**And Composite Validator**

Class Name: AndCompositeValidator

Attribute Name: ValidatorCompositionAttribute

Configuration tool name: And Composite Validator

[**Description**](javascript:void(0))

This validator creates a composite validator. Validation requires that all the validators that make up the composite validator be true. For example, you can use the And composite validator to require that the not null validator AND the date time range validator be True. Because the Validation Application Block's default behavior is to AND validators, you only need this validator to implement complex logic.

[**Properties**](javascript:void(0))

The And composite validator has a single property named Name. This is the name of the validator. The default name is And Composite Validator.

[**Example**](javascript:void(0))

The following code example shows the construction of an AndCompositeValidator instance using attributes. The validator combines a NotNullValidator and a StringLengthValidator. Note that this is the default behavior when more than one validator attribute is used. As a result, the ValidatorComposition attribute could be omitted.

public class Product

{

[ValidatorComposition(CompositionType.And)]

[NotNullValidator]

[StringLengthValidator(10)]

string \_productCode;

public string GetProductCode( )

{

return \_productCode;

}

// ...

}

Contains Characters Validator

Class Name: ContainsCharactersValidator

Attribute Name: ContainsCharactersValidatorAttribute

Configuration tool name: Contains Characters Validator

[**Description**](javascript:void(0))

This validator checks that an arbitrary string, such as a string entered by a user in a Web form, contains any or all of the characters that are specified by the CharacterSet property.

[**Properties**](javascript:void(0))

The following table lists the Contains Characters validator properties. The actual property names displayed in the configuration tools are shown in the description.

|  |  |
| --- | --- |
| Property | Description |
| CharacterSet | Character Set - This property defines the set of characters that must occur in the string you will validate. |
| ContainsCharacters | Contains Characters -This property determines whether the validator checks for Any of the characters defined in CharacterSet or for All of the characters. The default is Any. |
| MessageTemplate | Message Template - This property is a string containing template tokens that the validator replaces with values as it validates the target. Typically, it describes the validation result. |
| MessageTemplateResourceName | Template Resource Name - If you do not want to use the MessageTemplate property to hard-code a message template (perhaps for internationalization), you can use a template stored in the application resources. You must also specify a MessageTemplateResourceType value. If you include both a MessageTemplate value and a MessageTemeplateResourceName value, the MessageTemplate value takes precedence. |
| MessageTemplateResourceType | Template Resource Type - The resource type for the template you want to use. If you specify a MessageTemplateResourceName value, you must specify this value. |
| Name | Name – The name to use for this validator. |
| Negated | Negated - This is a Boolean property. If it is set to True, it changes the validator's behavior so that it will fail if the condition is met instead of when it is not met. The default is False. |
| Tag | Tag - This property is a user-supplied string. Typically, it is used to sort or categorize validation results. |
| TypeName | Type Name – The fully qualified name of the type configuration element. This property cannot be edited. |

**Message Template Tokens**

If the message template contains tokens (for example, "{0}"), the validator will replace these tokens with values when the ValidationResult is created. The following table lists the tokens that are supported by the contains characters validator.

|  |  |
| --- | --- |
| Token | Meaning |
| {0} | This token represents the value of the object that is being validated. Although it can be useful to show the original value as a part of the validation message, you must be careful to avoid injection attacks by escaping any characters that can be used to attack the system that conveys the message to the user. |
| {1} | This token represents the key of the object that is being validated. When the validator is attached to a member of a type such as a property or a field, the key is set to the member name. When the validator is attached to an object, the key is null and the token is replaced by an empty string. |
| {2} | This token represents the tag that is specified on the validator instance. If no tag is supplied, the token is replaced by an empty string. |
| {3} | The list of characters configured for this validator instance. |
| {4} | The type of character search (Any or All) configured for this validator instance. |

[**Example**](javascript:void(0))

The following code example uses attributes to ensure that the product code contains the character "f", the character "s" and the character "p".

public class Product

{

[ContainsCharactersValidator("fsp", ContainsCharacters.All)]

public string ProductCode

{

get

{

return \_productCode;

}

}

// ...

}

**Date Time Range Validator**

Class Name: DateTimeRangeValidator

Attribute Name: DateTimeRangeValidatorAttribute

Configuration tool name: Date Time Range Validator

[**Description**](javascript:void(0))

This validator checks that a DateTime object falls within the specified range.

[**Properties**](javascript:void(0))

The following table lists the date time range validator properties. The actual property names displayed in the configuration tools are listed in the table description.

|  |  |
| --- | --- |
| Property | Description |
| LowerBound | Lower Bound - This is the low-range boundary value. It must be of type DateTime. The compiler checks this requirement if you directly invoke the validator in your code. If you use attributes, the application block throws an exception at run time if the type is incorrect. If you use configuration, the configuration tool displays an error if you enter an incorrect type. |
| LowerBoundType | Lower Bound Type - This property determines how to evaluate the LowerBound value. Possible values for LowerBoundType are Ignore, Inclusive, and Exclusive. The Ignore value means that the validator ignores the LowerBound value. This is the default. The Inclusive value means that the validator allows values that are equal to the LowerBound value. The Exclusive value means that the validator does not allow values that are equal to the LowerBound value. |
| MessageTemplate | Message Template - This property is a string containing template tokens that the validator replaces with values as it validates the target. Typically, it describes the validation result. |
| MessageTemplateResourceName | Template Resource Name - If you do not want to use the MessageTemplate property to hard-code a message template (perhaps for internationalization), you can use a template stored in the application resources. You must also specify a MessageTemplateResourceType value. If you include both a MessageTemplate value and a MessageTemeplateResourceName value, the MessageTemplate value takes precedence. |
| MessageTemplateResourceType | Template Resource Type - The resource type for the template you want to use. If you specify a MessageTemplateResourceName value, you must specify this value. |
| Name | Name – The name to use for this validator. |
| Negated | Negated - This is a Boolean property. If it is set to True, it changes the validator's behavior so that it will fail if the condition is met instead of when it is not met. The default is False. |
| Tag | Tag - This property is a user-supplied string. Typically, it is used to sort or categorize validation results. |
| TypeName | Type Name – The fully qualified name of the type configuration element. This property cannot be edited. |
| UpperBound | Upper Bound - This is the upper-range boundary value. It must be of type DateTime. The compiler checks this requirement if you directly invoke the validator in your code. If you use attributes, the application block throws an exception at run time if the type is incorrect. If you use configuration, the configuration tool displays an error if you enter an incorrect type. |
| UpperBoundType | Upper Bound Type - This property determines how to evaluate the UpperBound value. Possible values for UpperBoundType are Ignore, Inclusive, and Exclusive. The Ignore value means that the validator ignores the UpperBound value. The Inclusive value means that the validator allows values that are equal to the UpperBound value. This is the default. The Exclusive value means that the validator does not allow values that are equal to the UpperBound value. |

