SharePoint 2013 Custom Editor Part

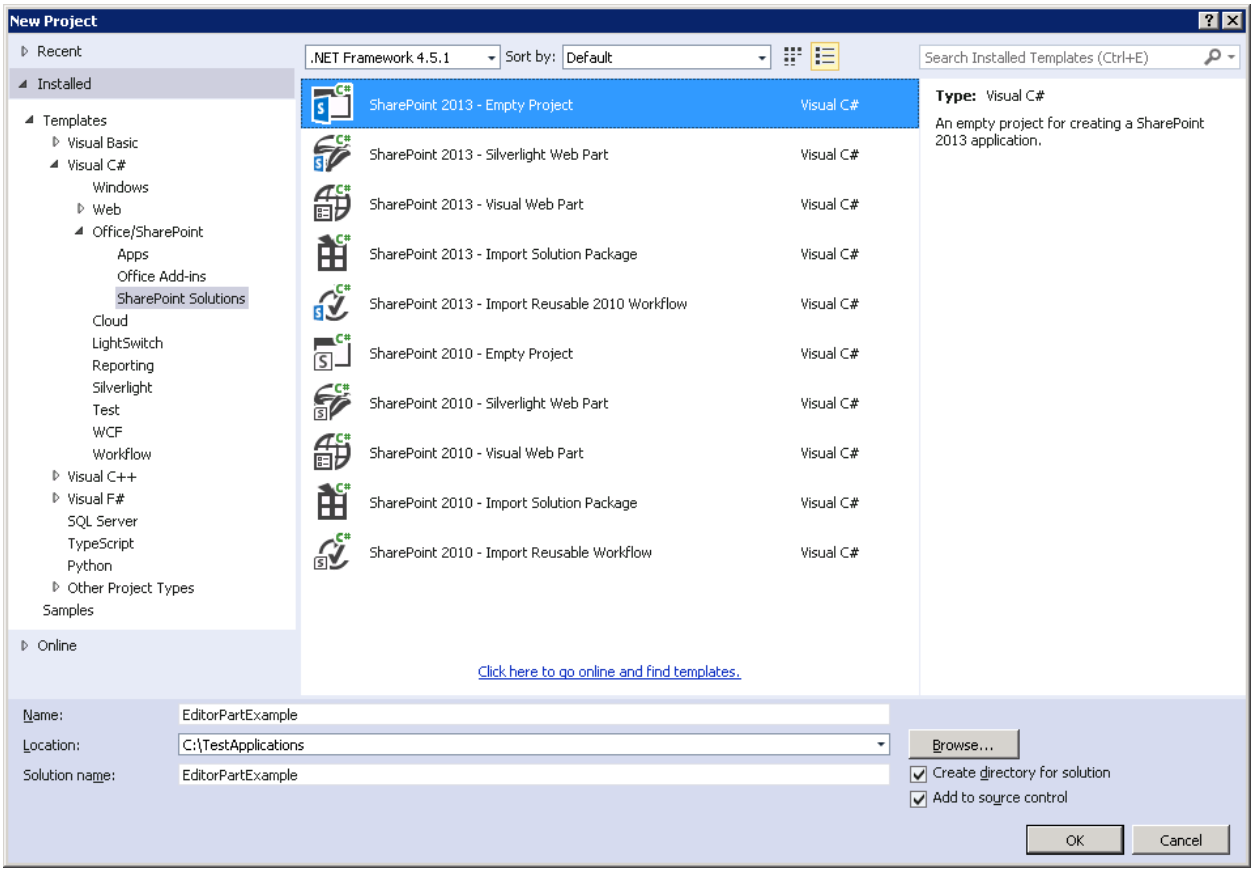
In SharePoint 2013 when you need to use ASP.NET controls for custom settings in the tool part pane for your WebPart which inherits from System.Web.UI.WebControls.WebParts.WebPart, we would need to create an EditorPart to achieve it.

Note:

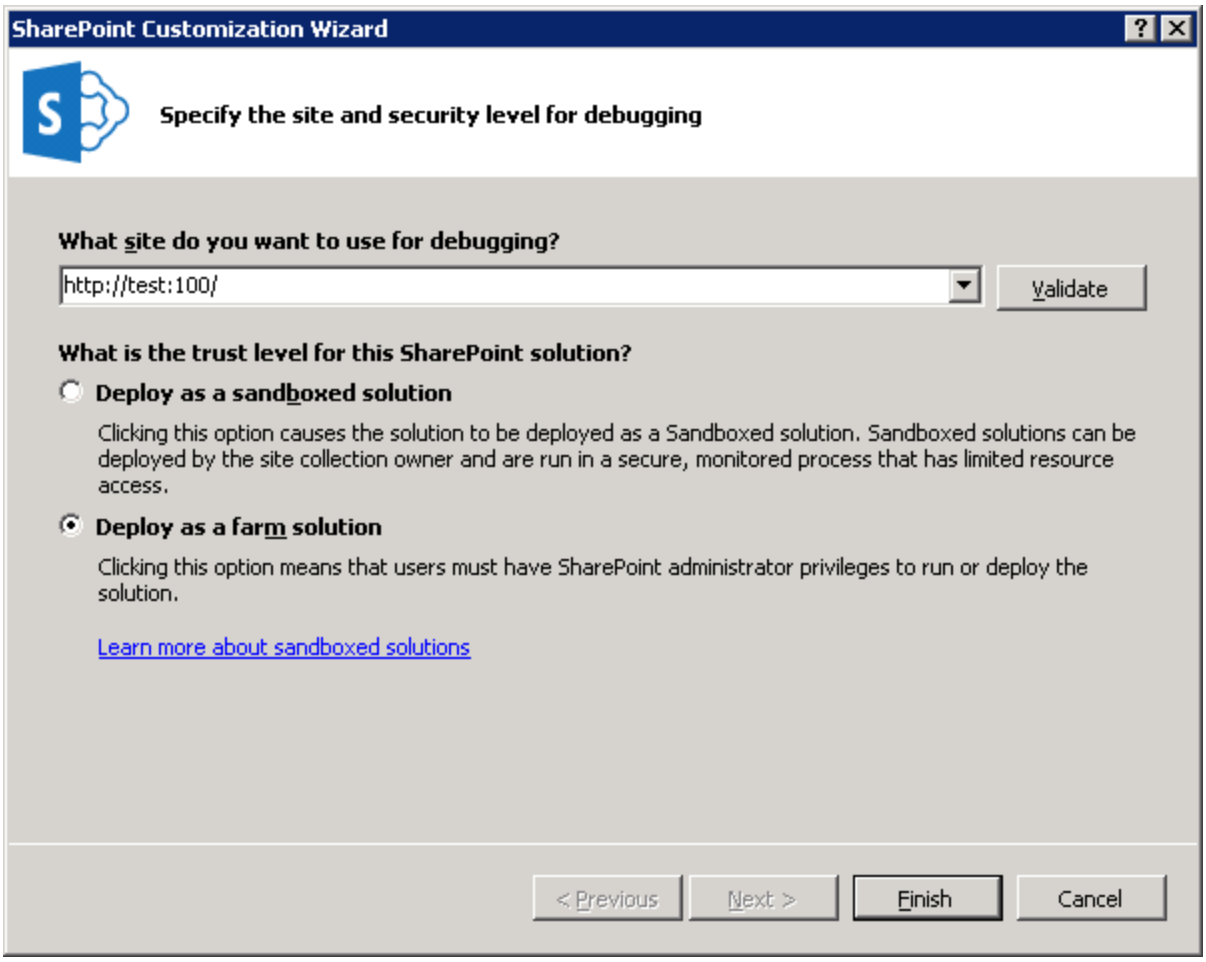
If the custom WebPart inherits from Microsoft.SharePoint.WebPartPages.WebPart then a ToolPart needs to be created.

This blog shows how a CheckBoxList control within EditorPart is used in the tool part pane to do some configuration.

Create a SharePoint 2013 - Empty Project



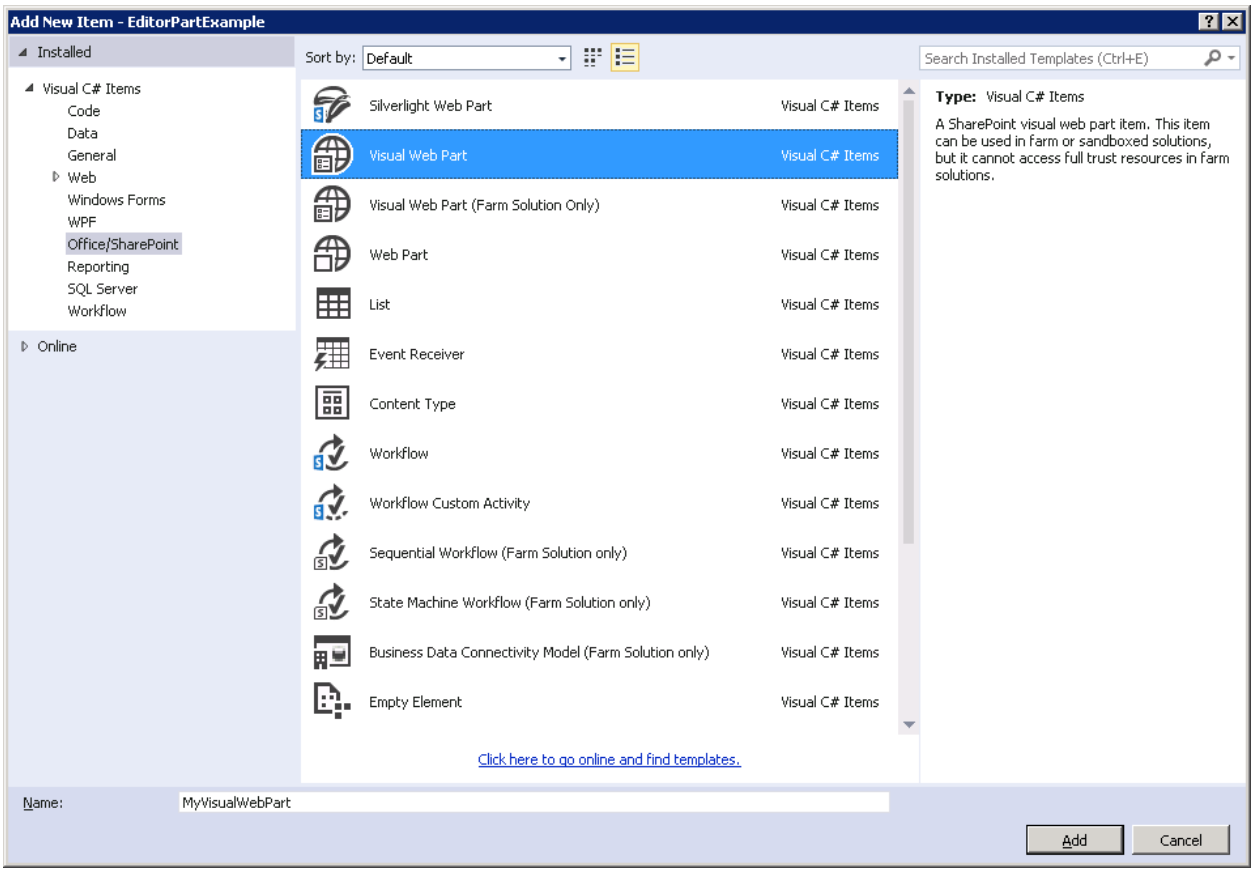
Click OK and choose a site for debugging



