Validating Data Using Data Annotations in .NET MAUI 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: Add Data Annotations to Entity Class

Open the **EntityClasses\User.cs** file in the AdventureWorks.EntityLayer project and replace the entire file with the following code.

```
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

using System.Text.Json.Serialization;
using Common.Library;

namespace AdventureWorks.EntityLayer;

/// <summary>
/// This class contains properties that map to each
field in the dbo.User table.
/// </summary>
[Table("User", Schema = "dbo")]
public partial class User : EntityBase
{
  #region Private Variables
  private int _UserId;
  private string _LoginId = string.Empty;
  private string _FirstName = string.Empty;
  private string _LastName = string.Empty;
  private string _Email = string.Empty;
  private string _Password = "P@ssW0Rd";
  private string _Phone = string.Empty;
  private string? _PhoneType;
  private bool? _IsFullTime;
  private bool? _IsEnrolledIn401k;
  private bool? _IsEnrolledInHealthCare;
  private bool? _IsEnrolledInHSA;
  private bool? _IsEnrolledInFlexTime;
  private bool? _IsActive;
  private DateTime? _BirthDate;
  private TimeSpan? _StartTime;
  #endregion

  #region Public Properties
  /// <summary>
  /// Get/Set User Id
  /// </summary>
  [Key]
  [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
  [Display(Name = "User Id")]
  [Required(ErrorMessage = "{0} must be filled in.")]
  [Column("UserId", TypeName = "int")]
  [JsonPropertyName("userId")]
  public int UserId
  {
    get { return _UserId; }
```

```
    set
    {
      _UserId = value;
      RaisePropertyChanged(nameof(UserId));
    }
  }

  /// <summary>
  /// Get/Set Login Id
  /// </summary>
  [Display(Name = "Login Id")]
  [Required(ErrorMessage = "{0} must be filled in.")]
  [Column("LoginId", TypeName = "nvarchar")]
  [StringLength(20, MinimumLength = 0, ErrorMessage =
"{0} must be between {2} and {1} characters long.")]
  [JsonPropertyName("loginId")]
  public string LoginId
  {
    get { return _LoginId; }
    set
    {
      _LoginId = value;
      RaisePropertyChanged(nameof(LoginId));
    }
  }

  /// <summary>
  /// Get/Set First Name
  /// </summary>
  [Display(Name = "First Name")]
  [Required(ErrorMessage = "{0} must be filled in.")]
  [Column("FirstName", TypeName = "nvarchar")]
  [StringLength(50, MinimumLength = 0, ErrorMessage =
"{0} must be between {2} and {1} characters long.")]
  [JsonPropertyName("firstName")]
  public string FirstName
  {
    get { return _FirstName; }
    set
    {
      _FirstName = value;
      RaisePropertyChanged(nameof(FirstName));
    }
  }

  /// <summary>
  /// Get/Set Last Name
```

```
  /// </summary>
  [Display(Name = "Last Name")]
  [Required(ErrorMessage = "{0} must be filled in.")]
  [Column("LastName", TypeName = "nvarchar")]
  [StringLength(50, MinimumLength = 0, ErrorMessage =
"{0} must be between {2} and {1} characters long.")]
  [JsonPropertyName("lastName")]
  public string LastName
  {
    get { return _LastName; }
    set
    {
      _LastName = value;
      RaisePropertyChanged(nameof(LastName));
    }
  }

  /// <summary>
  /// Get/Set Email
  /// </summary>
  [Display(Name = "Email")]
  [Required(ErrorMessage = "{0} must be filled in.")]
  [Column("Email", TypeName = "nvarchar")]
  [StringLength(256, MinimumLength = 0, ErrorMessage =
"{0} must be between {2} and {1} characters long.")]
  [EmailAddress()]
  [JsonPropertyName("email")]
  public string Email
  {
    get { return _Email; }
    set
    {
      _Email = value;
      RaisePropertyChanged(nameof(Email));
    }
  }

  /// <summary>
  /// Get/Set Password
  /// </summary>
  [Display(Name = "Password")]
  [Required(ErrorMessage = "{0} must be filled in.")]
  [Column("Password", TypeName = "nvarchar")]
  [StringLength(50, MinimumLength = 0, ErrorMessage =
"{0} must be between {2} and {1} characters long.")]
  [JsonPropertyName("password")]
  public string Password
```

```
  {
    get { return _Password; }
    set
    {
      _Password = value;
      RaisePropertyChanged(nameof(Password));
    }
  }

  /// <summary>
  /// Get/Set Phone
