DataBinding II

pluralsight
see what you can learn

Overview

- **DataSource parameters**
- **Two-way data binding**
- **Additional data binding controls**
  - FormView, ListView
- **Hierarchical data binding**
- **Binding to objects**
- **Typed DataSets**

DataSource parameters

- **Data sources support parameter collections**
  - Separate collection for Update, Insert, Select, Delete, and Filter
  - Parameter source can be control, cookie, form parameter, profile data, query string, session

Add Update and Delete commands with parameters

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
  ConnectionString="<%$ConnectionStrings:QuotesConnectionString1 %>"
  SelectCommand="SELECT [ID], [Author], [Quote] FROM [Quotes]"
  UpdateCommand="UPDATE Quotes SET Author=@Author, Quote=@Quote WHERE ID=@ID"
  DeleteCommand="DELETE FROM Quotes WHERE ID=@ID">
  <UpdateParameters>
    <asp:Parameter Name="Author" Type="String" />
    <asp:Parameter Name="Quote" Type="String" />
    <asp:Parameter Name="ID" Type="Int32" />
  </UpdateParameters>
  <DeleteParameters>
    <asp:Parameter Name="ID" Type="Int32" />
  </DeleteParameters>
</asp:SqlDataSource>
```

List parameter names and types

Note: Unbound parameters should always be named the same as the corresponding column names to work properly with the GridView and DetailsView controls

Parameter binding

- **Several different types of parameter sources are available**
  - Control Parameter
    - Value retrieved from property of any server control on the page
  - CookieParameter
    - Value retrieved from cookie in request
  - FormParameter
    - Value retrieved from HTTP POST variable
  - ProfileParameter
    - Value retrieved from client profile information
  - QueryStringParameter
    - Value retrieved from query string
  - SessionParameter
    - Value retrieved from client session

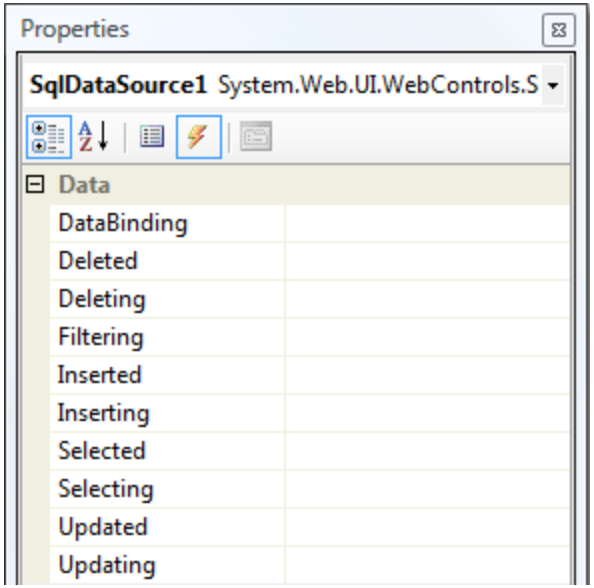
Specifying a bound parameter

- **Bound parameters can be used to populate any of the parameters in a data source**

```
<asp:SqlDataSource ID="patientDataSource" Runat="server"
  SelectCommand="SELECT patientid, firstname... FROM patients
                 WHERE firstname=@firstname"
  ConnectionString="..." >
 <SelectParameters>
    <asp:ControlParameter Name="firstname"
         ControlID="firstNameTextBox"
         PropertyName="Text" />
  </SelectParameters>
</asp:SqlDataSource>
```

Data source events

- **SqlDataSource exposes many events, useful for customizing data source interaction**
  - Adding parameters programmatically
  - Performing an operation after successful insert, update, delete
  - …



Programmatic Parameter Population

- **May be occasions where the existing set of parameters aren't sufficient**
  - You can build your own parameter class, or...
  - Populate the parameter programmatically in response to an event of the data source

```
<asp:SqlDataSource ID="_reviewsDataSource"
        runat="server"
        ConnectionString="<%$ ConnectionStrings:mrcs %>"
        InsertCommand="INSERT INTO reviews(movie_id, summary,
                       rating, review, reviewer) VALUES (@movie_id, @summary,
                       @rating, @review, @reviewer)"
        OnInserting="_reviewsDataSource_Inserting">
  <InsertParameters>
    <asp:Parameter Name="movie_id" />
    <asp:Parameter Name="summary" />
    <asp:Parameter Name="rating" />
    <asp:Parameter Name="review" />
    <asp:Parameter Name="reviewer" />
  </InsertParameters>
</asp:SqlDataSource>
```