Validate to verify that there are no connection errors; choose the trust level as farm solution and click on Finish.

Add a new Item

Right click the project and add new item, choose Visual Web Part, provide a name and click on Add.



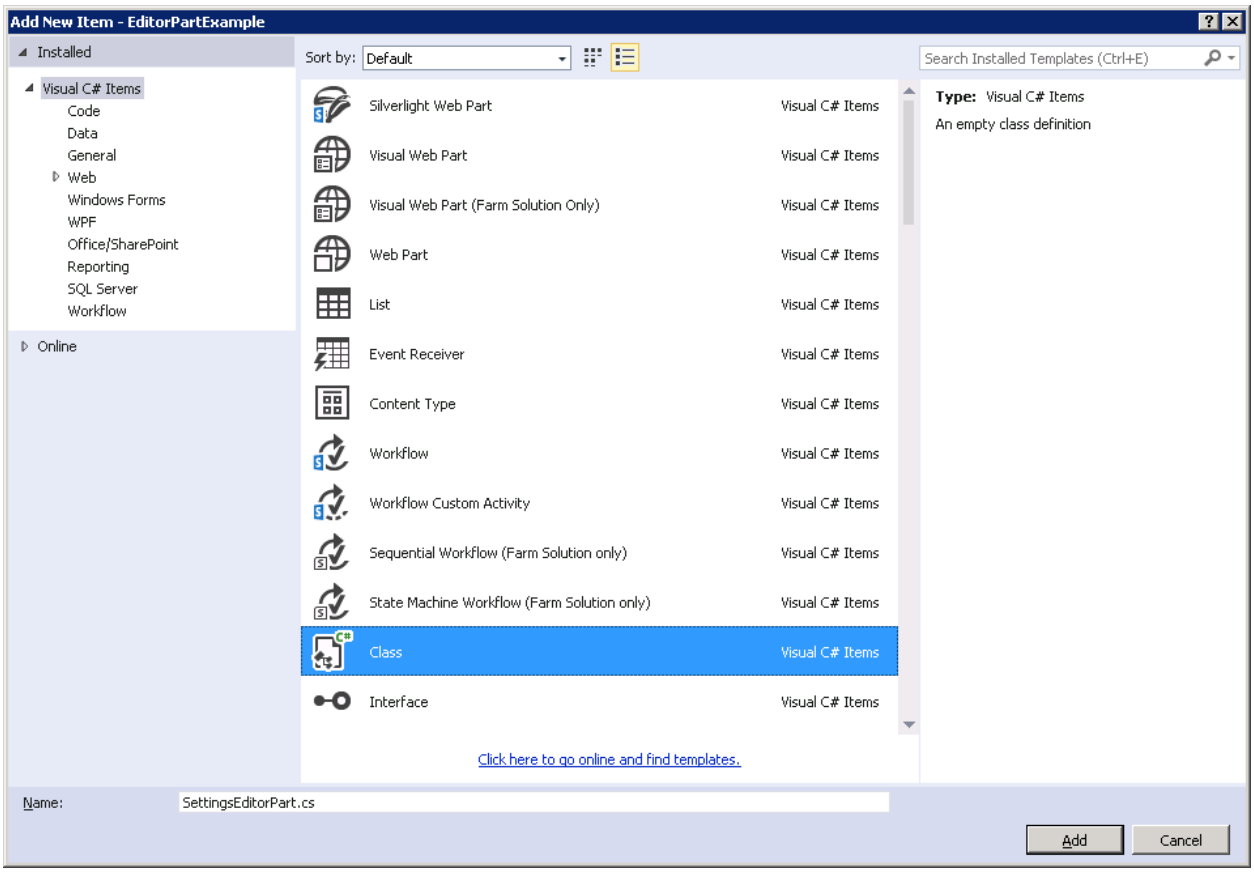
Add a GridView

Open the MyVisualWebPart.ascx, add a GridView as follows

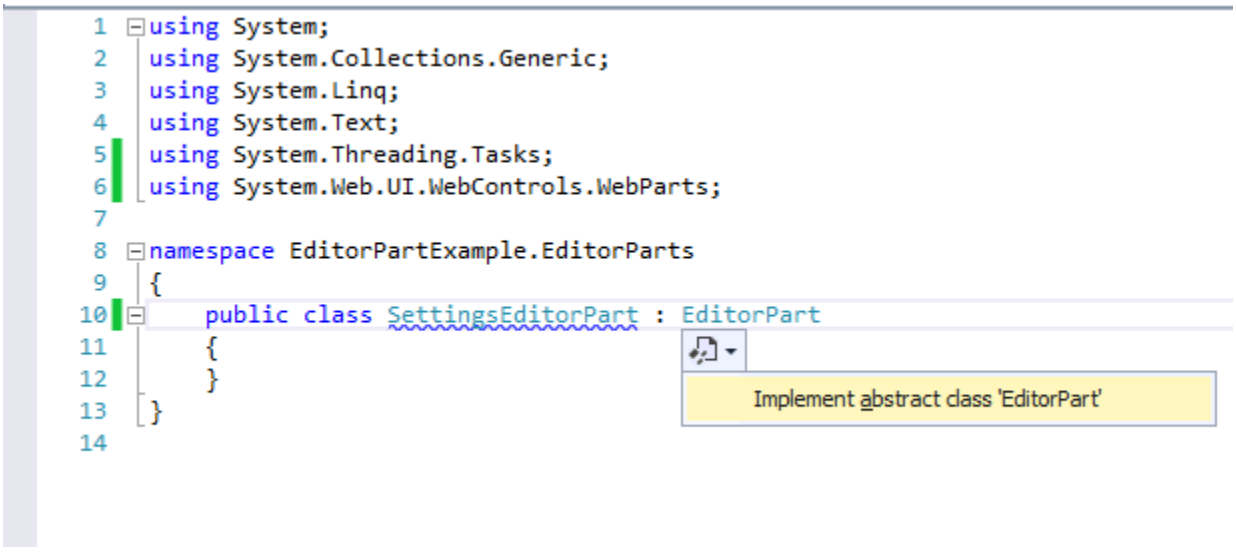
```
<asp:Panel runat="server" ID="Wrapper">
        <asp:GridView runat="server" ID="DocumentsGridView"
        AutoGenerateColumns="false" ShowHeaderWhenEmpty="true" CellPadding="10"
        BorderColor="Black" BorderStyle="Solid" BorderWidth="1">
                <Columns>
                        <asp:BoundField HeaderText="Name" DataField="Name" />
                        <asp:BoundField HeaderText="Modified" DataField="Modified" />
                </Columns>
        </asp:GridView>
</asp:Panel>
```

Add an EditorPart

Add a new folder to the project and name it as EditorParts and add a new class to it.



Inherit the class from EditorPart, implement abstract class EditorPart.



ApplyChanges() is used to apply the settings from custom settings to the Web Part and SyncChanges() is used to retrieve the current custom settings from the Web Part into the custom settings pane.

Create a CheckBoxList

To keep things simple, in SettingsEditorPart.cs we will create a CheckBoxList control to display a site (site URL is hardcoded) and its sub sites that are accessible to the user, from where the Web Part is going to pull the documents from all the Document Libraries.

Now declare a Panel & a CheckBoxList object, as follows

```
#region Fields

private Panel settingsPanel;

private CheckBoxList sitesCBL;

#endregion
```

Override CreateChildControls() method and add the following code in order to add the controls to our custom EditorPart.

```
protected override void CreateChildControls()
{
        this.Controls.Clear();
        this.settingsPanel = new Panel();
        this.sitesCBL = new CheckBoxList();
        this.sitesCBL.Items.AddRange(this.GetUrls().ToArray());
        this.settingsPanel.Controls.Add(this.sitesCBL);
        this.Controls.Add(this.settingsPanel);
        base.CreateChildControls();
}
```

Retrieve data from Document Libraries

Now add the GetUrls() method, as follows to retrieve items from Document Libraries.

```
private List<ListItem> GetUrls()
{
        List<ListItem> listItems = new List<ListItem>();
        try
        {
