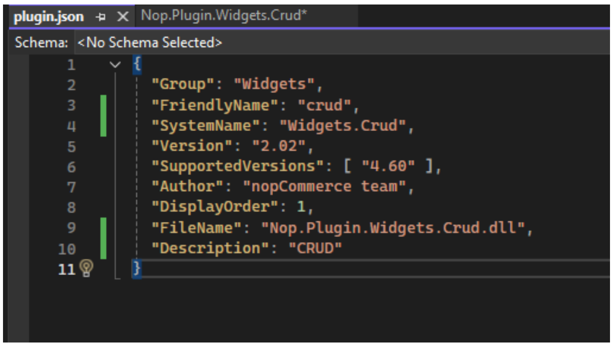
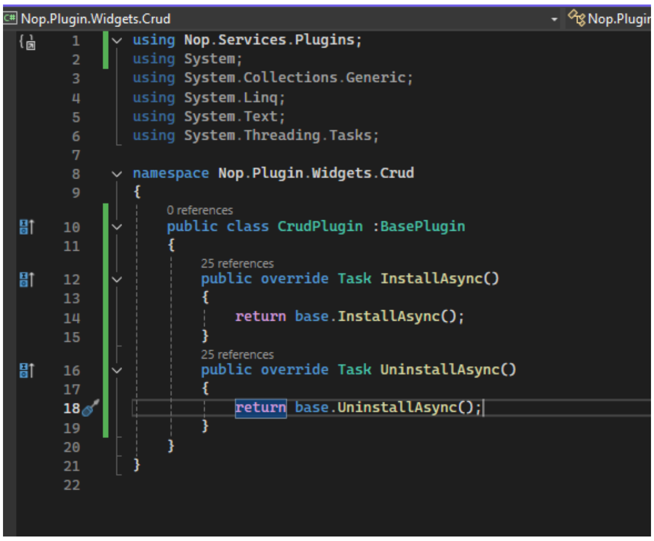
How to create CRUD plugin in nopCommerce 4.60.0

1. Download source 4.60.0 code from official website
2. Set database and run project
3. Create new plugin Nop.Plugin.Widgets.Crud and select .NET 7
4. add Nop.Web project reference
5. Create plugin.json or copy from other plugin and set dll



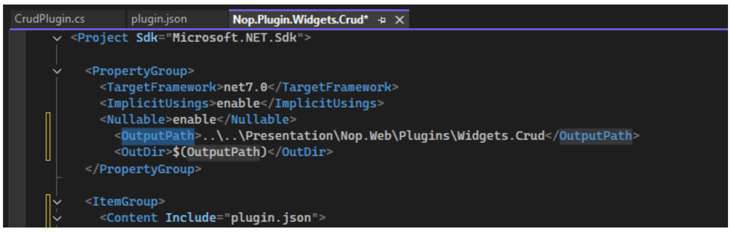
```json
{
  "Group": "Widgets",
  "FriendlyName": "crud",
  "SystemName": "Widgets.Crud",
  "Version": "2.02",
  "SupportedVersions": [ "4.60" ],
  "Author": "nopCommerce team",
  "DisplayOrder": 1,
  "FileName": "Nop.Plugin.Widgets.Crud.dll",
  "Description": "CRUD"
}
```

6. Create a class CrudPlugin.cs and make it public (note:most of the classes should public)



```csharp
using Nop.Services.Plugins;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Nop.Plugin.Widgets.Crud
{
    public class CrudPlugin :BasePlugin
    {
        public override Task InstallAsync()
        {
            return base.InstallAsync();
        }
        public override Task UninstallAsync()
        {
            return base.UninstallAsync();
        }
    }
}
```

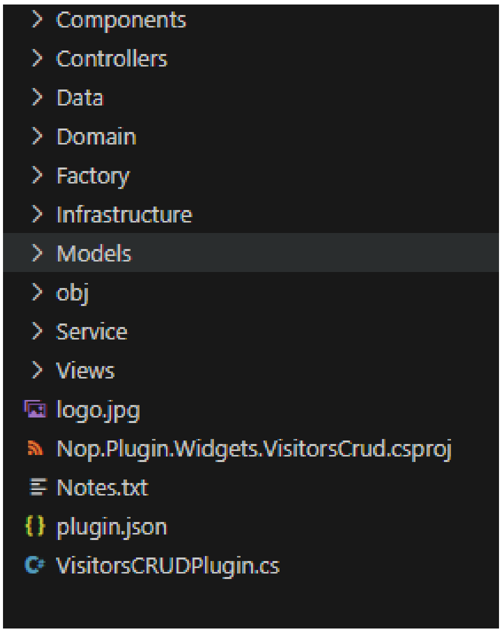
7. Now set OutputPath in



```xml
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>net7.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
      <OutputPath>..\..\Presentation\Nop.Web\Plugins\Widgets.Crud</OutputPath>
      <OutDir>$(OutputPath)</OutDir>
  </PropertyGroup>

  <ItemGroup>
    <Content Include="plugin.json">
```

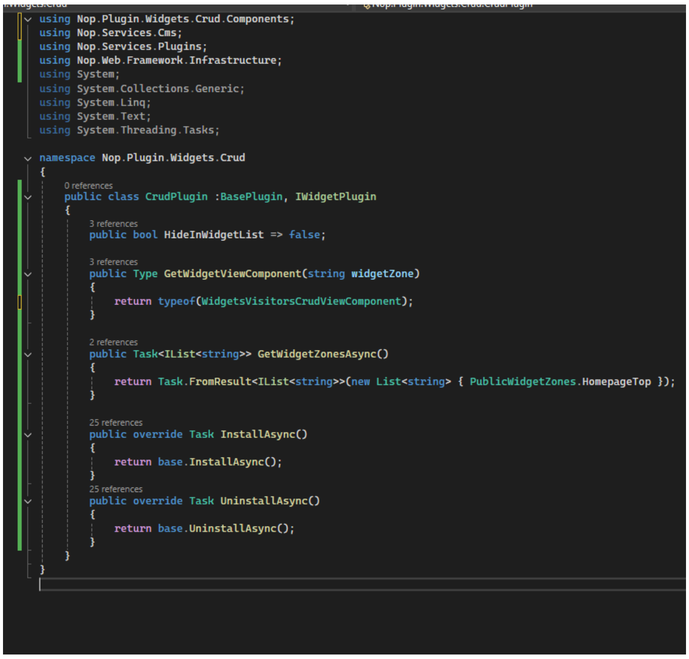
8. Noe build it and check it from admin side

9. Now create all folder



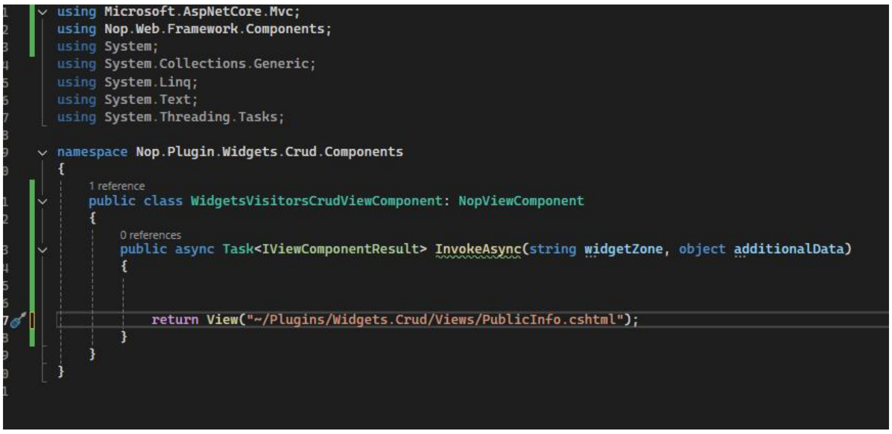
```
> Components
> Controllers
> Data
> Domain
> Factory
> Infrastructure
> Models
> obj
> Service
> Views
logo.jpg
Nop.Plugin.Widgets.VisitorsCrud.csproj
Notes.txt
plugin.json
VisitorsCRUDPlugin.cs
```

10. In CrudPlugin.cs inherit from IWidgetPlugin



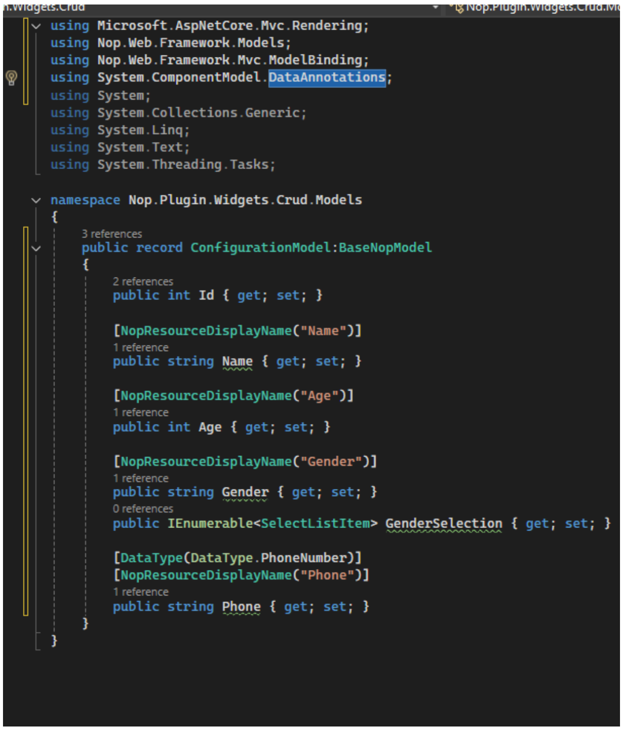
```csharp
using Nop.Plugin.Widgets.Crud.Components;
using Nop.Services.Cms;
using Nop.Services.Plugins;
using Nop.Web.Framework.Infrastructure;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Nop.Plugin.Widgets.Crud
{
    0 references
    public class CrudPlugin :BasePlugin, IWidgetPlugin
    {
        3 references
        public bool HideInWidgetList => false;

        3 references
        public Type GetWidgetViewComponent(string widgetZone)
        {
            return typeof(WidgetsVisitorsCrudViewComponent);
        }

        2 references
        public Task<IList<string>> GetWidgetZonesAsync()
        {
            return Task.FromResult<IList<string>>(new List<string> { PublicWidgetZones.HomepageTop });
        }

        25 references
        public override Task InstallAsync()
        {
            return base.InstallAsync();
        }
        25 references
        public override Task UninstallAsync()
        {
            return base.UninstallAsync();
        }
    }
}
```

11. Now create WidgetsVisitorsCrudViewComponent class inside component folder



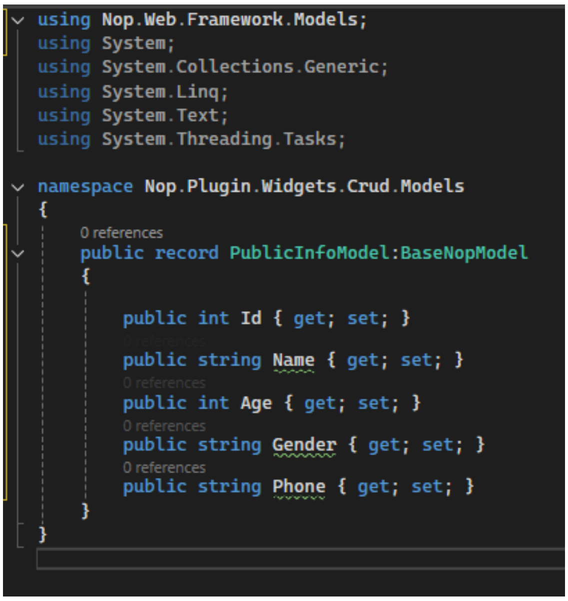
```csharp
using Microsoft.AspNetCore.Mvc;
using Nop.Web.Framework.Components;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Nop.Plugin.Widgets.Crud.Components
{
    1 reference
    public class WidgetsVisitorsCrudViewComponent: NopViewComponent
    {
        0 references
        public async Task<IViewComponentResult> InvokeAsync(string widgetZone, object additionalData)
        {


            return View("~/Plugins/Widgets.Crud/Views/PublicInfo.cshtml");
        }
    }
}
```

12. Now create PublicInfo.cshtml inside view filder
13. Add model PublicInfoModel.cs ConfigurationModel.cs
14. In ConfigurationModel.cs add DataAnnotations



