**Introduction**

The Repeater control is a really simple and powerful control which renders collection data into a web page within a short time of coding. Imagine you want to do the same in classic ASP or PHP, you may have to write a loop in ASP with combined HTML and server scripts.

Since we are going to **create** a control like DataRepeater, planning about the control is easy. I am going to change only the name of the control. The name of the control is MyRepeater, which consists of the following four templates:

* Header template
* Footer template
* Item template
* Alternate item template

As we already know, the Header and Footer should be rendered once, and the count of Item template and AlternateItemTemplate should be equal to the number of items in the DataSource.

**Planning the DataSource**

Like Repeater, our control should also support most of the generic types, like the ones given below:

* DataTable
* DataView
* List
* Collection
* ArrayList
* Array

All the above data types internally implement IEnumerable, so if we develop a control for IEnumerable, that will support all the above data sources. We are going to use the following important data types to develop our own Repeater:

* ITemplate
* IEnumerable
* **INamingContainer**
* IDataItem
* Control

**Creating the User Control**

**Create** a new user control named *MyRepeater.ascx*, and refer the same in *Default.aspx*. So *default.aspx* & *MyRepeater.ascx* will look like they are shown below:

**MyRepeater.ascx**

<%@ Control Language="C#" AutoEventWireup="true" **CodeFile="MyRepeater.ascx.cs"** Inherits="MyRepeater" %>

**Default.aspx**

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

**<%@ Register Src="MyRepeater.ascx" TagName="MyRepeater" TagPrefix="myOwn" %>**

<!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>Untitled Page</title>

</head>

<body>

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

<div>

**<myOwn:MyRepeater ID="MyRepeater1" runat="server"></myOwn:MyRepeater>**

</div>

</form>

</body>

</html>

**Creating Templates**

We have to **create** public properties to enable templates for the UserControl. **Create** the following four public properties inside the UserControl.

**MyRepeater.ascx.cs**

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/KB/user-controls/TemplatedControl.aspx)

using System;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.UI.HtmlControls;

using System.Collections;

public partial class MyRepeater : System.Web.UI.UserControl

{

private ITemplate \_ItemTemplate;

private Itemplate \_AlternateItemTemplate;

private ITemplate \_HeaderTemplate;

private ITemplate \_FooterTemplate;

private IEnumerable \_DataSource;

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

{

}

[TemplateContainer(typeof(SimpleTemplateItem))]

public ITemplate ItemTemplate

{

get { return \_ItemTemplate; }

set { \_ItemTemplate = value; }

}

[TemplateContainer(typeof(SimpleTemplateItem))]

public ITemplate AlternateItemTemplate

{

get { return \_AlternateItemTemplate; }

set { \_AlternateItemTemplate = value; }

}

[TemplateContainer(typeof(SimpleTemplateItem))]

public ITemplate HeaderTemplate

{

get { return \_HeaderTemplate; }

set { \_HeaderTemplate = value; }

}

[TemplateContainer(typeof(SimpleTemplateItem))]

public ITemplate FooterTemplate

{

get { return \_FooterTemplate; }

set { \_FooterTemplate = value; }

}

}

* ITemplate: This is an interface provided by ASP.NET, which is used to hold the specific template content.
* TemplateContainer: This attribute is used to provide the type of the template content.

For each template, runtime must **create** a UserControl instance to hold the contents, for instance, if your DataSource has 100 records, internally 100 instances of templates will be **create**d. We need to **create** another UserControl to render each template into that. This control is called the template container control. **Create** the class/control as follows:

**SimpleTemplateItem.cs**

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/KB/user-controls/TemplatedControl.aspx)

public class **SimpleTemplateItem** : Control, System.Web.UI.**INamingContainer**, IDataItemContainer

{

private object \_CurrentDataItem;

public SimpleTemplateItem(object currentItem)

{

\_CurrentDataItem = currentItem;

}

#region IDataItemContainer Members

public object DataItem

{

get { return \_CurrentDataItem; }

}

public int DataItemIndex

{

get { throw new Exception

("The method or operation is not implemented."); }

}

public int DisplayIndex

{

get { throw new Exception

("The method or operation is not implemented."); }

}

#endregion

}

* **INamingContainer**: This is a Marker interface � it doesn't have any methods, or properties inside � this is just **to provide control id in the runtime**.
* IDataItem: Properties of IDataItem are used to hold the single item of the DataSource with respect to the template instance.

