

Today's Agenda

Validation techniques

- Explicit Model Validation
- Displaying Validation Error Messages
- Property Validation Attributes
- Regular Expressions
- Custom Property Validation Attribute
- Model Validation Attribute
- Self-Validating Models
- Applying Client Side Validation
- Remote Validation (with Ajax)

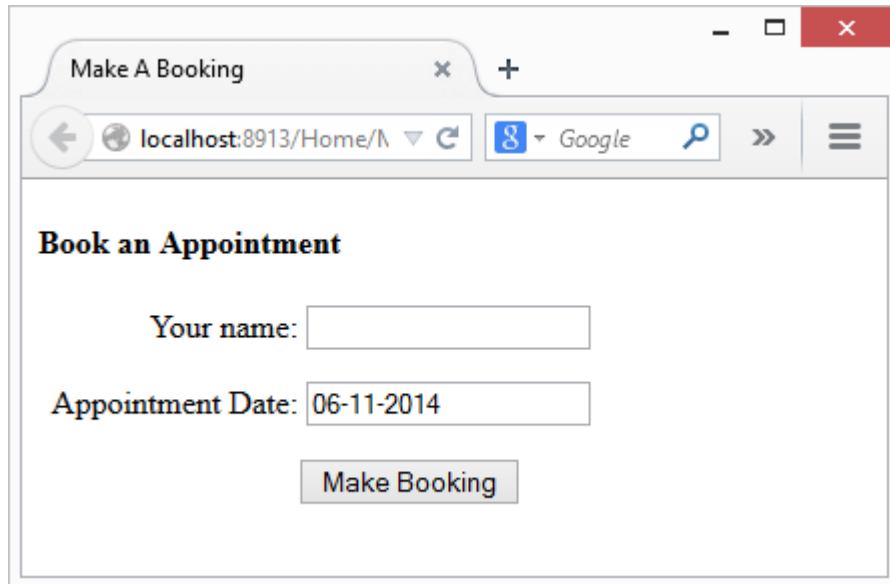
Exercises

Two types of server side validation

- Explicit validation of the Model
- Property validation attributes with data annotations

Explicit Model Validation

Explicit Model Validation (Model & View)



Make A Booking

Book an Appointment

Your name:

Appointment Date:

Strongly typed **Appointment** view

```
public class Appointment {  
    public string FullName { get; set; }  
  
    [DataType(DataType.Date)]  
    public DateTime Date { get; set; }  
}
```

Web.config

```
<system.web>  
    <globalization culture="da-DK" uiCulture="da-DK"/>  
</system.web>
```

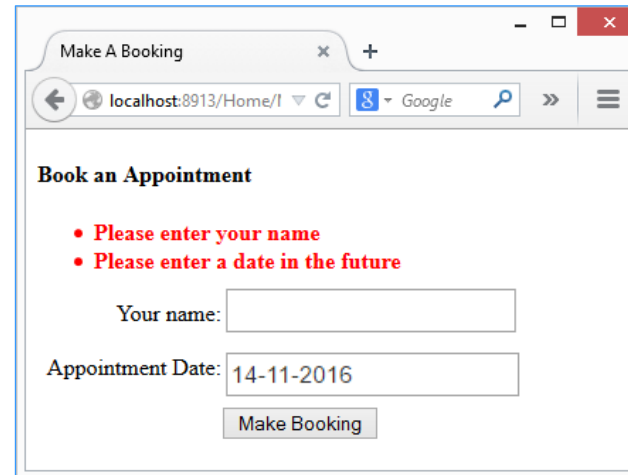
Explicit Model Validation (Action Method)

```
[HttpPost]
public ActionResult MakeBooking(Appointment appt) {
    if (string.IsNullOrEmpty(appt.FullName)) {
        ModelState.AddModelError("FullName", "Please enter your name"); // name of prop.
    }
    if (ModelState.IsValidField("Date") && DateTime.Now > appt.Date) {
        ModelState.AddModelError("Date", "Please enter a date in the future");
    }
    if (ModelState.IsValid) {
        // store new Appointment in repository
        return View("Completed", appt);
    }
    else {
        return View();
    }
}
```

