

Entity Framework

Data Annotations



Lesson Objectives

- Data Annotations Attributes
- DataAnnotations Schema
- Data Validation

- Code-First Conventions is default, simple, powerful but not cover advanced cases
 - ✓ More specific data type and constrains
 - Name of user is not exceed 100 characters, I don't want to create `nvarchar(Max)`
 - Age of user is in range 0 – 150, let's set the constrains to check this
 - ✓ Column name, table name in SQL have space
 - ✓ One-to-One, Many-to-Many relationships

- A simple attribute based configuration
- Can apply to domain classes and its properties
- Included in a separate namespace
 - ✓ `System.ComponentModel.DataAnnotations`
 - ✓ `System.ComponentModel.DataAnnotations.Schema`

- The Table attribute can be applied to a class to configure the corresponding table name in the database.
- It overrides the default convention.
 - ✓ What is default schema?
 - ✓ What is convention for table name?
- Format: `[Table(string name, Properties:[Schema = string])]`
 - ✓ name: Name of the Db table.
 - ✓ Schema: Name of the Db Schema in which a specified table should be created.
(Optional)

Data Annotations - Table Attribute

```
[Table("NewProduct", Schema = "Admin")]
```

2 references

```
public class Product
```

```
{
```

0 references

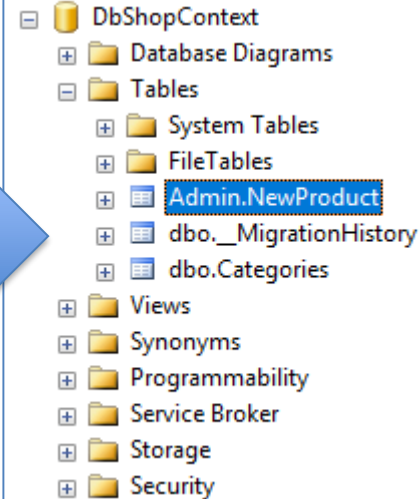
```
public int ID { get; set; }
```

```
[Required]
```

0 references

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

[Table("NewProduct",
Schema = "Admin")]



- Which are default conventions for column?
 - ✓ Column name
 - ✓ Data type
 - ✓ Order

- The Column attribute can be applied to one or more properties in an entity class
- Format: [Column (string name, Properties:[Order = int],[TypeName = string])
 - ✓ name: Name of a column in a SQL table.
 - ✓ Order: Order of a column, starting with zero index. (Optional)
 - ✓ TypeName: Data type of a column. (Optional)

- Name:
 - ✓ Can put space in SQL column name
 - ✓ Try to keep naming conventions
 - ✓ Similar to entity field to easy finding/understanding
 - ✓ Keep it simple, meaning full

- Order:
 - ✓ Use the zero-based
 - ✓ Difference values
 - ✓ Do not need to continuity
 - ✓ All ordered columns should come first (in order, of course)
 - ✓ All non-ordered columns should come along, in default order as conventions

- TypeName:
 - ✓ To change SQL data type for the column
 - ✓ Accept name of data type in SQL as a string

- Is used to create an index on a particular column in the database
- Format: [Index(string name, Properties:[IsClustered = bool],[IsUnique = bool] ,[Order = int])]
 - ✓ name: name of Index. Default value is: IX_{property name}
 - ✓ IsClustered: to set index is clustered or non-clustered.
 - ✓ IsUnique: to set constrain data is unique or not
 - ✓ Order: to set order of index

- Is used to configure a foreign key in the relationship between two entities.
- Format: [ForeignKey(name string)]
 - ✓ name: Name of the associated navigation property or the name of the associated foreign key(s).

- Used when we do not want to create corresponding columns in the database.
- Format: [NotMapped()]
- Discussion: Give an example to use this attribute?

- Is used when two entities have more than one relationship.
- Format: [InverseProperty(string property)]
 - ✓ property: The navigation property representing the other end of the same relationship.
- The attribute always set on the left-hand (set to property of the parent entity)

- What is default convention for PrimaryKey?
- How to set PrimaryKey for non-formatted field?
- How to set PrimaryKey for multiple columns?
- How to use Find method for PrimaryKey with multiple columns?

Data Annotations - Key Attribute

- Applied to a property in an entity class to make it a key property.
- Corresponding column to a PrimaryKey column in the database
- Format: [Key]

- In EF 6, the Key attribute can be applied to multiple properties of an entity class which will create composite primary key columns in the database,
- **BUT**, the Key attributes need to along with the [Column(Order=int)] attribute

- EF will create a NOT NULL column in a database table.
- Can be applied to one or more properties in an entity class.
- Format: [Required]
 - ✓ AllowEmptyStrings: to accept empty string or not
- Required attribute inherit from Validation attribute. Therefore:
 - ✓ ErrorMessage: The error message is thrown when the property associated with the validation control is invalid.

- Specifies the maximum length of data value allowed for a property which in turn sets the size of a corresponding column in the database.
- It can be applied to the **string** or **byte[]** properties of an entity.
- Format: [MaxLength(int)]
- MaxLength attribute inherit from Validation attribute

Example 1:

```
[MaxLength(50)]
```

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

Should create a field ProductName in SQL with data type is: nvarchar(50)

Example 2:

```
[MaxLength(1024)]
```

```
public byte[] FileContent { get; set; }
```

Should create a field FileContent in SQL with data type is: varbinary(1024)

- Same context as MaxLength
- Specifies the minimum length of data value allowed
- Question: How to specify fixed length of the data?
 - ✓ For example: ProductCode is always has 10 characters

- Can be applied to the string properties of an entity class.
- Specifies the minimum and maximum length of characters that are allowed in a data field.
- User often uses maximum length first, then minimum.
- Format: [StringLength(int, MinimumLength = int)]

- Specifies the numeric range constraints for the value of a data field.
- Apply the attribute to a data field of type integer.
 - ✓ [Range(int minimum, int maximum)]
- Apply the attribute to a data field of type double.
 - ✓ [Range(double minimum, double maximum)]

- Apply the attribute to a DateTime data field
 - ✓ Range(Type, String, String)
 - ✓ [Range(typeof(DateTime), string minimum, string maximum)]
- Apply the attribute to a custom data field
 - ✓ The object to validate must implement the **Comparable** interface.
 - ✓ Range(Type, String, String)

- It can only be applied once in an entity class to a byte array type property.
- It creates a column with timestamp data type in the SQL Server database.
- Entity Framework API automatically uses this Timestamp column in concurrency check on the UPDATE statement in the database
- Format: [Timestamp]

- The ConcurrencyCheck attribute can be applied to one or more properties in an entity class.
- When applied to a property, the corresponding column in the database table will be used in the optimistic concurrency check using the where clause.
- The ConcurrencyCheck attribute can be applied to any number of properties with any data type.

- <https://docs.microsoft.com/en-us/dotnet/api/system.componentmodel.dataannotations?view=netframework-4.8>

Thank you