```
protected void _reviewsDataSource_Inserting(object sender, SqlDataSourceCommandEventArgs e)
{
   e.Command.Parameters["@reviewer"].Value =
     User.Identity.IsAuthenticated ? User.Identity.Name : "Anonymous";
}
```

Two-way binding

- **Bind() syntax can be used in controls that support insert/update/delete**
  - Like Eval() but data goes both ways
  - During SELECT, field is retrieved just like 'Eval'
  - During INSERT/UPDATE field value is mapped onto parameter with the same name
  - Most common with GridView, DetailsView, FormView, ListView

```
<asp:FormView ID="FormView1" runat="server" DataKeyNames="myObjectId"
              DataSourceID="ds1">
  <EditItemTemplate>
    Date:
    <asp:TextBox ID="dateTextBox" runat="server"
         Text='<%# Bind("DateFiled") %>' />
...
```

FormView

- **Similar to the DetailsView, displays one row at a time**
- **No default rendering, must specify templates for display**
  - Useful when complete control of what to render is needed

```
<asp:FormView ID="fv1" runat="server" DataKeyNames="au_id"
              DataSourceID="sds1">
  <EditItemTemplate>
    Last name:
    <asp:TextBox ID="au_lnameTextBox" runat="server"
                 Text='<%# Bind("au_lname") %>' />
    <asp:LinkButton ID="UpdateButton" runat="server"
         CommandName="Update" Text="Update" />
    <asp:LinkButton ID="UpdateCancelButton" runat="server"
         CommandName="Cancel" Text="Cancel" />
 </EditItemTemplate>
 <InsertItemTemplate> ... </InsertItemTemplate>
 <ItemTemplate> ... </ItemTemplate>
 </asp:FormView>
```

ListView

- **Template-driven control**
  - Can replace *every* existing ASP.NET data-bound control

New LayoutTemplate

Placeholder indicates insertion point

Standard ItemTemplate