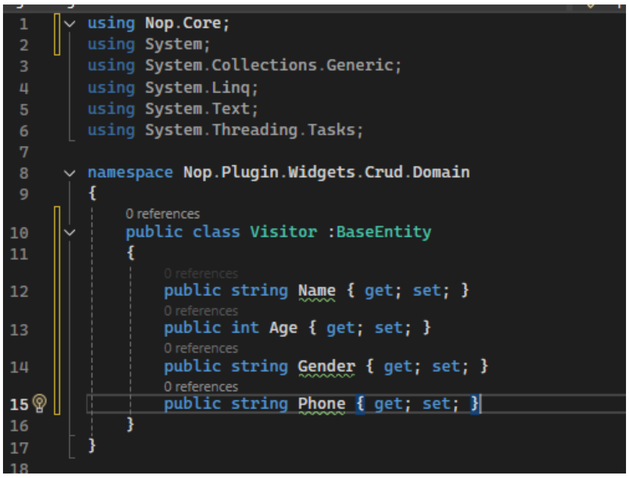
```
using Microsoft.AspNetCore.Mvc.Rendering;
using Nop.Web.Framework.Models;
using Nop.Web.Framework.Mvc.ModelBinding;
using System.ComponentModel.DataAnnotations;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Nop.Plugin.Widgets.Crud.Models
{
    3 references
    public record ConfigurationModel:BaseNopModel
    {
        2 references
        public int Id { get; set; }

        [NopResourceDisplayName("Name")]
        1 reference
        public string Name { get; set; }

        [NopResourceDisplayName("Age")]
        1 reference
        public int Age { get; set; }

        [NopResourceDisplayName("Gender")]
        1 reference
        public string Gender { get; set; }
        0 references
        public IEnumerable<SelectListItem> GenderSelection { get; set; }

        [DataType(DataType.PhoneNumber)]
        [NopResourceDisplayName("Phone")]
        1 reference
        public string Phone { get; set; }
    }
}
```

15. PublicInfoModel.cs



```
using Nop.Web.Framework.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Nop.Plugin.Widgets.Crud.Models
{
    0 references
    public record PublicInfoModel:BaseNopModel
    {

        public int Id { get; set; }
        0 references
        public string Name { get; set; }
        0 references
        public int Age { get; set; }
        0 references
        public string Gender { get; set; }
        0 references
        public string Phone { get; set; }
    }
}
```

16. Add a class in Domain Visitor.cs



```
using Nop.Core;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Nop.Plugin.Widgets.Crud.Domain
{
    0 references
    public class Visitor :BaseEntity
    {
        0 references
        public string Name { get; set; }
        0 references
        public int Age { get; set; }
        0 references
        public string Gender { get; set; }
        0 references
        public string Phone { get; set; }
    }
}
```

17. difference between Visitor and PublicInfoModel class

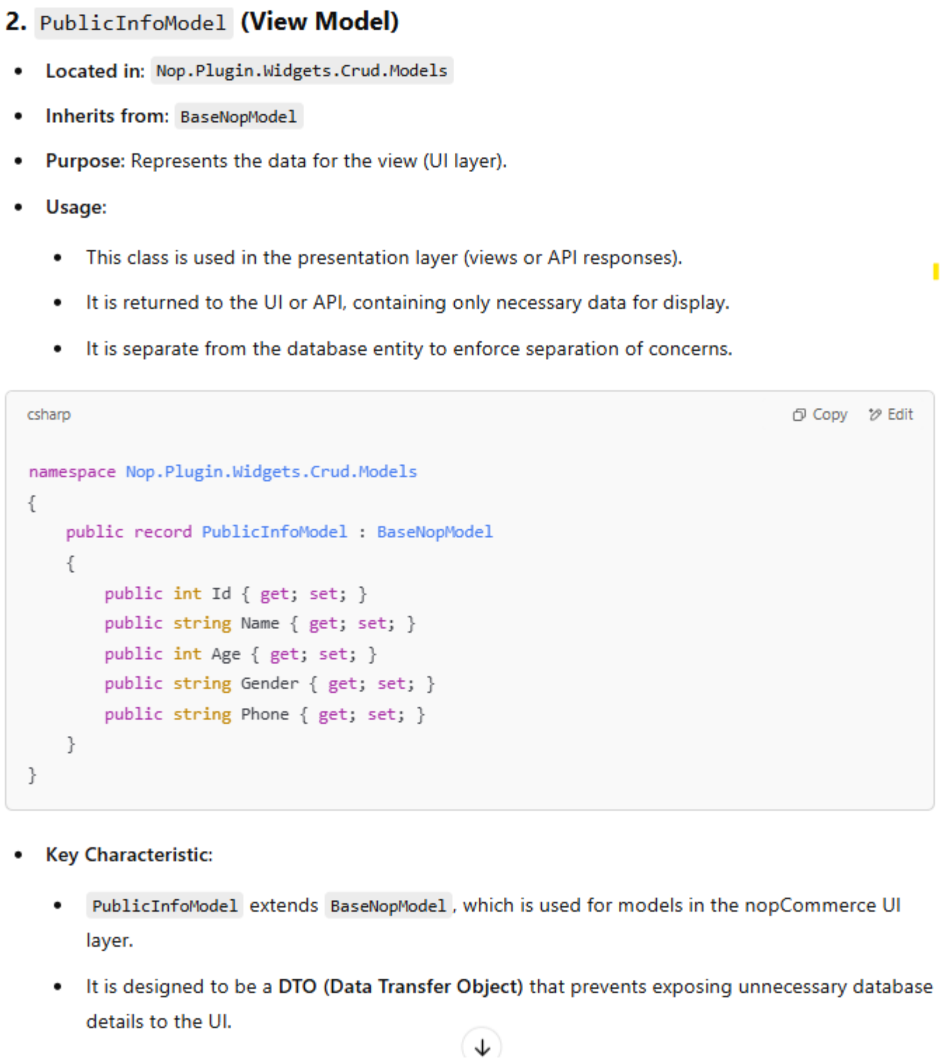
1. `Visitor` (Domain Model)

- **Located in:** `Nop.Plugin.Widgets.Crud.Domain`
- **Inherits from:** `BaseEntity`
- **Purpose**: Represents the database entity.
- **Usage**:
  - This class is mapped to the database table.
  - It is used by the data access layer (repositories and services).
  - It interacts with Entity Framework Core to perform CRUD operations.

```csharp
namespace Nop.Plugin.Widgets.Crud.Domain
{
    public class Visitor : BaseEntity
    {
        public string Name { get; set; }
        public int Age { get; set; }
        public string Gender { get; set; }
        public string Phone { get; set; }
    }
}
```

- **Key Characteristic:**
  - The `Visitor` class extends `BaseEntity`, which typically includes properties like `Id`, `CreatedOnUtc`, `UpdatedOnUtc`, etc.
  - It is responsible for persistence (saving data to and retrieving data from the database).

18.



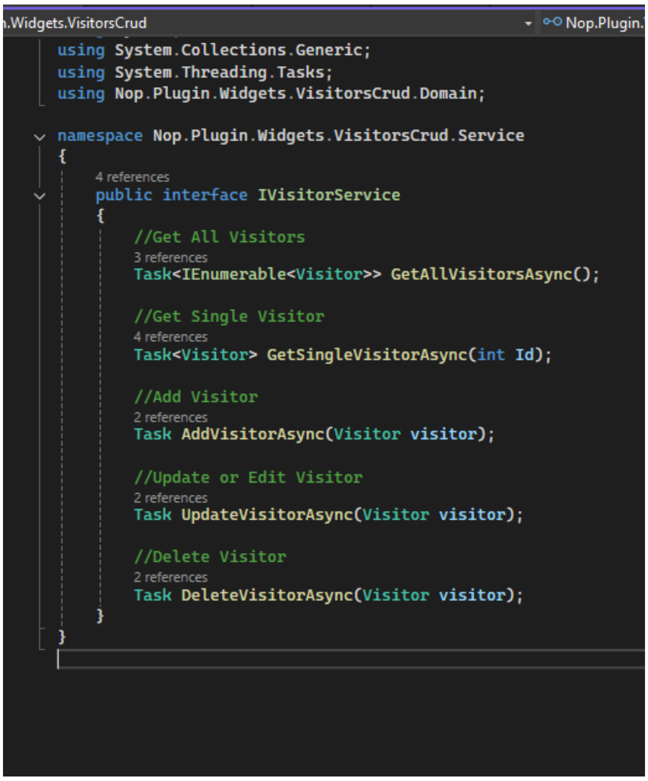
2. `PublicInfoModel` (View Model)

- **Located in:** `Nop.Plugin.Widgets.Crud.Models`
- **Inherits from:** `BaseNopModel`
- **Purpose:** Represents the data for the view (UI layer).
- **Usage:**
  - This class is used in the presentation layer (views or API responses).
  - It is returned to the UI or API, containing only necessary data for display.
  - It is separate from the database entity to enforce separation of concerns.

```csharp
namespace Nop.Plugin.Widgets.Crud.Models
{
    public record PublicInfoModel : BaseNopModel
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public int Age { get; set; }
        public string Gender { get; set; }
        public string Phone { get; set; }
    }
}
```

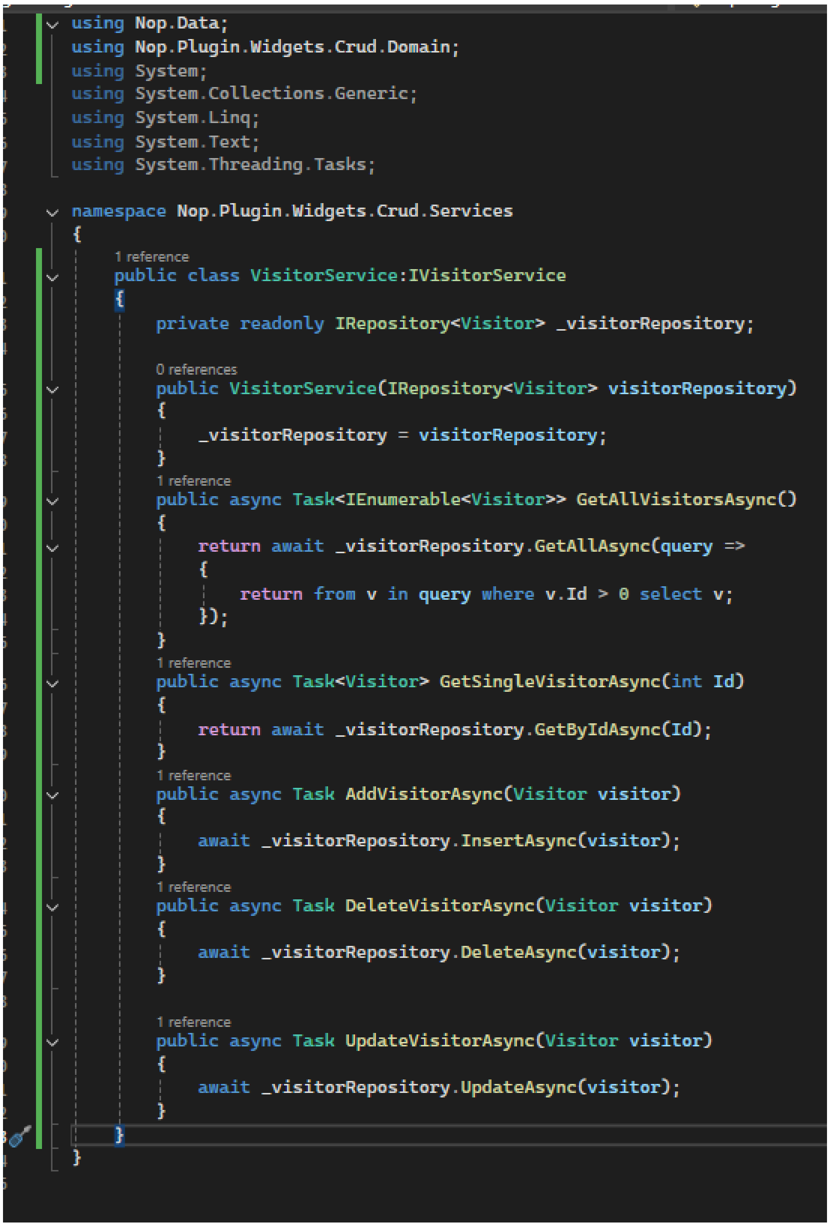
- **Key Characteristic:**
  - `PublicInfoModel` extends `BaseNopModel`, which is used for models in the nopCommerce UI layer.
  - It is designed to be a **DTO (Data Transfer Object)** that prevents exposing unnecessary database details to the UI.