**Message Template Tokens**

If the message template contains tokens (for example, "{0}"), the validator will replace these tokens with values when the ValidationResult is created. The following table lists the tokens that are supported by the date time range validator.

|  |  |
| --- | --- |
| Token | Meaning |
| {0} | This token represents the value of the object that is being validated. Although it can be useful to show the original value as a part of the validation message, you must be careful to avoid injection attacks by escaping any characters that can be used to attack the system that conveys the message to the user. |
| {1} | This token represents the key of the object that is being validated. When the validator is attached to a member of a type such as a property or a field, the key is set to the member name. When the validator is attached to an object, the key is null and the token is replaced by an empty string. |
| {2} | This token represents the tag that is specified on the validator instance. If no tag is supplied, the token is replaced by an empty string. |
| {3} | The lower bound configured for the validator instance. |
| {4} | The lower bound type (Inclusive, Exclusive, or Ignore) configured for the validator instance. |
| {5} | The upper bound configured for the validator instance. |
| {6} | The upper bound type (Inclusive, Exclusive, or Ignore) configured for the validator instance. |

[**Example**](javascript:void(0))

The following code example checks that the expiration date is before January 20, 2010.

public class Product

{

[DateTimeRangeValidator("2010-01-20T00:00:00")]

public DateTime ExpirationDate

{

get

{

return expirationDate;

}

}

// ...

}

**Domain Validator**

Class Name: DomainValidator<T>

Attribute Name: DomainValidatorAttribute

Configuration tool name: Domain Validator

[**Description**](javascript:void(0))

This validator checks that a value is one of the specified values in a specified set. For example, it can check that a name is "Tom," "Dick," "Harry," or "George" or that an integer is 2, 3, 5, 7, or 11. If the set only contains one value, you can use this validator to check for equality.

[**Properties**](javascript:void(0))

The following table lists the domain validator properties. The actual property names shown in the configuration tools are listed in the table description.

|  |  |
| --- | --- |
| Property | Description |
| Domain | Domain - The set of values that specifies the acceptable elements. The value that you want to validate must be of the same type as the values in the list. The compiler checks this requirement if you directly invoke the validator in your code. If you use attributes, the application block throws an exception at run time if the type is incorrect. If you use configuration, the configuration tool displays an error if you enter an incorrect type. This value is required. |
| MessageTemplate | Message Template - This property is a string containing template tokens that the validator replaces with values as it validates the target. Typically, it describes the validation result. |
| MessageTemplateResourceName | Template Resource Name - If you do not want to use the MessageTemplate property to hard-code a message template (perhaps for internationalization), you can use a template stored in the application resources. You must also specify a MessageTemplateResourceType value. If you include both a MessageTemplate value and a MessageTemeplateResourceName value, the MessageTemplate value takes precedence. |
| MessageTemplateResourceType | Template Resource Type - The resource type for the template you want to use. If you specify a MessageTemplateResourceName value, you must specify this value. |
| Name | Name – The name to use for this validator. |
| Negated | Negated - This is a Boolean property. If it is set to True, it changes the validator's behavior so that it will fail if the condition is met instead of when it is not met. The default is False. |
| Tag | Tag - This property is a user-supplied string. Typically, it is used to sort or categorize validation results. |
| TypeName | Type Name – The fully qualified name of the type configuration element. This property cannot be edited. |

**Message Template Tokens**

If the message template contains tokens (for example, "{0}"), the validator will replace these tokens with values when the ValidationResult is created. The tokens supported by the domain validator are listed in the following table.

|  |  |
| --- | --- |
| Token | Meaning |
| {0} | This token represents the value of the object that is being validated. Although it can be useful to show the original value as a part of the validation message, you must be careful to avoid injection attacks by escaping any characters that can be used to attack the system that conveys the message to the user. |
| {1} | This token represents the key of the object that is being validated. When the validator is attached to a member of a type such as a property or a field, the key is set to the member name. When the validator is attached to an object, the key is null and the token is replaced by an empty string. |
| {2} | This token represents the tag that is specified on the validator instance. If no tag is supplied, the token is replaced by an empty string. |

[**Example**](javascript:void(0))

The following code example shows how to use the validator with attributes to ensure that a value is within a specified list of values.

public class Product

{

[DomainValidator("each", "dozen", "gross")]

public string SalesUnit

{

get

{

return salesUnit;

}

}

// ...

}

**Enum Conversion Validator**

Class Name: EnumConversionValidator

Attribute Name: EnumConversionValidatorAttribute

Configuration tool name: Enum Conversion Validator

[**Description**](javascript:void(0))

This validator checks that a string can be converted to a value in a specified enum type. For example, it can check that "Blue" can be converted to a value in the Color enumeration.

[**Properties**](javascript:void(0))

The following table lists the enum conversion validator properties. The actual property names shown in the configuration tools are listed in the table description.

|  |  |
| --- | --- |
| Property | Definition |
| EnumType | Enumeration Type - This property is the type of the enumeration. You can either enter the type or you can select it with the type selector dialog box. |
| MessageTemplate | Message Template - This property is a string containing template tokens that the validator replaces with values as it validates the target. Typically, it describes the validation result. |
| MessageTemplateResourceName | Template Resource Name - If you do not want to use the MessageTemplate property to hard-code a message template (perhaps for internationalization), you can use a template stored in the application resources. You must also specify a MessageTemplateResourceType value. If you include both a MessageTemplate value and a MessageTemeplateResourceName value, the MessageTemplate value takes precedence. |
| MessageTemplateResourceType | Template Resource Type - The resource type for the template you want to use. If you specify a MessageTemplateResourceName value, you must specify this value. |
| Name | Name – The name to use for this validator. |
| Negated | Negated - This is a Boolean property. If it is set to True, it changes the validator's behavior so that it will fail if the condition is met instead of when it is not met. The default is False. |
| Tag | Tag - This property is a user-supplied string. Typically, it is used to sort or categorize validation results. |
| TypeName | Type Name – The fully qualified name of the type configuration element. This property cannot be edited. |

**Message Template Tokens**

If the message template contains tokens (for example, "{0}"), the validator will replace these tokens with values when the ValidationResult is created. The following table lists the tokens that are supported by the enum conversion validator.

|  |  |
| --- | --- |
| Token | Meaning |
| {0} | This token represents the value of the object that is being validated. Although it can be useful to show the original value as a part of the validation message, you must be careful to avoid injection attacks by escaping any characters that can be used to attack the system that conveys the message to the user. |
| {1} | This token represents the key of the object that is being validated. When the validator is attached to a member of a type such as a property or a field, the key is set to the member name. When the validator is attached to an object, the key is null and the token is replaced by an empty string. |
| {2} | This token represents the tag that is specified on the validator instance. If no tag is supplied, the token is replaced by an empty string. |
| {3} | The type of enumeration to which the validator will attempt to convert. |

[**Examples**](javascript:void(0))

The following code examples show how to use the validator with attributes and with code.