```
<asp:ListView runat="server" ID="_simpleTableListView"
              DataSourceID="_moviesDataSource">
  <LayoutTemplate>
    <table>
      <thead>
        <tr>
          <th>ID</th>
          <th>Title</th>
          <th>Release Date</th>
        </tr>
      </thead>
      <tbody>
        <asp:PlaceHolder runat="server" ID="itemPlaceholder" />
      </tbody>
    </table>
  </LayoutTemplate>
  <ItemTemplate>
    <tr>
      <td><%# Eval("movie_id") %></td>
      <td><%# Eval("title") %></td>
      <td><%# Eval("release_date", "{0:d}") %></td>
    </tr>
  </ItemTemplate>
</asp:ListView>
```

ListView + CSS

- **You are in complete control of what the ListView renders**
  - Trivial to inject CSS styles
  - Common task of taking a designer-created CSS styled piece of HTML and building a control that renders it becomes simple!

DataPager

- **DataPager class decouples paging UI from ListView**
  - Can place paging UI anywhere you like
  - Can create multiple paging interfaces attached to the same control

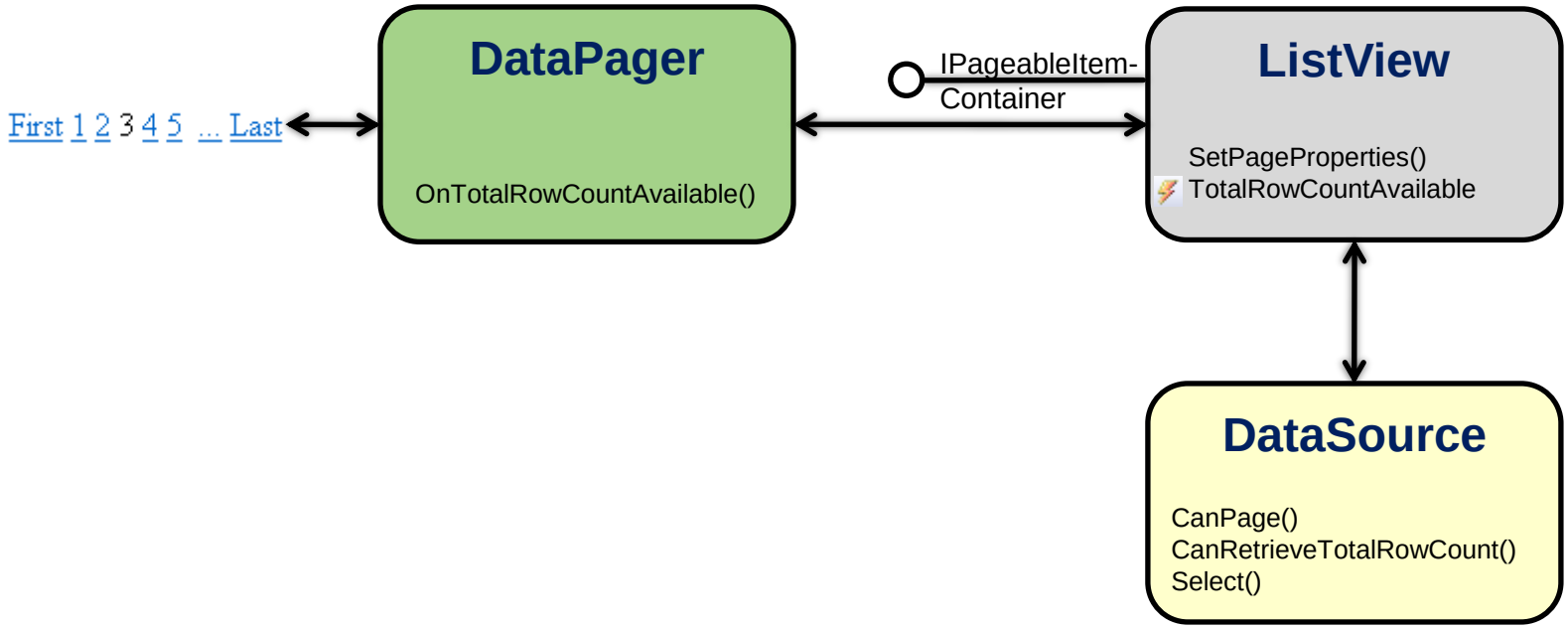
```
<asp:ListView ID="_moviesGrid" runat="server" DataKeyNames="movie_id"
            DataSourceID="_moviesDataSource">
  <LayoutTemplate>
    <!-- ... -->
    <div class="Pagination">
      <asp:DataPager ID="_moviesGridDataPager" runat="server">
        <Fields>
         <asp:NumericPagerField />
        </Fields>
      </asp:DataPager>
    </div>
  </LayoutTemplate>
</asp:ListView>
```

DataPager Implementation

- **DataPager provides paging support to any control that implements IPageableItemContainer**
  - Currently only implemented by ListView ☺

