## **验证属性**

下面是几个常用的内置验证属性：

[CreditCard]：验证属性是否具有信用卡格式。

[Compare]：验证某个模型中的两个属性是否匹配。

[EmailAddress]：验证属性是否具有电子邮件格式。

[Phone]：验证属性是否具有电话格式。

[Range]：验证属性值是否落在给定范围内。

[RegularExpression]：验证数据是否与指定的正则表达式匹配。

[Required]：将属性设置为必需属性。

[StringLength]：验证字符串属性是否最多具有给定的最大长度。

[Url]：验证属性是否具有 URL 格式。

MVC 支持从 ValidationAttribute 派生的所有用于验证的属性。

## **必需属性的使用说明**

从本质上来说，需要不可以为 null 的[值类型](https://docs.microsoft.com/zh-cn/dotnet/csharp/language-reference/keywords/value-types" \t "https://blog.csdn.net/boonya/article/details/_blank)（如 decimal、int、float 和 DateTime），但不需要 Required 特性。

[BindRequired 特性](https://docs.microsoft.com/zh-cn/dotnet/api/microsoft.aspnetcore.mvc.modelbinding.bindrequiredattribute" \t "https://blog.csdn.net/boonya/article/details/_blank)（另请参阅 [ASP.NET Core 中的模型绑定](https://docs.microsoft.com/zh-cn/aspnet/core/mvc/models/model-binding?view=aspnetcore-2.1" \l "customize-model-binding-behavior-with-attributes" \t "https://blog.csdn.net/boonya/article/details/_blank)）可用于确保表单数据完整。 当应用于某个属性时，模型绑定系统要求该属性具有值。 当应用于某个类型时，模型绑定系统要求该类型的所有属性都具有值

## **处理模型状态错误**

在执行控制器操作之前进行模型验证。

该操作负责检查 ModelState.IsValid 并做出相应响应。 在许多情况下，正确的反应是返回错误响应，理想状况下会详细说明模型验证失败的原因。

如果在使用 [ApiController] 属性的 web API 控制器中，ModelState.IsValid 的计算结果为 false，将返回包含问题详细信息的自动 HTTP 400 响应。 有关详细信息，请参阅[自动 HTTP 400 响应](https://docs.microsoft.com/zh-cn/aspnet/core/web-api/index?view=aspnetcore-2.1" \l "automatic-http-400-responses" \t "https://blog.csdn.net/boonya/article/details/_blank)。

## **自定义验证**

在 MVC 中创建你自己的自定义验证属性很简单。 只需从 ValidationAttribute 继承并重写 IsValid 方法。 IsValid 方法采用两个参数，第一个是名为 *value* 的对象，第二个是名为 *validationContext* 的 ValidationContext 对象。 *Value* 引用自定义验证程序要验证的字段中的实际值。

public class ClassicMovieAttribute : ValidationAttribute, IClientModelValidator

{

private int \_year;

public ClassicMovieAttribute(int year)

{

\_year = year;

}

protected override ValidationResult IsValid(object value, ValidationContext validationContext)

{

Movie movie = (Movie)validationContext.ObjectInstance;

if (movie.Genre == Genre.Classic && movie.ReleaseDate.Year > \_year)

{

return new ValidationResult(GetErrorMessage());

}

return ValidationResult.Success;

}