Работа с атрибутами валидации

Edit was unsuccessful. Please correct the errors and try again.

- **The Name field is required.**
- **The field Color must be a string with a maximum length of 15.**
- **The field Weight must be between 0 and 9999.**

Fields

Name:

*

Color:

Black - Very dark with fading edge *

Weight:

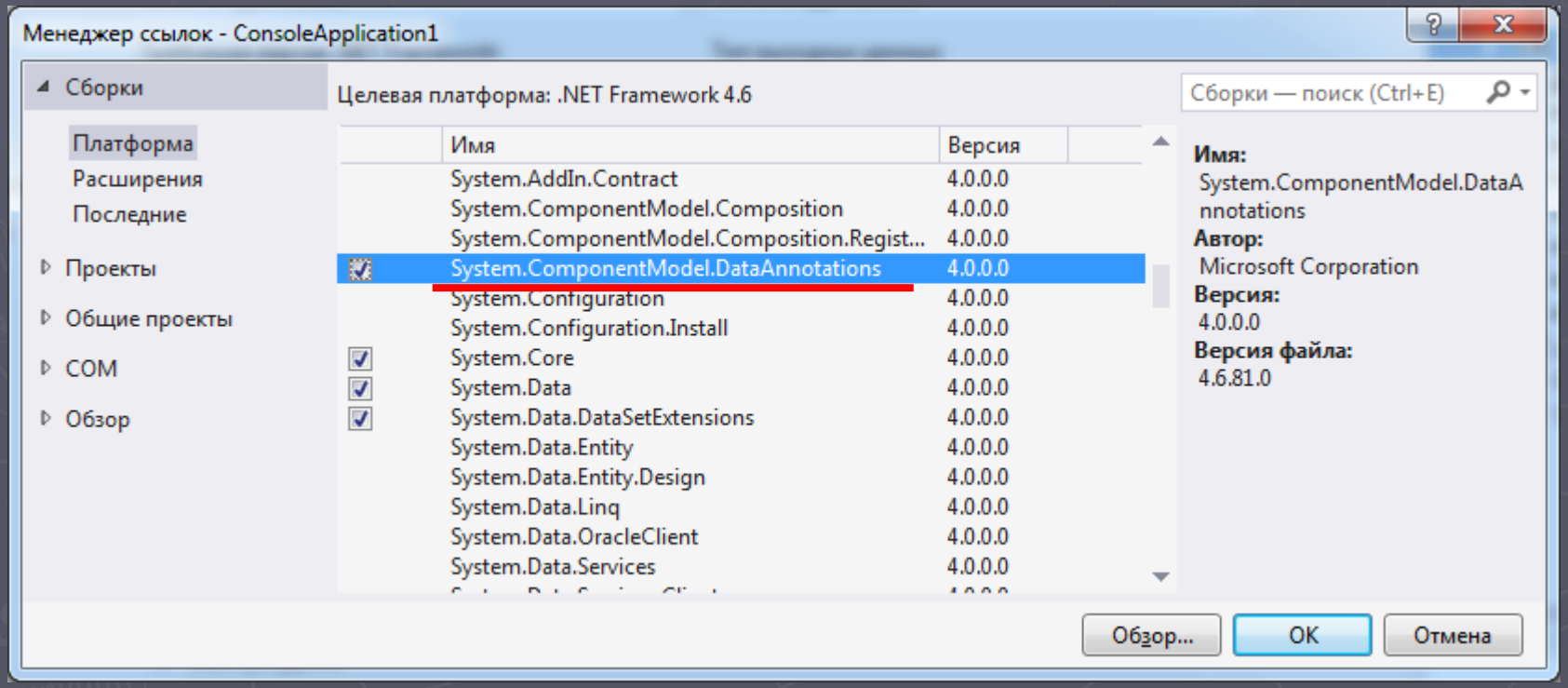
-16.38 *

Save

Атрибуты валидации модели

проверка вводимых данных на корректность

- встроенные атрибуты конкретизируют правило валидации



```
public class Student
    {
        [Required]
        public string Id { get; set; }
        [Required]
        public string Name { get; set; }
        [Required]
        [Range(16, 55)]
        public int Age { get; set; }
    }
```

класс RequiredAttribute - требует обязательного наличия

RangeAttribute - диапазон приемлимых значений

атрибуты имееют ряд общих и собственных свойств

```
public class Student
    {
        [Required(ErrorMessage = "Отсутвует ID")]
        public string Id { get; set; }


       [Required(ErrorMessage = "Отсутвует имя",
                                AllowEmptyStrings = true)]
        public string Name { get; set; }


        [Required(ErrorMessage = "Отсутвует возраст")]
        [Range(16, 55,ErrorMessage =
                                "Диапазон возраста от 16 до 55")]
        public int Age { get; set; }
    }
```

```
[RegularExpression(@"[0-9]{2}", ErrorMessage = "Неверный формат")]

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

указывает на регулярное выражение, которому должно соответствовать значение свойства

```
[StringLength(10, MinimumLength = 2,
                        ErrorMessage = "Недопустимая длина имени")]

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

определяет допустимую длину для строковых свойств.

позволяет сравнить значение текущего свойства со значением другого свойства

```
 [Compare("Name")]

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

Управление валидацией

```
Student dima = new Student { Name = "Dima", Age = 13 };

    var results = new List<ValidationResult>();
```

список объектов ValidationResult оказывается заполенным если валидация не пройдена

```
    var context = new ValidationContext(dima);
```

контекст валидации - передается валидируемый объект,

```
if (!Validator.TryValidateObject(dima, context, results, true))
            {
                foreach (var error in results)
                {
                    string strWithErrroe  = error.ErrorMessage;
                }
            }
```

`MemberNames` - список свойств, для которых возникла ошибка,
`ErrorMessage` - сообщение об ошибке

Создание атрибутов валидации

```
public class PassportAttribute : ValidationAttribute
    {
        public override bool IsValid(object value)
        {
            if (value != null)
            {
                string passport = value.ToString();
                if (passport.StartsWith("MP"))
                    return true;
                else
                    this.ErrorMessage = "номер должен начинаться с MP";
            }
            return false;
        }
```

```
[Passport]
        public string Passport { get; set; }
```

Самовалидация модели

```
public class Student : IValidatableObject
    {

        public string Id { get; set; }

        public string Name { get; set; }

        public int Age { get; set; }
        public IEnumerable<ValidationResult> Validate
                                        (ValidationContext validationContext
        {
            throw new NotImplementedException();
        }
    }
```