```
public interface IPageableItemContainer
{
    event EventHandler<PageEventArgs> TotalRowCountAvailable;
    void SetPageProperties(int startRowIndex, int maximumRows,
                           bool databind);
    int MaximumRows { get; }
    int StartRowIndex { get; }
}
```

Paging relationships



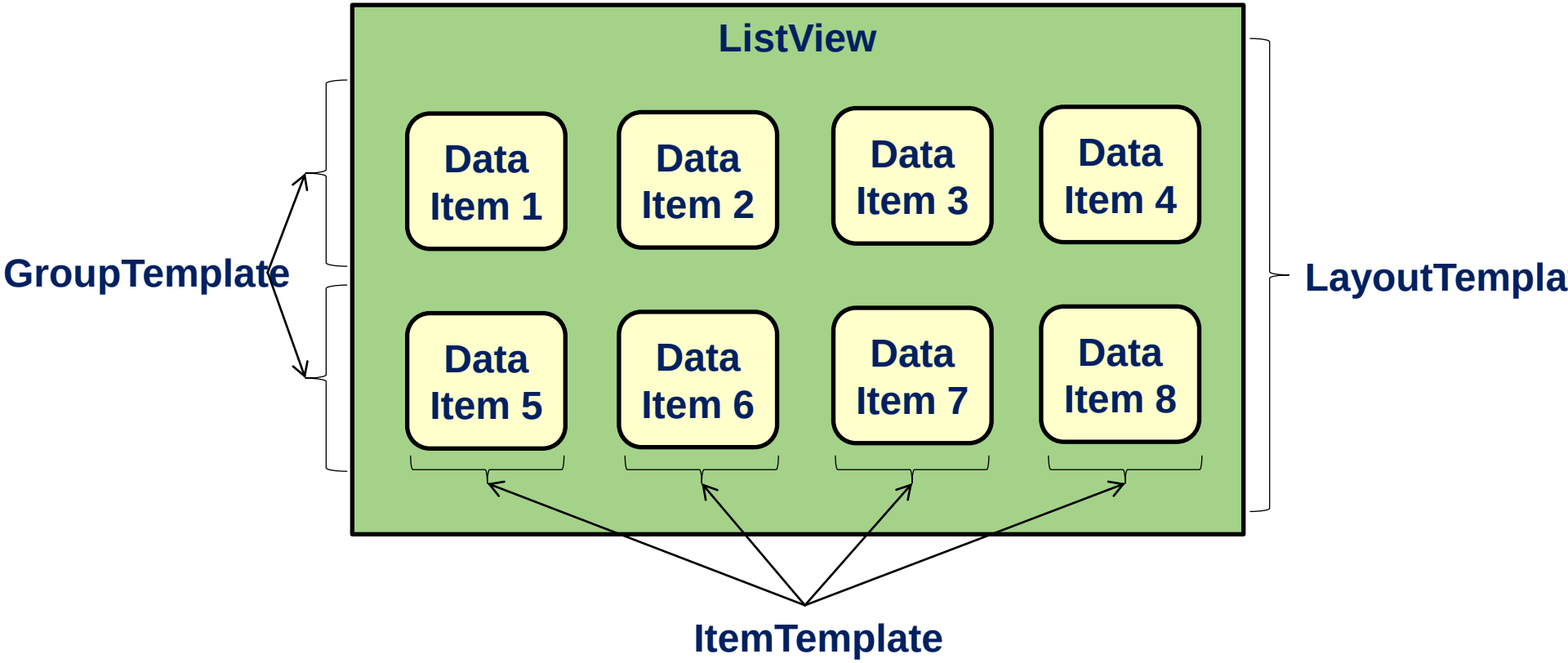
Sorting, Editing, Inserting, Deleting

- **CRUD operations supplied in similar fashion to FormView**
  - Add command button with CommandName set to Sort, Cancel, Delete, Select, Edit, Insert, Update

```
<asp:ListView ID="_moviesGrid" runat="server" DataKeyNames="movie_id" DataSourceID="_moviesDataSource">
  <LayoutTemplate>
    <div class="PrettyGrid">
      <table cellpadding="0" cellspacing="0" summary="">
        <thead><tr><th scope="col">
              <asp:LinkButton ID="_movieIdSortLink"
                   CommandName="Sort" CommandArgument="movie_id"
                   runat="server">ID</asp:LinkButton> </th>
           <th scope="col"> <asp:LinkButton ID="_titleSortLink"
                   CommandName="Sort" CommandArgument="title"
                   runat="server">Title</asp:LinkButton> </th>
           <th scope="col"><asp:LinkButton ID="_releaseDateSortLink"
                   CommandName="Sort" CommandArgument="release_date"
                   runat="server">Release date</asp:LinkButton> </th>
          </tr>
        </thead>
    <!-- ... -->
  </LayoutTemplate>
</asp:ListView>
```

Grouping

- **ListView supports grouping of data items**
  - Similar to the grouping mechanism used by a DataList
  - New GroupTemplate sits above LayoutTemplate



ListView GroupTemplate Example