                using (SPSite site = new SPSite("http://vpntest:22000/"))
                {
                        SPWebCollection webs = site.RootWeb.GetSubwebsForCurrentUser();
```

```
                    List<SPWeb> spWebs = webs.ToList();
                    spWebs.Add(site.RootWeb);
                    for (int i = 0; i < spWebs.Count; i++)
                    {
                        using (SPWeb web = spWebs[i])
                        {
                            SPListCollection lists =
                            web.GetListsOfType(SPBaseType.DocumentLibrary);
                            foreach (SPList list in lists)
                            {
                                if (list.BaseTemplate ==
                                SPListTemplateType.DocumentLibrary &&
                                list.Title != "Style Library")
                                {
                                    List<SPListItem> total = (from SPListItem
                                    item in list.Items where item["Created
                                    By"].ToString() ==
                                    SPContext.Current.Web.CurrentUser.ID +
                                    ";#" +
                                    SPContext.Current.Web.CurrentUser.Nam
                                    e select item).ToList();
                                    if (total.Any())
                                    {
                                        ListItem listItem = new ListItem
                                        {
                                            Text = web.Title,
                                            Value = web.ID.ToString()
                                        };

                                        listItem.Attributes["Title"] =
                                        web.Url;
                                        listItems.Add(listItem);

                                        break;
                                    }
                                }
                            }
                        }
                    }
                }
                catch
                {
                    throw;
                }

                return listItems;
            }
```

Create settings model

While creating child controls we need to sync data from Web Part to Custom Editor Part for this we need a Serializable model. Serializable model is required in order to save our custom settings to the database.

