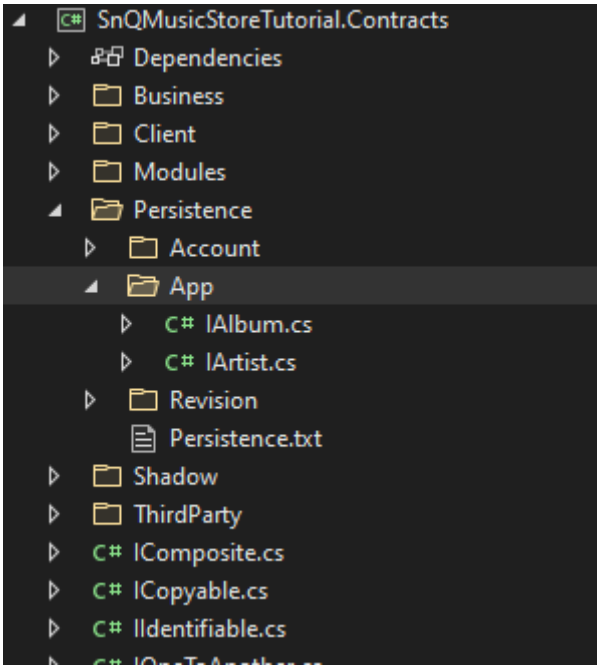
Tutorial

Schritt 1:



Schritt 2:

```
namespace SnQMusicStoreTutorial.Contracts.Persistence.App
{
    1 reference
    public interface IAlbum : ICopyable<IAlbum>, IVersionable
    {
        [ContractPropertyInfo(MaxLength = 128, Required = true)]
        0 references
        string Name { get; set; }
        [ContractPropertyInfo(MaxLength = 512)]
        0 references
        string Description { get; set; }
    }
}
```

Schritt 3:

```
PM> add-migration initDb
Build started...
Build succeeded.
To undo this action, use Remove-Migration.
PM> update-database
Build started...
```

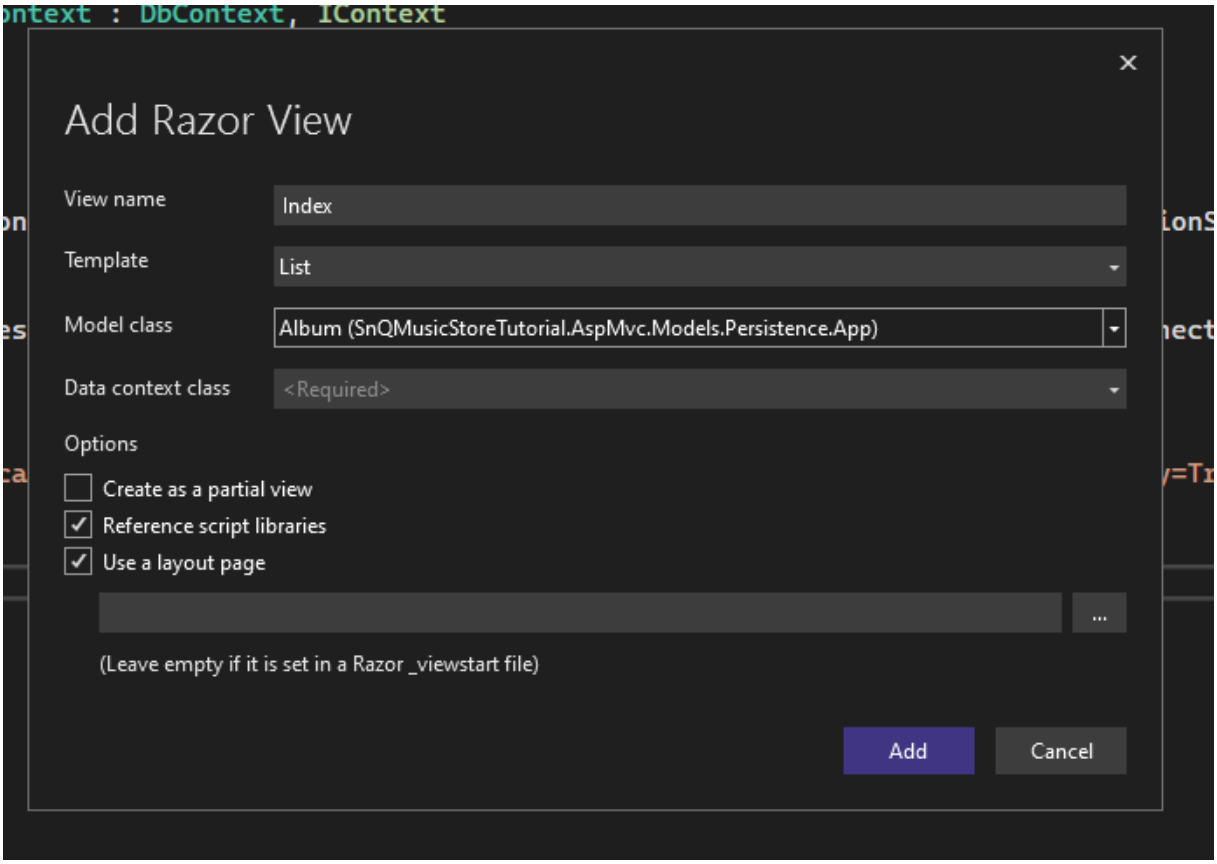
Schritt 4:

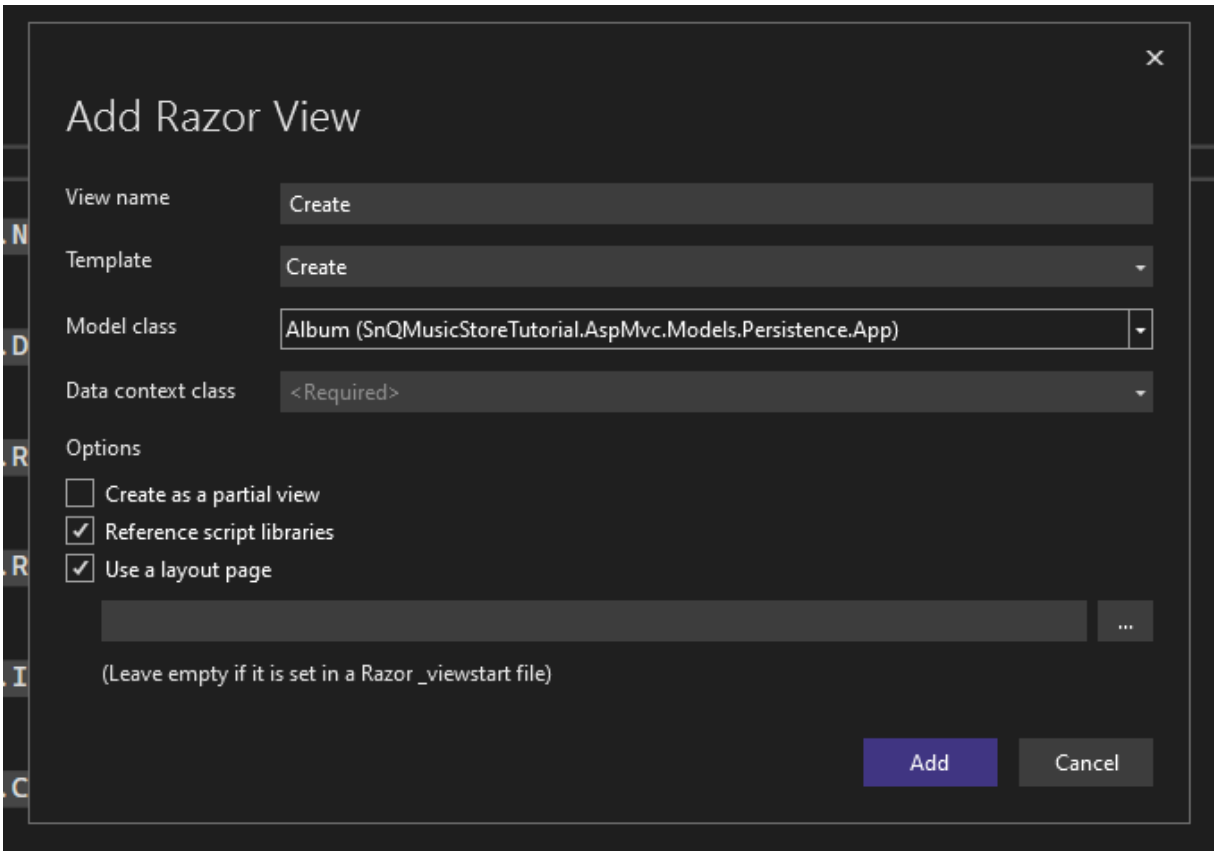
```
appsettings.Development.json   appsettings.json   SnQMusicStor...IDbContext.cs   IntelliCode   20220208204441_initDb.cs   IArtist.cs   IAlbum.cs
SnQMusicStoreTutorial.Logic   SnQMusicStoreTutorial.Logic.DataContext.SnQMusicStoreTutorialDbContext   SnQMusicStoreTutorialDbContext()

4   using SnQMusicStoreTutorial.Logic.Entities;
5   using System.Linq;
6   using System.Linq.Dynamic.Core;
7   using System.Threading.Tasks;
8
9   namespace SnQMusicStoreTutorial.Logic.DataContext
10  {
        6 references
11      internal partial class SnQMusicStoreTutorialDbContext : DbContext, IContext
12      {
            0 references
13          static SnQMusicStoreTutorialDbContext()
14          {
15              ClassConstructing();
16              ConnectionString = CommonBase.Modules.Configuration.AppSettings.Configuration[StaticLiterals.EnvironmentConnectionStringKey];
17              if (ConnectionString.IsNullOrEmpty())
18              {
19                  ConnectionString = CommonBase.Modules.Configuration.AppSettings.Configuration[StaticLiterals.AppSettingsConnectionStringKey];
20              }
21              if (ConnectionString.IsNullOrEmpty())
22              {
23                  ConnectionString = "Data Source=(localdb)\\MSSQLLocalDB;Database=SnQMusicStoreTutorialDb;Integrated Security=True";
24              }
25              ClassConstructed();
```