19. Create IVisitorService inside service folder



```
using System.Collections.Generic;
using System.Threading.Tasks;
using Nop.Plugin.Widgets.VisitorsCrud.Domain;

namespace Nop.Plugin.Widgets.VisitorsCrud.Service
{
    4 references
    public interface IVisitorService
    {
        //Get All Visitors
        3 references
        Task<IEnumerable<Visitor>> GetAllVisitorsAsync();

        //Get Single Visitor
        4 references
        Task<Visitor> GetSingleVisitorAsync(int Id);

        //Add Visitor
        2 references
        Task AddVisitorAsync(Visitor visitor);

        //Update or Edit Visitor
        2 references
        Task UpdateVisitorAsync(Visitor visitor);

        //Delete Visitor
        2 references
        Task DeleteVisitorAsync(Visitor visitor);
    }
}
```

20. Create VisitorService inside service folder



```
using Nop.Data;
using Nop.Plugin.Widgets.Crud.Domain;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Nop.Plugin.Widgets.Crud.Services
{
    1 reference
    public class VisitorService:IVisitorService
    {
        private readonly IRepository<Visitor> _visitorRepository;

        0 references
        public VisitorService(IRepository<Visitor> visitorRepository)
        {
            _visitorRepository = visitorRepository;
        }
        1 reference
        public async Task<IEnumerable<Visitor>> GetAllVisitorsAsync()
        {
            return await _visitorRepository.GetAllAsync(query =>
            {
                return from v in query where v.Id > 0 select v;
            });
        }
        1 reference
        public async Task<Visitor> GetSingleVisitorAsync(int Id)
        {
            return await _visitorRepository.GetByIdAsync(Id);
        }
        1 reference
        public async Task AddVisitorAsync(Visitor visitor)
        {
            await _visitorRepository.InsertAsync(visitor);
        }
        1 reference
        public async Task DeleteVisitorAsync(Visitor visitor)
        {
            await _visitorRepository.DeleteAsync(visitor);
        }

        1 reference
        public async Task UpdateVisitorAsync(Visitor visitor)
        {
            await _visitorRepository.UpdateAsync(visitor);
        }
    }
}
```

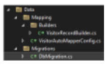
steps for Data folder

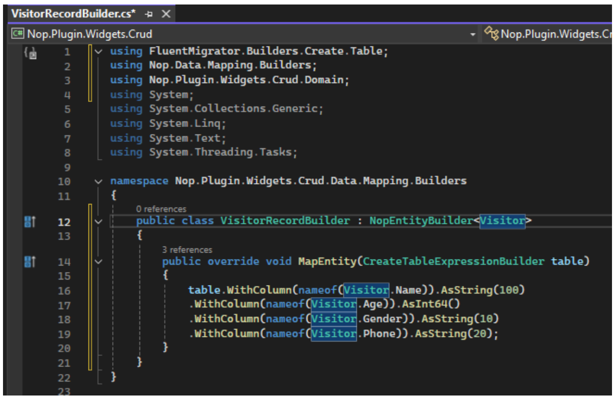
21. Add Mapping Migrations folder inside Data folder

22. Add Builders folder and VisitorAutoMapperConfig.cs inside Mapping folder

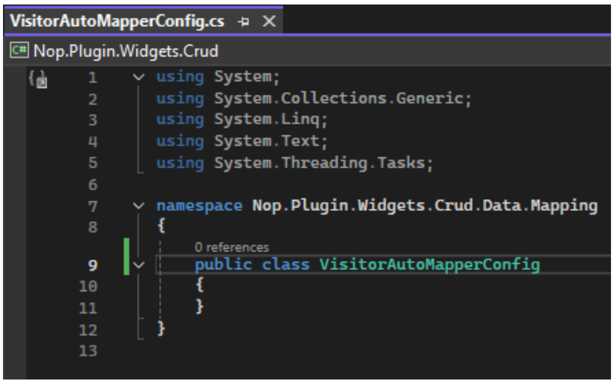
23. Add VisitorRecordBuilder.cs inside Builders folder

24. Add DbMigration inside Migrations folder

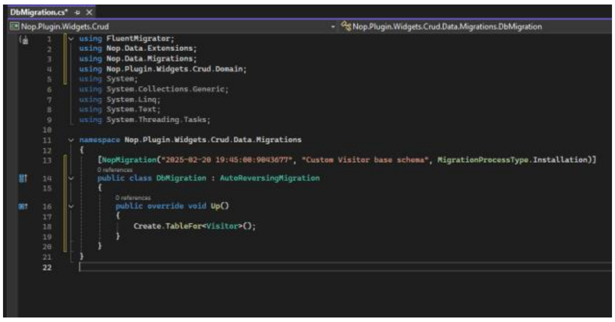


25. 

```csharp
using FluentMigrator.Builders.Create.Table;
using Nop.Data.Mapping.Builders;
using Nop.Plugin.Widgets.Crud.Domain;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Nop.Plugin.Widgets.Crud.Data.Mapping.Builders
{
    public class VisitorRecordBuilder : NopEntityBuilder<Visitor>
    {
        public override void MapEntity(CreateTableExpressionBuilder table)
        {
            table.WithColumn(nameof(Visitor.Name)).AsString(100)
            .WithColumn(nameof(Visitor.Age)).AsInt64()
            .WithColumn(nameof(Visitor.Gender)).AsString(10)
            .WithColumn(nameof(Visitor.Phone)).AsString(20);
        }
    }
}
```

26. 

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Nop.Plugin.Widgets.Crud.Data.Mapping
{
    public class VisitorAutoMapperConfig
    {
    }
}
```

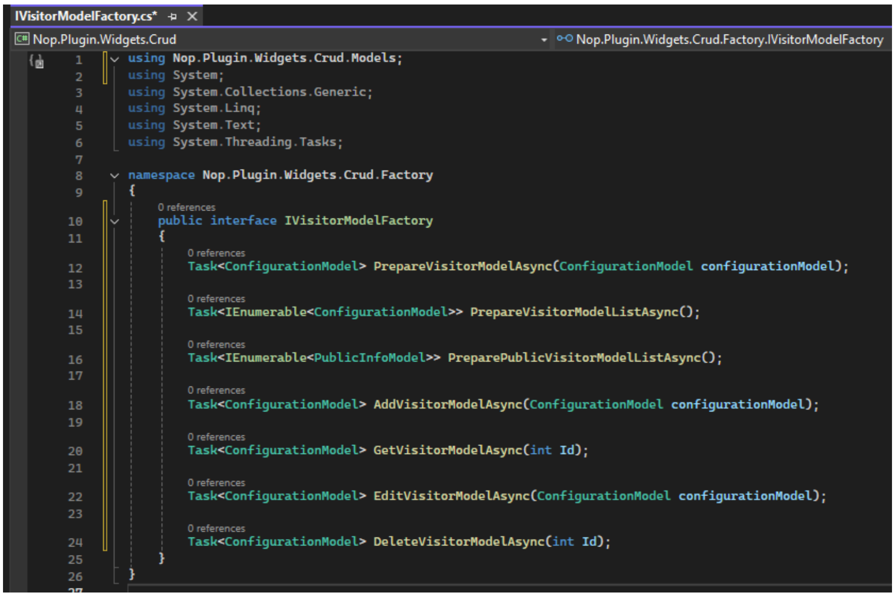
27. 

