**FormFieldValidation**

С този пример ще покажем базова функционалност за валидация на полета на форма. Ще се използват библиотеките jquery.validate.js и jquery.validate.unobtrusive.js.

Също така ще създадем и на server-side валидация.

Не ползваме html 5 валидацията!

------------------------------------

Първоначално създаваме празен проект:

VS2015 -> File -> New -> Project -> Web -> ASP.NET Web Application

с името FormFieldValidation, избираме template: Empty + отметката за MVC

------------------------------------

Създаваме празен контролер в директорията Controllers

Controllers -> Add -> Controller… -> MVC 5 Controller – Empty

с името HomeController

---

Създаваме viewmodel клас в директорията Models

Models -> Add -> Class…

с името RegisterViewModel

Този клас ще съдържа полетата, които ще се попълваме от форма във дадено View. Също в този клас ще зададем с атрибути кои полета искаме да валидираме. За начало може да добавим само два атрибута:

[Required] – чрез който указваме, че даденото поле не трябва да бъде празно.

[Display(Name ="Име на полето") ] – с какво име да се извежда във формата

Ето и класа как ще изглежда:

---

public class RegisterViewModel

{

[Required]

[Display(Name = "First Name")]

public string FirstName { get; set; }

[Required]

[Display(Name = "Last Name")]

public string LastName { get; set; }

[Required]

[Display(Name = "Age")]

public int Age { get; set; }

[Required]

[Display(Name = "Email")]

public string Email { get; set; }

}

---

За да ползваме атрибутите трябва да добавим с Ctrl +. Библиотеката:

using System.ComponentModel.DataAnnotations;

--------------------------------------

Добавяме в HomeController още един action с името Register,който ще бъде за формата за попълване.

---

[HttpGet]

public ActionResult Register()

{

return View();

}

---

И добавяме view() -> Add View със същото име Register. Това View ще бъде типизирано и ще използва за модел обект от типа RegisterViewModel

---

@using FormFieldValidation.Models

@model RegisterViewModel

@{

ViewBag.Title = "Register";

}

<h2>Register</h2>

@using (Html.BeginForm("Register", "Home", FormMethod.Post))

{

<div>

@Html.LabelFor(model => model.FirstName)

@Html.EditorFor(model => model.FirstName)

@Html.ValidationMessageFor(model => model.FirstName, "", new { @class = "text-danger" })

</div>

<div>

@Html.LabelFor(model => model.LastName)

@Html.EditorFor(model => model.LastName)

@Html.ValidationMessageFor(model => model.LastName, "", new { @class = "text-danger" })

</div>

<div>

@Html.LabelFor(model => model.Age)

@Html.TextBoxFor(model => model.Age)

@Html.ValidationMessageFor(model => model.Age, "", new { @class = "text-danger" })

</div>

<div>

@Html.LabelFor(model => model.Email)

@Html.EditorFor(model => model.Email)

@Html.ValidationMessageFor(model => model.Email, "", new { @class = "text-danger" })

</div>

<div>

<input type="submit" value="Save" />

</div>

}

---

----------------------------------

Тъй като при стартиране се отваря Index от контролера Home, трябва да го създадем това view

Add View ->Index, което ще се намира в \Views\Home\Index.cshtml

В него добавяме линк към view-то Register:

---

@{

ViewBag.Title = "Index";

}

<h2>Index</h2>

@ViewBag.Message @\* Actually there is no need to check if it is null or "" \*@

@Html.ActionLink("Register", "Register", "Home")

---

--------------------------------------

Пробваме и виждаме, че не работи валидацията още (тъй като нямаме action за POST).

Затова добавяме в HomeContorller още един Action с името Edit

---

[HttpPost]

public ActionResult Register(RegisterViewModel viewModel)

{

// Do not save to database when the ModelState is not valid !

if (ModelState.IsValid)

{

// save the information to the database

// ...

TempData["Message"] = "You registered succesfully!";

return RedirectToAction("Index");

}

else

{

return View();

}

}

---

Вече валидацията трябва да работи, но само server-side

Обясняваме ModelState, ModelState.IsValid, ModelState.AddModelError()   
Дебъгваме и показаме quickwatch -> ModelState и къде са грешките

Чрез пропъртито ModelState.IsValid се проверява дали модела, който е подаден от формата е валидиран.

Това е задължително да се прави преди да правим каквото и да е с даннине (например записване в база данни).