Create a folder called Models and create a class called CustomListItemModel.cs and add the following code.

```
[Serializable]
public class CustomListItemModel
{
        [DefaultValue(false)]
        public bool Selected { get; set; }

        [DefaultValue("")]
        [Localizable(true)]
        public string Value { get; set; }

        public string WebUrl { get; set; }
}
```

Now open MyVisualWebPart.ascx file and add the following code.

```
#region Properties

[WebBrowsable(false), Personalizable(PersonalizationScope.Shared)]
public List<CustomListItemModel> SiteUrls { get; set; }

public override object WebBrowsableObject
{
        get
        {
                return this;
        }
}

#endregion
```

In the constructor instantiate the SiteUrls object, if PersonalizationScope is set to Shared, then same settings are applied to all users, else if set to User, then each user can apply their own settings.

```
public MyVisualWebPart()
{
        this.SiteUrls = new List<CustomListItemModel>();
}
```

Creating behavior to update settings

Now back to SettingsEditorPart.cs, create a method called ApplyOrSyncChanges() to perform the sync/apply any changes, which can be invoked in CreateChildControls() method, as well as in ApplyChanges() and SyncChanges() methods.

```
private void ApplyOrSyncChanges(string mode = "Sync")
{
        try
        {
                using (MyVisualWebPart.MyVisualWebPart webPart =
                (MyVisualWebPart.MyVisualWebPart)this.WebPartToEdit)
                {
                        if (webPart != null)
                        {
                                if (mode == "Sync")
                                {
                                        for (int i = 0; i < webPart.SiteUrls.Count; i++)
                                        {
                                                for (int j = 0; j < this.sitesCBL.Items.Count;
                                                j++)
                                                {
                                                        // SPWeb's ID is stored in Value property
                                                        if (webPart.SiteUrls[i].Value ==
                                                        this.sitesCBL.Items[j].Value)
                                                        {
                                                                this.sitesCBL.Items[j].Selected =
                                                                webPart.SiteUrls[i].Selected;

                                                                break;
                                                        }
                                                }
                                        }
                                }
                                else
                                {
                                        for (int j = 0; j < this.sitesCBL.Items.Count; j++)
                                        {
                                                for (int i = 0; i < webPart.SiteUrls.Count; i++)
                                                {
                                                        // SPWeb's ID is stored in Value property
                                                        if (webPart.SiteUrls[i].Value ==
                                                        this.sitesCBL.Items[j].Value)
                                                        {
                                                                webPart.SiteUrls[i].Selected =
                                                                this.sitesCBL.Items[j].Selected;
                                                                break;
                                                        }
                                                }
                                        }
                                }
                        }
                }
```

```
        }
        catch
        {
                throw;
        }
}
```

In ApplyOrSyncChanges() method we are type casting WebPartToEdit to our Web Part object and if mode is "Sync" then we are getting the values from the Web Part and marking the items as checked/unchecked in the CheckBoxList in the custom EditorPart and when the mode is not "Sync" then we are doing the reverse of it.

Now call this ApplyOrSyncChanges() method in CreateChildControls() method right before the base.CreateChildControls(); statement

```
// code omitted for brevity
this.ApplyOrSyncChanges();

base.CreateChildControls();
// code omitted for brevity
```

Default settings

When there are no settings saved yet then we need an initial stage for the CheckBoxList control such that all items are checked by default and the same data is copied to the Web Part's model, for this we are going to use the following method

```
private void SetDefaultSettings()
{
        try
        {
                using (MyVisualWebPart.MyVisualWebPart webPart =
                (MyVisualWebPart.MyVisualWebPart)this.WebPartToEdit)
                {
                        for (int i = 0; i < this.sitesCBL.Items.Count; i++)
                        {
                                this.sitesCBL.Items[i].Selected = true;
                                webPart.SiteUrls.Add(new CustomListItemModel
                                {
                                        Selected = true,
                                        Value = this.sitesCBL.Items[i].Value,
                                        WebUrl = this.sitesCBL.Items[i].Attributes["Title"]
                                });
                        }
                }
        }
        catch
        {
```

```
                throw;
        }
}
```

Invoke the above SetDefaultSettings() method in CreateChildControls() method right after the statement - this.sitesCBL.Items.AddRange(this.GetUrls().ToArray()); as shown below

```
using (MyVisualWebPart.MyVisualWebPart webPart =
(MyVisualWebPart.MyVisualWebPart)this.WebPartToEdit)
{
    if (webPart != null && webPart.SiteUrls.Count == 0)
    {
        this.SetDefaultSettings();
     }
}
```

Invoking the behavior ApplyOrSyncChanges() to update settings

Update ApplyChanges() and SyncChanges() as follows,

```
public override bool ApplyChanges()
{
        try
        {
                this.EnsureChildControls();
                this.ApplyOrSyncChanges("Apply");
        }
        catch
        {
                throw;
        }

          return true;
}

public override void SyncChanges()
{
        try
        {
                this.EnsureChildControls();
                this.ApplyOrSyncChanges();
        }
        catch
        {
                throw;
        }
}
```

Attaching EditorPart to tool part pane

Now we need to attach the custom EditorPart to our Web Part, by overriding CreateEditorParts() method

```
public override EditorPartCollection CreateEditorParts()
{
        EditorPartCollection editorPartCollection = null;
        try
        {
                EditorPart[] editorParts = new EditorPart[] { new SettingsEditorPart() };
                editorParts[0].ID = this.ID + "SettingsEditorPart";
                editorParts[0].Title = "Custom Settings";
                editorPartCollection = new EditorPartCollection(base.CreateEditorParts(),
                editorParts);
        }
        catch
        {
                throw;
        }

        return editorPartCollection;
}
```

Creating the GridView model

The process until now will save the settings now we need to use the saved settings to perform some task.
First we need a model to bind the data to the GridView in the Web Part and the GridView will be populated with some data filtered by using the data from CheckBoxList control from the custom EditorPart.