```csharp
using FluentMigrator;
using Nop.Data.Extensions;
using Nop.Data.Migrations;
using Nop.Plugin.Widgets.Crud.Domain;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Nop.Plugin.Widgets.Crud.Data.Migrations
{
    [NopMigration("2025-02-20 19:45:00:9843677", "Custom Visitor base schema", MigrationProcessType.Installation)]
    public class DbMigration : AutoReversingMigration
    {
        public override void Up()
        {
            Create.TableFor<Visitor>();
        }
    }
}
```

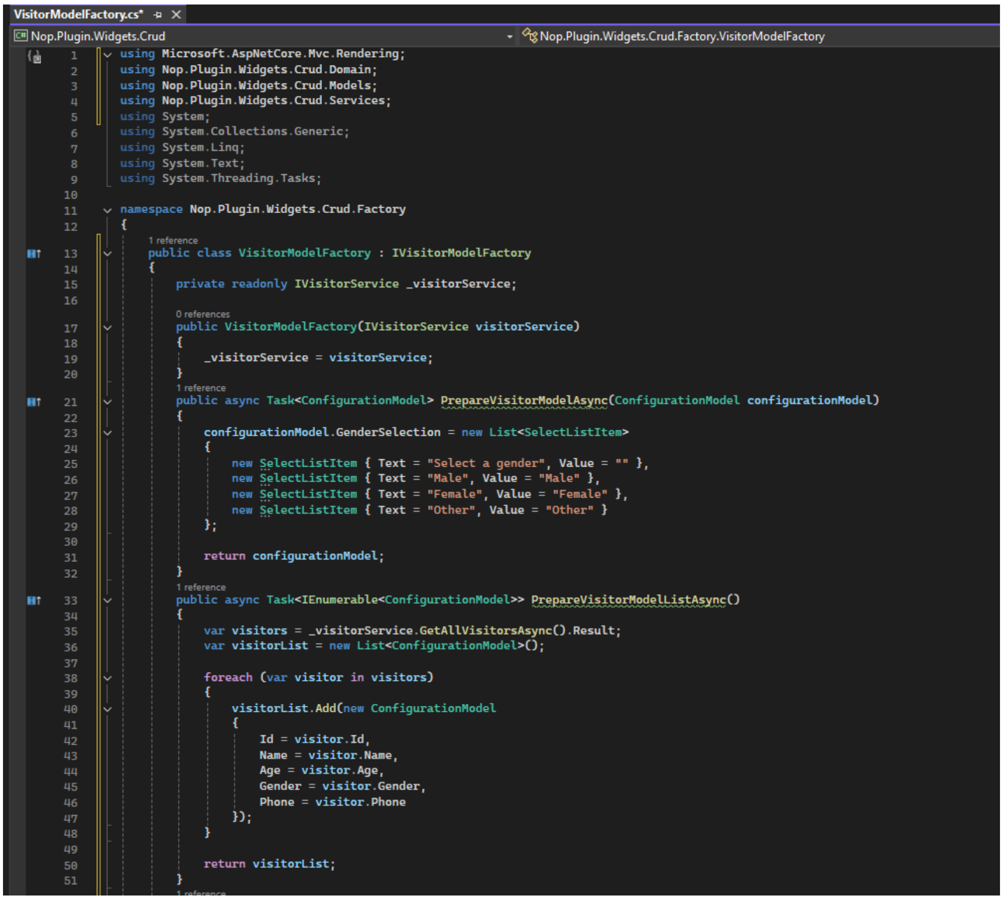
----

28. Now time to Factory

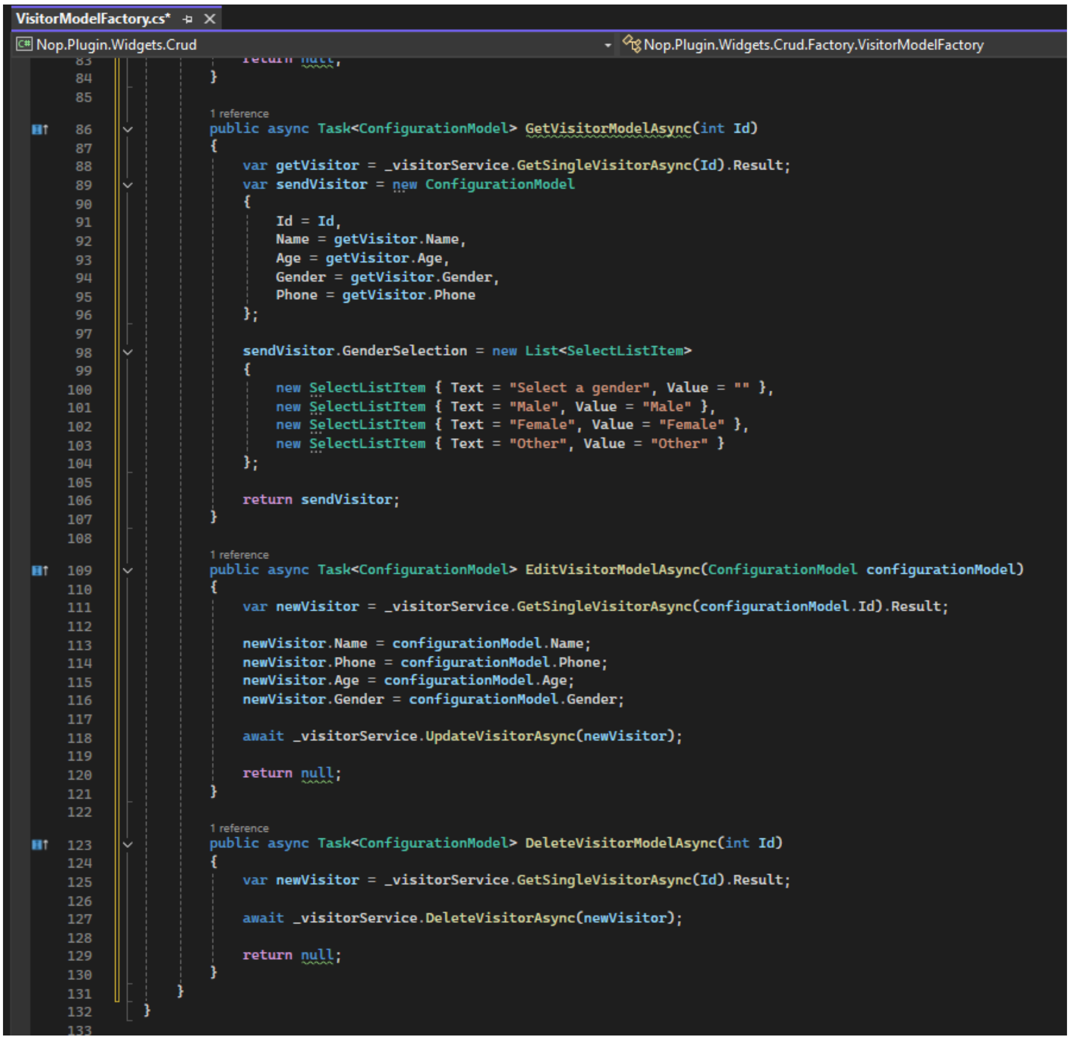
29. Add VisitorModelFactory.cs IVisitorModelFactory.cs inside Factory folder



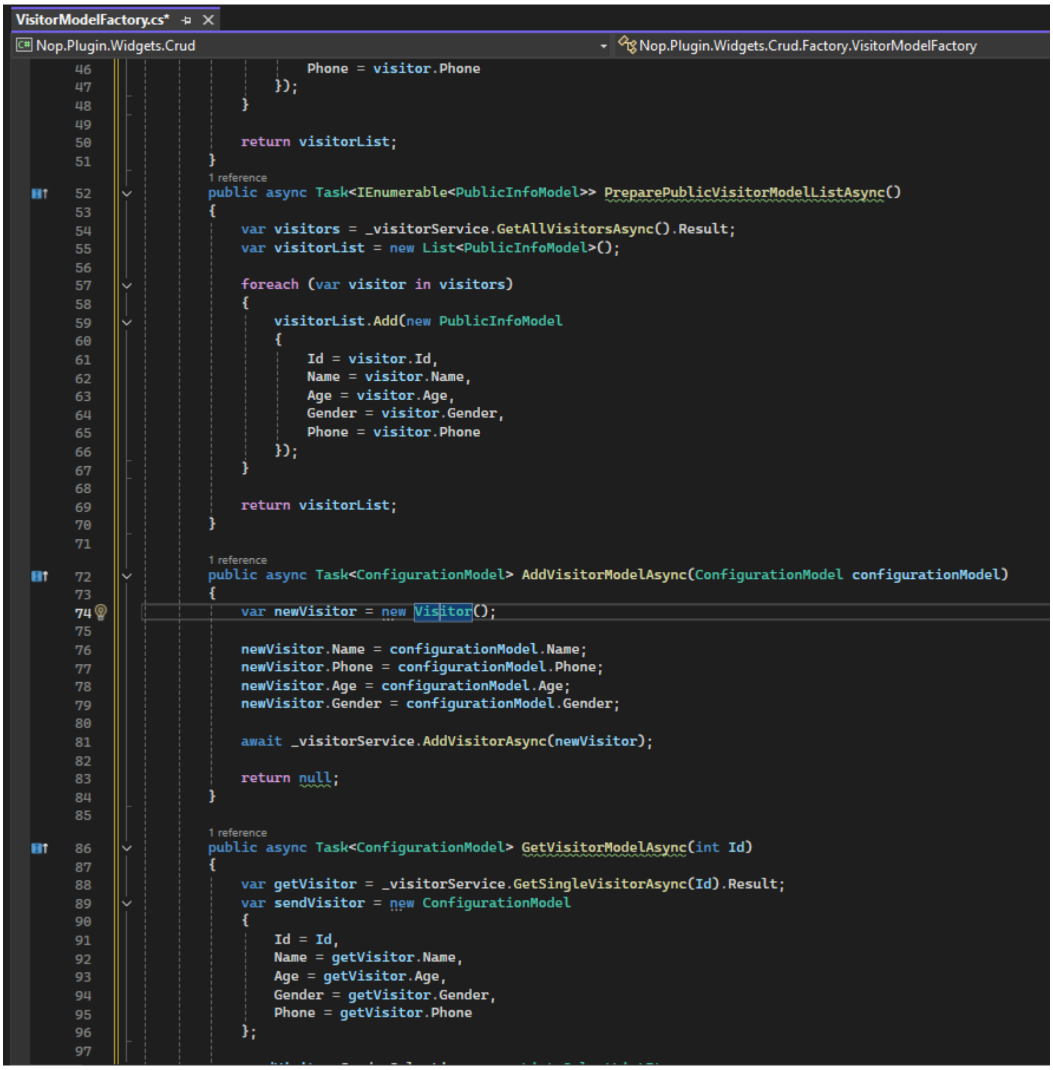
```csharp
using Nop.Plugin.Widgets.Crud.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace Nop.Plugin.Widgets.Crud.Factory
{

    public interface IVisitorModelFactory
    {
        Task<ConfigurationModel> PrepareVisitorModelAsync(ConfigurationModel configurationModel);

        Task<IEnumerable<ConfigurationModel>> PrepareVisitorModelListAsync();

        Task<IEnumerable<PublicInfoModel>> PreparePublicVisitorModelListAsync();

        Task<ConfigurationModel> AddVisitorModelAsync(ConfigurationModel configurationModel);

        Task<ConfigurationModel> GetVisitorModelAsync(int Id);

        Task<ConfigurationModel> EditVisitorModelAsync(ConfigurationModel configurationModel);

        Task<ConfigurationModel> DeleteVisitorModelAsync(int Id);
    }
}
```



```csharp
using Microsoft.AspNetCore.Mvc.Rendering;
using Nop.Plugin.Widgets.Crud.Domain;
using Nop.Plugin.Widgets.Crud.Models;
using Nop.Plugin.Widgets.Crud.Services;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Nop.Plugin.Widgets.Crud.Factory
{
    public class VisitorModelFactory : IVisitorModelFactory
    {
        private readonly IVisitorService _visitorService;

        public VisitorModelFactory(IVisitorService visitorService)
        {
            _visitorService = visitorService;
        }
        public async Task<ConfigurationModel> PrepareVisitorModelAsync(ConfigurationModel configurationModel)
        {
            configurationModel.GenderSelection = new List<SelectListItem>
            {
                new SelectListItem { Text = "Select a gender", Value = "" },
                new SelectListItem { Text = "Male", Value = "Male" },
                new SelectListItem { Text = "Female", Value = "Female" },
                new SelectListItem { Text = "Other", Value = "Other" }
            };

            return configurationModel;
        }
        public async Task<IEnumerable<ConfigurationModel>> PrepareVisitorModelListAsync()
        {
            var visitors = _visitorService.GetAllVisitorsAsync().Result;
            var visitorList = new List<ConfigurationModel>();

            foreach (var visitor in visitors)
            {
                visitorList.Add(new ConfigurationModel
                {
                    Id = visitor.Id,
                    Name = visitor.Name,
                    Age = visitor.Age,
                    Gender = visitor.Gender,
                    Phone = visitor.Phone
                });
            }

            return visitorList;
        }
```

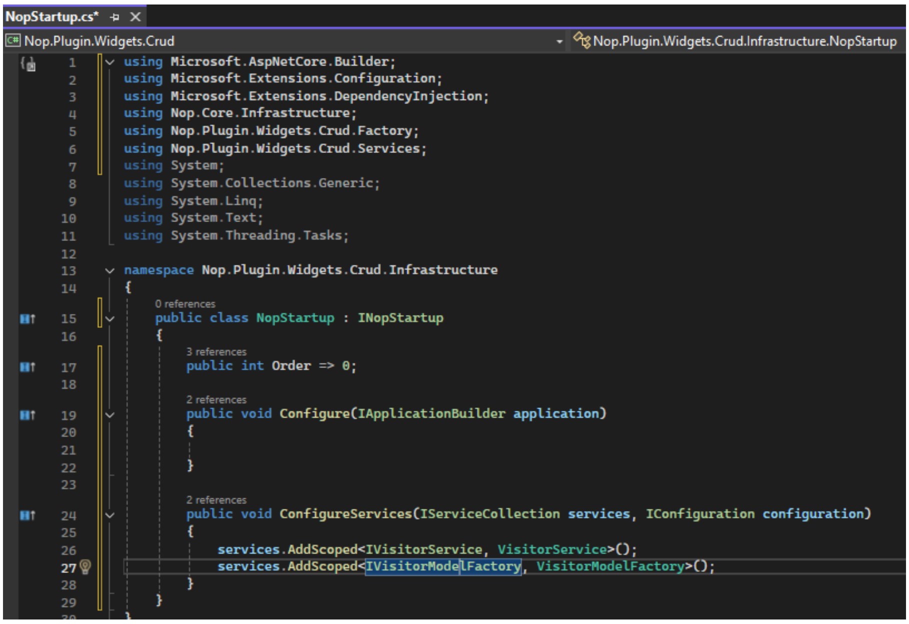


```
            return null;
        }

        public async Task<ConfigurationModel> GetVisitorModelAsync(int Id)
        {
            var getVisitor = _visitorService.GetSingleVisitorAsync(Id).Result;
            var sendVisitor = new ConfigurationModel
            {
                Id = Id,
                Name = getVisitor.Name,
                Age = getVisitor.Age,
                Gender = getVisitor.Gender,
                Phone = getVisitor.Phone
            };

            sendVisitor.GenderSelection = new List<SelectListItem>
            {
                new SelectListItem { Text = "Select a gender", Value = "" },
                new SelectListItem { Text = "Male", Value = "Male" },
                new SelectListItem { Text = "Female", Value = "Female" },
                new SelectListItem { Text = "Other", Value = "Other" }
            };

            return sendVisitor;
        }

        public async Task<ConfigurationModel> EditVisitorModelAsync(ConfigurationModel configurationModel)
        {
            var newVisitor = _visitorService.GetSingleVisitorAsync(configurationModel.Id).Result;

            newVisitor.Name = configurationModel.Name;
            newVisitor.Phone = configurationModel.Phone;
            newVisitor.Age = configurationModel.Age;
            newVisitor.Gender = configurationModel.Gender;

            await _visitorService.UpdateVisitorAsync(newVisitor);

            return null;
        }

        public async Task<ConfigurationModel> DeleteVisitorModelAsync(int Id)
        {
            var newVisitor = _visitorService.GetSingleVisitorAsync(Id).Result;

            await _visitorService.DeleteVisitorAsync(newVisitor);