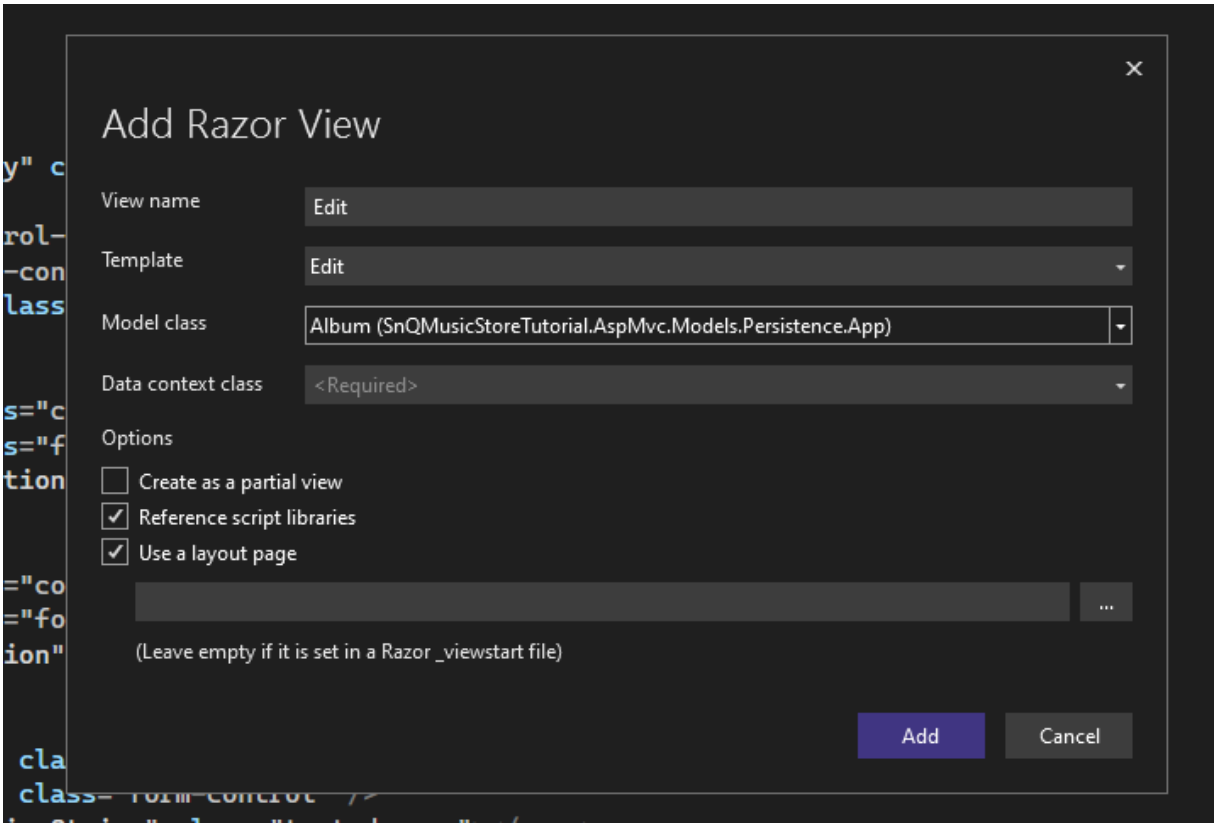
Einfügen

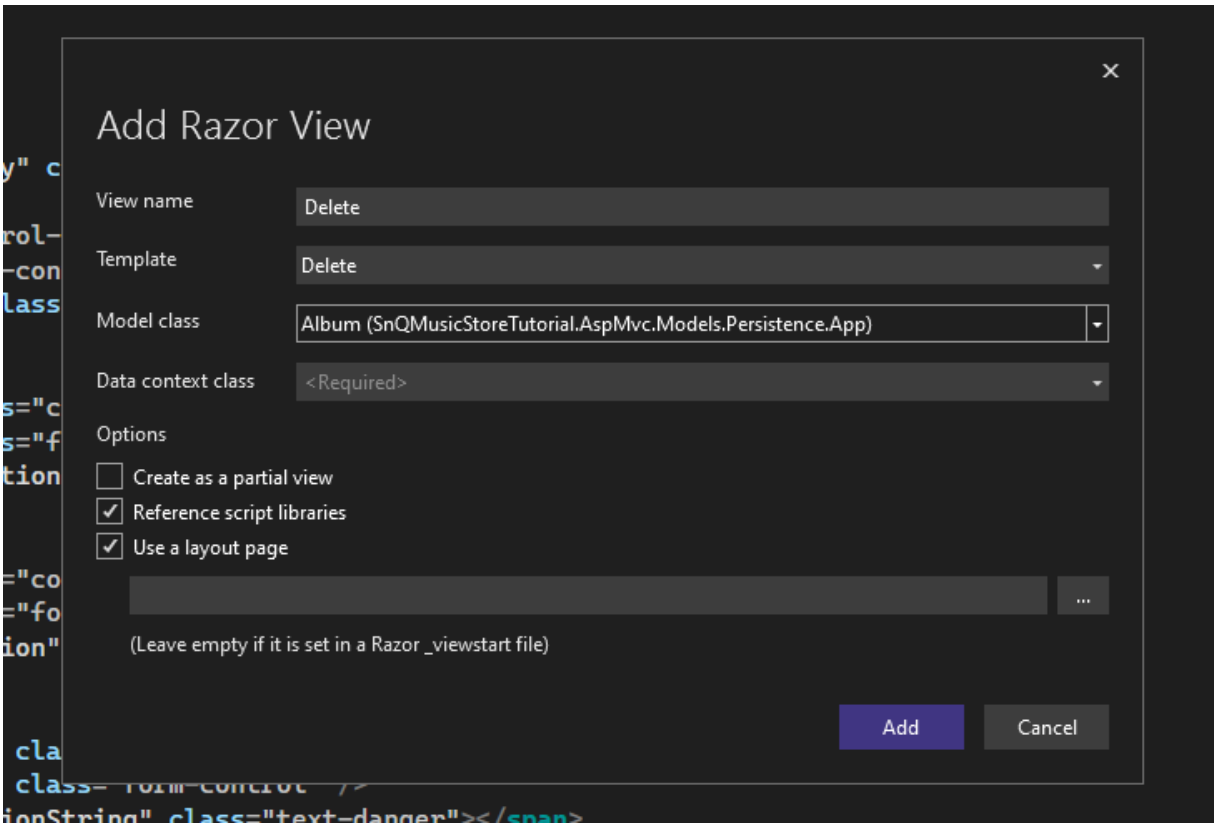
```
appsettings.D...elopment.json   appsettings.json   SnQMusicStore...IDbContext.cs   IntelliCode   20220208204441_initDb.cs   IArtist.cs   IAlbum.cs
Schema: <No Schema Selected>

1   {
2     "ConnectionStrings": {
3       "DefaultConnection": "Data Source=(localdb)\\MSSQLLocalDB;Database=SnQMusicStoreTutorialDb;Integrated Security=True"
4     }
5   }
6
```

