Edit

Creating a New Person

Next step is to create a modal to add a new item to phone book.

Add a CreatePerson Method to PersonAppService

We first define **CreatePerson** method in **IPersonAppService** interface:

```
Task CreatePerson(CreatePersonInput input);
```

Then we create **CreatePersonInput** DTO that defines parameters of the method:

```
public class CreatePersonInput
{
    [Required]
    [MaxLength(PersonConsts.MaxNameLength)]
    public string Name { get; set; }

    [Required]
    [MaxLength(PersonConsts.MaxSurnameLength)]
    public string Surname { get; set; }

    [EmailAddress]
    [MaxLength(PersonConsts.MaxEmailAddressLength)]
    public string EmailAddress { get; set; }
}
```

Then we add configuration for AutoMapper into CustomDtoMapper.cs like below:

Feedback

CreatePersonInput is mapped to **Person** entity (comment out related line in CustomDtoMapper.cs and we will use mapping below). All properties are decorated with **data annotation attributes** to provide automatic **validation**. Notice that we use same consts defined in **PersonConsts.cs** in **.Core.Shared** project for **MaxLength** properties. After adding this class, you can remove consts from **Person** entity and use this new consts class.

```
public class PersonConsts
{
    public const int MaxNameLength = 32;
    public const int MaxSurnameLength = 32;
    public const int MaxEmailAddressLength = 255;
}
```

Here, the implementation of CreatePerson method:

```
public async Task CreatePerson(CreatePersonInput input)
{
    var person = ObjectMapper.Map<Person>(input);
    await _personRepository.InsertAsync(person);
}
```

A Person entity is created by mapping given input, then inserted to database. We used **async/await** pattern here. All methods in ASP.NET Zero startup project is **async**. It's advised to use async/await wherever possible.

Next

- Testing CreatePerson Method