Displaying Validation Error Messages

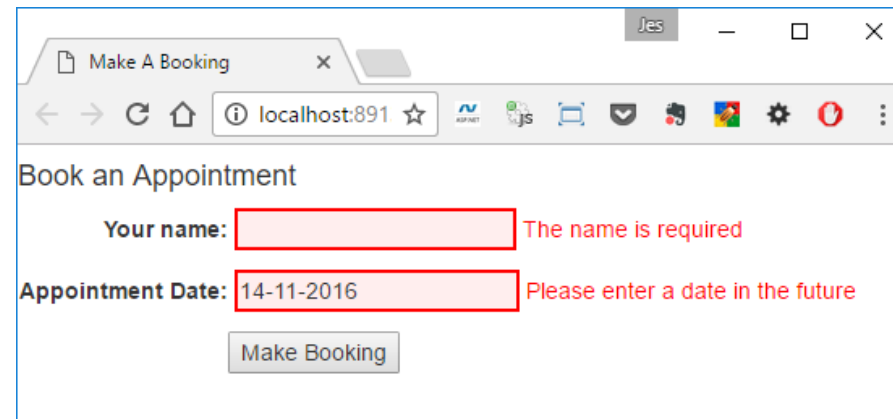
Options for displaying validation errors

1. Model level



A screenshot of a web browser window titled 'Make A Booking'. The address bar shows 'localhost:8913/Home/1'. The page content is titled 'Book an Appointment'. It displays two validation errors in red text: 'Please enter your name' and 'Please enter a date in the future'. Below the errors, there is a text input field for 'Your name:' and a date input field for 'Appointment Date:' containing '14-11-2016'. A 'Make Booking' button is at the bottom.

2. Property level



A screenshot of a web browser window titled 'Make A Booking'. The address bar shows 'localhost:891'. The page content is titled 'Book an Appointment'. It displays two validation errors in red text, each associated with a specific input field: 'The name is required' for the 'Your name:' field and 'Please enter a date in the future' for the 'Appointment Date:' field (which contains '14-11-2016'). A 'Make Booking' button is at the bottom.

Displaying validation messages (model-level)

The screenshot shows a web browser window titled 'Make A Booking' at the URL 'localhost:8913/Home/1'. The page has a heading 'Book an Appointment' and two red bullet points indicating validation errors: 'Please enter your name' and 'Please enter a date in the future'. Below these, there are two input fields: 'Your name:' and 'Appointment Date:'. The 'Your name' field is empty and has a red border. The 'Appointment Date' field contains '11-11-2013' and also has a red border. A 'Make Booking' button is at the bottom.

_Layout.cshtml

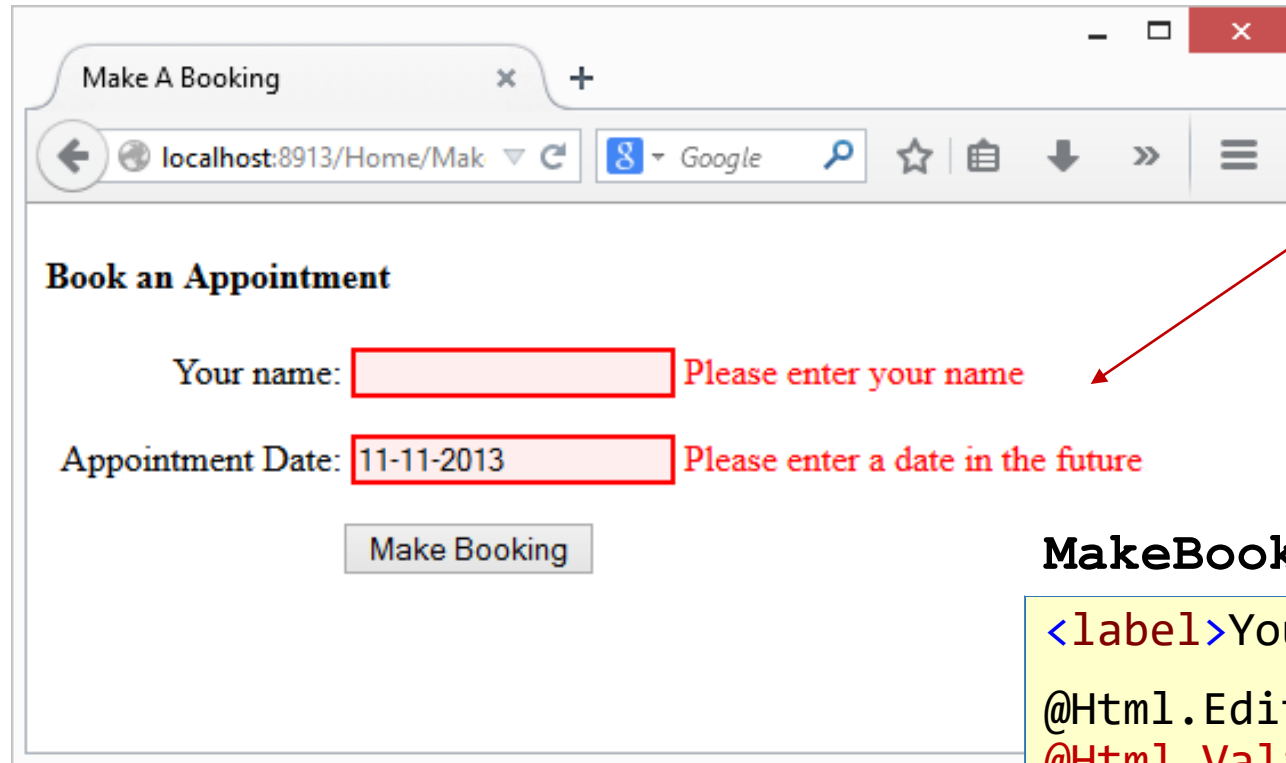
```
.validation-summary-errors {  
    color: #f00;  
    font-weight: bold;  
}
```

```
.input-validation-error {  
    border: 2px solid #f00;  
    background-color: #fee;  
}
```