            return null;
        }
    }
}
```



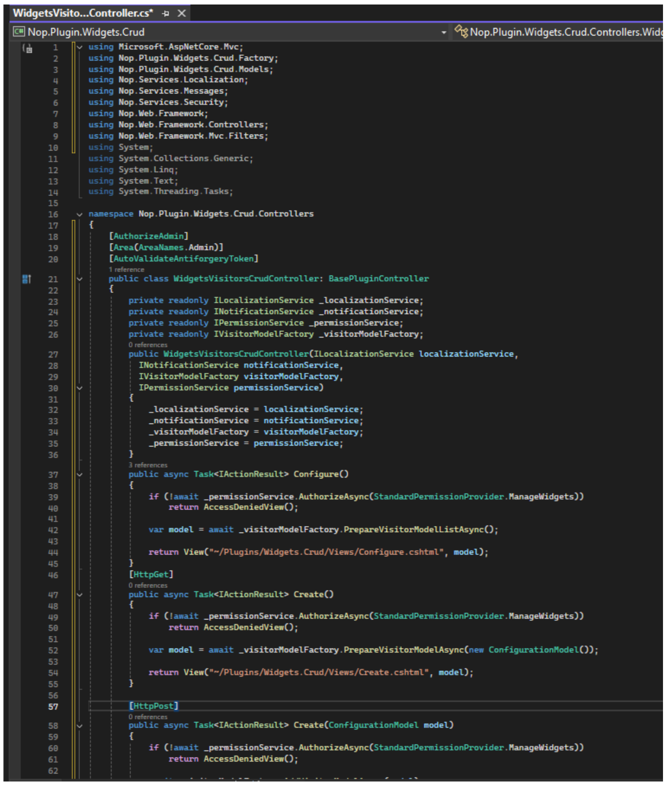
```
                    Phone = visitor.Phone
                });
            }

            return visitorList;
        }
        1 reference
        public async Task<IEnumerable<PublicInfoModel>> PreparePublicVisitorModelListAsync()
        {
            var visitors = _visitorService.GetAllVisitorsAsync().Result;
            var visitorList = new List<PublicInfoModel>();

            foreach (var visitor in visitors)
            {
                visitorList.Add(new PublicInfoModel
                {
                    Id = visitor.Id,
                    Name = visitor.Name,
                    Age = visitor.Age,
                    Gender = visitor.Gender,
                    Phone = visitor.Phone
                });
            }

            return visitorList;
        }

        1 reference
        public async Task<ConfigurationModel> AddVisitorModelAsync(ConfigurationModel configurationModel)
        {
            var newVisitor = new Visitor();

            newVisitor.Name = configurationModel.Name;
            newVisitor.Phone = configurationModel.Phone;
            newVisitor.Age = configurationModel.Age;
            newVisitor.Gender = configurationModel.Gender;

            await _visitorService.AddVisitorAsync(newVisitor);

            return null;
        }

        1 reference
        public async Task<ConfigurationModel> GetVisitorModelAsync(int Id)
        {
            var getVisitor = _visitorService.GetSingleVisitorAsync(Id).Result;
            var sendVisitor = new ConfigurationModel
            {
                Id = Id,
                Name = getVisitor.Name,
                Age = getVisitor.Age,
                Gender = getVisitor.Gender,
                Phone = getVisitor.Phone
            };
```

30. Add NopStartup.cs inside infrastructure folder

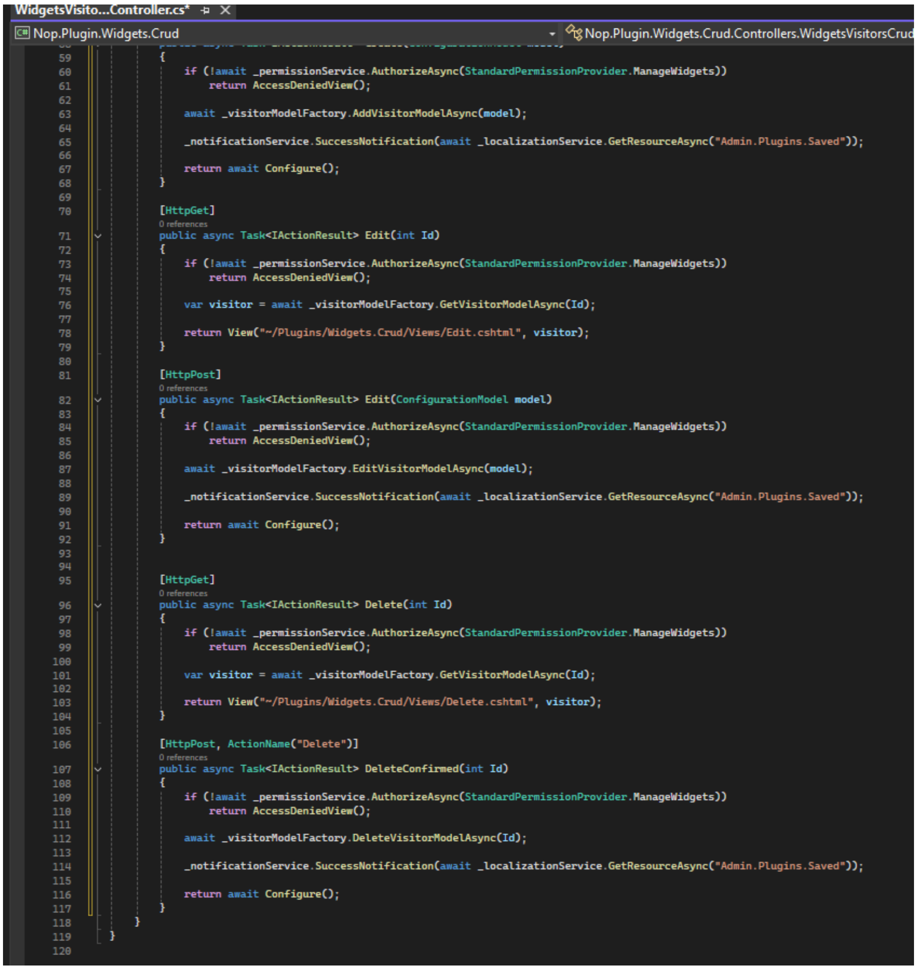


```
using Microsoft.AspNetCore.Builder;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Nop.Core.Infrastructure;
using Nop.Plugin.Widgets.Crud.Factory;
using Nop.Plugin.Widgets.Crud.Services;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Nop.Plugin.Widgets.Crud.Infrastructure
{
    0 references
    public class NopStartup : INopStartup
    {
        3 references
        public int Order => 0;

        2 references
        public void Configure(IApplicationBuilder application)
        {

        }

        2 references
        public void ConfigureServices(IServiceCollection services, IConfiguration configuration)
        {
            services.AddScoped<IVisitorService, VisitorService>();
            services.AddScoped<IVisitorModelFactory, VisitorModelFactory>();
        }
    }
```

31. **WidgetsVisitorsCrudController.cs** add in Controller folder

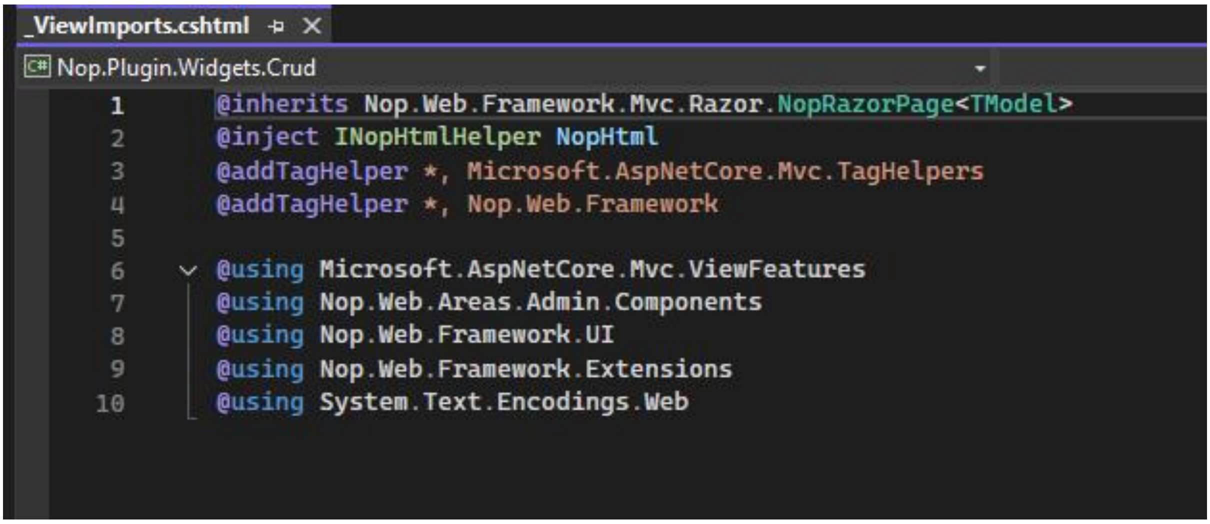

WidgetsVisito...Controller.cs*
Nop.Plugin.Widgets.Crud
Nop.Plugin.Widgets.Crud.Controllers.Widg

```
using Microsoft.AspNetCore.Mvc;
using Nop.Plugin.Widgets.Crud.Factory;
using Nop.Plugin.Widgets.Crud.Models;
using Nop.Services.Localization;
using Nop.Services.Messages;
using Nop.Services.Security;
using Nop.Web.Framework;
using Nop.Web.Framework.Controllers;
using Nop.Web.Framework.Mvc.Filters;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Nop.Plugin.Widgets.Crud.Controllers
{
    [AuthorizeAdmin]
    [Area(AreaNames.Admin)]
    [AutoValidateAntiforgeryToken]
    1 reference
    public class WidgetsVisitorsCrudController: BasePluginController
    {
        private readonly ILocalizationService _localizationService;
        private readonly INotificationService _notificationService;
        private readonly IPermissionService _permissionService;
        private readonly IVisitorModelFactory _visitorModelFactory;
        0 references
        public WidgetsVisitorsCrudController(ILocalizationService localizationService,
          INotificationService notificationService,
          IVisitorModelFactory visitorModelFactory,
          IPermissionService permissionService)
        {
            _localizationService = localizationService;
            _notificationService = notificationService;
            _visitorModelFactory = visitorModelFactory;
            _permissionService = permissionService;
        }
        3 references
        public async Task<IActionResult> Configure()
        {
            if (!await _permissionService.AuthorizeAsync(StandardPermissionProvider.ManageWidgets))
                return AccessDeniedView();

            var model = await _visitorModelFactory.PrepareVisitorModelListAsync();

            return View("~/Plugins/Widgets.Crud/Views/Configure.cshtml", model);
        }
        [HttpGet]
        0 references
        public async Task<IActionResult> Create()
        {
            if (!await _permissionService.AuthorizeAsync(StandardPermissionProvider.ManageWidgets))
                return AccessDeniedView();

            var model = await _visitorModelFactory.PrepareVisitorModelAsync(new ConfigurationModel());