Using Attributes

The following code example checks that the sales unit is a string that can be converted to a value of the Unit enumeration. For example, the string "Each" is allowed.

public class Product

{

enum Unit

{

Each, Dozen, Gross

}

[EnumConversionValidator(typeof(Unit))]

public string SalesUnit

{

get

{

return salesUnit;

}

}

// ...

}

Using Code

The following code example constructs an EnumConversionValidator instance that can be used to check whether the sales unit is a string that is convertible to a value of the Unit enumeration. For example, the string "Each" would be allowed.

public class Product

{

enum Unit

{

Each, Dozen, Gross

}

Validator unitValidator = new EnumConversionValidator(typeof(Unit));

// ...

}

**Not Null Validator**

Class Name: NotNullValidator

Attribute Name: NotNullValidatorAttribute

Configuration tool name: Not Null Validator

[**Description**](javascript:void(0))

This validator checks that the value is not null (C#) or Nothing (Visual Basic).

[**Properties**](javascript:void(0))

The following table lists the Not Null validator properties. The actual property names shown in the configuration tools are listed in the table description.

|  |  |
| --- | --- |
| Property | Description |
| MessageTemplate | Message Template - This property is a string containing template tokens that the validator replaces with values as it validates the target. Typically, it describes the validation result. |
| MessageTemplateResourceName | Template Resource Name - If you do not want to use the MessageTemplate property to hard-code a message template (perhaps for internationalization), you can use a template stored in the application resources. You must also specify a MessageTemplateResourceType value. If you include both a MessageTemplate value and a MessageTemeplateResourceName value, the MessageTemplate value takes precedence. |
| MessageTemplateResourceType | Template Resource Type - The resource type for the template you want to use. If you specify a MessageTemplateResourceName value, you must specify this value. |
| Name | Name – The name to use for this validator. |
| Negated | Negated - This is a Boolean property. If it is set to True, it changes the validator's behavior so that it will fail if the condition is met instead of when it is not met. The default is False. |
| Tag | Tag - This property is a user-supplied string. Typically, it is used to sort or categorize validation results. |
| TypeName | Type Name – The fully qualified name of the type configuration element. This property cannot be edited. |

**Message Template Tokens**

If the message template contains tokens (for example, "{0}"), the validator will replace these tokens with values when the ValidationResult is created. The tokens supported by the Not Null validator are listed in the following table.

|  |  |
| --- | --- |
| Token | Meaning |
| {0} | This token represents the value of the object that is being validated. Although it can be useful to show the original value as a part of the validation message, you must be careful to avoid injection attacks by escaping any characters that can be used to attack the system that conveys the message to the user. |
| {1} | This token represents the key of the object that is being validated. When the validator is attached to a member of a type such as a property or a field, the key is set to the member name. When the validator is attached to an object, the key is null and the token is replaced by an empty string. |
| {2} | This token represents the tag that is specified on the validator instance. If no tag is supplied, the token is replaced by an empty string. |

[**Example**](javascript:void(0))

The following code example shows how to use the validator with attributes to ensure that a value is not null.

public class Product

{

[NotNullValidator]

public Company Manufacturer

{

get

{

return manufacturer;

}

}

// ...

}

**Object Collection Validator**

Class Name: ObjectCollectionValidator

Attribute Name: ObjectCollectionValidatorAttribute

Configuration tool name: Object Collection Validator

[**Description**](javascript:void(0))

This validator checks that the object is a collection of the specified type and then invokes validation on each element of the collection. If the object you want to validate is null, the validation is ignored. If the object you want to validate is not a collection, then the validation fails and the rule set is not applied. If there are elements in the collection that are of a different type than the one you specified for the object, then the validation for these elements fails but

this does not affect validation for the other elements.

[**Properties**](javascript:void(0))

The following table lists the object collection validator properties. The actual property names shown in the configuration tools are listed in the table description.

|  |  |
| --- | --- |
| Property | Description |
| MessageTemplate | Message Template - This property is a string containing template tokens that the validator replaces with values as it validates the target. Typically, it describes the validation result. |
| MessageTemplateResourceName | Template Resource Name - If you do not want to use the MessageTemplate property to hard-code a message template (perhaps for internationalization), you can use a template stored in the application resources. You must also specify a MessageTemplateResourceType value. If you include both a MessageTemplate value and a MessageTemeplateResourceName value, the MessageTemplate value takes precedence. |
| MessageTemplateResourceType | Template Resource Type - The resource type for the template you want to use. If you specify a MessageTemplateResourceName value, you must specify this value. |
| Name | Name – The name to use for this validator. |
| Tag | This property is a user-supplied string. Typically, it is used to sort or categorize validation results. |
| TargetRuleset | Target Ruleset - This is the name of the rule set that will be applied to each element in the collection. |
| TargetType | Target Type - This property is the type of the object that you want to validate. You can either enter the type or you can select it with the Type Selector – System.Object dialog box. |
| TypeName | Type Name – The fully qualified name of the type configuration element. This property cannot be edited. |

**Message Template Tokens**

If the message template contains tokens (for example, "{0}"), the validator will replace these tokens with values when the ValidationResult is created. The tokens supported by the object collection validator are listed in the following table.

|  |  |
| --- | --- |
| Token | Meaning |
| {0} | This token represents the value of the object that is being validated. Although it can be useful to show the original value as a part of the validation message, you must be careful to avoid injection attacks by escaping any characters that can be used to attack the system that conveys the message to the user. |
| {1} | This token represents the key of the object that is being validated. When the validator is attached to a member of a type such as a property or a field, the key is set to the member name. When the validator is attached to an object, the key is null and the token is replaced by an empty string. |
| {2} | This token represents the tag that is specified on the validator instance. If no tag is supplied, the token is replaced by an empty string. |

[**Example**](javascript:void(0))

The following example shows how to use the object collection validator with attributes to check that each part in a kit's part list meets the requirements of the Part type's validator. The Part type's validator checks that the name is fewer than 20 characters and that the manufacturer's URL is well formed according to a particular regular expression pattern.

public class Part

{

[StringLengthValidator(20)]

string name;

[RegexValidator(@"http://(www\.)?([^\.]+)\.com")]

string manufacturerUrl;

// ...

