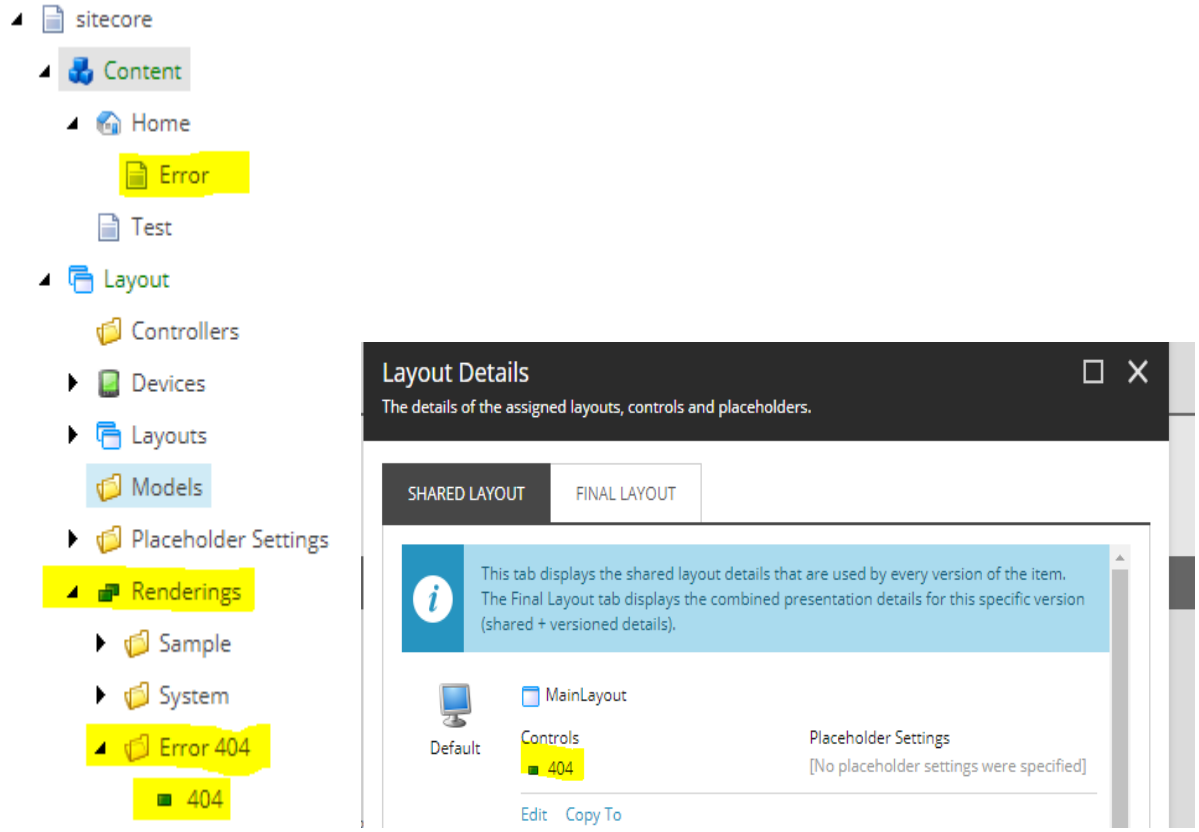
How to create a Custom Pipeline for Item not Found

Steps:

1. In Sitecore, we need to either have a Controller Rendering or a View Rendering that has code to set a few different properties to allow us to properly return a 404-status code then create an Error Item under the Home node and add the rendering on it. Publish the items.



2. Add the Controller in the Visual Studio.

Code:

