Automatic Route All Pages in ASP.NET WebForms

adriancs
14 Nov 2022 CPOL

Easily route all pages at once in single line.

A quick and easy way to route all pages in ASP.NET WebForms

Download source code - 17 KB

Introduction

There are a few factors that affect the path of URL in ASP.NET.

In order to organize the pages, they are normally being grouped with folders.

Here is one of the typical URLs for a page.

```
/pages/admin/setup/user/v2/EditUser.aspx
```

The page "*EditUser.aspx*" located within the folder of "*v2*", which falls in the folder "*user*", which falls in the folder "*setup*", which falls in another folder "*admin*", which... again falls into yet another folder called "*pages*".

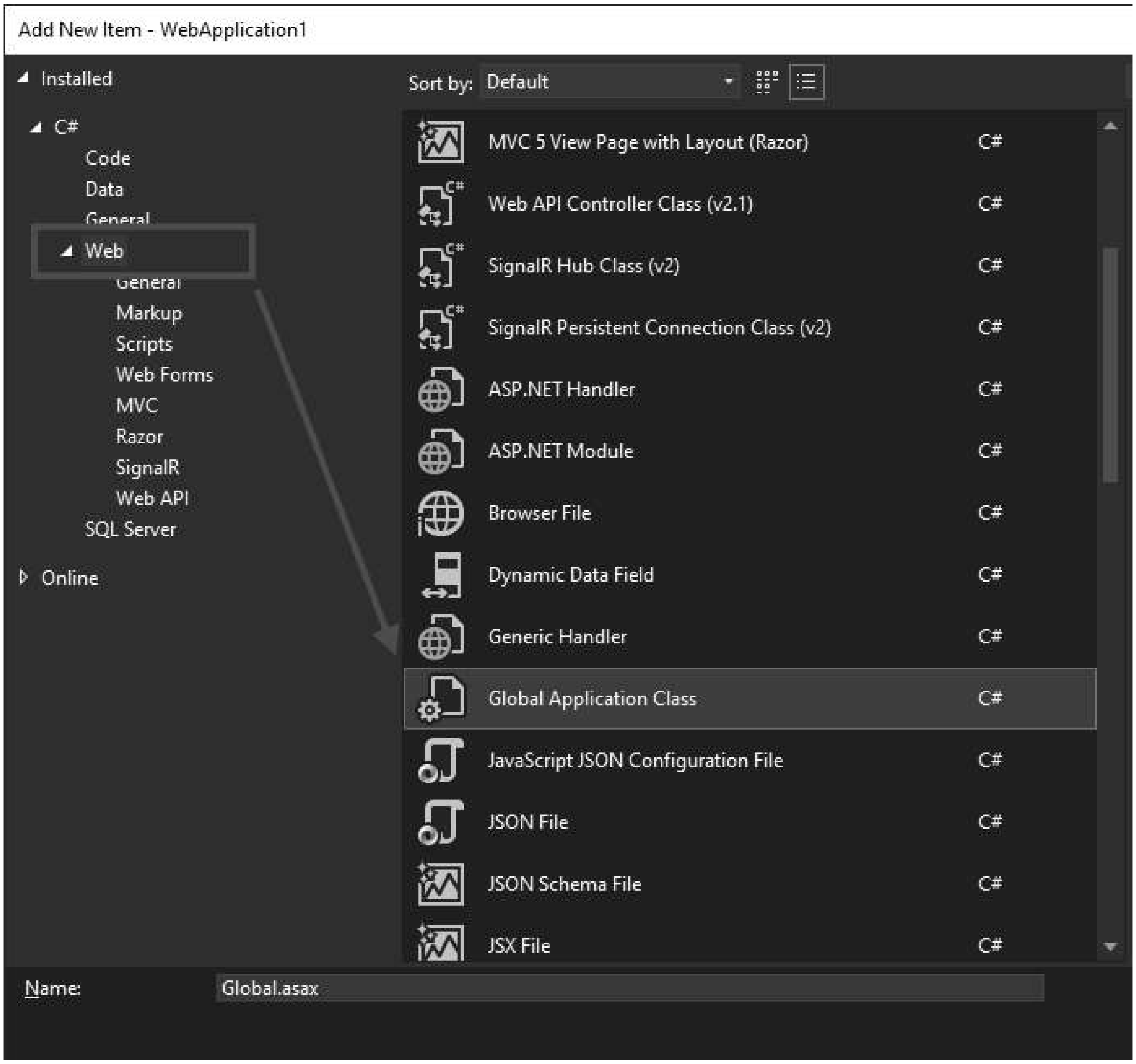
As the project grows larger, more and more root folders will be scattered all over within the projects.