}

public class Kit

{

[ObjectCollectionValidator(typeof(Part))]

public Part[] PartList

{

get

{

return partList;

}

}

// ...

}

|  |  |
| --- | --- |
| **Ff664585.note(en-us,PandP.50).gifNote:** | |
| When validating a collection, the object collection validator is inheritance-aware. However, if you specify the target type, an exception is raised if the collection is recursive. To resolve this, do not specify the target object type for the object collection validator. At run time, the validator will determine the appropriate type. This will incur a minor performance penalty. | |

**Object Validator**

Class Name: **ObjectValidator**

Attribute Name: **ObjectValidatorAttribute**

Configuration tool name: **Object Validator**

[**Description**](javascript:void(0))

This validator causes validation to occur on an object reference. All validators defined for the object's type will be invoked, just as if the **Validation.Validate** method had been called on the object. If the object you want to validate is null, the validation is ignored. If the reference is to an instance of a type that is not compatible with the configured target's type, the validation fails. This validator is helpful for validating tree-structured data.

[**Properties**](javascript:void(0))

The following table lists the object validator properties. The actual property names shown in the configuration tools are listed in the table description.

|  |  |
| --- | --- |
| Property | Description |
| **MessageTemplate** | **Message Template** - This property is a string containing template tokens that the validator replaces with values as it validates the target. Typically, it describes the validation result. |
| **MessageTemplateResourceName** | **Template Resource Name** - If you do not want to use the **MessageTemplate** property to hard-code a message template (perhaps for internationalization), you can use a template stored in the application resources. You must also specify a **MessageTemplateResourceType** value. If you include both a **MessageTemplate** value and a **MessageTemeplateResourceName** value, the **MessageTemplate** value takes precedence. |
| **MessageTemplateResourceType** | **Template Resource Type -** The resource type for the template you want to use. If you specify a **MessageTemplateResourceName** value, you must specify this value. |
| **Name** | **Name** – The name to use for this validator. |
| **Tag** | This property is a user-supplied string. Typically, it is used to sort or categorize validation results. |
| **TargetRuleset** | **Target Ruleset** This is the name of the rule set that will be applied to the object. |
| **TypeName** | **Type Name** – The fully qualified name of the type configuration element. This property cannot be edited. |
| **ValidateActualType** | **Validate Actual Type**- Indicates whether to validate based on the static type or the actual type. |

**Message Template Tokens.**

If the message template contains tokens (for example, "{0}"), the validator will replace these tokens with values when the **ValidationResult** is created. The tokens supported by the object validator are listed in the following table.

|  |  |
| --- | --- |
| Token | Meaning |
| **{0}** | This token represents the value of the object that is being validated. Although it can be useful to show the original value as a part of the validation message, you must be careful to avoid injection attacks by escaping any characters that can be used to attack the system that conveys the message to the user. |
| **{1}** | This token represents the key of the object that is being validated. When the validator is attached to a member of a type such as a property or a field, the key is set to the member name. When the validator is attached to an object, the key is null and the token is replaced by an empty string. |
| **{2}** | This token represents the tag that is specified on the validator instance. If no tag is supplied, the token is replaced by an empty string. |

[**Example**](javascript:void(0))

The following example shows how to use the object validator with attributes to check that the **Part** object that is returned by the **OrderedPart** property meets the requirements of the **Part** class validator. The **Part** class validator checks that the name is fewer than 20 characters and that the manufacturer URL is well formed according to a particular regular expression pattern.

public class Part

{

[StringLengthValidator(20)]

string name;

[RegexValidator(@"http://(www\.)?([^\.]+)\.com")]

string manufacturerUrl;

// ...

}

public class OrderLineItem

{

[NotNullValidator]

[ObjectValidator]

public Part OrderedPart

{

get

{

return partList;

}

}

// ...

}

|  |  |
| --- | --- |
| **Ff664406.note(en-us,PandP.50).gifNote:** | |
| When validating a collection, the object validator and object collection validator are inheritance-aware. However, if the collection is recursive the object validator will raise an exception. To resolve this, set the **ValidateActualType** named parameter of the object validator attribute to true. | |

[**Differences between the Object Validator and the Factory-Created Validators**](javascript:void(0))

You can use an object validator instead of creating a validator for a type using the **CreateValidator** method of the **ValidatorFactory**. While the two approaches achieve the same result, there are some differences in their behavior:

If you do not specify a target type when you create an object validator programmatically, you can use it to validate any type. When you call the **Validate** method, you specify the target instance, and the object validator creates a type-specific validator for the type of the target instance. In contrast, the validator you obtain from a factory can only be used to validate instances of the type you specify when you obtain the validator. However, it can also be used to validate subclasses of the specified type, but it will use the rules defined for the specified target type.

The object validator will always use rules in configuration for the type of the target object, and attributes and self validation methods within the target instance. In contrast, you can use a specific factory class type to obtain validators that only validate the target instance using one type of rule source (in other words, just configuration rule sets, or just one type of attributes).

The object validator will acquire a type specific validator of the appropriate type each time you call the **Validate** method, even if you use the same instance of the object validator every time. In contrast, a validator obtained from one of the factory classes does not need to do this, and will offer improved performance.

You should generally consider using the factory approach for creating validators to validate objects to benefit from the increased flexibility and performance.

|  |  |
| --- | --- |
| **Ff664406.note(en-us,PandP.50).gifNote:** | |
| When you use an object validator attribute within a class to validate a member such as a property, the validator will, by default, use the rules defined for the type of that property. However, if the property is defined as an interface or base type, you can specify that the object validator use the rules defined for the actual concrete type that the property contains by setting the **ValidateActualType** property of the attribute as shown here**: [ObjectValidator(ValidateActualType = true)]** | |

**Or Composite Validator**

Class Name: **OrCompositeValidator**

Attribute Name: **ValidatorCompositionAttribute**

Configuration tool name: **Or Composite Validator**

[**Description**](javascript:void(0))

This validator creates a composite validator. Validation requires that at least one of the validators that make up the composite validator be **True**. For example, you can use the Or composite validator to require that the Not Null validator **OR** the date time range validator be **True**.