            return View("~/Plugins/Widgets.Crud/Views/Create.cshtml", model);
        }

        [HttpPost]
        0 references
        public async Task<IActionResult> Create(ConfigurationModel model)
        {
            if (!await _permissionService.AuthorizeAsync(StandardPermissionProvider.ManageWidgets))
                return AccessDeniedView();

```

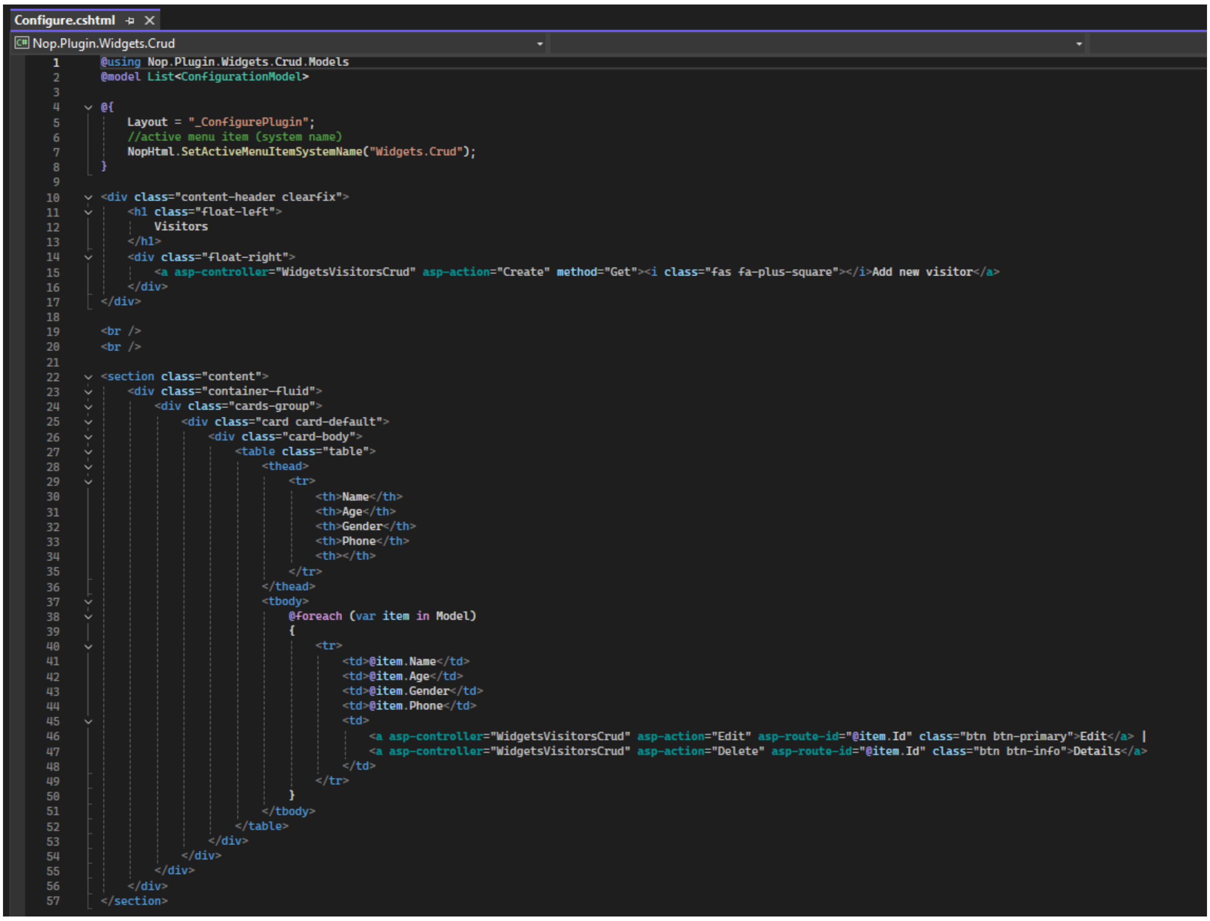

WidgetsVisito...Controller.cs*
Nop.Plugin.Widgets.Crud
Nop.Plugin.Widgets.Crud.Controllers.WidgetsVisitorsCrud

```
        {
            if (!await _permissionService.AuthorizeAsync(StandardPermissionProvider.ManageWidgets))
                return AccessDeniedView();

            await _visitorModelFactory.AddVisitorModelAsync(model);

            _notificationService.SuccessNotification(await _localizationService.GetResourceAsync("Admin.Plugins.Saved"));

            return await Configure();
        }

        [HttpGet]
        0 references
        public async Task<IActionResult> Edit(int Id)
        {
            if (!await _permissionService.AuthorizeAsync(StandardPermissionProvider.ManageWidgets))
                return AccessDeniedView();

            var visitor = await _visitorModelFactory.GetVisitorModelAsync(Id);

            return View("~/Plugins/Widgets.Crud/Views/Edit.cshtml", visitor);
        }

        [HttpPost]
        0 references
        public async Task<IActionResult> Edit(ConfigurationModel model)
        {
            if (!await _permissionService.AuthorizeAsync(StandardPermissionProvider.ManageWidgets))
                return AccessDeniedView();

            await _visitorModelFactory.EditVisitorModelAsync(model);

            _notificationService.SuccessNotification(await _localizationService.GetResourceAsync("Admin.Plugins.Saved"));

            return await Configure();
        }


        [HttpGet]
        0 references
        public async Task<IActionResult> Delete(int Id)
        {
            if (!await _permissionService.AuthorizeAsync(StandardPermissionProvider.ManageWidgets))
                return AccessDeniedView();

            var visitor = await _visitorModelFactory.GetVisitorModelAsync(Id);

            return View("~/Plugins/Widgets.Crud/Views/Delete.cshtml", visitor);
        }

        [HttpPost, ActionName("Delete")]
        0 references
        public async Task<IActionResult> DeleteConfirmed(int Id)
        {
            if (!await _permissionService.AuthorizeAsync(StandardPermissionProvider.ManageWidgets))
                return AccessDeniedView();

            await _visitorModelFactory.DeleteVisitorModelAsync(Id);

            _notificationService.SuccessNotification(await _localizationService.GetResourceAsync("Admin.Plugins.Saved"));

            return await Configure();
        }
    }
}

```

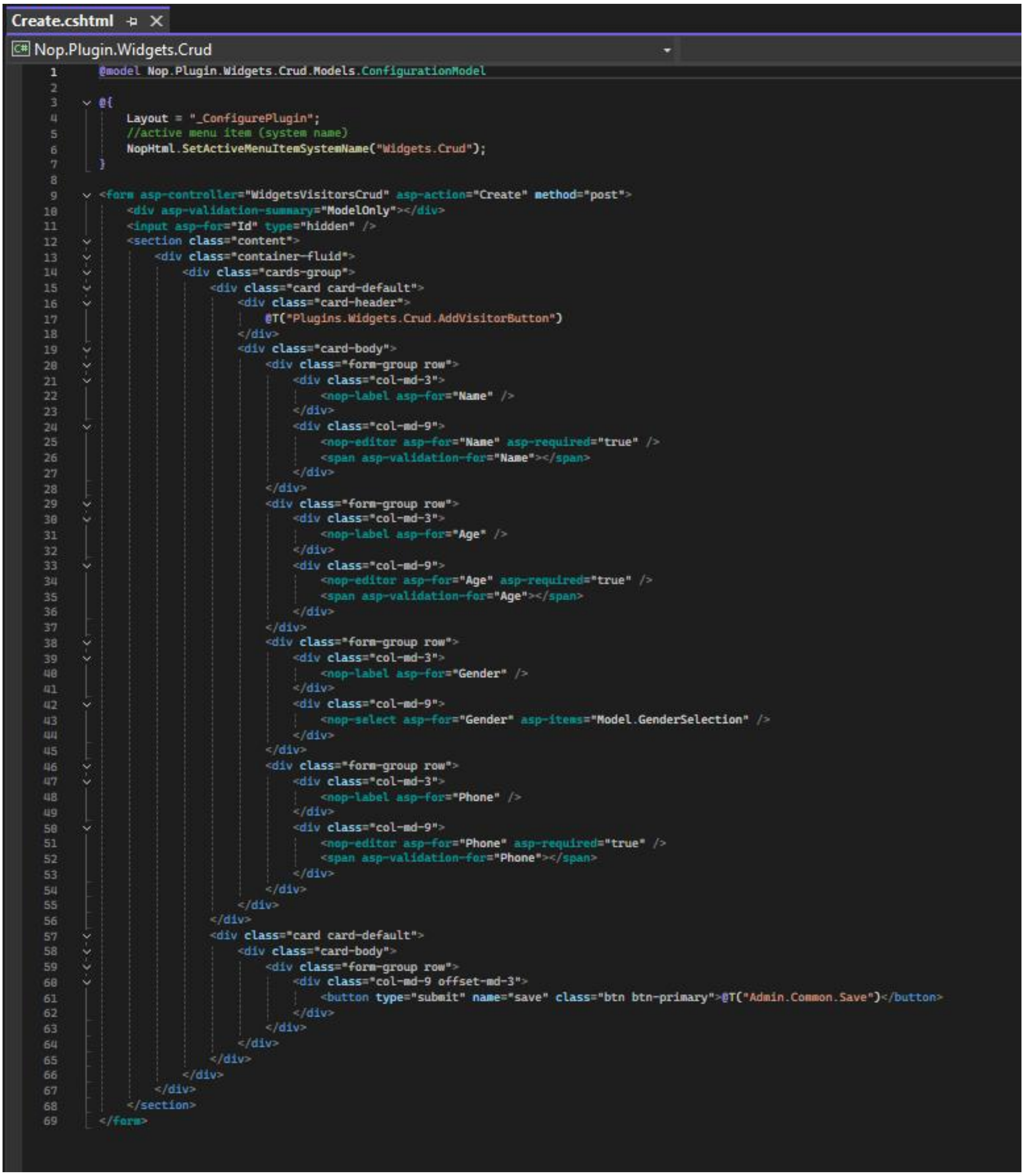
32. Now create all view according to controller and also add _ViewImports.cshtml



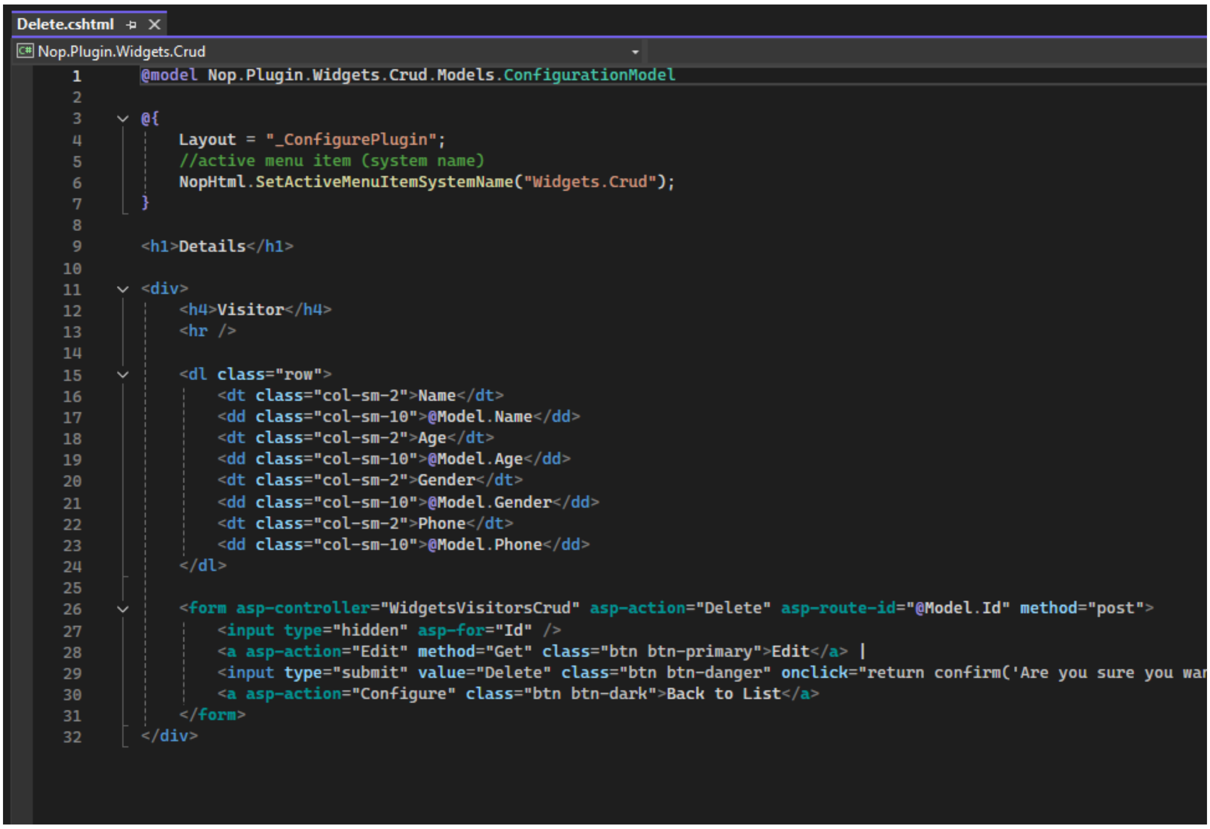
```
@inherits Nop.Web.Framework.Mvc.Razor.NopRazorPage<TModel>
@inject INopHtmlHelper NopHtml
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
@addTagHelper *, Nop.Web.Framework