Since the folder path ties strictly to the path of URL of the file, it will make the URL become very long.

ASP.NET provides a useful function call "**Routing**" that can solve the long URL problem.

Here's how it is implemented.

At the root of the project, add a "*Global Application Class*" file, so called "*Global.asax*".



Name: Global.asax

Open the file "*Global.asax*".

}

Import the library of routing by adding a `using` statement:

C#

```
using System.Web.Routing;
```

At the method `Application_Start`:

C#

```
protected void Application_Start(object sender, EventArgs e)
{

}
```

Type the routing code:

C#

```
protected void Application_Start(object sender, EventArgs e)
{
    RouteTable.Routes.MapPageRoute("EditUser", "EditUser",
                                    "~/admin/setup/user/v2/EditUser.aspx");
}
```

This will make the following URL:

```
/admin/setup/user/v2/EditUser.aspx
```

shorten to:

```
/EditUser
```

Looks good so far....

What if... there are a lots (yeah... a huge lots) of pages?

Well, ends up you are going to route them all one by one... for example:

```
RouteTable.Routes.MapPageRoute("UserEdit", "UserEdit",
"~/admin/setup/user/v2/UserEdit.aspx");

RouteTable.Routes.MapPageRoute("UserList", "UserList",
"~/admin/setup/user/v2/UserList.aspx");

RouteTable.Routes.MapPageRoute("UserCategory", "UserCategory",
"~/admin/setup/user/v2/UserCategory.aspx");

RouteTable.Routes.MapPageRoute("UserCategoryEdit", "UserCategoryEdit",
"~/admin/setup/user/v2/UserCategoryEdit.aspx");

RouteTable.Routes.MapPageRoute("InvoicePreview", "InvoicePreview",
"~/invoice/user/v3/InvoicePreview.aspx");

RouteTable.Routes.MapPageRoute("InvoiceSarch", "InvoiceSarch",
"~/invoice/user/v3/InvoiceSarch.aspx");

// ...
// continue for yet another lots of... lots of pages...
// ....
```

And yeah, you get the idea. That would be a hell lot of lines. The huge lines of routing code will make the maintenance work painful. Bugs can be hard to be monitored.

But, how about... route all the pages automatically in a single line?

Here Is What We Are Going to Do

Use only one single root folder to organize all pages.

For example, instead of having multiple root folders like this:

```
/settings
/user
/activity
/invoice
/member
/inventory
....
```

Put them altogether in a single folder like this:

```
/pages/settings
/pages/user
/pages/activity
/pages/invoice
/pages/member
/pages/inventory
```

Then, go back to the "*Global.asax*" file to code the routing commands.

At the `Application_Start` method, type the following routing command:

C#

```
protected void Application_Start(object sender, EventArgs e)
{
   RouteFolder("~/pages");
}
```

Here's the content of `RouteFolder` method:

C#

```
public static void RouteFolder(string folder)
{

}
```

Obtain the root directory path:

C#

```
public static void RouteFolder(string folder)
{
    string rootFolder = HttpContext.Current.Server.MapPath("~/");
}
```

Double confirm the parameter `folder` prefix with "~/":