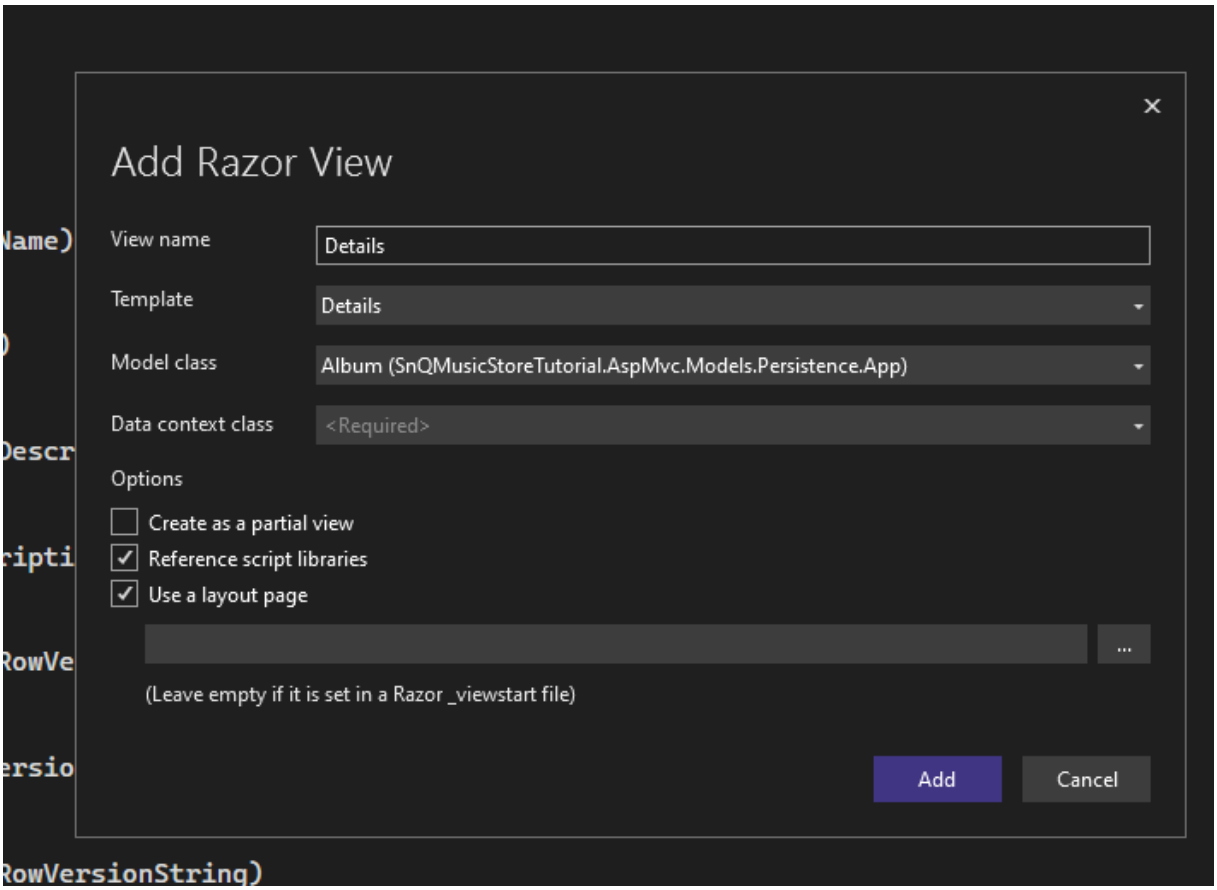
Schritt 5:




Add Razor View
View name
Create
Template
Create
Model class
Album (SnQMusicStoreTutorial.AspMvc.Models.Persistence.App)
Data context class
<Required>
Options
Create as a partial view
Reference script libraries
Use a layout page
...
(Leave empty if it is set in a Razor _viewstart file)
Add
Cancel


Add Razor View
View name
Edit
Template
Edit
Model class
Album (SnQMusicStoreTutorial.AspMvc.Models.Persistence.App)
Data context class
<Required>
Options
Create as a partial view
Reference script libraries
Use a layout page
...
(Leave empty if it is set in a Razor _viewstart file)
Add
Cancel


Add Razor View
View name
Delete
Template
Delete
Model class
Album (SnQMusicStoreTutorial.AspMvc.Models.Persistence.App)
Data context class
<Required>
Options
Create as a partial view
Reference script libraries
Use a layout page
(Leave empty if it is set in a Razor _viewstart file)
Add
Cancel


Add Razor View
View name
Details
Template
Details
Model class
Album (SnQMusicStoreTutorial.AspMvc.Models.Persistence.App)
Data context class
<Required>
Options
Create as a partial view
Reference script libraries
Use a layout page
(Leave empty if it is set in a Razor _viewstart file)
Add
Cancel

Schritt 6:

```
_PortalMenu.cshtml  Details.cshtml  Delete.cshtml  Edit.cshtml  Create.cshtml  Index.cshtml  appsettings.Development.json  appsettings.json  SnQMusicStore...IDbContext.cs  IntelliCode
1   @*@CodeCopy*@
2   @{
3       @using SnQMusicStoreTutorial.AspMvc.Modules.Session
4       @using SnQMusicStoreTutorial.AspMvc.Modules.Language
5
6       var sessionWrapper = new SessionWrapper(Context.Session);
7       Func<string, string> translate = Translator.TranslateIt;
8       Func<string, string> translateFor = pn => translate($"PortalMenu.{pn}");
9
10      <ul class="navbar-nav flex-grow-1">
11          <li class="nav-item">
12              <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">@translateFor("Home")</a>
13          </li>
14          <li class="nav-item">
15              <a class="nav-link text-dark" asp-area="" asp-controller="Albums" asp-action="Index">@translateFor("Albums")</a>
16          </li>
17  @*#if ACCOUNT_ON*@@*
```

Schritt 7:

```
GenericController.cs  Details.cshtml  Delete.cshtml  Edit.cshtml  Create.cshtml  Index.cshtml  appsettings.Development.json
1   @model SnQMusicStoreTutorial.AspMvc.Models.Persistence.App.Album
2
3   @{
4       ViewData["Title"] = "Create";
5   }
6
7   <h1>Create</h1>
8
9   <h4>Album</h4>
10  <hr />
11  <div class="row">
12      <div class="col-md-4">
13          <form asp-action="Insert" method="post">
14              <div asp-validation-summary="ModelOnly" class="text-danger"></div>
15              <div class="form-group">
16                  <label asp-for="Name" class="control-label"></label>
17                  <input asp-for="Name" class="form-control" />
18                  <span asp-validation-for="Name" class="text-danger"></span>
19              </div>
20              <div class="form-group">
21                  <label asp-for="Description" class="control-label"></label>
22                  <input asp-for="Description" class="form-control" />
23                  <span asp-validation-for="Description" class="text-danger"></span>
24              </div>
25              <div hidden class="form-group">
26                  <label asp-for="RowVersion" class="control-label"></label>
27                  <input asp-for="RowVersion" class="form-control" />
28                  <span asp-validation-for="RowVersion" class="text-danger"></span>
29              </div>
```

Schritt 8:

In Index id=item.Id ändern

```
            @Html.DisplayFor(modelItem => item.Id)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.CommandMode)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.Id }) |
            @Html.ActionLink("Details", "Details", new { id=item.Id }) |
            @Html.ActionLink("Delete", "Delete", new { id=item.Id })
        </td>
    </tr>

</tbody>
```