MakeBooking.cshtml

```
@using (Html.BeginForm()) {  
  
    @Html.ValidationSummary()  
  
    <p><label>Your name:</label> @Html.EditorFor(m => m.FullName)</p>  
    <p><label>Appointment Date:</label> @Html.EditorFor(m => m.Date)</p>  
    <p><label> </label><input type="submit" value="Make Booking" /> </p>  
}
```


Displaying validation messages (property-level)



The screenshot shows a web browser window titled 'Make A Booking' at the URL 'localhost:8913/Home/Mak'. The page has a heading 'Book an Appointment'. Below it, there are two input fields. The first is labeled 'Your name:' and contains a red border and a red message 'Please enter your name'. The second is labeled 'Appointment Date:' and contains the value '11-11-2013' with a red border and a red message 'Please enter a date in the future'. Below these fields is a button labeled 'Make Booking'.

_Layout.cshtml

```
.field-validation-error {  
    color: #f00;  
}
```

MakeBooking.cshtml

```
<label>Your name:</label>  
@Html.EditorFor(m => m.FullName)  
@Html.ValidationMessageFor(m => m.FullName)
```

Property Validation Attributes

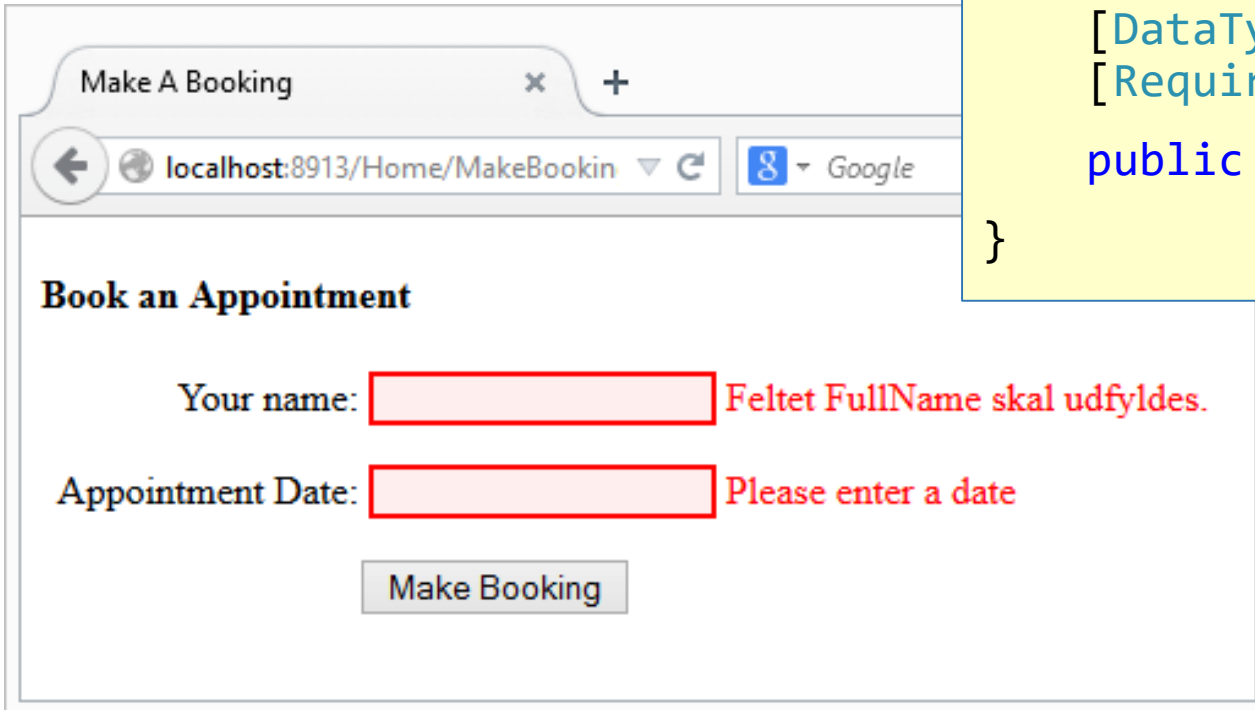
Built-in Validation Attributes

Attribute	Example	Description
Compare	<code>[Compare("MyOtherProperty")]</code>	Two properties must have the same value. This is useful when you ask the user to provide the same information twice, such as an e-mail address or a password.
Range	<code>[Range(10, 20)]</code>	A numeric value (or any property type that implement <code>IComparable</code>) must not lie beyond the specified minimum and maximum values. To specify a boundary on only one side, use a <code>MinValue</code> or <code>MaxValue</code> constant—for example, <code>[Range(int.MinValue, 50)]</code> .