@using Microsoft.AspNetCore.Mvc.ViewFeatures
@using Nop.Web.Areas.Admin.Components
@using Nop.Web.Framework.UI
@using Nop.Web.Framework.Extensions
@using System.Text.Encodings.Web
```



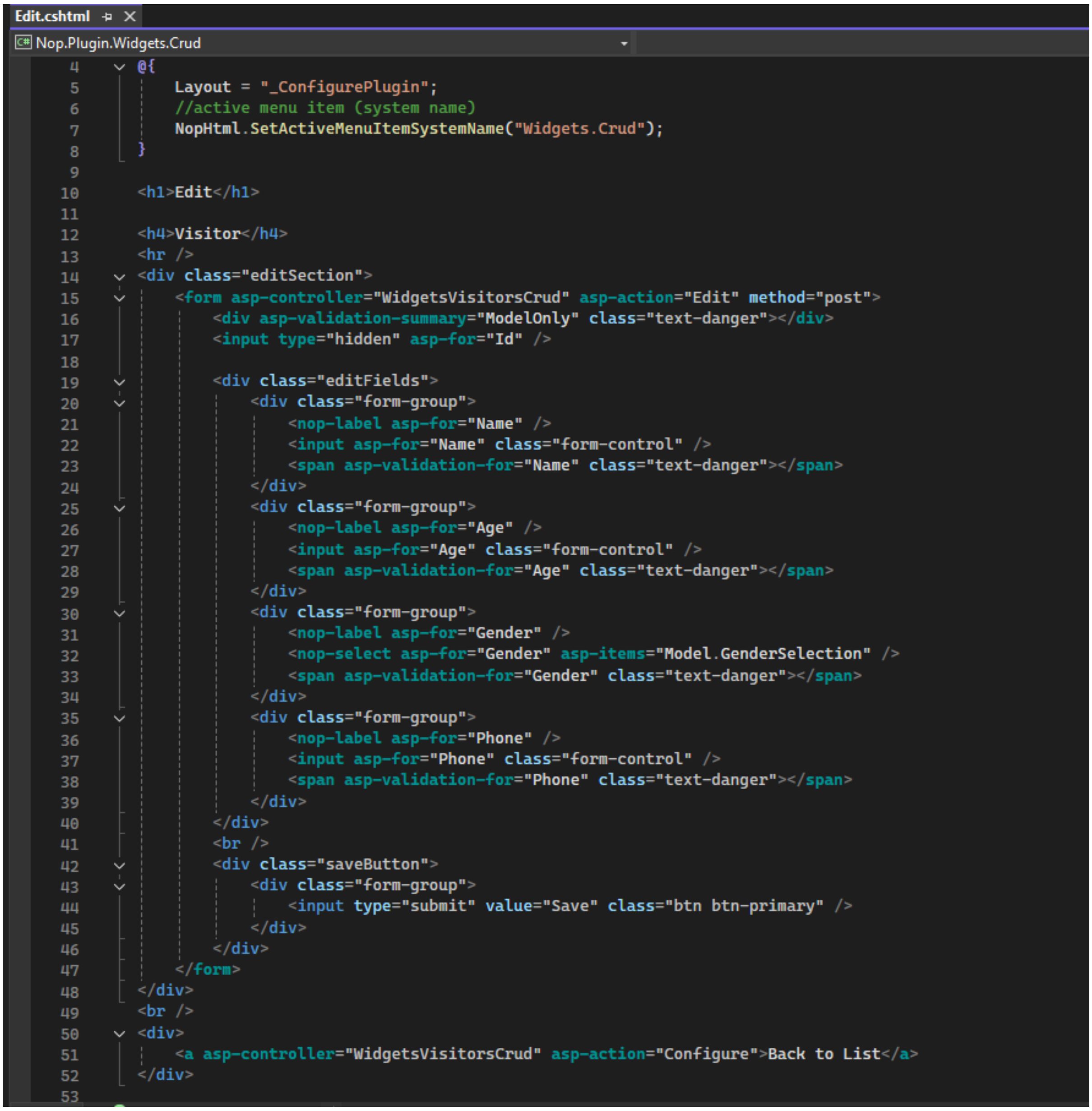
```
@using Nop.Plugin.Widgets.Crud.Models
@model List<ConfigurationModel>

@{
    Layout = "_ConfigurePlugin";
    //active menu item (system name)
    NopHtml.SetActiveMenuItemSystemName("Widgets.Crud");
}

<div class="content-header clearfix">
    <h1 class="float-left">
        Visitors
    </h1>
    <div class="float-right">
        <a asp-controller="WidgetsVisitorsCrud" asp-action="Create" method="Get"><i class="fas fa-plus-square"></i>Add new visitor</a>
    </div>
</div>

<br />
<br />

<section class="content">
    <div class="container-fluid">
        <div class="cards-group">
            <div class="card card-default">
                <div class="card-body">
                    <table class="table">
                        <thead>
                            <tr>
                                <th>Name</th>
                                <th>Age</th>
                                <th>Gender</th>
                                <th>Phone</th>
                                <th></th>
                            </tr>
                        </thead>
                        <tbody>
                            @foreach (var item in Model)
                            {
                                <tr>
                                    <td>@item.Name</td>
                                    <td>@item.Age</td>
                                    <td>@item.Gender</td>
                                    <td>@item.Phone</td>
                                    <td>
                                        <a asp-controller="WidgetsVisitorsCrud" asp-action="Edit" asp-route-id="@item.Id" class="btn btn-primary">Edit</a> |
                                        <a asp-controller="WidgetsVisitorsCrud" asp-action="Delete" asp-route-id="@item.Id" class="btn btn-info">Details</a>
                                    </td>
                                </tr>
                            }
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>
</section>
```



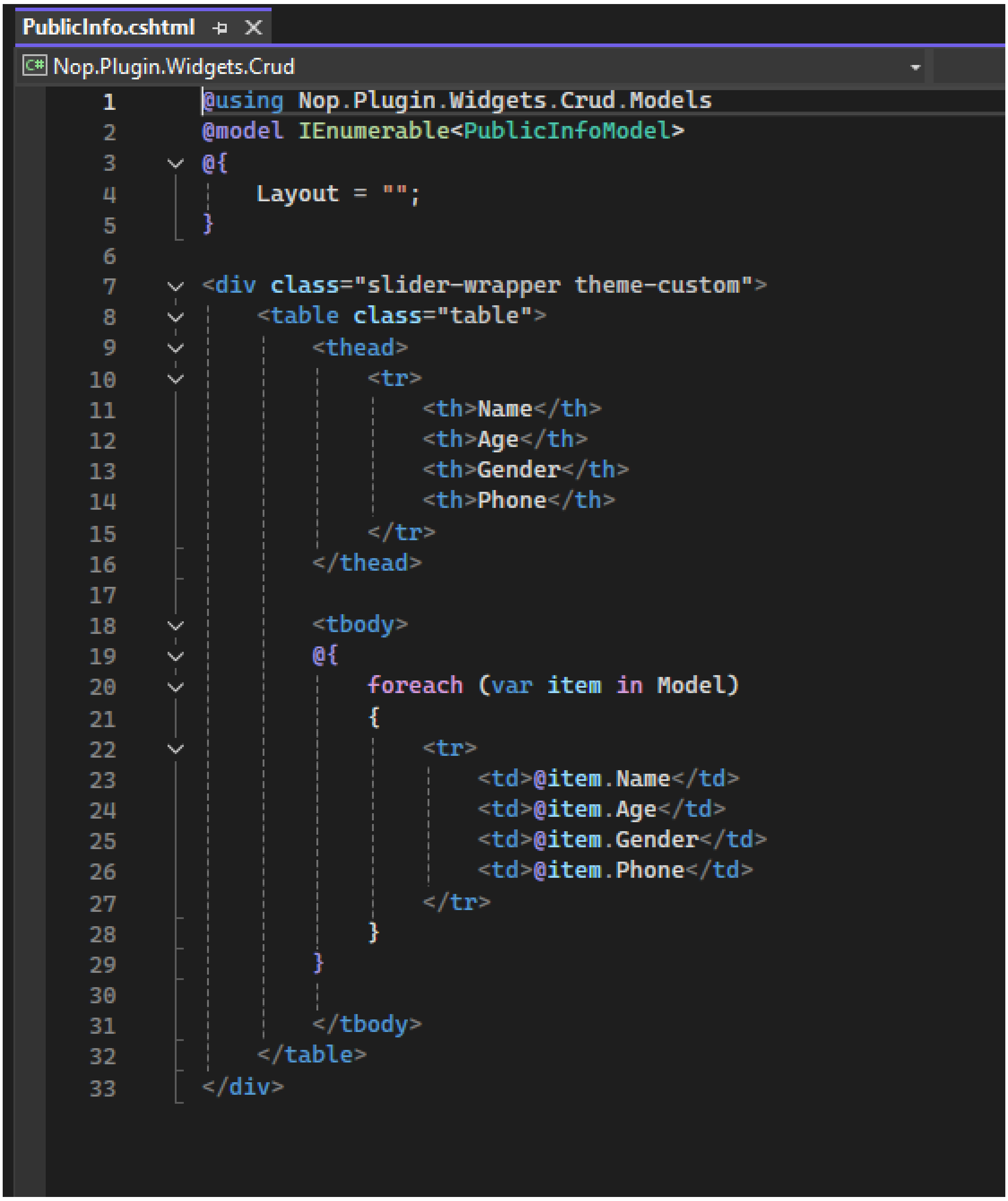
```
@model Nop.Plugin.Widgets.Crud.Models.ConfigurationModel

@{
    Layout = "_ConfigurePlugin";
    //active menu item (system name)
    NopHtml.SetActiveMenuItemSystemName("Widgets.Crud");
}

<form asp-controller="WidgetsVisitorsCrud" asp-action="Create" method="post">
    <div asp-validation-summary="ModelOnly"></div>
    <input asp-for="Id" type="hidden" />
    <section class="content">
        <div class="container-fluid">
            <div class="cards-group">
                <div class="card card-default">
                    <div class="card-header">
                        @T("Plugins.Widgets.Crud.AddVisitorButton")
                    </div>
                    <div class="card-body">
                        <div class="form-group row">
                            <div class="col-md-3">
                                <nop-label asp-for="Name" />
                            </div>
                            <div class="col-md-9">
                                <nop-editor asp-for="Name" asp-required="true" />
                                <span asp-validation-for="Name"></span>
                            </div>
                        </div>
                        <div class="form-group row">
                            <div class="col-md-3">
                                <nop-label asp-for="Age" />
                            </div>
                            <div class="col-md-9">
                                <nop-editor asp-for="Age" asp-required="true" />
                                <span asp-validation-for="Age"></span>
                            </div>
                        </div>
                        <div class="form-group row">
                            <div class="col-md-3">
                                <nop-label asp-for="Gender" />
                            </div>
                            <div class="col-md-9">
                                <nop-select asp-for="Gender" asp-items="Model.GenderSelection" />
                            </div>
                        </div>
                        <div class="form-group row">
                            <div class="col-md-3">
                                <nop-label asp-for="Phone" />
                            </div>
                            <div class="col-md-9">
                                <nop-editor asp-for="Phone" asp-required="true" />
                                <span asp-validation-for="Phone"></span>
                            </div>
                        </div>
                    </div>
                </div>
                <div class="card card-default">
                    <div class="card-body">
                        <div class="form-group row">
                            <div class="col-md-9 offset-md-3">
                                <button type="submit" name="save" class="btn btn-primary">@T("Admin.Common.Save")</button>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </section>
</form>
```



```
@model Nop.Plugin.Widgets.Crud.Models.ConfigurationModel

@{
    Layout = "_ConfigurePlugin";
    //active menu item (system name)
    NopHtml.SetActiveMenuItemSystemName("Widgets.Crud");
}

<h1>Details</h1>

<div>
    <h4>Visitor</h4>
    <hr />

    <dl class="row">
        <dt class="col-sm-2">Name</dt>
        <dd class="col-sm-10">@Model.Name</dd>
        <dt class="col-sm-2">Age</dt>
        <dd class="col-sm-10">@Model.Age</dd>
        <dt class="col-sm-2">Gender</dt>
        <dd class="col-sm-10">@Model.Gender</dd>
        <dt class="col-sm-2">Phone</dt>
        <dd class="col-sm-10">@Model.Phone</dd>
    </dl>