Добре е да дебъгнем примера, като сложим breakpoint на:

**О** if (ModelState.IsValid)

И разгледаме на ModelState.Values[].Errors

---

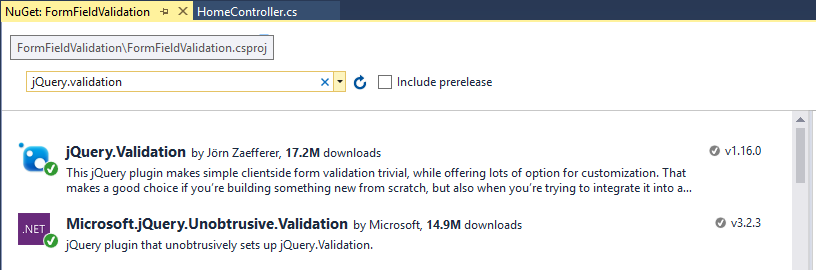
Сега ще добавим js библиотеките за client-side валидация

Даваме десен бутон на проекта -> Manage NuGet Packages… ->

Browse: jquery.validation

И го инсталираме.

Добавяме пак от NuGet и jQuery.Validation.Unobtrusive, което също инсталираме.



За да ползваме тези библиотеки трябва да ги добавим в \Views\Shared\\_Layout.cshtml

В самия \_Layout, библотеките се добавят в края, точно преди затварящия таг </body>.

А добавянето става с drag&drop като хващаме съответната библиотека от директорията \Scripts

-----------------------------------------------

Вече client-side валидацията трябва да работи

----

За да извеждаме подаденото съобщение от Register във Index, трябва да допишем Action-a в HomeController:

---

public ActionResult Index()

{

// add Message to the Viewbag if another Action has added it to the TempData

if (TempData["Message"] != null)

{

ViewBag.Message = TempData["Message"];

}

return View();

}

---

--------------------------------

Нека добавим още проверки за валидация чрез допълнителни атрибути в класа RegisterViewModel

public class RegisterViewModel

{

[Required]

[MinLength(4, ErrorMessage = "First Name is at least 4 symbols")]

[MaxLength(10, ErrorMessage = "First Name is max 10 symbols")]

[Display(Name = "First Name")]

public string FirstName { get; set; }

[Required]

[StringLength(20, MinimumLength = 5, ErrorMessage = "The field {0} should be between 5 and 10 characters")]

[RegularExpression("([A-Za-z]+)", ErrorMessage = "last name should contain only letters")]

[Display(Name = "Last Name")]

public string LastName { get; set; }

[Required]

[Range(12, 120, ErrorMessage = "You should be above 12 years to use this site!")]

[Display(Name = "Age")]

public int Age { get; set; }

[Required]

[EmailAddress]

[Display(Name = "Email")]

public string Email { get; set; }

}

И отново да пробваме.

---------------------------------------

Засега направихме само валидация с атрибути дефинирани в DataAnnotations които създават както сървърна, така и клиенткса валидация.

Можем да добавим допълнителна по специфична сървърна валидация за водните данни.

Например нека добавим валидация за полето FirstName да започва с главна буква и валидация за блокираните имена за потребители. Това става в Action-a Register

[HttpPost]

public ActionResult Register(RegisterViewModel viewModel)

{

// server validation for Firstname to be with capital letter

if (string.IsNullOrEmpty(viewModel.FirstName) == false)

{

char firstLetter = viewModel.FirstName[0];

if (char.IsUpper(firstLetter) == false)

{

ModelState.AddModelError("FirstName", "First name should start with a capital letter!");

}

}

// server validation for the banned users

if (viewModel.FirstName == "Ivan" && viewModel.LastName == "Ivanov")

{

ModelState.AddModelError("", "Sorry, this user is banned from our web server!");

}

// Do not save to database when the ModelState is not valid !

if (ModelState.IsValid)

{

// save the information to the database

// ...

TempData["Message"] = "You registered succesfully!";

return RedirectToAction("Index");

}

else

{

return View();

}

}

---

Обръщаме внимание, че към ModelState могат да се добавят допълнителни грешки, които ще променят IsValid.

---------------------------------------

Накрая можем да добавим и Summary на грешките в Register.cshtml, точно след отваряне на формата:

@using (Html.BeginForm("Register", "Home", FormMethod.Post))

{

<div class="text-danger">

@Html.ValidationSummary(true)

</div>

. . .

---

Показваме примера за [Remote()] атрибут за валидация.:

..

[Remote("ValidateEmail", "Home", ErrorMessage = "This email is already used.")]

public string Email { get; set; }

и action-а:

public ActionResult ValidateEmail(string email)

{

bool isEmailUsed = false;

if (string.IsNullOrEmpty(email) == false)

{

isEmailUsed = (email == "trendo@abv.bg") || (email == "venelinmonev@gmail.com");

}

return Json(!isEmailUsed, JsonRequestBehavior.AllowGet);

}

Това е много добре описано в <http://www.devtrends.co.uk/blog/the-complete-guide-to-validation-in-asp.net-mvc-3-part-1>