[EmailAddress], [Phone], [Url], and [CreditCard] Annotations 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: Validate Emails Using the [EmailAddress] Attribute

Open the **User.cs** file and remove the [RegularExpression] attribute from the **EmailAddress** property. Apply the **[EmailAddress]** attribute to the **EmailAddress** property as shown in the following code.

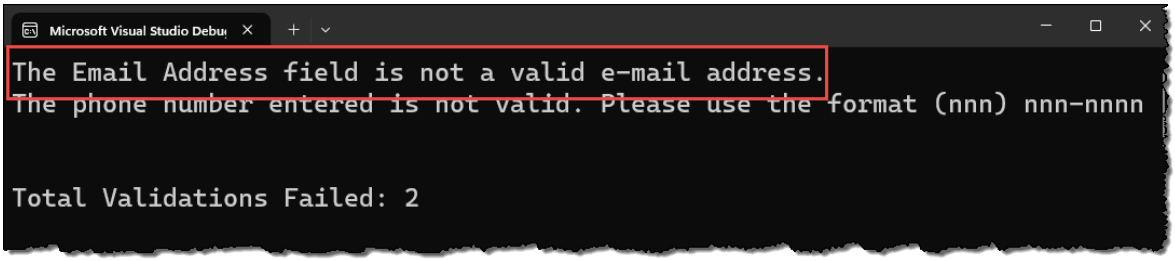
```
[EmailAddress]
public string EmailAddress { get; set; } = string.Empty;
```

Try it Out

Open the **Program.cs** file and modify the initialization of the entity object to look like the following code. Notice there is an invalid format for the **EmailAddress** property.

```
User entity = new() {
  UserId = 1,
  LoginId = "JoeSmith",
  Password = "Joe!Smith@2024",
  ConfirmPassword = "Joesmith2024",
  EmailAddress = "john!smith.com",
  Phone = "xxx-xxx-xxxx"
};
```

Run the application and you should see the appropriate error message for the email address property as shown below.



Lab 2: Validate Phone Numbers using the [Phone] Attribute

Open the **User.cs** file and remove the [RegularExpression] attribute from the **Phone** property. Apply the **[Phone]** attribute to the **Phone** property as shown in the following code.

```
[Phone]
public string Phone { get; set; } = string.Empty;
```

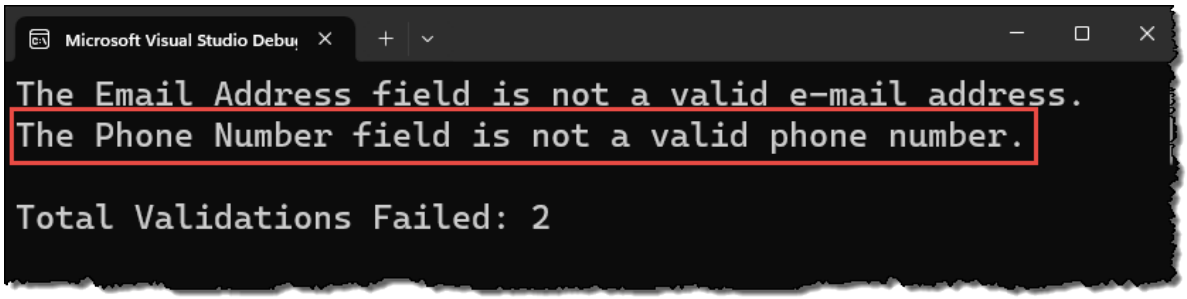
NOTE: Valid phone formats are the following.

- (999) 999-9999
- 999-999-9999
- 999.999.9999

Try it Out

Run the application and you should see the error message for the Phone number as shown below.

Copyright © 2024 by Paul D. Sheriff
All rights reserved worldwide. Reproduction is strictly prohibited.



Lab 3: Validate URLs Using the [Url] Attribute

If you have a URL property in your class, you can use the [Url] attribute to ensure the data contained within that URL is valid. Be aware that the URL entered into your property must start with either http://, https://, or ftp://. If you don't want these prefixes, you will not be able to use the [Url] attribute. Open the **Product.cs** file and add a **ProductUrl** property and add a **[Url]** data annotation to it as shown below.

```
[Display(Name = "Product URL")]
[Url]
public string ProductUrl { get; set; } = string.Empty;
```