[**Properties**](javascript:void(0))

The following table lists the Or composite validator properties. The actual property names shown in the configuration tools are listed in the table description.

|  |  |
| --- | --- |
| Property | Description |
| **MessageTemplate** | **MessageTemplate** - This property is a string containing template tokens that the validator replaces with values as it validates the target. Typically, it describes the validation result. |
| **MessageTemplateResourceName** | **Template Resource Name** - If you do not want to use the **MessageTemplate** property to hard-code a message template (perhaps for internationalization), you can use a template stored in the application resources. You must also specify a **MessageTemplateResourceType** value. If you include both a **MessageTemplate** value and a **MessageTemeplateResourceName** value, the **MessageTemplate** value takes precedence. |
| **MessageTemplateResourceType** | **Template Resource Type -** The resource type for the template you want to use. If you specify a **MessageTemplateResourceName** value, you must specify this value. |
| **Name** | **Name** – The name to use for this validator. |
| **Tag** | This property is a user-supplied string. Typically, it is used to sort or categorize validation results. |
| **TypeName** | **Type Name** – The fully qualified name of the type configuration element. This property cannot be edited. |

[**Example**](javascript:void(0))

The following example shows how to use the validators with attributes. The validator combines a **NotNullValidator** and a **StringLengthValidator**.

public class Product

{

[ValidatorComposition(CompositionType.Or)]

[NotNullValidator]

[StringLengthValidator(10)]

public string ProductCode

{

get

{

return \_productCode;

}

}

// ...

}

**Property Comparison Validator**

Class Name: **PropertyComparisonValidator**

Attribute Name: **PropertyComparisonAttribute**

Configuration tool name: **Property Comparison Validator**

[**Description**](javascript:void(0))

This validator compares the value to be checked with the value of a property.

[**Properties**](javascript:void(0))

The following table lists the property comparison validator properties. The actual property names shown in the configuration tools are listed in the table description.

|  |  |
| --- | --- |
| Property | Description |
| **ComparisonOperator** | **Comparison Operator** - This property determines how the two properties are compared. The possible comparisons are **Equal**, **NotEqual**, **GreaterThan**, **GreaterThanEqual**, **LessThan**, and **LessThanEqual**. |
| **MessageTemplate** | **Message Template** - This property is a string containing template tokens that the validator replaces with values as it validates the target. Typically, it describes the validation result. |
| **MessageTemplateResourceName** | **Template Resource Name** - If you do not want to use the **MessageTemplate** property to hard-code a message template (perhaps for internationalization), you can use a template stored in the application resources. You must also specify a **MessageTemplateResourceType** value. If you include both a **MessageTemplate** value and a **MessageTemeplateResourceName** value, the **MessageTemplate** value takes precedence. |
| **MessageTemplateResourceType** | **Template Resource Type -** The resource type for the template you want to use. If you specify a **MessageTemplateResourceName** value, you must specify this value. |
| **Name** | **Name** – The name to use for this validator. |
| **Negated** | This is a Boolean property. If it is set to **True**, it changes the validator's behavior so that it will fail if the condition is met instead of when it is not met. The default is **False**. |
| **PropertyToCompare** | **Property To Compare** - This property is a the name of the property to compare |
| **Tag** | This property is a user-supplied string. Typically, it is used to sort or categorize validation results. |
| **TypeName** | **Type Name** – The fully qualified name of the type configuration element. This property cannot be edited. |

**Message Template Tokens**

If the message template contains tokens (for example, "{0}"), the validator will replace these tokens with values when the **ValidationResult** is created. The following table lists the tokens that are supported by the property comparison validator.

|  |  |
| --- | --- |
| Token | Meaning |
| **{0}** | This token represents the value of the object that is being validated. Although it can be useful to show the original value as a part of the validation message, you must be careful to avoid injection attacks by escaping any characters that can be used to attack the system that conveys the message to the user. |
| **{1}** | This token represents the key of the object that is being validated. When the validator is attached to a member of a type such as a property or a field, the key is set to the member name. When the validator is attached to an object, the key is null and the token is replaced by an empty string. |
| **{2}** | This token represents the tag that is specified on the validator instance. If no tag is supplied, the token is replaced by an empty string. |
| **{3}** | The value against which the current value is being compared. |
| **{4}** | The name of the property against which the current value is being compared. |
| **{5}** | The operator being used for comparison (**Equal**, **NotEqual**, **GreaterThan**, **GreaterThanEqual**, **LessThan**, **LessThanEqual**). |

[**Example**](javascript:void(0))

The following example shows how to use the validator with attributes to check that, for an **AuctionItem** object, the current bid is greater than or equal to the minimum bid.

public class AuctionItem

{

public double MinimimumBid

{

get

{

return minimumBid;

}

}

[PropertyComparisonValidator("MinimumBid", ComparisonOperator.GreaterThanEqual)]

public double CurrentBid

{

get

{

return currentBid;

}

}

// ...

}

**Range Validator**

Class Name: **RangeValidator<T>**

Attribute Name: **RangeValidatorAttribute**

Configuration tool name: **Range Validator**

[**Description**](javascript:void(0))

This validator checks that a value falls within a specified range. The range may be either closed, which means it has both a lower and an upper bound specified, or open, which means that it only has one bound specified.

The range validator can be used with any type that implements the **IComparable** interface. This includes all numeric types and strings. While it is possible, in code, to use this validator with **DateTime** types, the date time range validator may be a better choice because it allows you to take advantage of attributes and configuration.

[**Properties**](javascript:void(0))

The following table lists the range validator properties. The actual property names displayed in the configuration tools are listed in the table description.