```
<asp:ListView ID="_groupListView" runat="server"
              DataKeyNames="movie_id" DataSourceID="_moviesDataSource"
              GroupItemCount="4" >
  <GroupTemplate>
    <tr>
      <asp:PlaceHolder runat="server" ID="itemPlaceholder" />
    </tr>
  </GroupTemplate>
  <LayoutTemplate>
    <table>
      <asp:PlaceHolder ID="groupPlaceholder" runat="server" />
    </table>
  </LayoutTemplate>
  <ItemTemplate>
    <td>
      movie_id:
      <asp:Label ID="_movie_idLabel" runat="server"
                 Text='<%# Eval("movie_id") %>' /> <br />
      title:
      <asp:Label ID="_titleLabel" runat="server"
                 Text='<%# Eval("title") %>' /> <br />
      release_date:
      <asp:Label ID="_release_dateLabel" runat="server"
                 Text='<%# Eval("release_date", "{0:d}") %>' /> <br />
    </td>
  </ItemTemplate>
</asp:ListView>
```

Hierarchical data binding

- **XmlDataSource serves any xml content**
  - New XPath data binding expression

```
 <asp:XmlDataSource ID="psRssDataSource" Runat="server"
    DataFile="http://www.pluralsight.com/community/blogs/MainFeed.aspx"
    XPath="/rss/channel/item" />

<asp:DataList ID="rssDataList" Runat="server"
              DataSourceID="psRssDataSource">
    <ItemTemplate>
      <b><%# XPath("pubDate")%> - <%# XPath("title") %></b>
      <br />
      <%# XPath("description") %>
    </ItemTemplate>
</asp:DataList>
```

Specifying a transform file

- **XmlDataSource supports a transform file to use an XSL transform prior to binding**
  - XPath expression available to select nodeset as well

```
<asp:XmlDataSource ID="XmlDataSource1" runat="server"
        DataFile="~/App_Data/Bookstore2.xml"
        TransformFile="~/App_Data/Bookstore.xsl" />
```

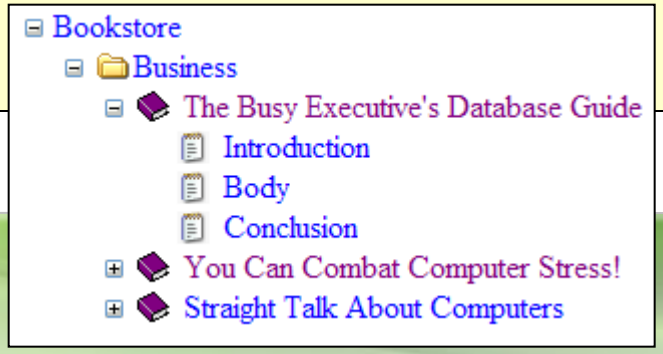
```
<Bookstore>
  <genre name="Business">
    <book>
      <ISBN>BU1032</ISBN>
      <title>The Busy Executive's Database Guide</title>
      <price>19.99</price>
      <chapters>
        <chapter num="1" name="Introduction">
          Abstract...
        </chapter>
        <chapter num="2" name="Body">
          Abstract...
        </chapter>
      </chapters>
    </book>
...
```

```
<Bookstore>
  <genre name="Business">
    <book ISBN="BU1032"
          Title="The Busy Executive's Database Guide"
          Price="19.99">
      <chapter num="1" name="Introduction">
        Abstract...
      </chapter>
      <chapter num="2" name="Body">
        Abstract...
      </chapter>
    </book>
...
```

Binding to a TreeView

- **The TreeView is a truly hierarchical control**
  - Ideal candidate for binding to an XmlDataSource
  - Can specify TreeNodeBinding for particular elements