Create a class file, MyDocumentsModel.cs in Models folder.

```
public class MyDocumentsModel
{
        public string Name { get; set; }

        public string Modified { get; set; }
}
```

Create the property public List<MyDocumentsModel> MyDocuments { get; set; } in MyVisualWebPart.cs, to set data into the MyDocuments property and to bind it to the GridView.

Add behaviors to set GridView model data

Add the following four methods, to control the items being shown in the GridView based the settings applied in the custom EditorPart.

```
private void DataBindDocumentsGridView()
{
        try
```

```
        {
                this.MyDocuments = this.GetDocuments();
                this.DocumentsGridView.DataSource = this.MyDocuments;
                this.DocumentsGridView.DataBind();
        }
        catch
        {
                throw;
        }
}

private List<MyDocumentsModel> GetDocuments()
{
        List<MyDocumentsModel> data = null;
        try
        {
                if (this.SiteUrls.All(x => x.Selected == true))
                {
                        data = this.GetAllDocuments();
                }
                else if (this.SiteUrls.All(x => x.Selected == false))
                {
                        return null;
                }
                else
                {
                        data = this.GetDocumentsByFilter();
                }
        }
        catch
        {
                throw;
        }

        return data;
}

private List<MyDocumentsModel> GetAllDocuments()
{
        List<MyDocumentsModel> urls = new List<MyDocumentsModel>();
        try
        {
                using (SPSite site = new SPSite("http://vpntest:22000/"))
                {
                        SPWebCollection webs = site.RootWeb.GetSubwebsForCurrentUser();
                        List<SPWeb> spWebs = webs.ToList();
                        spWebs.Add(site.RootWeb);
                        for (int i = 0; i < spWebs.Count; i++)
                        {
                                using (SPWeb web = spWebs[i])
                                {
                                        SPListCollection lists =
```

```
                                    web.GetListsOfType(SPBaseType.DocumentLibrary);
                                    foreach (SPList list in lists)
                                    {
                                        if (list.BaseTemplate ==
                                        SPListTemplateType.DocumentLibrary &&
                                        list.Title != "Style Library")
                                        {
                                            List<SPListItem> total = (from SPListItem
                                            item in list.Items where item["Created
                                            By"].ToString() ==
                                            SPContext.Current.Web.CurrentUser.ID +
                                            ";#" +
                                            SPContext.Current.Web.CurrentUser.Nam
                                            e select item).ToList();
                                            foreach (var item in total)
                                            {
                                                urls.Add(new MyDocumentsModel
                                                {
                                                Name = item.Name,
                                                Modified = item["Modified"] != null
                                                ? item["Modified"].ToString() :
                                                string.Empty
                                                });
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                    catch
                    {
                        throw;
                    }

                    return urls;
                }

                private List<MyDocumentsModel> GetDocumentsByFilter()
                {
                    List<MyDocumentsModel> urls = new List<MyDocumentsModel>();
                    try
                    {
                        for (int i = 0; i < this.SiteUrls.Count; i++)
                        {
                            if (this.SiteUrls[i].Selected)
                            {
                                using (SPSite site = new SPSite(this.SiteUrls[i].WebUrl))
                                {
                                    using (SPWeb web = site.OpenWeb())
                                    {
                                        SPListCollection lists =
                                        web.GetListsOfType(SPBaseType.DocumentLibrar
```

```
                                        y);
                                        foreach (SPList list in lists)
                                        {
                                                if (list.BaseTemplate ==
                                                SPListTemplateType.DocumentLibrary &&
                                                list.Title != "Style Library")
                                                {
                                                        List<SPListItem> total = (from
                                                        SPListItem item in list.Items where
                                                        item["Created By"].ToString() ==
                                                        SPContext.Current.Web.CurrentUs
                                                        er.ID + ";#" +
                                                        SPContext.Current.Web.CurrentUs
                                                        er.Name select item).ToList();
                                                        foreach (var item in total)
                                                        {
                                                                urls.Add(new
                                                                MyDocumentsModel
                                                                {
                                                                Name = item.Name,
                                                                Modified = item["Modified"]
                                                                != null ?
                                                                item["Modified"].ToString()
                                                                : string.Empty
                                                                });
                                                        }
                                                }
                                        }
                                }
                        }
                }
        }
        catch
        {
                throw;
        }

        return urls;
}
```

The GetAllDocuments() method is similar to GetUrls() method from SettingsEditorPart.cs except that here we are looping through the list items created by the user and adding them to our model i.e. MyDocuments.

The GetDocumentsByFilter() method is adding the list items created by the user from the filtered choice present in SiteUrls property instead of looping through all sites in the specified site collection.