**Adding Code to Render the Templates**

We **create**d

* a UserControl with 4 template properties
* and another UserControl (template container) to render the templates on it.

Now we need to add the code logic to render the templates. ITemplate interface internally has a method called InstantiateIn, which is used to render the template content into the desired control. Add the following codes into the databind method of myrepeater class, which will add the templates to the UserControl.

**MyRepeater.ascx.cs [Databind Method]**

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/KB/user-controls/TemplatedControl.aspx)

public override void DataBind()

{

*//Rendering Header template into current control*

AddTemplateAsControl(HeaderTemplate, null);

IEnumerator ie = DataSource.GetEnumerator();

bool renderAlternateTemplate = false;

while (ie.MoveNext())

{

if (renderAlternateTemplate && AlternateItemTemplate != null)

{

AddTemplateAsControl(ItemTemplate, ie.Current);

}

else if (AlternateItemTemplate != null)

{

AddTemplateAsControl(AlternateItemTemplate,ie.Current);

}

else

{

*//don't render anything*

}

renderAlternateTemplate = !renderAlternateTemplate;

}

*//Rendering footer template into current control*

AddTemplateAsControl(FooterTemplate, null);

*//Always better to call base class implementation*

base.DataBind();

}

private void AddTemplateAsControl(ITemplate anyTemplate,object cuurentItem)

{

SimpleTemplateItem templateContentHolder = new SimpleTemplateItem(cuurentItem);

anyTemplate.InstantiateIn(templateContentHolder);

this.Controls.Add(templateContentHolder);

}

**Testing MyRepeater**

To test the Repeater, **create** any DataSource and bind to the UserControl. Below is the sample code to test the Repeater.

**Default.aspx**

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/KB/user-controls/TemplatedControl.aspx)

<myOwn:MyRepeater ID="MyRepeater1" runat="server">

<HeaderTemplate>

Customer List:

<table border="1">

<tr><th>Id</th><th>Name</th><th>Location</th></tr>

</HeaderTemplate>

<ItemTemplate>

<tr style="background-color:gold">

<td><%#DataBinder.Eval(Container.DataItem,"Id") %></td>

<td><%#DataBinder.Eval(Container.DataItem,"Name")%></td>

<td><%#DataBinder.Eval(Container.DataItem,"Location") %></td>

</tr>

</ItemTemplate>

<AlternateItemTemplate>

<tr><td><%#DataBinder.Eval(Container.DataItem,"Id") %></td>

<td><%#DataBinder.Eval(Container.DataItem,"Name")%></td>

<td><%#DataBinder.Eval(Container.DataItem,"Location") %></td>

</tr>

</AlternateItemTemplate>

<FooterTemplate>

</table>

</FooterTemplate>

</myOwn:MyRepeater>

**Default.aspx.cs**

http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/KB/user-controls/TemplatedControl.aspx)

using System;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.UI.HtmlControls;

using System.Collections.Generic;

public partial class \_Default : System.Web.UI.Page

{

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

{

List<**Customer**> customerList = new List<**Customer**>();

customerList.Add(new Customer("Prem", "New york"));

customerList.Add(new Customer("Jhon", "Amsterdam"));

customerList.Add(new Customer("Peter", "London"));

customerList.Add(new Customer("Mani", "Chennai"));

customerList.Add(new Customer("Paul", "Paris"));

MyRepeater1.DataSource = customerList;

MyRepeater1.DataBind();

}

}

public class **Customer**

{

private string \_Name;

private Guid \_Id;

private string \_Location;

public Customer(string name, string location)

{

this.Name = name;

this.Location = location;

this.Id = Guid.NewGuid();

}

public string Name

{

get { return \_Name; }

set { \_Name = value; }

}

public Guid Id

{

get { return \_Id; }

set { \_Id = value; }

}

public string Location

{

get { return \_Location; }

set { \_Location = value; }

}

}

**Conclusion**

I hope you understood how to develop your own Templated control like DataGrid, DataList etc. with the addition of your own features. Instead of the SimpleTemplateItem class, you can also use the existing RepeaterItem class if you do not require any additional features. The control that we **create**d is very lightweight, like a normal Repeater. If you want, you can also add some extra features like paging, sorting, etc.