```
<asp:TreeView ID="TreeView1" runat="server"
              DataSourceID="XmlDataSource1">
  <DataBindings>
     <asp:TreeNodeBinding DataMember="chapter"
                  ImageUrl="~/img/notepad.gif" ValueField="name" />
     <asp:TreeNodeBinding DataMember="book"
                  ImageUrl="~/img/closedbook.gif" ValueField="Title" />
     <asp:TreeNodeBinding DataMember="genre"
                  ImageUrl="~/img/folder.gif" ValueField="name" />
  </DataBindings>
</asp:TreeView>
```



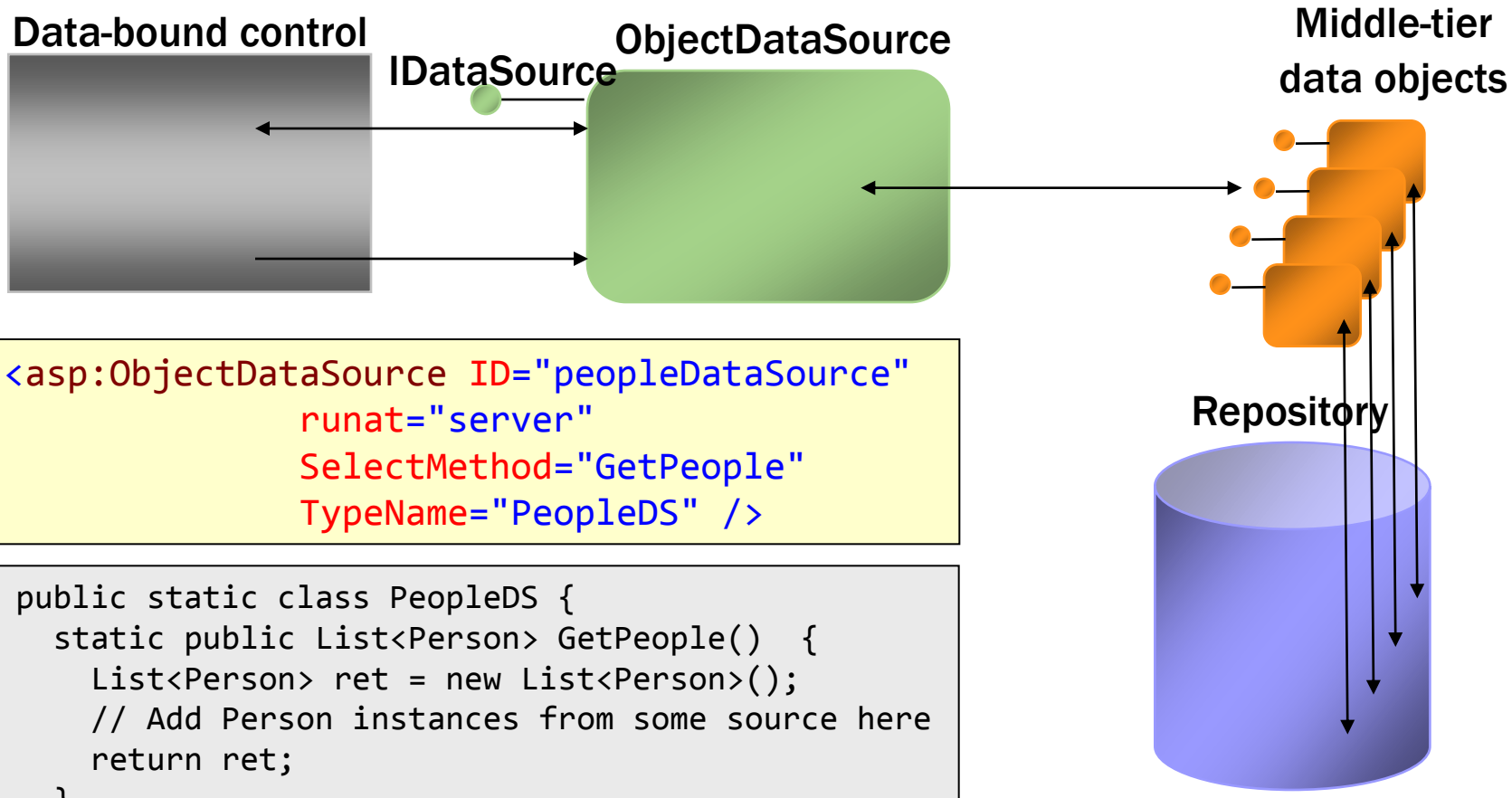
Nesting hierarchies

- **The XPathSelect data binding expression enables nested binding**
  - Useful for binding XmlDataSources to rectangular controls

```
<asp:DataList ID="DataList1" runat="server"
              DataSourceID="XmlDataSource1">
  <ItemTemplate>
    Title: <%# Eval("Title") %><br />
    <asp:DataList runat="server" ID="nestedDastaList"
         DataSource='<%# XPathSelect("chapter") %>'>
      <ItemTemplate>
        <h4>Chapternum: <%# XPath("@num") %></h4>
        <h4>Chapter name: <%# XPath("@name") %></h4>
        <%# XPath(".") %>
        <br />
      </ItemTemplate>
    </asp:DataList>
    <br />
  </ItemTemplate>
</asp:DataList>
```

Binding to objects

- **ObjectDataSource supports binding to middle-tier objects**




