Валидация модели и ModelState

Валидация позволяет проверить входные данные на наличие неправильных, корректных значений и должным образом обработать эти значения. Для рассмотрения валидации создадим модель - класс Person:

```
using System.ComponentModel.DataAnnotations;
namespace ValidationApp.Models
{
    public class Person
    {
        [Required]
        public string Name { get; set; }
        [Required]
        public string Email { get; set; }
        [Required]
        public string Password { get; set; }
        public int Age { get; set; }
    }
}
```

Здесь каждое свойство модели помечено атрибутом Required, который находится в пространстве имен System.ComponentModel.DataAnnotations.

Затем определим в контроллере HomeController действие Create, через которое мы будем добавлять на сервер объект модели Person:

```
public class HomeController: Controller
{
    public IActionResult Create()
    {
        return View();
    }
    [HttpPost]
    public IActionResult Create(Person person)
    {
        if (ModelState.IsValid)
            return Content($"{person.Name} - {person.Email}");
        else
            return View(person);
    }
}
```

Если данные для модели были введены правильно, то через метод Content() эти данные будут выведены в браузере. Если же были введены некорректные данные, то возвращаем объект Person в метод View. Объект ModelState сохраняет все значения, которые пользователь ввел для свойств модели, а также все ошибки, связанные с каждым свойством и с моделью в целом. Если в объекте ModelState имеются какие-нибудь ошибки, то свойство ModelState.IsValid возвратит false.

Далее определим само представление Create.cshtml, которое будет содержать форму для ввода данных:

```
@model ValidationApp.Models.Person

<form asp-antiforgery="true" asp-action="Create" asp-
controller="Home">
    <div>
        <div class="validation" asp-validation-
summary="ModelOnly"></div>
        <div>
            <label asp-for="Name"></label><br />
            <input type="text" asp-for="Name" />
            <span asp-validation-for="Name"></span>
        </div>
        <div>
            <label asp-for="Email"></label><br />
            <input asp-for="Email" />
            <span asp-validation-for="Email"></span>
        </div>
        <div>
            <label asp-for="Password"></label><br />
            <input asp-for="Password" />
            <span asp-validation-for="Password"></span>
        </div>
        <div>
            <label asp-for="Age"></label><br />
            <input asp-for="Age" />
            <span asp-validation-for="Age"></span>
        </div>
        <div>
            <input type="submit" value="Save" />
        </div>
    </div>
</form>
```

Это стандартное строго типизированное представление, которое использует tag-хелперы для создания полей для свойств модели Person.

Основные моменты валидации в данном случае:

1. Мы указали для каждого свойства атрибут Required, благодаря чему фреймворк знает, что данное свойство обязательно должно содержать некоторое значение.
2. При каждом свойстве мы используем хелпер валидации :

   ```
   <div>
     <label asp-for="Name"></label><br />
     <input type="text" asp-for="Name" />
     <span asp-validation-for="Name" />
   </div>
   ```

   Благодаря чему и отображается сообщение об ошибке.
3. Во фреймворке предусмотрена валидация на стороне сервера:

   ```
   if (ModelState.IsValid)
        return Content($"{person.Name} - {person.Email}");
   else
        return View(person);
   ```

С помощью свойства ModelState.IsValid мы узнаем, проходит модель валидацию или нет, и зависимости от результата совершаем те или иные действия.

Мы также можем проверять корректность значений отдельных свойств модели или указать ошибки уровня модели:

```
[HttpPost]
public IActionResult Create(Person person)
{
    if (string.IsNullOrEmpty(person.Name))
    {
        ModelState.AddModelError("Name", "Некорректное имя");
    }
    if (person.Name == person.Password)
    {
        ModelState.AddModelError("", "Имя и пароль не должны
     совпадать");
    }
    if (ModelState.IsValid)
    {
        return Content($"{person.Name} - {person.Email}");
    }

    return View(person);
}
```

Для ошибок уровня модели в представлении используется отдельный хелпер:

```
<div class="validation" asp-validation-summary="ModelOnly"></div>
```

Самые часто используемые атрибуты валидации:

`[Required]`
данное свойство должно быть обязательно установлено

`[RegularExpression]`
вводимое значение должно соответствовать указанному в этом атрибуте регулярному выражению

`[StringLength]`

Значение не должно превышать указанное количество символов, допольнительно можно задать минимальное значение с помощью параметра MinimumLength

```
[StringLength(50, MinimumLength = 3)]
```

`[Range]`
атрибут Range определяет минимальные и максимальные ограничения для числовых данных

`[Compare]`
атрибут Compare гарантирует, что два свойства объекта модели имеют одно и то же значение

`[EmailAddress]`
проверяет на корректность email-адрес

`[Phone]`
проверяет на корректность телефон

С помощью свойства ErrorMessage этого атрибута можно настроить выводимое при валидации сообщение. А если мы явным образом не установим текст сообщения, то при выводе ошибки будет отображаться стандартный текст сообщения.

```
[Required (ErrorMessage = "Не указан электронный адрес")]
```

Подробнее: https://metanit.com/sharp/aspnet5/19.1.php