|  |  |
| --- | --- |
| Property | Description |
| **Culture** | Indicates the name of the culture that will be used to readthe lower and upper bounds from the configuration file. |
| **LowerBound** | **Lower Bound** - This is the low-range boundary value. It can be any **IComparable** type that is compatible with the target. The compiler checks this requirement if you directly invoke the validator in your code. If you use attributes or configuration, the application block attempts to convert the supplied string representation of the value to the correct type at run time. |
| **LowerBoundType** | **Lower Bound Type** - This property determines how to evaluate the **LowerBound** value. Possible values for **LowerBoundType** are **Ignore**, **Inclusive**, and **Exclusive**. The **Ignore** value means that the validator ignores the **LowerBound** value. This is the default. The **Inclusive** value means that the validator allows values that are equal to the **LowerBound** value. The **Exclusive** value means that the validator does not allow values that are equal to the **LowerBound** value. |
| **MessageTemplate** | **Message Template** - This property is a string containing template tokens that the validator replaces with values as it validates the target. Typically, it describes the validation result. |
| **MessageTemplateResourceName** | **Template Resource Name** - If you do not want to use the **MessageTemplate** property to hard-code a message template (perhaps for internationalization), you can use a template stored in the application resources. You must also specify a **MessageTemplateResourceType** value. If you include both a **MessageTemplate** value and a **MessageTemeplateResourceName** value, the **MessageTemplate** value takes precedence. |
| **MessageTemplateResourceType** | **Template Resource Type -** The resource type for the template you want to use. If you specify a **MessageTemplateResourceName** value, you must specify this value. |
| **Name** | **Name** – The name to use for this validator. |
| **Negated** | This is a **Boolean** property. If it is set to **True**, it changes the validator's behavior so that it will fail if the condition is met, rather than when it is not met. The default is **False**. |
| **Tag** | This property is a user-supplied string. Typically, it is used to sort or categorize validation results. |
| **TypeName** | **Type Name** – The fully qualified name of the type configuration element. This property cannot be edited. |
| **UpperBound** | **Upper Bound** - This is the upper-range boundary value. It can be any **IComparable** type that is compatible with the target. The compiler checks this requirement if you directly invoke the validator in your code. If you use attributes or configuration, the application block attempts to convert the supplied string representation of the value to the correct type at run time. |
| **UpperBoundType** | **Upper Bound Type** - This property determines how to evaluate the **UpperBound** value. Possible values for **UpperBoundType** are **Ignore**, **Inclusive**, and **Exclusive**. The **Ignore** value means that the validator ignores the **UpperBound** value. The **Inclusive** value means that the validator allows values that are equal to the **UpperBound** value. This is the default. The **Exclusive** value means that the validator does not allow values that are equal to the **UpperBound** value. |

**Message Template Tokens**

If the message template contains tokens (for example, "{0}"), the validator will replace these tokens with values when the **ValidationResult** is created. The tokens supported by the range validator are shown in the following table.

|  |  |
| --- | --- |
| Token | Meaning |
| **{0}** | This token represents the value of the object that is being validated. Although it can be useful to show the original value as a part of the validation message, you must be careful to avoid injection attacks by escaping any characters that can be used to attack the system that conveys the message to the user. |
| **{1}** | This token represents the key of the object that is being validated. When the validator is attached to a member of a type such as a property or a field, the key is set to the member name. When the validator is attached to an object, the key is null and the token is replaced by an empty string. |
| **{2}** | This token represents the tag that is specified on the validator instance. If no tag is supplied, the token is replaced by an empty string. |
| **{3}** | The lower bound configured for the validator instance. |
| **{4}** | The lower bound type (**Inclusive**, **Exclusive**, or **Ignore**) configured for the validator instance. |
| **{5}** | The upper bound configured for the validator instance. |
| **{6}** | The upper bound type (**Inclusive**, **Exclusive**, or **Ignore**) configured for the validator instance. |

[**Example**](javascript:void(0))

The following example shows how to use the range validator with attributes to check that the **Age** property is between 0 and 110, inclusive.

public class Person

{

[RangeValidator(0, RangeBoundaryType.Inclusive, 110, RangeBoundaryType.Inclusive)]

int Age

{

get

{

return this.CalculateAge();

}

}

// ...

}

**Regular Expression Validator**

Class Name: **RegexValidator**

Attribute Name: **RegexValidatorAttribute**

Configuration tool name: **Regular Expression Validator**

[Description](javascript:void(0))

This validator checks that the value matches the pattern specified by a regular expression.

[Properties](javascript:void(0))

The following table lists the regular expression validator properties. The actual property names displayed in the configuration tools are listed in the table description.

|  |  |
| --- | --- |
| Property | Description |
| **MessageTemplate** | **Message Template** - This property is a string containing template tokens that the validator replaces with values as it validates the target. Typically, it describes the validation result. |
| **MessageTemplateResourceName** | **Template Resource Name** - If you do not want to use the **MessageTemplate** property to hard-code a message template (perhaps for internationalization), you can use a template stored in the application resources. You must also specify a **MessageTemplateResourceType** value. If you include both a **MessageTemplate** value and a **MessageTemeplateResourceName** value, the **MessageTemplate** value takes precedence. |
| **MessageTemplateResourceType** | **Template Resource Type -** The resource type for the template you want to use. If you specify a **MessageTemplateResourceName** value, you must specify this value. |
| **Name** | **Name** – The name to use for this validator. |
| **Options** | This property sets the option to use when validating a pattern. Possible values are **None**, **IgnoreCase**, **Multiline**, **ExplicitCapture**, **Compiled**, **Singleline**, **ECMAScript**, **CultureInvariant**, **IgnorePatternWhiteSpace**, and **RightToLeft**. The default is **None**. |
| **Pattern** | This property defines the regular expression. It is required. You can either enter the expression or use the **Regular Expression Editor** dialog box. |
| **PatternResourceName** | **RegEx Resource Name** - If you do not want to use the **Pattern** property to hard-code a regular expression (perhaps for internationalization), you can use a regular expression stored in the application resources. You must also specify a **PatternResourceName** value. If you include both a **Pattern** value and a **PatternResourceName** value, the **Pattern** value takes precedence. |
| **PatternResourceType** | **RegEx Resource Type** - The enumeration element type. |
| **PatternResourceTypeName** | **RegEx Resource Type Name** - The resource type for the pattern you want to use. If you specify a **PatternResourceName** value, you must specify this value. |
| **RegularExpression** | **Regular Expression** - The regular expression to use to validate the value. |
| **Tag** | This property is a user-supplied string. Typically, it is used to sort or categorize validation results. |
| **TypeName** | **Type Name** – The fully qualified name of the type configuration element. This property cannot be edited. |

Message Template Tokens

If the message template contains tokens (for example, "{0}"), the validator will replace these tokens with values when the **ValidationResult** is created. The tokens supported by the regular expression validator are listed in the following table.

|  |  |
| --- | --- |
| Token | Meaning |
| **{0}** | This token represents the value of the object that is being validated. Although it can be useful to show the original value as a part of the validation message, you must be careful to avoid injection attacks by escaping any characters that can be used to attack the system that conveys the message to the user. |
| **{1}** | This token represents the key of the object that is being validated. When the validator is attached to a member of a type such as a property or a field, the key is set to the member name. When the validator is attached to an object, the key is null and the token is replaced by an empty string. |
| **{2}** | This token represents the tag that is specified on the validator instance. If no tag is supplied, the token is replaced by an empty string. |
| **{3}** | The regular expression pattern configured for the validator instance. |
| **{4}** | The regular expression options configured for the validator instance. |

[Example](javascript:void(0))

The following example shows how to use the regular expression validator with attributes to check that an e-mail address is formed according to the pattern given by the regular expression.

public class Person

{

[RegexValidator(@"\w+([-+.']\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*")]

public string EmailAddress

{

get

{

return emailAddress;

}

}

// ...