```
<asp:ObjectDataSource ID="peopleDataSource"
                runat="server"
                SelectMethod="GetPeople"
                TypeName="PeopleDS" />
```

```
public static class PeopleDS {
  static public List<Person> GetPeople()  {
    List<Person> ret = new List<Person>();
    // Add Person instances from some source here
    return ret;
  }
}
```

ObjectDataSource properties

- **Number of 'helper' properties that let you tell the data source how to use your methods**

```
public class ObjectDataSource : DataSourceControl
 {
   public string TypeName { get; set; }
   public string DataObjectTypeName { get; set; }

   public string DeleteMethod { get; set; }
   public string InsertMethod { get; set; }
   public string SelectCountMethod { get; set; }
   public string SelectMethod { get; set; }
   public string SortParameterName { get; set; }
   public string UpdateMethod { get; set; }
   public string MaximumRowsParameterName { get; set; }
   public string StartRowIndexParameterName { get; set; }
   //...
```

Sample ObjectDataSource class

```
 namespace EssentialAspDotNet2.DataBinding
{
  public static class MovieReviewsData
  {
    public static ICollection<Movie> GetMovies()
    { /*… */ }

    public static void UpdateMovie(Movie m)
    { /* … */ }

    public static void DeleteMovie(Movie m)
    { /* … */ }

    public static void InsertMovie(Movie m)
    { /* … */ }
  }
}
```

Wiring up a data access layer

```
<asp:ObjectDataSource ID="_moviesObjectDataSource" runat="server"
          TypeName="EssentialAspDotNet2.DataBinding.MovieReviewsData"
          DataObjectTypeName="EssentialAspDotNet2.DataBinding.Movie"
          DeleteMethod="InsertMovie"
          InsertMethod="InsertMovie"
          SelectMethod="GetMovies"
          UpdateMethod="UpdateMovie" />
```

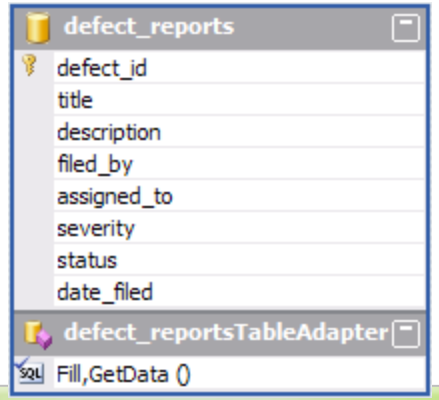
Must specify entity class (if in use)

```
<asp:GridView ID="_moviesGrid" runat="server" AllowPaging="True"
             DataSourceID="_moviesObjectDataSource"
             DataKeyNames="MovieId" />
```

Must specify primary key(s) – retained in control state for update/insert/delete

Binding to typed DataSets

- **ASP.NET 2.0 supports TableAdapter**
  - Generated as part of typed DataSet
  - Generates Fill / GetData methods for DataSet
- **Typed DataSets bound using ObjectDataSource**
  - Data source sees when return type is DataTable
  - Sorting, filtering, paging automatically enabled



Typed DataSets

- **Example of interacting with a strongly-typed DataSet**
  - Note that the TableAdapter is now strongly typed as well

```
defect_reportsTableAdapter drta =
                            new defect_reportsTableAdapter();
Defects.defect_reportsDataTable dt =
                            new Defects.defect_reportsDataTable();

drta.Fill(dt);

foreach (Defects.defect_reportsRow r in dt.Rows)
    Response.Output.Write("<h3>{0} {1:d} {2}</h3><br />",
                          r.title, r.date_filed, r.filed_by);
```

Summary

- **DataSource parameters**
- **Two-way data binding**
  - Support for populating parameters from controls
- **FormView**
  - Like DetailsView but completely template-driven
- **ListView, DataPager**
  - Powerful databinding controls that fit many scenarios
- **Hierarchical data binding**
- **Binding to objects**
- **Typed DataSets**