Invoke DataBindDocumentsGridView() in Page_Load() method

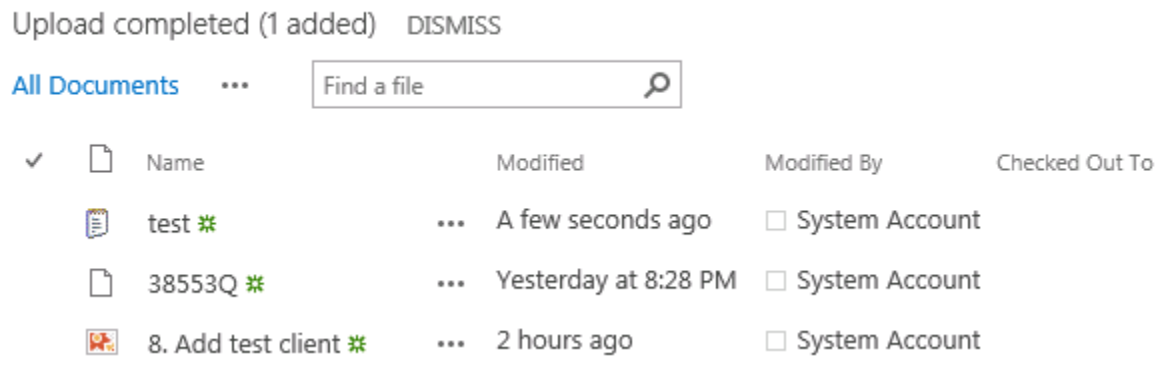
```
protected void Page_Load(object sender, EventArgs e)
{
        try
        {
                if (!this.Page.IsPostBack)
                {
                        this.DataBindDocumentsGridView();
                }
        }
        catch
        {
                throw;
        }
}
```

Deploy solution

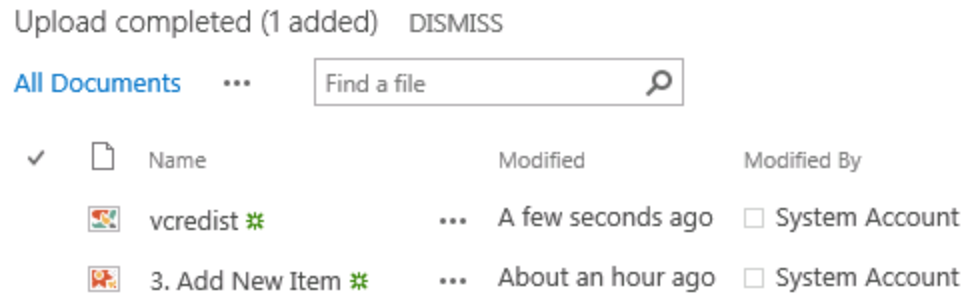
Now rebuild the project and deploy it.

The site and its sub site from where the documents are being pulled each has a document library, and there are some sample documents uploaded in these libraries by using the same user account with which the custom Web Part will be tested in order to retrieve the documents as we're retrieving them using "Created By" field.

The site's document library contains,

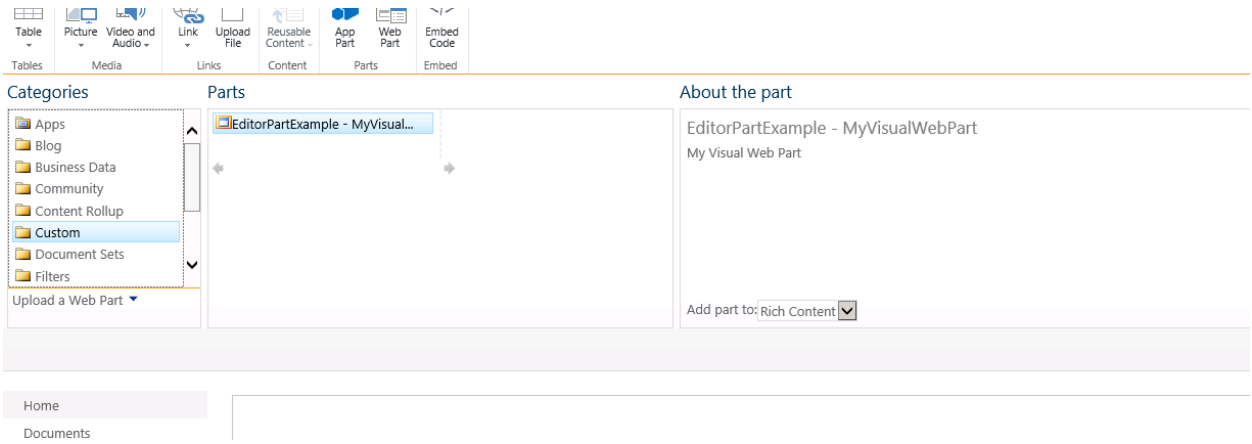


The sub site's document library contains,



Add WebPart to a Page

Go to the site where the Web Part was deployed, Edit the page (appropriate permissions are required to Edit a page), add the Web Part from the Custom category to a page and save the page.