RegularExpression	<code>[RegularExpression("pattern")]</code>	A string value must match the specified regular expression pattern. Note that the pattern has to match the <i>entire</i> user-supplied value, not just a substring within it. By default, it matches case sensitively, but you can make it case insensitive by applying the <code>(?i)</code> modifier—that is, <code>[RegularExpression("(?i)mypattern")]</code> .

Attribute	Example	Description
Required	<code>[Required]</code>	The value must not be empty or be a string consisting only of spaces. If you want to treat whitespace as valid, use <code>[Required(AllowEmptyStrings = true)]</code> .
StringLength	<code>[StringLength(10)]</code>	A string value must not be longer than the specified maximum length. You can also specify a minimum length: <code>[StringLength(10, MinimumLength=2)]</code> .

Specifying Validation Rules with Metadata

```
public class Appointment {  
    [Required]  
    public string FullName { get; set; }  
  
    [DataType(DataType.Date)]  
    [Required(ErrorMessage="Please enter a date")]  
    public DateTime Date { get; set; }  
}
```

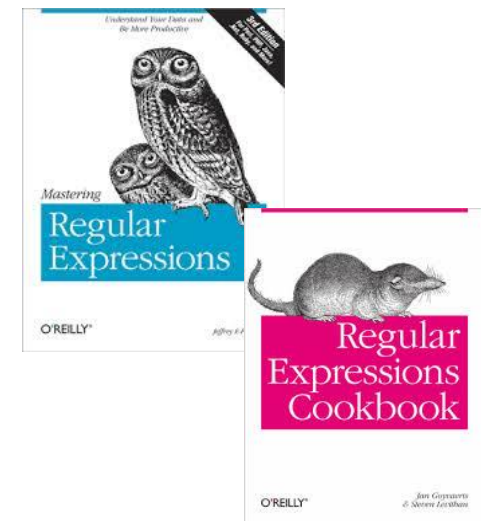


The screenshot shows a web browser window with the title 'Make A Booking'. The address bar displays 'localhost:8913/Home/MakeBookin'. The page content includes a heading 'Book an Appointment' and two input fields. The first field, labeled 'Your name:', is empty and has a red border with the error message 'Feltet FullName skal udfyldes.' to its right. The second field, labeled 'Appointment Date:', is also empty and has a red border with the error message 'Please enter a date' to its right. Below these fields is a 'Make Booking' button.