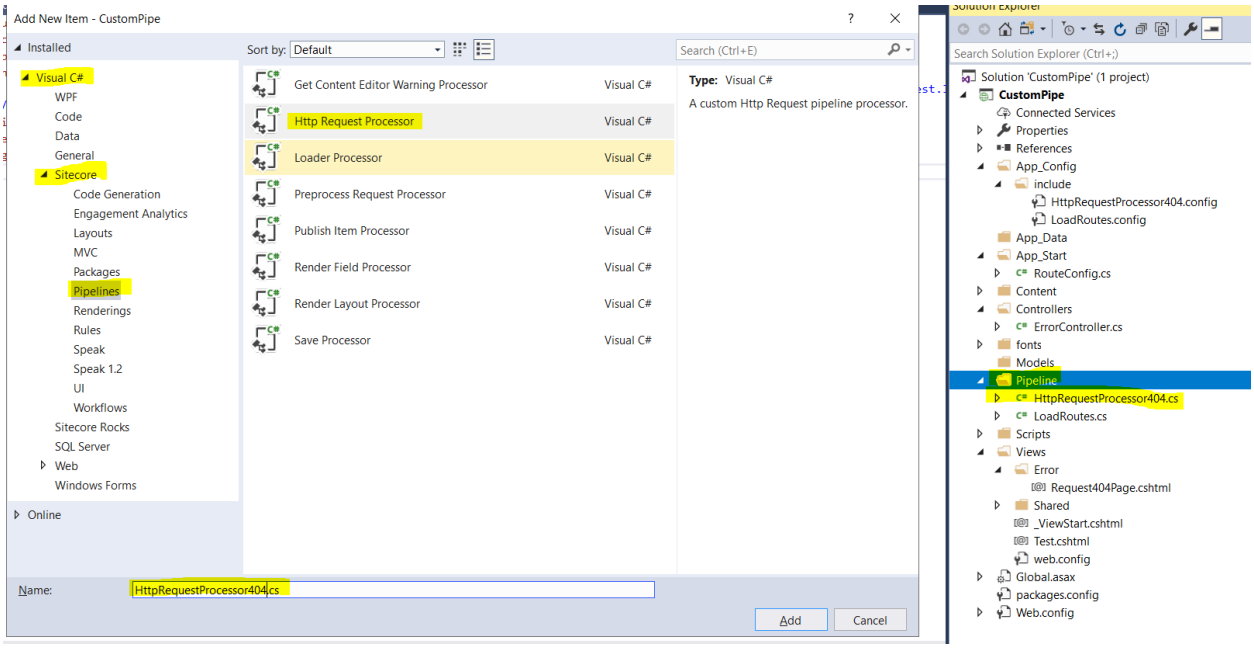
```
using System.Web.Mvc;

namespace CustomPipe.Controllers
{
    public class ErrorController : Controller
    {
        // GET: Error
        public ActionResult Request404Page()
        {
            System.Web.HttpContext.Current.Response.StatusCode = 404;
            System.Web.HttpContext.Current.Response.TrySkipIisCustomErrors = true;
            System.Web.HttpContext.Current.Response.StatusDescription = "404 File Not
Found";
```

```
            return View("~/Views/Error/Request404Page.cshtml");
        }
    }
}
```

3. Add the View in the Visual Studio.
   Code:

```
@{
    ViewBag.Title = "Error Page";
}

<h1>Error Page</h1>
```

4. **Create a 404 Processor**
   a. Create a Folder in the Solution by named as “Pipeline” if exists then use the same folder.
   b. Add a Pipeline by using Http Request Processor, Refer the below screenshot.



   c. By default, VS create a class.

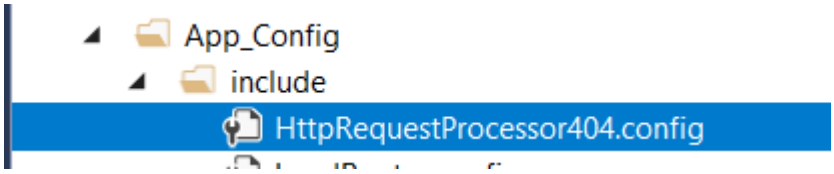
   Code:

```
namespace CustomPipe.Pipeline
{
    using Sitecore.Configuration;
    using Sitecore.Data.Items;
    using Sitecore.Pipelines.HttpRequest;
    using System;
```

```
    // TODO: \App_Config\include\HttpRequestProcessor404.config created
automatically when creating HttpRequestProcessor404 class.

    public class HttpRequestProcessor404 : HttpRequestProcessor
    {
        public override void Process(HttpRequestArgs args)
        {
            if (Sitecore.Context.Item != null || Sitecore.Context.Site == null
|| Sitecore.Context.Database == null || Sitecore.Context.Database.Name ==
"core")
            {
                return;
            }


            var pageNotFound = this.Get404PageItem();
            if (pageNotFound == null)
                return;
            args.ProcessorItem = pageNotFound;
            Sitecore.Context.Item = pageNotFound;
        }

        protected Item Get404PageItem()
        {
            // This is largely up to how the project in general is setup.
            // My solutions are heavily Fortis dependent, but for Vanilla
            // Sitecore setups you could just use the following setting:
            // Settings.GetSetting("ItemNotFoundUrl", "/errors/404");
            // and pull the Item object from that path

            //var path = Sitecore.Context.Site.RootPath + "/Home/Error";

            var item = Sitecore.Context.Database.GetItem(@"{E8698D9A-0721-44D9-
8F19-4DEEB410DB49}");

            return item;
        }
    }
}
```

d. Replace the above GUID with the current Error item GUID which you have created.

e. If you see the related config file is already added in your solution.



Code:

```
<configuration xmlns:patch="http://www.sitecore.net/xmlconfig/">
  <sitecore>
    <pipelines>
      <httpRequestBegin>
```

```
        <processor type="CustomPipe.Pipeline.HttpRequestProcessor404,CustomPipe"
patch:after="processor[@type='Sitecore.Pipelines.HttpRequest.ItemResolver,
Sitecore.Kernel']"/>
      </httpRequestBegin>
    </pipelines>
  </sitecore>
</configuration>
```

f. Build and Publish the Solution and the configs.

5. Open the web browser and put the “Domain Name” / (Anything). See the Output