Default WebPart state

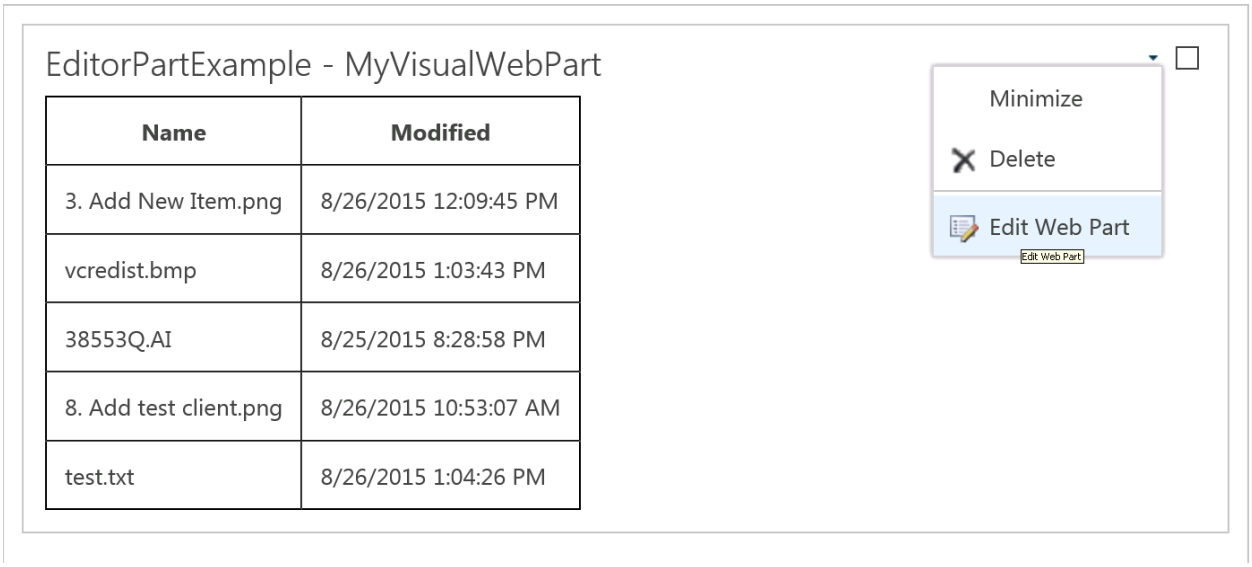
The following output is displayed.

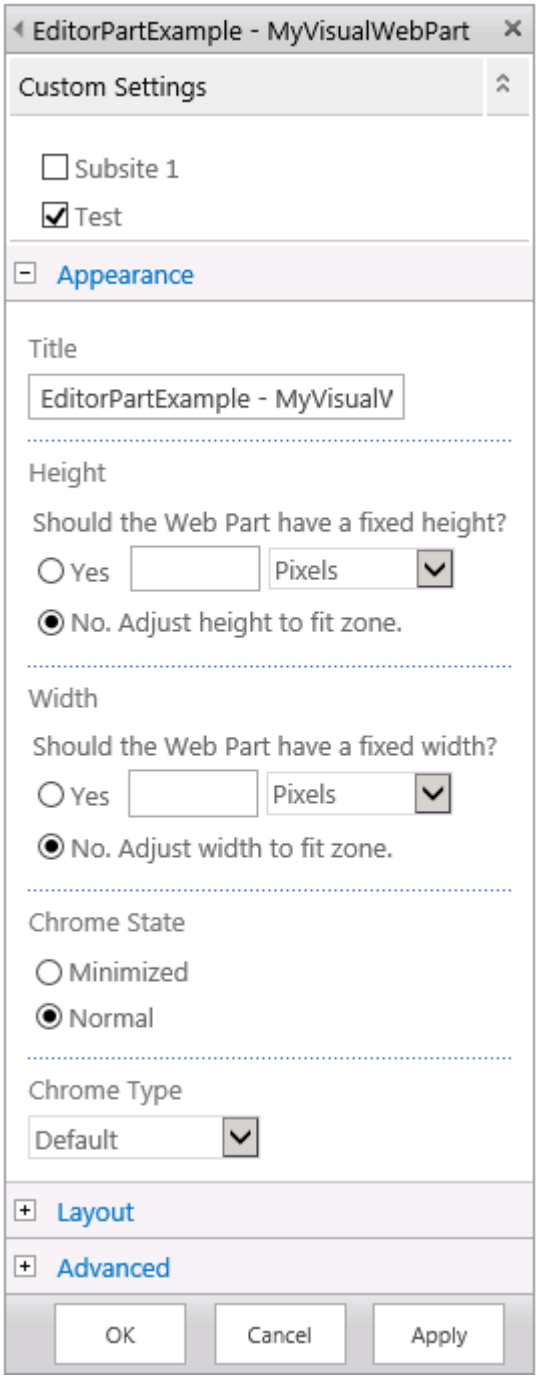
EditorPartExample - MyVisualWebPart

Name	Modified
3. Add New Item.png	8/26/2015 12:09:45 PM
vcredist.bmp	8/26/2015 1:03:43 PM
38553Q.AI	8/25/2015 8:28:58 PM
8. Add test client.png	8/26/2015 10:53:07 AM
test.txt	8/26/2015 1:04:26 PM

Filters results

To filter these results in the GridView, Edit the page and select our custom Web Part and click Edit Web Part





Uncheck any site in Custom Settings section and click on Ok and Save the page.

Filtered WebPart state

Now the results in the GridView are filtered.

EditorPartExample - MyVisualWebPart

Name	Modified
3. Add New Item.png	8/26/2015 12:09:45 PM
vcredist.bmp	8/26/2015 1:03:43 PM