}

**Relative Date Time Validator**

Class Name: **RelativeDateTimeValidator**

Attribute Name: **RelativeDateTimeValidatorAttribute**

Configuration tool name: **Relative Date Time Validator**

[Description](javascript:void(0))

This validator checks that the **DateTime** value falls within a specified range using relative times and dates.

[Properties](javascript:void(0))

The following table lists the relative date time validator properties. The actual property names displayed in the configuration tools are listed in the table description.

|  |  |
| --- | --- |
| Property | Description |
| **LowerBound** | **Lower Bound** - This is the low-range boundary value representing the number of a specific unit of time (such as days) before the date being validated. It must be of type **Integer**. The compiler checks this requirement if you directly invoke the validator in your code. If you use attributes or configuration, the application block throws an exception at run time if the type is incorrect. |
| **LowerBoundType** | **Lower Bound Type** - This property determines how to evaluate the **LowerBound** value. Possible values for **LowerBoundType** are **Ignore**, **Inclusive**, and **Exclusive**. The **Ignore** value means that the validator ignores the **LowerBound** value. This is the default. The **Inclusive** value means that the validator allows values that are equal to the **LowerBound** value. The **Exclusive** value means that the validator does not allow values that are equal to the **LowerBound** value. |
| **LowerUnit** | **Lower Bound Unit** - This property sets the unit of time for the lower boundary. The possible values are **None**, **Second**, **Minute**, **Hour**, **Day**, **Month**, and **Year**. The default is **None**. |
| **MessageTemplate** | **Message Template** - This property is a string containing template tokens that the validator replaces with values as it validates the target. Typically, it describes the validation result. |
| **MessageTemplateResourceName** | **Template Resource Name** - If you do not want to use the **MessageTemplate** property to hard-code a message template (perhaps for internationalization), you can use a template stored in the application resources. You must also specify a **MessageTemplateResourceType** value. If you include both a **MessageTemplate** value and a **MessageTemeplateResourceName** value, the **MessageTemplate** value takes precedence. |
| **MessageTemplateResourceType** | **Template Resource Type -** The resource type for the template you want to use. If you specify a **MessageTemplateResourceName** value, you must specify this value. |
| **Name** | **Name** – The name to use for this validator. |
| **Negated** | This is a **Boolean** property. If it is set to **True**, it changes the validator's behavior so that it will fail if the condition is met, rather than when it is not met. The default is **False**. |
| **Tag** | This property is a user-supplied string. Typically, it is used to sort or categorize validation results. |
| **TypeName** | **Type Name** – The fully qualified name of the type configuration element. This property cannot be edited. |
| **UpperBound** | **Upper Bound** - This is the upper-range boundary value representing the number of a specific unit of time (such as days) after the date being validated. It must be type **Integer**. The compiler checks this requirement if you directly invoke the validator in your code. If you use attributes or configuration, the application block throws an exception at run time if the type is incorrect. |
| **UpperBoundType** | **Upper Bound Type** - This property determines how to evaluate the **UpperBound** value. Possible values for **UpperBoundType** are **Ignore**, **Inclusive**, and **Exclusive**. The **Ignore** value means that the validator ignores the **UpperBound** value. **Inclusive** is the default. The **Inclusive** value means that the validator allows values that are equal to the **UpperBound** value. The **Exclusive** value means that the validator does not allow values that are equal to the **UpperBound** value. |
| **UpperUnit** | **Upper Bound Unit** - This property sets the unit of time for the upper boundary. The possible values are **None**, **Second**, **Minute**, **Hour**, **Day**, **Month**, and **Year**. The default is **None**. |

Message Template Tokens

If the message template contains tokens (for example, "{0}"), the validator will replace these tokens with values when the **ValidationResult** is created. The tokens supported by the relative date time validator are listed in the following table.

|  |  |
| --- | --- |
| Token | Meaning |
| **{0}** | This token represents the value of the object that is being validated. Although it can be useful to show the original value as a part of the validation message, you must be careful to avoid injection attacks by escaping any characters that can be used to attack the system that conveys the message to the user. |
| **{1}** | This token represents the key of the object that is being validated. When the validator is attached to a member of a type such as a property or a field, the key is set to the member name. When the validator is attached to an object, the key is null and the token is replaced by an empty string. |
| **{2}** | This token represents the tag that is specified on the validator instance. If no tag is supplied, the token is replaced by an empty string. |

[Example](javascript:void(0))

The following example shows how to use the relative date time validator with attributes. It attaches the **RelativeDateTimeValidator** attribute to the **DateOfBirth** property and checks to see if the user is 18 years or older.

public class Person

{

[RelativeDateTimeValidator(-120, DateTimeUnit.Year, -18, DateTimeUnit.Year,

Ruleset="RuleSetA", MessageTemplate="Must be 18 years or older.")]

public DateTime DateOfBirth

{

get

{

return dateOfBirth;

}

}

}

**String Length Validator**

Class Name: **StringLengthValidator**

Attribute Name: **StringLengthValidatorAttribute**

Configuration tool name: **String Length Validator**

[Description](javascript:void(0))

This validator checks that the length of the string is within the specified range. The range may include or exclude the endpoints by omitting the lower or upper bound.

[Properties](javascript:void(0))

The following table lists the string length validator properties. The actual property names displayed in the configuration tools are listed in the table description.

|  |  |
| --- | --- |
| Property | Description |
| **LowerBound** | **Lower Bound** - This is the minimum length of the target string. It must be of type **Integer**. The compiler checks this requirement if you directly invoke the validator in your code. |
| **LowerBoundType** | **Lower Bound Type** - This property determines how to evaluate the **LowerBound** value. Possible values for **LowerBoundType** are **Ignore**, **Inclusive**, and **Exclusive**. The **Ignore** value means that the validator ignores the **LowerBound** value. This is the default. The **Inclusive** value means that the validator allows values that are equal to the **LowerBound** value. The **Exclusive** value means that the validator does not allow values that are equal to the **LowerBound** value. |
| **MessageTemplate** | **Message Template** - This property is a string containing template tokens that the validator replaces with values as it validates the target. Typically, it describes the validation result. |
| **MessageTemplateResourceName** | **Template Resource Name** - If you do not want to use the **MessageTemplate** property to hard-code a message template (perhaps for internationalization), you can use a template stored in the application resources. You must also specify a **MessageTemplateResourceType** value. If you include both a **MessageTemplate** value and a **MessageTemeplateResourceName** value, the **MessageTemplate** value takes precedence. |
| **MessageTemplateResourceType** | **Template Resource Type -** The resource type for the template you want to use. If you specify a **MessageTemplateResourceName** value, you must specify this value. |
| **Name** | **Name** – The name to use for this validator. |
| **Negated** | This is a Boolean property. If it is set to **True**, it changes the validator's behavior so that it will fail if the condition is met, rather than when it is not met. The default is **False**. |
| **Tag** | This property is a user-supplied string. Typically, it is used to sort or categorize validation results. |
| **TypeName** | **Type Name** – The fully qualified name of the type configuration element. This property cannot be edited. |
| **UpperBound** | **Upper Bound** - This is the maximum length of the target string. It must be type **Integer**. The compiler checks this requirement if you directly invoke the validator in your code. |
| **UpperBoundType** | **Upper Bound Type** - This property determines how to evaluate the **UpperBound** value. Possible values for **UpperBoundType** are **Ignore**, **Inclusive**, and **Exclusive**. The **Ignore** value means that the validator ignores the **UpperBound** value. **Inclusive** is the default. The **Inclusive** value means that the validator allows values that are equal to the **UpperBound** value. The **Exclusive** value means that the validator does not allow values that are equal to the **UpperBound** value. |