    <form asp-controller="WidgetsVisitorsCrud" asp-action="Delete" asp-route-id="@Model.Id" method="post">
        <input type="hidden" asp-for="Id" />
        <a asp-action="Edit" method="Get" class="btn btn-primary">Edit</a>
        <input type="submit" value="Delete" class="btn btn-danger" onclick="return confirm('Are you sure you war
        <a asp-action="Configure" class="btn btn-dark">Back to List</a>
    </form>
</div>
```

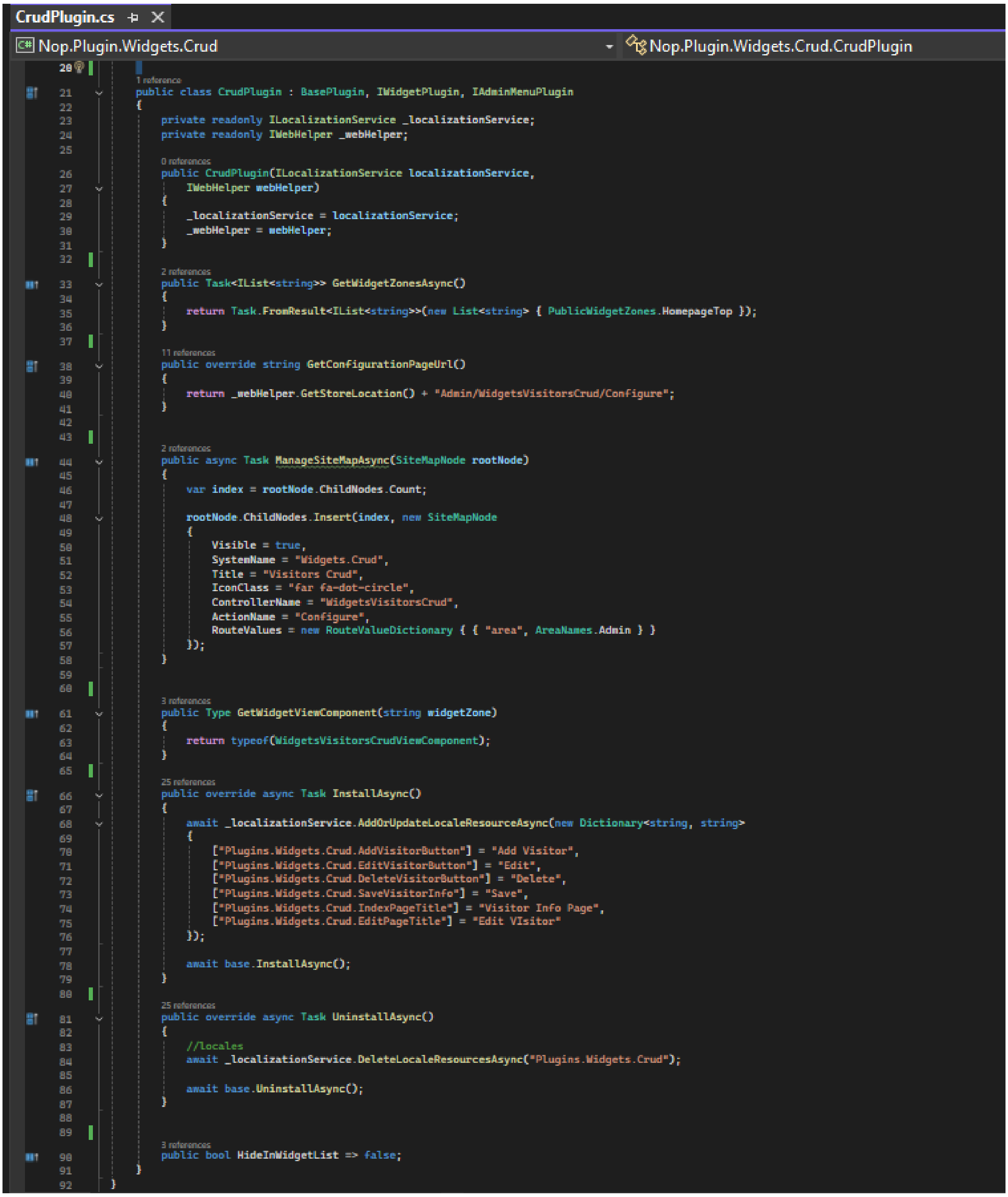


```
@{
    Layout = "_ConfigurePlugin";
    //active menu item (system name)
    NopHtml.SetActiveMenuItemSystemName("Widgets.Crud");
}

<h1>Edit</h1>

<h4>Visitor</h4>
<hr />
<div class="editSection">
    <form asp-controller="WidgetsVisitorsCrud" asp-action="Edit" method="post">
        <div asp-validation-summary="ModelOnly" class="text-danger"></div>
        <input type="hidden" asp-for="Id" />

        <div class="editFields">
            <div class="form-group">
                <nop-label asp-for="Name" />
                <input asp-for="Name" class="form-control" />
                <span asp-validation-for="Name" class="text-danger"></span>
            </div>
            <div class="form-group">
                <nop-label asp-for="Age" />
                <input asp-for="Age" class="form-control" />
                <span asp-validation-for="Age" class="text-danger"></span>
            </div>
            <div class="form-group">
                <nop-label asp-for="Gender" />
                <nop-select asp-for="Gender" asp-items="Model.GenderSelection" />
                <span asp-validation-for="Gender" class="text-danger"></span>
            </div>
            <div class="form-group">
                <nop-label asp-for="Phone" />
                <input asp-for="Phone" class="form-control" />
                <span asp-validation-for="Phone" class="text-danger"></span>
            </div>
        </div>
        <br />
        <div class="saveButton">
            <div class="form-group">
                <input type="submit" value="Save" class="btn btn-primary" />
            </div>
        </div>
    </form>
</div>
<br />
<div>
    <a asp-controller="WidgetsVisitorsCrud" asp-action="Configure">Back to List</a>
</div>
```



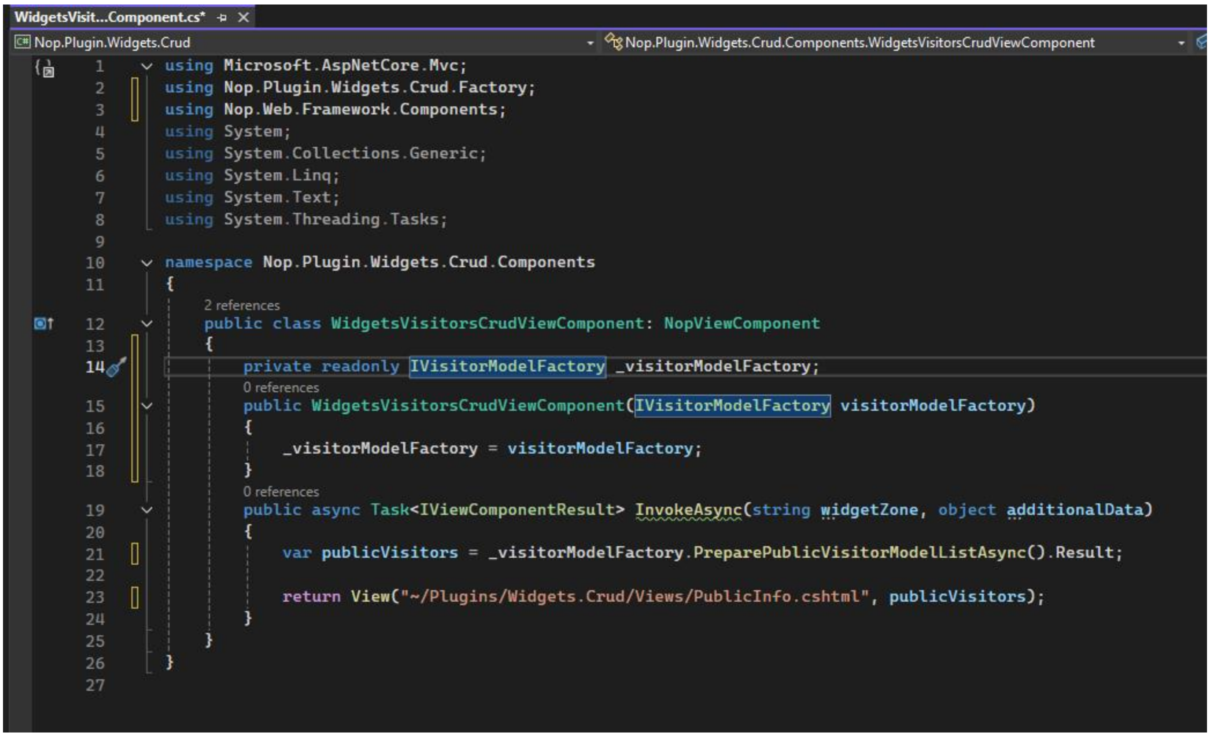
```
@using Nop.Plugin.Widgets.Crud.Models
@model IEnumerable<PublicInfoModel>
@{
    Layout = "";
}

<div class="slider-wrapper theme-custom">
    <table class="table">
        <thead>
            <tr>
                <th>Name</th>
                <th>Age</th>
                <th>Gender</th>
                <th>Phone</th>
            </tr>
        </thead>

        <tbody>
        @{
            foreach (var item in Model)
            {
                <tr>
                    <td>@item.Name</td>
                    <td>@item.Age</td>
                    <td>@item.Gender</td>
                    <td>@item.Phone</td>
                </tr>
            }
        }

        </tbody>
    </table>
</div>
```

33. CrudPlugin inherit from IAdminMenuPlugin



```
public class CrudPlugin : BasePlugin, IWidgetPlugin, IAdminMenuPlugin
{
    private readonly ILocalizationService _localizationService;
    private readonly IWebHelper _webHelper;

    public CrudPlugin(ILocalizationService localizationService,
        IWebHelper webHelper)
    {
        _localizationService = localizationService;
        _webHelper = webHelper;
    }

    public Task<IList<string>> GetWidgetZonesAsync()
    {
        return Task.FromResult<IList<string>>(new List<string> { PublicWidgetZones.HomepageTop });
    }

    public override string GetConfigurationPageUrl()
    {
        return _webHelper.GetStoreLocation() + "Admin/WidgetsVisitorsCrud/Configure";
    }


    public async Task ManageSiteMapAsync(SiteMapNode rootNode)
    {
        var index = rootNode.ChildNodes.Count;

        rootNode.ChildNodes.Insert(index, new SiteMapNode
        {
            Visible = true,
            SystemName = "Widgets.Crud",
            Title = "Visitors Crud",
            IconClass = "far fa-dot-circle",
            ControllerName = "WidgetsVisitorsCrud",
            ActionName = "Configure",
            RouteValues = new RouteValueDictionary { { "area", AreaNames.Admin } }
        });
    }


    public Type GetWidgetViewComponent(string widgetZone)
    {
        return typeof(WidgetsVisitorsCrudViewComponent);
    }

    public override async Task InstallAsync()
    {
        await _localizationService.AddOrUpdateLocaleResourceAsync(new Dictionary<string, string>
        {
            ["Plugins.Widgets.Crud.AddVisitorButton"] = "Add Visitor",
            ["Plugins.Widgets.Crud.EditVisitorButton"] = "Edit",
            ["Plugins.Widgets.Crud.DeleteVisitorButton"] = "Delete",
            ["Plugins.Widgets.Crud.SaveVisitorInfo"] = "Save",
            ["Plugins.Widgets.Crud.IndexPageTitle"] = "Visitor Info Page",
            ["Plugins.Widgets.Crud.EditPageTitle"] = "Edit VIsitor"
        });

        await base.InstallAsync();
    }

    public override async Task UninstallAsync()
    {
        //locales
        await _localizationService.DeleteLocaleResourcesAsync("Plugins.Widgets.Crud");

        await base.UninstallAsync();
    }


    public bool HideInWidgetList => false;
}
}
```

34. Now time to go WidgetsVisitorsCrudViewComponent.cs inside components


WidgetsVisit...Component.cs*
Nop.Plugin.Widgets.Crud
Nop.Plugin.Widgets.Crud.Components.WidgetsVisitorsCrudViewComponent
2 references
0 references
0 references

```csharp
using Microsoft.AspNetCore.Mvc;
using Nop.Plugin.Widgets.Crud.Factory;
using Nop.Web.Framework.Components;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Nop.Plugin.Widgets.Crud.Components
{
    public class WidgetsVisitorsCrudViewComponent: NopViewComponent
    {
        private readonly IVisitorModelFactory _visitorModelFactory;
        public WidgetsVisitorsCrudViewComponent(IVisitorModelFactory visitorModelFactory)
        {
            _visitorModelFactory = visitorModelFactory;
        }
        public async Task<IViewComponentResult> InvokeAsync(string widgetZone, object additionalData)
        {
            var publicVisitors = _visitorModelFactory.PreparePublicVisitorModelListAsync().Result;

            return View("~/Plugins/Widgets.Crud/Views/PublicInfo.cshtml", publicVisitors);
        }
    }
}

```