

<asp:Parameter Name="MIN\_TEM" Type="Double" />

<asp:Parameter Name="MAX\_TEM" Type="Double" />

<asp:Parameter Name="WEA" Type="String" />

<asp:Parameter Name="AREA" Type="String" />

</UpdateParameters>

<InsertParameters>

<asp:Parameter Name="AREA" Type="String" />

<asp:Parameter Name="DATE" Type="DateTime" />

<asp:Parameter Name="MIN\_TEM" Type="Double" />

<asp:Parameter Name="MAX\_TEM" Type="Double" />

<asp:Parameter Name="WEA" Type="String" />

</InsertParameters>

</asp:SqlDataSource>

</div>

<asp:DetailsView ID="DetailsView1" runat="server" AutoGenerateRows="False" DataKeyNames="AREA"

DataSourceID="SqlDataSource1" Height="50px" Width="125px">

<Fields>

<asp:BoundField DataField="AREA" HeaderText="AREA" ReadOnly="True" SortExpression="AREA" />

<asp:BoundField DataField="DATE" HeaderText="DATE" SortExpression="DATE" />

<asp:BoundField DataField="MIN\_TEM" HeaderText="MIN\_TEM" SortExpression="MIN\_TEM" />

<asp:BoundField DataField="MAX\_TEM" HeaderText="MAX\_TEM" SortExpression="MAX\_TEM" />

<asp:BoundField DataField="WEA" HeaderText="WEA" SortExpression="WEA" />

<asp:CommandField ShowDeleteButton="True" ShowEditButton="True" ShowInsertButton="True" />

</Fields>

</asp:DetailsView>

</form>

</body>

</html>

예제 확인2) GridView와 DetailsView를 이용한 예제

<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head id="Head1" runat="server">

<title>제목 없음</title>

</head>

<body>

<form id="form1" runat="server">

<div>

<h3>GridView와 DetailsView를 사용한 예제</h3>

<table>

<tr>

<td>

<asp:GridView ID="GridView1" runat="server" AllowPaging="True" AutoGenerateColumns="False" AutoGenerateSelectButton="True" DataSourceID="SqlDataSource1" DataKeyNames="emp\_id">

<Columns>

<asp:BoundField DataField="emp\_id" HeaderText="emp\_id" ReadOnly="True" SortExpression="emp\_id" />

<asp:BoundField DataField="fname" HeaderText="fname" SortExpression="fname" />

<asp:BoundField DataField="lname" HeaderText="lname" SortExpression="lname" />

</Columns>

</asp:GridView>

</td>

<td>

<asp:DetailsView ID="DetailsView1" runat="server" AutoGenerateRows="False" DataSourceID="SqlDataSource2" DataKeyNames="emp\_id">

<Fields>

<asp:BoundField DataField="emp\_id" HeaderText="emp\_id" ReadOnly="True" SortExpression="emp\_id" />

<asp:BoundField DataField="fname" HeaderText="fname" SortExpression="fname" />

<asp:BoundField DataField="lname" HeaderText="lname" SortExpression="lname" />

<asp:BoundField DataField="job\_lvl" HeaderText="job\_lvl" SortExpression="job\_lvl" />

<asp:BoundField DataField="hire\_date" HeaderText="hire\_date" SortExpression="hire\_date" />

</Fields>

</asp:DetailsView>

</td>

</tr>

</table>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:pubsConnectionString %>" SelectCommand="SELECT [emp\_id], [fname], [lname] FROM [employee]"></asp:SqlDataSource>

<asp:SqlDataSource ID="SqlDataSource2" runat="server" ConnectionString="<%$ ConnectionStrings:pubsConnectionString %>" SelectCommand="SELECT [emp\_id], [fname], [lname], [job\_lvl], [hire\_date] FROM [employee] WHERE ([emp\_id] = @emp\_id)">

<SelectParameters>

<asp:ControlParameter ControlID="GridView1" Name="emp\_id" PropertyName="SelectedValue"

Type="String" />

</SelectParameters>

</asp:SqlDataSource>

</div>

</form>

</body>

</html>

**[ FormView ] New in 2.0**

DetailsView와 비슷하며 한번에 하나의 레코드를 랜더링하고 레코드 삽입, 수정, 삭제 및 페이징 기능을 제공함

DeatilsView 컨트롤 : 데이터 레코드의 각 열에 대해 테이블 기반 레이아웃을 제공

FormView 컨트롤 : 레코드를 표시할 때 미리 정의된 레이아웃을 제공하지 않으므로 개발자가 직접 템플릿을 사용하여 래코드의 개별 필드를 표시해야 함

예제 확인1)

<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head id="Head1" runat="server">

<title>제목 없음</title>

</head>

<body>

<form id="form1" runat="server">

<div>

<h3>SqlDataSource와 FormView를 사용한 예제</h3>

<asp:FormView ID="FormView1" runat="server" DataSourceID="SqlDataSource1" AllowPaging="True">

<ItemTemplate>

책 제목 : <%# Eval("title") %><br />

가격 : <%# Eval("price") %><br />

비고 : <%# Eval("notes") %><br />

출간일자 : <%# Eval("pubdate") %>

</ItemTemplate>

</asp:FormView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:pubsConnectionString %>" SelectCommand="SELECT \* FROM [titles]"></asp:SqlDataSource>

</div>

</form>

</body>

</html>