Default code generation of your entity model will automatically add some validation attributes to your entity properties. You can also have code generation **generate custom validation attributes** onto your entity properties using a metadata "buddy" class.

These following verifier attributes are automatically added to the code that DevForce generates based on the property attributes defined in the Entity Framework Designer:

- if a property is non-nullable the *RequiredValueVerifier* is added
- if a property has a backing column with a maximum string length, a *StringLengthVerifier* is added.

These attributes appear as follows in the generated code:

```
[IbVal.RequiredValueVerifier( ErrorMessageResourceName="Employee_Id")]
public long Id { ... }
[IbVal.StringLengthVerifier(MaxValue=30, IsRequired=true, ErrorMessageResourceName="Employee_LastName")]
public string LastName { ... }
```

```
<IbVal.RequiredValueVerifier(ErrorMessageResourceName:="Employee_Id")> _
Public ReadOnly Property Id() As Long
  ...
End Property
<IbVal.StringLengthVerifier(MaxValue:=30, IsRequired:=True, ErrorMessageResourceName:="Employee_LastName")> _
Public ReadOnly Property LastName() As String
```

To add additional verifiers to your entities, you can programmatically add verifiers, as we've seen earlier. But you have another option too: you can use a "buddy" class containing metadata to be applied to entity properties. In the buddy class you define additional validation attributes for your entity properties, and "link" the buddy class to the entity class by placing the MetadataType attribute on the class. Here's an example to make this clearer:

```
[MetadataType(typeof(EmployeeMetadata))]
public partial class Employee { ... }
public class EmployeeMetadata {
 [RequiredValueVerifier()]
 public static string City;
 [DateTimeRangeVerifier(MinValue="1/1/2010", MaxValue="12/31/2020")]
 public static DateTime? HireDate;
 [RegexVerifier("Home phone", "USPhone")]
 public static string HomePhone;
}
```

```
<MetadataType(GetType(EmployeeMetadata))> _
Public Partial Class Employee
End Class
Public Class EmployeeMetadata
 <RequiredValueVerifier> _
Public Shared City As String
 <DateTimeRangeVerifier(MinValue := "1/1/2010", MaxValue := "12/31/2020")> _
Public Shared HireDate As System.Nullable(Of DateTime)
 <RegexVerifier("Home phone", "USPhone")> _
Public Shared HomePhone As String
End Class
```

The verifier attributes defined in the buddy class will take precedence and override the same verifier attribute for the same property in the generated code. For example, if you have a generated *Employee.LastName StringLengthVerifierAttribute* with *IsRequired* = true, and you create an Employee buddy class and add a *StringLengthVerifierAttribute* with *IsRequired* = false to the *LastName* property, then the *LastName* will no longer be required.
The precedence described above will only take place for standard DevForce verifier attributes. If you have created your own custom verifier attribute, and define it in the buddy class, code generation will still discover it. But the existing generated DevForce verifier attributes will not be overridden.

The verifiers may be a bit contrived here (unless you really do want to hire only US employees in this decade), but should give you an idea of what you can do. Any of the DevForce validation attributes, such as any of the range verifiers *Int32RangeVerifierAttribute*, *Int64RangeVerifierAttribute*, *DecimalRangeVerifierAttribute*, *DoubleRangeVerifierAttribute*, *DateTimeRangeVerifierAttribute*, etc.) or *RegexVerifierAttribute*, or any custom validation attributes you write which extend a DevForce verifier attribute, can be used.

There are a few rules to follow, to ensure that your attributes are found:

- Be sure to add the *MetadataType* attribute to the class you will be providing metadata for. Here we've decorated the Employee class with the attribute.
- The buddy class can be standalone or nested, and should be public or internal, or private if used only in Desktop applications.

- The buddy class should define public fields or properties having the same name as the property in the class for which it's defined. These members can be either static or defined on the buddy class instance. In the case above, the Employee class should have properties named "City", "HireDate", and "HomePhone".
- The *MetadataType* attribute is defined in *IdeaBlade.Core.ComponentModel.DataAnnotations* for Silverlight applications, and in *System.ComponentModel.DataAnnotations* for desktop applications. The attribute functions the same in either environment.
- DevForce will look for the *MetadataType* attribute as it is generating code for your entities.

Always remember to regenerate your models to ensure these new attributes are discovered and included in the generated code.