C#

```
public static void RouteFolder(string folder)
{
    string rootFolder = HttpContext.Current.Server.MapPath("~/");

    if (folder.StartsWith("~/"))
    { }
    else if (folder.StartsWith("/"))
    {
        folder = "~" + folder;
    }
    else
    {
        folder = "~/" + folder;
    }
}
```

Start routing the folder in another separate method:

C#

```
public static void RouteFolder(string folder)
{
    string rootFolder = HttpContext.Current.Server.MapPath("~/");

    if (folder.StartsWith("~/"))
    { }
    else if (folder.StartsWith("/"))
    {
        folder = "~" + folder;
    }
    else
    {
        folder = "~/" + folder;
    }

    folder = HttpContext.Current.Server.MapPath(folder);

    MapPageRoute(folder, rootFolder);
}
```

The `MapPageRoute` method. First obtain the sub-folders:

C#

```
static void MapPageRoute(string folder, string rootFolder)
{
    // obtain sub-folders
    string[] folders = Directory.GetDirectories(folder);
}
```

Do a recursive loop for routing the sub-folders:

C#

```
static void MapPageRoute(string folder, string rootFolder)
{
    // obtain sub-folders
    string[] folders = Directory.GetDirectories(folder);

    foreach (var subFolder in folders)
    {
        MapPageRoute(subFolder, rootFolder);
    }
}
```

Get all the files within the folder:

C#

```
static void MapPageRoute(string folder, string rootFolder)
{
    // obtain sub-folders
    string[] folders = Directory.GetDirectories(folder);

    foreach (var subFolder in folders)
    {
        MapPageRoute(subFolder, rootFolder);
    }

    string[] files = Directory.GetFiles(folder);

    foreach (var file in files)
    {

    }
}
```

Loop through all the files, if the page is not an ASP.NET page, skip it.

C#

```
static void MapPageRoute(string folder, string rootFolder)
{
    // obtain sub-folders
    string[] folders = Directory.GetDirectories(folder);

    foreach (var subFolder in folders)
    {
        MapPageRoute(subFolder, rootFolder);
    }

    string[] files = Directory.GetFiles(folder);

    foreach (var file in files)
    {
        if (!file.EndsWith(".aspx"))
            continue;
    }
}
```

Obtain the web URL path of the file:

C#

```
static void MapPageRoute(string folder, string rootFolder)
{
    // obtain sub-folders
    string[] folders = Directory.GetDirectories(folder);

    foreach (var subFolder in folders)
    {
        MapPageRoute(subFolder, rootFolder);
    }

    string[] files = Directory.GetFiles(folder);

    foreach (var file in files)
    {
        if (!file.EndsWith(".aspx"))
            continue;

        string webPath = file.Replace(rootFolder, "~/").Replace("\\", "/");
    }
}
```

Example: The following file path:

```
file = "D:\wwwroot\website1\pages\user\UserProfile.aspx"
```

will become:

```
webPath = "~/pages/user/UserProfile.aspx"
```

Next, obtain the filename:

C#

```
static void MapPageRoute(string folder, string rootFolder)
{
    // obtain sub-folders
    string[] folders = Directory.GetDirectories(folder);

    foreach (var subFolder in folders)
    {
        MapPageRoute(subFolder, rootFolder);
    }
    string[] files = Directory.GetFiles(folder);

    foreach (var file in files)
    {
        // not a page, skip action
        if (!file.EndsWith(".aspx"))
            continue;

        string webPath = file.Replace(rootFolder, "~/").Replace("\\", "/");

        var filename = Path.GetFileNameWithoutExtension(file);
    }
}
```

Output:

```
filename = "UserProfile"
```

Next, if it was a `default` page, skip it. Always put empty content for `default` page. Use `default` page as landing page for redirecting the user to correspondence/appropriate/real page.

