Add, Save, and Delete Data Labs

Perform these labs on your own computer using Visual Studio 2022 or later to ensure you understand the lessons presented in the corresponding videos and lectures.

Lab 1: Display a Blank Record

Open the **Views\UserListView.xaml** file and within the Informational Message Area, just before the closing </HorizontalStackLayout> tag, add the following.

```
<Button Text="Add"
        HorizontalOptions="Start"
        Command="{Binding AddCommand}" />
```

Open the **CommandClasses\UserViewModelCommands.cs** file and add a new ICommand definition.

```
public ICommand? AddCommand { get; private set; }
```

Create a new instance of the ICommand object within the Init() method.

```
AddCommand = new Command(async () => await AddAsync(),
() => true);
```

Add a new method in this class named **AddAsync**().

```
#region AddAsync Method
protected async Task AddAsync()
{
  await
Shell.Current.GoToAsync($"{nameof(Views.UserDetailView)}
?id={0}&isAdding={true}");
}
#endregion
```

Open the **ViewModelClasses\UserViewModel.cs** file in the **AdventureWorks.ViewModelLayer** project and add a new method named **CreateEmpty**().

```
#region CreateEmpty Method
public User CreateEmpty()
{
  User ret = new() {
    UserId = 0,
    LoginId = "BugsBunny123",
    FirstName = "Bugs",
    LastName = "Bunny",
    Email = "BugsBunny@acme.com",
    Password = "P@ssw0rd!",
    Phone = "(615) 123-9876",
    PhoneType = "Mobile",
    IsFullTime = true,
    IsEnrolledIn401k = true,
    IsEnrolledInHealthCare = false,
    IsEnrolledInHSA = false,
    IsEnrolledInFlexTime = true,
    IsActive = true,
    BirthDate = Convert.ToDateTime("7/27/1940"),
    StartTime = new TimeSpan(6, 0, 0),
  };

  return ret;
}
#endregion
```

Open the **Views\UserDetailView.xaml.cs** file and within the OnAppearing() event, add the code shown in bold below.

```
ViewModel.IsAdding = IsAdding;
if (IsAdding) {
  ViewModel.CurrentEntity = ViewModel.CreateEmpty();
}
else {
  // Retrieve a User
  await ViewModel.GetAsync(UserId);
}
```

Try It Out

Run the application and click on the **Users** menu.

Click the **Add** button and you should see a "Bugs Bunny" user record appear on the User Information page.

NOTE:	The Save button will NOT work yet.

Lab 2: Save Data

Open the **Interfaces\IRepository.cs** file in the **Common.Library** project and add three new interface methods.

```
Task<TEntity?> InsertAsync(TEntity? entity);
Task<TEntity?> UpdateAsync(TEntity? entity);
Task<TEntity?> DeleteAsync(TEntity? entity);
```

Copy Files

Copy the files from the **_SampleCode\RepositoryClasses-API** into the **AdventureWorks.DataLayer.API\RepositoryClasses** folder.

Copy the files from the **_SampleCode\RepositoryClasses-EF** into the **AdventureWorks.DataLayer.EF\RepositoryClasses** folder.

Copy the files from the **_SampleCode\RepositoryClasses-Mock** into the **AdventureWorks.DataLayer.Mock\RepositoryClasses** folder.

Copy the files from the **_SampleCode\RouterClasses** into the **AdventureWorks.MinWebAPI\RouterClasses** folder.

Copy the files from the **_SampleCode\ViewModelClasses** into the **AdventureWorks.ViewModelLayer\ViewModelClasses** folder.

Try It Out

Run the application and click on the **Users** menu.

Click the **Add** button and you should see a blank record appear on the User Information page.

Fill in valid data, then click on the **Save** button and the information should appear at the end of the list.

Click on the **Edit** button for the data you just added and modify some information. Click on the **Save** button to see the data changed.

Lab 3: Delete Data

Open the **Views\UserListView.xaml** file and locate the Delete button.

```
<Button Text="Delete"
        CommandParameter="{Binding UserId}"
        Command="{Binding Source={x:Reference
UserListPage}, Path=BindingContext.DeleteCommand}" />
```

Open the **CommandClasses\UserViewModelCommands.cs** file and add a new using statement.

```
using AdventureWorks.MAUI.Views;
```

Add a public property.

```
#region Public Properties
public UserListView? TheView { get; set; }
#endregion
```

Add a new ICommand.

```
public ICommand? DeleteCommand { get; private set; }
```

Map the DeleteCommand in the Init() method.

```
DeleteCommand = new Command<int>(async (int id) => await
DeleteAsync(id), (id) => true);
```

Add a new method.

```
#region DeleteAsync Method
protected new async Task DeleteAsync(int id)
{
  if (TheView != null) {
    // Ask user for confirmation
    string action = await
TheView.DisplayActionSheet("Delete this User?",
"Cancel", "Delete");
    if (action == "Delete") {
      // Perform delete
      User? response = await base.DeleteAsync(id);
      if (response != null) {
        InfoMessage = "User Deleted";
      }

      // Redisplay list
      await GetAsync();
    }
  }
}
#endregion
```

Open the **Views\UserListView.xaml.cs** file and modify the OnAppearing() method to set the **TheView** property.

```
protected async override void OnAppearing()
{
  base.OnAppearing();

  // Set 'TheView' property to 'this' so you can display
alerts
  ViewModel.TheView = this;

  BindingContext = ViewModel;

  await ViewModel.GetAsync();
}
```

Try It Out

Run the application and click on the **Users** menu.

Click on the **Delete** button for the "Bugs Bunny" user you added, and you should get a prompt asking if you really wish to delete this user. Click the **Delete** button.