DATA ANNOTATION

- Data Annotations attributes are .NET attributes which can be applied on an entity class or properties to override default conventions in EF 6 and EF Core.
- Data Annotation in .NET Framework means add extra meaning to the data by adding attribute tags.
- The advantage is that by applying Data Attributes, we can manage the data definition in a single place and do not need re-write the same rules in multiple places.
- The **System.ComponentModel.DataAnnotations** namespace provides attribute classes that are used to define metadata of ASP.NET.

CONTD…

- The Data Annotation attributes falls into three categories:
- **Validation Attributes**: Used to enforce validation rules.
- **Display Attributes**: Used to specify how data from a class /member is displayed in the UI.
- **Modelling Attributes**: Used to specify the intended use of class member and the relationship between classes.

CONTD…

```
public class Person
{
        [Required]
        [Display(Name = “First Name")]
        [StringLength(50, ErrorMessage = "First name cannot be longer than 50 characters.")]
        [Column("FirstName")]
        public string FirstName { get; set; }
        [DataType(DataType.EmailAddress)]
        [EmailAddress]
        public string Email { get; set; }
}
```

SYSTEM.COMPONENTMODEL.DATAANNOTATIONS ATTRIBUTES:

Attribute	Description
Key	Can be applied to a property to specify a key property in an entity and make the corresponding column a PrimaryKey column in the database.
Timestamp	Can be applied to a property to specify the data type of a corresponding column in the database as rowversion.
ConcurrencyCheck	Can be applied to a property to specify that the corresponding column should be included in the optimistic concurrency check.
Required	Can be applied to a property to specify that the corresponding column is a NotNull column in the database.
MinLength	Can be applied to a property to specify the minimum string length allowed in the corresponding column in the database.
MaxLength	Can be applied to a property to specify the maximum string length allowed in the corresponding column in the database.
StringLength	Can be applied to a property to specify the maximum string length allowed in the corresponding column in the database.

SYSTEM.COMPONENTMODEL.DATAANNOTATIONS.SCHEMA ATTRIBUTES:

Attribute	Description
Table	Can be applied to an entity class to configure the corresponding table name and schema in the database.
Column	Can be applied to a property to configure the corresponding column name, order and data type in the database.
Index	Can be applied to a property to configure that the corresponding column should have an Index in the database. (EF 6.1 onwards only)
ForeignKey	Can be applied to a property to mark it as a foreign key property.
NotMapped	Can be applied to a property or entity class which should be excluded from the model and should not generate a corresponding column or table in the database.
DatabaseGenerated	Can be applied to a property to configure how the underlying database should generate the value for the corresponding column e.g. identity, computed or none.
InverseProperty	Can be applied to a property to specify the inverse of a navigation property that represents the other end of the same relationship.
ComplexType	Marks the class as complex type in EF 6. EF Core 2.0 does not support this attribute.

Course class to represent Course entity

```
public class Course
{       public int Id
        { get; set; }
        public string Name
        { get; set; }
        public virtual ICollection<Student> Students
        { get; set; }
}
```

- Course has many students.

Student class to represent Student entity

```
public class Student
{
        public int Id
        { get; set; }
        public string Name
        { get; set; }
        public string Email
        { get; set; }
        public virtual Course Course
        { get; set; }
}
```

To get reference between two objects

- **To include Course(Parent table) under Student object**

Example:

```
_context.Student.Include(s=>s.Course).ToList();
```

- **To include Students(Child table) under Course object**

Example:

```
context.Course.Include(c=>c.Students)
```

Note:

Specifies the related objects to include in the query results.