  /// </summary>
  [Display(Name = "Phone")]
  [Required(ErrorMessage = "{0} must be filled in.")]
  [Column("Phone", TypeName = "nvarchar")]
  [StringLength(25, MinimumLength = 0, ErrorMessage =
"{0} must be between {2} and {1} characters long.")]
  [Phone()]
  [JsonPropertyName("phone")]
  public string Phone
  {
    get { return _Phone; }
    set
    {
      _Phone = value;
      RaisePropertyChanged(nameof(Phone));
    }
  }

  /// <summary>
  /// Get/Set Phone Type
  /// </summary>
  [Display(Name = "Phone Type")]
  [Column("PhoneType", TypeName = "nvarchar")]
  [StringLength(10, MinimumLength = 0, ErrorMessage =
"{0} must be between {2} and {1} characters long.")]
  [JsonPropertyName("phoneType")]
  public string? PhoneType
  {
    get { return _PhoneType; }
    set
    {
      _PhoneType = value;
      RaisePropertyChanged(nameof(PhoneType));
    }
  }
```

```
  /// <summary>
  /// Get/Set Is Full Time
  /// </summary>
  [Display(Name = "Is Full Time")]
  [Column("IsFullTime", TypeName = "bit")]
  [JsonPropertyName("isFullTime")]
  public bool? IsFullTime
  {
    get { return _IsFullTime; }
    set
    {
      _IsFullTime = value;
      RaisePropertyChanged(nameof(IsFullTime));
    }
  }

  /// <summary>
  /// Get/Set Is Enrolled In 40 1K
  /// </summary>
  [Display(Name = "Is Enrolled In 40 1K")]
  [Column("IsEnrolledIn401k", TypeName = "bit")]
  [JsonPropertyName("isEnrolledIn401k")]
  public bool? IsEnrolledIn401k
  {
    get { return _IsEnrolledIn401k; }
    set
    {
      _IsEnrolledIn401k = value;
      RaisePropertyChanged(nameof(IsEnrolledIn401k));
    }
  }

  /// <summary>
  /// Get/Set Is Enrolled In Health Care
  /// </summary>
  [Display(Name = "Is Enrolled In Health Care")]
  [Column("IsEnrolledInHealthCare", TypeName = "bit")]
  [JsonPropertyName("isEnrolledInHealthCare")]
  public bool? IsEnrolledInHealthCare
  {
    get { return _IsEnrolledInHealthCare; }
    set
    {
      _IsEnrolledInHealthCare = value;

RaisePropertyChanged(nameof(IsEnrolledInHealthCare));
    }
```

```
  }

  /// <summary>
  /// Get/Set Is Enrolled In Hsa
  /// </summary>
  [Display(Name = "Is Enrolled In Hsa")]
  [Column("IsEnrolledInHSA", TypeName = "bit")]
  [JsonPropertyName("isEnrolledInHSA")]
  public bool? IsEnrolledInHSA
  {
    get { return _IsEnrolledInHSA; }
    set
    {
      _IsEnrolledInHSA = value;
      RaisePropertyChanged(nameof(IsEnrolledInHSA));
    }
  }

  /// <summary>
  /// Get/Set Is Enrolled In Flex Time
  /// </summary>
  [Display(Name = "Is Enrolled In Flex Time")]
  [Column("IsEnrolledInFlexTime", TypeName = "bit")]
  [JsonPropertyName("isEnrolledInFlexTime")]
  public bool? IsEnrolledInFlexTime
  {
    get { return _IsEnrolledInFlexTime; }
    set
    {
      _IsEnrolledInFlexTime = value;

RaisePropertyChanged(nameof(IsEnrolledInFlexTime));
    }
  }

  /// <summary>
  /// Get/Set Is Active
  /// </summary>
  [Display(Name = "Is Active")]
  [Column("IsActive", TypeName = "bit")]
  [JsonPropertyName("isActive")]
  public bool? IsActive
  {
    get { return _IsActive; }
    set
    {
      _IsActive = value;
```

```
      RaisePropertyChanged(nameof(IsActive));
    }
  }

  /// <summary>
  /// Get/Set Birth Date
  /// </summary>
  [Display(Name = "Birth Date")]
  [Column("BirthDate", TypeName = "date")]
  [DataType(DataType.Date)]
  [JsonPropertyName("birthDate")]
  public DateTime? BirthDate
  {
    get { return _BirthDate; }
    set
    {
      _BirthDate = value;
      RaisePropertyChanged(nameof(BirthDate));
    }
  }

  /// <summary>
  /// Get/Set Start Time
  /// </summary>
  [Display(Name = "Start Time")]
  [Column("StartTime", TypeName = "time")]
  [JsonPropertyName("startTime")]
  public TimeSpan? StartTime
  {
    get { return _StartTime; }
    set
    {
      _StartTime = value;
      RaisePropertyChanged(nameof(StartTime));
    }
  }

  public string FullName
  {
    get { return FirstName + " " + LastName; }
  }

  public string LastNameFirstName
  {
    get { return LastName + ", " + FirstName; }
  }
  #endregion
```

```
  #region ToString Override
  public override string ToString()
  {
    return $"{LastName}";
  }
  #endregion
}
```

