Using .NET Core Data Annotation

Thiago Vivas Jan 19, 2023 127.1k 5 10

In this article, I will be explaining how to use some properties of data annotation in order to make it easier to model your database and also to save your time with front-end validations.

How to create your database?

- Check it here

What is Data Annotation?

"Data Annotation provides attribute classes that are used to define metadata for ASP.NET MVC and ASP.NET data controls."

Why would I use Data Annotation?

There are 3 main areas where you may use it; two of them are related to your data presentation to your end user and one is to design your database.

1. *Front-End: Validation Attributes*
   Used to enforce validation rules in your view. It can be used to validate email, data, fields with masks, etc.

2. *Front-End: Display Attributes*
   Used to specify how your properties from your Model will be displayed. It can be used within Resources to display a different value depending on the user language.

3. *Back-End: Modelling Attributes*
   Used to specify the table limitations and the relationship between classes. It can be used to set field type, length, mask, etc.

How to do it?

Let's see it now. In this example, we are going to use a very simple trip manager model, TripManager class.

```
1  public class TripManager
2  {
3      [Key]
4      public int Id { get; set; }
5
6      [ForeignKey( "HotelID" )]
```

```
7       public Hotel Hotel { get; set; }
8
9       public int HotelID { get; set; }
10
11      [ForeignKey( "CarID" )]
12      public Car Car { get; set; }
13
14      public int CarID { get; set; }
15
16      [DataType( DataType.Date )]
17      public DateTime CheckInDate { get; set; }
18
19      [DataType( DataType.Date )]
20      public DateTime CheckOutDate { get; set; }
21
22      [Column( "TodaysPrice" )]
23      [Range( 10.30, 46.60 )]
24      public double Price { get; set; }
25
26      public Person Responsable { get; set; }
27  }
```

Hotel Class

```
1   public class Hotel
2   {
3       public int Id { get; set; }
4
5       [DisplayFormat( NullDisplayText = "Null name" )]
6       public string Name { get; set; }
7   }
```

Person Class

```
1   public class Person
2   {
3       public int Id { get; set; }
4
5       [StringLength( 50, ErrorMessage = "Name cannot be longer than 50 chara
6       public string Name { get; set; }
7
8       [NotMapped]
9       public List<Place> VisitedPlaces { get; set; }
10  }
```

Car Class

```
public class Car
{
    public int Id { get; set; }

    [Required]
    public string Model { get; set; }

    public string Brand { get; set; }
}
```

Place Class

```
public class Place
{
    public int Id { get; set; }
    public string Location { get; set; }
}
```

Now, let's understand the application of each one data annotation attribute used here.

Key

```
[Key]
public int Id { get; set; }
```

Explanation

Sets the key of the table.

Result

Foreign Key

```
[ForeignKey( "HotelID" )]
public Hotel Hotel { get; set; }

public int HotelID { get; set; }
```

Explanation

Sets the foreign key of this relationship as the property described.

Result in the database

Foreign Key

```
[ForeignKey( "CarID" )]
public Car Car { get; set; }

public int CarID { get; set; }
```

Explanation

Same as before. Sets the foreign key of this relationship as the property described.

Result in the database

Data type

```
[DataType( DataType.Date )]
public DateTime CheckInDate { get; set; }
```

Explanation

Creates a UI mask based on your provided type. It's very useful and commonly seen used with data and password types. Find more types here.

Result

Column and Range

```
[Column( "TodaysPrice" )]
[Range( 10.30, 46.60 )]
public double Price { get; set; }
```

Column Explanation

Used to define the name of the column in the database of its respective table for the specified property.

Column Result in the database

Range Explanation

Used to validate the input on the client side according to the specified range of value.

Range Result

Required

```
1  [Required]
2  public string Model { get; set; }
```

Explanation

Specifies a property as required and does not accept null at the database.

Result in the database

String length

```
1  [StringLength( 50, ErrorMessage = "Name cannot be longer than 50 character
2  public string Name { get; set; }
```

Explanation

Sets the max length of its field in the database and validates the input in the UI.

Result in the database.

Result in the UI.

Not mapped

```
1  [NotMapped]
2  public List<Place> VisitedPlaces { get; set; }
```

Explanation

The property is ignored by the database.

Result in the database

Display format

```
1  [DisplayFormat( NullDisplayText = "Null name" )]
2  public string Name { get; set; }
```

Explanation

Configures how this field is going to be displayed to the end user. It's used with language resources and you can find more options here.

Result

Congratulations! You have successfully implemented Data Annotation with your ASP.NET Core application.

- Project on GitHub
- Microsoft Data Annotation Page

.Net Core | .Net Core Data Annotation | Data Annotation

SIMILAR ARTICLES

Data Validation with Annotations, Custom Attributes, and Fluent Validation

Validation Using Data Annotations Attribute

Validation in ASP.NET MVC Using the Data Annotations

Custom Validations With Data Annotations

Options Pattern Validation in .NET Core with Examples

Thiago Vivas TOP 500

Interested about improving my .Net Core, Azure and DevOps skills.

128 2.3m 4

5

View All Comments

Type your comment here and press Enter Key (Minimum 10 characters)