C#

```
static void MapPageRoute(string folder, string rootFolder)
{
    // obtain sub-folders
    string[] folders = Directory.GetDirectories(folder);

    foreach (var subFolder in folders)
    {
        MapPageRoute(subFolder, rootFolder);
    }

    string[] files = Directory.GetFiles(folder);

    foreach (var file in files)
    {
        // not a page, skip action
        if (!file.EndsWith(".aspx"))
            continue;

        string webPath = file.Replace(rootFolder, "~/").Replace("\\", "/");

        var filename = Path.GetFileNameWithoutExtension(file);

        if (filename.ToLower() == "default")
        {
            continue;
        }
    }
}
```

Finally, do the routing:

C#

```csharp
static void MapPageRoute(string folder, string rootFolder)
{
    // obtain sub-folders
    string[] folders = Directory.GetDirectories(folder);

    foreach (var subFolder in folders)
    {
        MapPageRoute(subFolder, rootFolder);
    }

    string[] files = Directory.GetFiles(folder);

    foreach (var file in files)
    {
        // not a page, skip action
        if (!file.EndsWith(".aspx"))
            continue;

        string webPath = file.Replace(rootFolder, "~/").Replace("\\", "/");

        var filename = Path.GetFileNameWithoutExtension(file);

        if (filename.ToLower() == "default")
        {
            continue;
        }

        RouteTable.Routes.MapPageRoute(filename, filename, webPath);
    }
}
```

So now, instead of doing this:

C#

```csharp
protected void Application_Start(object sender, EventArgs e)
{
    RouteTable.Routes.MapPageRoute("UserEdit", "UserEdit",
    "~/pages/admin/setup/user/v2/UserEdit.aspx");
    RouteTable.Routes.MapPageRoute("UserList", "UserList",
    "~/pages/admin/setup/user/v2/UserList.aspx");
    RouteTable.Routes.MapPageRoute("UserCategory", "UserCategory",
    "~/pages/admin/setup/user/v2/UserCategory.aspx");
    RouteTable.Routes.MapPageRoute("UserCategoryEdit", "UserCategoryEdit",
    "~/pages/admin/setup/user/v2/UserCategoryEdit.aspx");
    RouteTable.Routes.MapPageRoute("InvoicePreview", "InvoicePreview",
    "~/pages/invoice/user/v3/InvoicePreview.aspx");
    RouteTable.Routes.MapPageRoute("InvoiceSarch", "InvoiceSarch",
    "~/pages/invoice/user/v3/InvoiceSarch.aspx");

    // ...
    // continue for yet another lots of... lots of pages...
    // ....
}
```

All you need is just one single line:

C#

```
protected void Application_Start(object sender, EventArgs e)
{
   RouteFolder("~/pages");
}
```

and Voila! It's done, like magic. All pages are routed without supervision (coding the route command 1 by 1).

All these pages with confusing, complicated URL path (due to organizing pages):

```
/pages/admin/setup/user/v2/EditUser.aspx
/pages/admin/system/settings/AppConfig.aspx
/pages/user/ViewUserProfile.aspx
/pages/activity/ActivityEventList.aspx
/pages/invoice/v3/PrintInvoice.aspx
/pages/member/group/EditMemberList.aspx
/pages/departments/inventory/setup/InventoryCategory.aspx
```

will now all shorten to:

```
/EditUser
/AppConfig
/ViewUserProfile
/ActivityEventList
/PrintInvoice
/EditMemberList
/InventoryCategory
```

You just need to be careful not to repeat (re-use) the same filename in a different folder.

For example:

```
/pages/member/Search.aspx
/pages/team/Search.aspx
```

Instead of this, put the section name within the filename, like this:

```
/pages/member/SearchMember.aspx
/pages/team/SearchTeam.aspx
```

and the pages will be routed to as follow:

```
/SearchMember
/SearchTeam
```

Okay, that's all for this article.

Thanks for reading and happy coding.

History

- 14th November, 2022: Initial version

License

Written By

adriancs

Software Developer
Other Other

Programming is an art.

Comments and Discussions

0 messages have been posted for this article Visit **https://www.codeproject.com/Articles/5346909/Automatic-Route-All-Pages-in-ASP-NET-WebForms** to post and view comments on this article, or click **here** to get a print view with messages.