Lab 2: Add Generic Validation Classes to the Common Library

Right mouse-click on the **Common.Library** project and add a new folder named **ValidationClasses**.

Right mouse-click on the ValidationClasses folder and add a new class named **ValidationMessage**. Replace the entire contents of this new file with the following code.

```
namespace Common.Library;

public class ValidationMessage : CommonBase
{
  #region Private Variables
  private string _PropertyName = string.Empty;
  private string _Message = string.Empty;
  #endregion

  #region Public Properties
  /// <summary>
  /// Get/Set PropertyName
  /// </summary>
  public string PropertyName
  {
    get { return _PropertyName; }
    set
    {
      _PropertyName = value;
      RaisePropertyChanged(nameof(PropertyName));
    }
  }

  /// <summary>
  /// Get/Set Message
  /// </summary>
  public string Message
  {
    get { return _Message; }
    set
    {
      _Message = value;
      RaisePropertyChanged(nameof(Message));
    }
  }
  #endregion

  #region ToString() Override
  public override string ToString()
  {
    if (string.IsNullOrEmpty(PropertyName)) {
      return $"{Message}";
    }
    else {
      return $"{Message} ({PropertyName})";
    }
  }
```

```
  #endregion
}
```

Open the **BaseClasses\ViewModelBase.cs** file and add two private variables.

```
private bool _HasValidationErrors;
private ObservableCollection<ValidationMessage>
_ValidationMessages = new();
```

Add two public properties.

```
/// <summary>
/// Get/Set if the current selected entity has passed
all validation
/// </summary>
public bool HasValidationErrors
{
  get { return _HasValidationErrors; }
  set
  {
    _HasValidationErrors = value;
    RaisePropertyChanged(nameof(HasValidationErrors));
  }
}

/// <summary>
/// Get/Set a list of validation messages to display
/// </summary>
public ObservableCollection<ValidationMessage>
ValidationMessages
{
  get { return _ValidationMessages; }
  set
  {
    _ValidationMessages = value;
    RaisePropertyChanged(nameof(ValidationMessages));
  }
}
```

