Edit

Multi Tenancy

We have built a fully functional application until here. Now, we will see how to convert it to a multi-tenant application easily. Logout from the application before any change.

Enable Multi Tenancy

We disabled multi-tenancy at the beginning of this document. Now, re-enabling it in **PhoneBookDemoConsts** class:

```
public const bool MultiTenancyEnabled = true;
```


Make Entities Multi Tenant

In a multi-tenant application, a tenant's entities should be isolated by other tenants. For this example project, every tenant should have own phone book with isolated people and phone numbers.

When we implement IMustHaveTenant interface, ABP automatically filters data based on current Tenant, while retrieving entities from database. So, we should declare that Person entity must have a tenant using **IMustHaveTenant** interface:

```
public class Person : FullAuditedEntity, IMustHaveTenant
{
    public virtual int TenantId { get; set; }

    //...other properties
}
```

We may want to add IMustHaveTenant interface to also Phone entity. This is needed if we directly use phone repository to get phones. In this sample project, it's not needed.

This command creates a new code-first database migration. The migration class adds an annotation this is needed for automatic filtering. We don't have to know what it is since it's done automatically. It also adds a **TenantId** column to PbPersons table as shown below:

```
migrationBuilder.AddColumn<int>(name: "TenantId",table: "PbPersons",nullable: false
```

I added **defaultValue as 1** to AddColumn options. Thus, current people are automatically assigned to **default tenant** (default tenant's id is always 1).

Now, we can update the database again:

```
Update-Database
```

Next

- Running the Application