Message Template Tokens

If the message template contains tokens (for example, "{0}"), the validator will replace these tokens with values when the **ValidationResult** is created. The tokens supported by the string length validator are listed in the following table.

|  |  |  |
| --- | --- | --- |
| Token | Meaning | |
| **{0}** | This token represents the value of the object that is being validated. Although it can be useful to show the original value as a part of the validation message, you must be careful to avoid injection attacks by escaping any characters that can be used to attack the system that conveys the message to the user. |  |
| **{1}** | This token represents the key of the object that is being validated. When the validator is attached to a member of a type such as a property or a field, the key is set to the member name. When the validator is attached to an object, the key is null and the token is replaced by an empty string. |  |
| **{2}** | This token represents the tag that is specified on the validator instance. If no tag is supplied, the token is replaced by an empty string. |  |
| **{3}** | The lower bound configured for the validator instance. | |
| **{4}** | The lower bound type (**Inclusive**, **Exclusive**, or **Ignore**) configured for the validator instance. | |
| **{5}** | The upper bound configured for the validator instance. | |
| **{6}** | The upper bound type (**Inclusive**, **Exclusive** or **Ignore**) configured for the validator instance. | |

[Example](javascript:void(0))

The following example shows how to use the validator with attributes to check that the product code is a string between 1 and 10 characters long.

public class Product

{

[StringLengthValidator(1, 10)]

public string ProductCode

{

get

{

return productCode;

}

}

// ...

}

**Type Conversion Validator**

Class Name: **TypeConversionValidator**

Attribute Name: **TypeConversionValidatorAttribute**

Configuration tool name: **Type Conversion Validator**

[Description](javascript:void(0))

This validator checks that a string can be converted to a specific type. For example, the validator can check that "6.32" can be converted to a **Double** type or that "2007-02-09" can be converted to a **DateTime** type.

[Properties](javascript:void(0))

The following table lists the type conversion validator properties. The actual property names displayed in the configuration tools are listed in the table description.

|  |  |
| --- | --- |
| Property | Description |
| **MessageTemplate** | **Message Template** - This property is a string containing template tokens that the validator replaces with values as it validates the target. Typically, it describes the validation result. |
| **MessageTemplateResourceName** | **Template Resource Name** - If you do not want to use the **MessageTemplate** property to hard-code a message template (perhaps for internationalization), you can use a template stored in the application resources. You must also specify a **MessageTemplateResourceType** value. If you include both a **MessageTemplate** value and a **MessageTemeplateResourceName** value, the **MessageTemplate** value takes precedence. |
| **MessageTemplateResourceType** | **Template Resource Type -** The resource type for the template you want to use. If you specify a **MessageTemplateResourceName** value, you must specify this value. |
| **Name** | **Name** – The name to use for this validator. |
| **Negated** | This is a **Boolean** property. If it is set to **True**, it changes the validator's behavior so that it will fail if the condition is met, rather than when it is not met. The default is **False**. |
| **Tag** | This property is a user-supplied string. Typically, it is used to sort or categorize validation results. |
| **TargetType** | **Target Type** - The type to which you want to convert the string. You can either enter the type or select it with the **Type Selector – System.Object** dialog box. |
| **TypeName** | **Type Name** – The fully qualified name of the type configuration element. This property cannot be edited. |

Message Template Tokens

If the message template contains tokens (for example, "{0}"), the validator will replace these tokens with values when the **ValidationResult** is created. The tokens supported by the type conversion validator are listed in the following table.

|  |  |
| --- | --- |
| Token | Meaning |
| **{0}** | This token represents the value of the object that is being validated. Although it can be useful to show the original value as a part of the validation message, you must be careful to avoid injection attacks by escaping any characters that can be used to attack the system that conveys the message to the user. |
| **{1}** | This token represents the key of the object that is being validated. When the validator is attached to a member of a type such as a property or a field, the key is set to the member name. When the validator is attached to an object, the key is null and the token is replaced by an empty string. |
| **{2}** | This token represents the tag that is specified on the validator instance. If no tag is supplied, the token is replaced by an empty string. |
| **{3}** | The full name of the type to which the validator will attempt to convert. |

[Example](javascript:void(0))

The following example shows how to use the type conversion validator with attributes to check if the string **DiscountString** can be converted to type **Double**.

public class Product

{

[TypeConversionValidator(typeof(double))]

public string DiscountString

{

get

{

return discountString;

}

}

// ...

}

**Single Member Validators**

The Validation Application Block contains three validators that you can use to validate individual members of types, instead of validating the entire type using attributes or rule sets. While not a common scenario, this technique may be useful when integrating with other frameworks such as WPF and Windows Forms. The three validators are:

FieldValueValidator. Use this validator to validate a field of a type.

MethodReturnValueValidator. Use this validator to validate the return value of a method of a type.

PropertyValueValidator. Use this validator to validate the value of a property of a type.

For example, you can programmatically create a validator for an instance of a class named MyClass that validates the value of a property named MyProperty using a regular expression validator as shown here.

Validator propValidator = new PropertyValueValidator<MyClass>("MyProperty",

new RegexValidator("some-regular-expression"));

MyClass myInstance = new MyClass();

myInstance.MyProperty = "Some value";

ValidationResults results = propValidator.Validate(myInstance);

That second parameter to the constructor is the validator to use for the property value. You can also create a composite validator from a combination of validators, and specify this composite validator in the code above. A similar technique can be used with the FieldValueValidator and MethodReturnValueValidator.