Add a Validate() method.

```
#region Validate Method
public bool Validate<T>(T entity)
{
  ValidationMessages.Clear();

  if (entity != null) {
    // Create instance of ValidationContext object
    ValidationContext context = new(entity,
serviceProvider: null, items: null);
    List<ValidationResult> results = new();

    // Call TryValidateObject() method
    if (!Validator.TryValidateObject(entity, context,
results, true)) {
      // Get validation results
      foreach (ValidationResult item in results) {
        string propName = string.Empty;
        if (item.MemberNames.Any()) {
          propName = ((string[])item.MemberNames)[0];
        }
        // Build new ValidationMessage object
        ValidationMessage msg = new() {
          Message = item.ErrorMessage ?? string.Empty,
          PropertyName = propName
        };

        // Add validation object to list
        ValidationMessages.Add(msg);
      }
    }
  }

  HasValidationErrors = ValidationMessages.Count > 0;

  return !HasValidationErrors;
}
#endregion
```

In the BeginProcessing() method, initialize both of these new properties.

```
HasValidationErrors = false;
ValidationMessages.Clear();
```

In the EndProcessing() method, check to see if Validation Errors have occurred.

```
HasValidationErrors = ValidationMessages.Count > 0;
```

Lab 3: Add Validate() Method to User View Model Class

Open the **ViewModelClasses\UserViewModel.cs** file and modify the **SaveAsync**() method to look like the following.

```
public async virtual Task<User?> SaveAsync()
{
  User? ret;

  if (Validate(CurrentEntity)) {
    // TODO: Write code to save data

    ret = new User();
  }
  else {
    ret = null;
  }

  return await Task.FromResult(ret);
}
```

Add Styles

Open the **Resources\Styles\AppStyles.xaml** file and add two new keyed styles.

```
<Style TargetType="Label"
       x:Key="ValidationMessage">
  <Setter Property="FontAttributes"
          Value="Bold" />
  <Setter Property="FontSize"
          Value="Small" />
  <Setter Property="TextColor"
          Value="Red" />
</Style>
<Style TargetType="Border"
       x:Key="ValidationErrorArea">
  <Setter Property="Stroke"
          Value="Gray" />
  <Setter Property="Margin"
          Value="10" />
  <Setter Property="Padding"
          Value="10" />
</Style>
```

Modify the Detail View

Open the **Views\UserDetailView.xaml** file and add a new XML namespace.

```
xmlns:val="clr-
namespace:Common.Library;assembly=Common.Library"
```

Add a new "**Auto**" to the RowDefinitions attribute of the <Grid>.

Scroll down to closing </Grid> element and just before this element add the following XAML.

```
<!-- ******************************** -->
<!-- Validation Failure Messages Area -->
<!-- ******************************** -->
<Border Grid.Row="13"
        Grid.Column="0"
        Grid.ColumnSpan="2"
        IsVisible="{Binding HasValidationErrors}"
        Style="{StaticResource ValidationErrorArea}">
  <VerticalStackLayout>
    <Label FontSize="Medium"
           Text="Please Fix These Validation Data Entry
Errors" />
    <ListView ItemsSource="{Binding
ValidationMessages}">
      <ListView.ItemTemplate>
        <DataTemplate
x:DataType="val:ValidationMessage">
          <ViewCell>
            <HorizontalStackLayout>
              <Label Text="{Binding Message}"
                     Style="{StaticResource
ValidationMessage}" />
            </HorizontalStackLayout>
          </ViewCell>
        </DataTemplate>
      </ListView.ItemTemplate>
    </ListView>
  </VerticalStackLayout>
</Border>
```

Try It Out

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

Click the **Edit** button for one of the users.

Remove the data from the Login ID, First Name, Last Name, and Email Address entry controls.

Click the **Save** button.

You should see the list of validation errors appear just below the Save button.