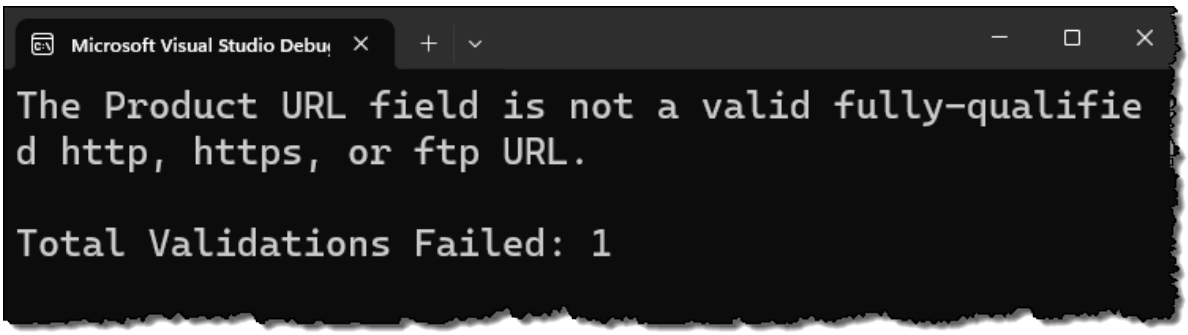
Try it Out

Open the **Program.cs** file and change the entity object to that shown in the following code.

```
Product entity = new() {
  ProductID = 1,
  Name = "A New Product",
  ProductNumber = "NEW-001",
  ProductUrl = "www.badurl-com",
  Color = "Red",
  StandardCost = 1,
  ListPrice = 10,
  SellStartDate = Convert.ToDateTime("12/31/2023"),
  SellEndDate = Convert.ToDateTime("12/31/2025"),
  DiscontinuedDate = DateTime.Now
};
```

Run the application and you should see the error message shown below.

Copyright © 2024 by Paul D. Sheriff
All rights reserved. Reproduction is strictly prohibited.



Lab 4: Validate Credit Cards Using the [CreditCard] Attribute

Another common business rule is to check for valid credit card data entered by a user. To try this out, right mouse-click on the **EntityClasses** folder and add a new class named **CreditCard**. In the new CreditCard class add the code shown below. Notice the use of the [CreditCard] attribute decorating the **CardNumber** property.

Copyright © 2024 by Paul D. Sheriff
All rights reserved worldwide. Reproduction is strictly prohibited.

```
using System.ComponentModel.DataAnnotations;

namespace DataAnnotationsSamples;

public class CreditCard
{
  [Required]
  [Display(Name = "Credit Card Type")]
  public string CardType { get; set; } = string.Empty;
  [Required]
  [Display(Name = "Name on Card")]
  public string NameOnCard { get; set; } = string.Empty;
  [CreditCard]
  [Display(Name = "Credit Card Number")]
  public string CardNumber { get; set; } = string.Empty;
  [Required]
  [Display(Name = "Security Code")]
  public string SecurityCode { get; set; } =
string.Empty;
  [Range(1, 12)]
  [Display(Name = "Expiration Month")]
  public int ExpMonth { get; set; } = 1;
  [Range(2024, 2030)]
  [Display(Name = "Expiration Year")]
  public int ExpYear { get; set; } = DateTime.Now.Year;
  [Required]
  [Display(Name = "Card Billing Postal Code")]
  public string BillingPostalCode { get; set; } =
string.Empty;
}
```

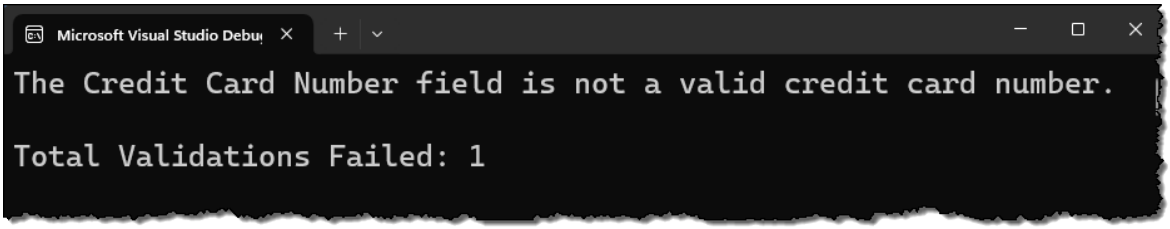
Try it Out

Open the **Program.cs** file and create a new instance of the CreditCard class and set the appropriate properties of the entity object as shown in the following code.

```
CreditCard entity = new() {
  CardType = "Visa",
  CardNumber = "12 13 123 1234",
  NameOnCard = "Joe Smith",
  BillingPostalCode = "99999",
  ExpMonth = 01,
  ExpYear = 2026,
  SecurityCode = "000"
};
```


Copyright © 2024 by Paul D. Sheriff
All rights reserved. Reproduction is strictly prohibited.

Run the application and you should see an error message informing you that the **CardNumber** property is not a valid credit card number as shown below.



Copyright © 2024 by Paul D. Sheriff
All rights reserved worldwide. Reproduction is strictly prohibited.