Schritt 9:

In Edit auf Update ändern

```
    <h1>Edit</h1>

    <h4>Album</h4>
    <hr />
    <div class="row">
        <div class="col-md-4">
            <form asp-action="Update">
                <div asp-validation-summary="ModelOnly" class="text-danger"></div>
                <div class="form-group">
                    <label asp-for="Name" class="control-label"></label>
                    <input asp-for="Name" class="form-control" />
                    <span asp-validation-for="Name" class="text-danger"></span>
```

Schritt 10:

GenericController -> STRG-f -> indexasync ->return View(models) ändern

```
            models = await QueryModelPageListAsync(pageIndex, pageSize, true).ConfigureAwait(false
        }
        catch (Exception ex)
        {
            LastViewError = ex.GetError();
        }
    }
    AfterIndex(models);
    return View(models);
}
4 references
partial void BeforeIndex(ref IEnumerable<TModel> models, ref bool handled);
```

Schritt 11:

GenericController -> STRG-f -> createasync ->return View(“Create ”models) ändern

```
            LastViewError = ex.GetError();
        }
    }
    AfterCreate(model);
    if (HasError == false)
    {
        model = BeforeView(model, ActionMode.Create);
        model = await BeforeViewAsync(model, ActionMode.Create).ConfigureAwait(false);
    }
    return HasError ? RedirectToAction("Index") : View("Create", model);

[HttpGet]
```

Schritt 12:

GenericController -> STRG-f -> insertasync ->return Return RedirectToAction(“index”) ändern

```
        }
        AfterInsertModel(model);
        if (HasError == false)
        {
            model = BeforeView(model, ActionMode.Create);
            model = await BeforeViewAsync(model, ActionMode.Create).ConfigureAwait(false);
        }
        return RedirectToAction("Index");
    }
    1 reference
```

Schritt 13:

GenericController -> STRG-f -> editasync->return View(“Edit” ,models) ändern

```
            {
                LastViewError = ex.GetError();
            }
        }
        AfterEdit(model);
        if (HasError == false)
        {
            model = BeforeView(model, ActionMode.Edit);
            model = await BeforeViewAsync(model, ActionMode.Edit).ConfigureAwait(false);
        }
        return HasError ? RedirectToAction("Index") : View("Edit",model);
    }
    1 Verweis
```

Schritt 14:

GenericController -> STRG-f -> updateAsync->Return RedirectToAction(“index”) ändern

```
        catch (Exception ex)
        {
            LastViewError = ex.GetError();
        }
        if (HasError == false)
        {
            model = BeforeView(model, ActionMode.Edit);
            model = await BeforeViewAsync(model, ActionMode.Edit).ConfigureAwait(false);
        }
        return RedirectToAction("Index");
    }
```

Schritt 14:

GenericController -> STRG-f -> updateAsync->Return ReturnDeleteView(model)

```
        }
    }
    AfterDelete(model);
    if (HasError == false)
    {
        model = BeforeView(model, ActionMode.Delete);
        model = await BeforeViewAsync(model, ActionMode.Delete).ConfigureAwait(false);
    }
    return HasError ? RedirectToAction("Index") : View("Delete", model);
}
```

1. Entity Framework ist ein ORMapper
2. Es gibt Zwei Typen: Value Type (kann nicht Null sein und wird am Stack angelegt), Reference Type wird am (Heap angelegt und kann null sein).
3. Reference Datentyp kann Null sein, aber er muss nicht Null sein aber wenn er Null sein kann muss man es spezifizieren (Man muss es kennzeichnen und der Compiler muss das jedes Mal prüfen)
4. sealed bei Klassen: Man kann nicht mehr ableiten
5. not Mapped für ein Feld, dann befindet sich das Feld nicht auf einer Datenbank.
6. Public get und Internal set für ein Feld welches berechnet werden soll in der Logik.
7. OrMapper :Eigenschaften: Transferiert ein Objekt in eine Datenbank und umgekehrt

8. Jedes Entity muss identifieziert sein
9. Klassen mit Nomen bennant werden
10. Internal nur in dme Projekt zugreifbar