Advantage of using metadata

- Validation rules are enforced **anywhere** that **the binding process is applied throughout the application**, not just in a single action method.

Regular expressions



What is regular expressions?

- Regular expressions
 - pattern recognition
- Consists of:
 - **Ordinary characters** (**literals**), that represent their face values (a means a and £ stands for £)
 - **Meta characters**, who has a different meaning than their nominal (for example does . stand for any character and **\d** stands for any number)

Special characters in regular expressions

Meta character defined		Exp	Maches
*	The asterisk is used to match 0 or more occurrences of the preceding character.	ab*c	ac, abc, abbc, abbbc, ...
+	1 or more of previous character og expression.	ab+c	abc, abbc, abbbc, ...
()	Logical grouping of part of an expression.	(32)+	32, 3232, 323232, 32323232; But not 32322 or 32323
{n,m}	Explicit quantifier notation	A{2,4}	AA, AAA, AAAA
	Alternation. One of the two characters or sub expressions must match.	A B	A, B
[]	Explicit set of characters to match.	[A-Ca-c]	A, B, C, a, b eller c.
[^]	Matches any single character not in the specified set of characters.	[^A-C]	All characters that are nor A, B or C

More Special characters

Meta character defined		Exp	Maches
.	Matches any character except \n (new line).	.at	hat, cat, rat, bat, fat, vat
\s	Matches any white-space character like space, tab, new line	ab\s c	ab c
\S	Matches any non-white-space character.		
\d	Matches any number		
\D	Matches any character wich are not numbers		
\w	Matches any word character like letter, number and underscore	\w	sand, Cars 2, see_you_soon
\W	Matches any non-word character	\W	£, \$, ;, *, :, ?

Examples

Field		
Email address	<code>\S+@\S+\.\S{2,3}</code>	Check for @ and . and allow only nonwhitespace characters
Password	<code>\w+</code>	Allow only word characters (letters, numbers and underscore)
Password	<code>\w{6,12}</code>	The same as previous but min. 6, max. 12 characters
Password	<code>[a-zA-Z]\w*\d+\w*</code>	Starts with a-z or A-Z contains word characters and one number at least
Phone	<code>\+[0,1][\d\s]{8,14}</code>	Might start with +. 8 -14 numbers and whitespaces are allowed

An example



```
[RegularExpression(@"\S+@(\S+\.)+\w{2,4}",  
ErrorMessage = "There is a problem with the email")]  
[Required(ErrorMessage = "You must enter email")]  
public string Email { get; set; }
```



RegexOne

Learn regular expressions with simple,
interactive examples.

<http://regexone.com>

Exercises

Create a Custom Property Validation Attribute

The **FutureDateAttribute** class

- located in the infrastructure folder

```
public class FutureDateAttribute : ValidationAttribute {  
    public override bool IsValid(object value) {  
        DateTime dt;  
        return (DateTime.TryParse(value.ToString(), out dt) &&  
  
                (DateTime)value > DateTime.Now);  
    }  
}
```

```
public class Appointment {  
    ...  
    [DataType(DataType.Date)]  
    [FutureDate(ErrorMessage="Please enter a date in the future")]  
    public DateTime Date { get; set; }  
}
```


Create a Custom Model Validation Attribute

The Model Example

```
public class CarRent {  
    [Required]  
    public string FullName{ get; set; }  
    [Required]  
    public string Manufacturer { get; set; }  
    [Required]  
    public string Color { get; set; }  
}
```

The Model Validation Attribute

located in the `infrastructure` folder

```
public class ManufacturerColorAttribute : ValidationAttribute {  
    public ManufacturerColorAttribute() {  
        ErrorMessage = "The Manufacturer does not support this color";  
    }  
    public override bool IsValid(object value) {  
        CarRent carRent = value as CarRent;  
        if (carRent == null || string.IsNullOrEmpty(carRent.Manufacturer)  
            || string.IsNullOrEmpty(carRent.Color)) {  
            return true; // I don't have a model the properties I require  
        } else { // Ford only have have blue cars  
            if (carRent.Manufacturer == "Ford") {  
                return carRent.Color == "Blue"; }  
            else {return true; }  
        }  
    }  
}
```

The validation attribute

```
[ManufacturerColor] ←  
public class CarRent {  
    [Required]  
    public string FullName{ get; set; }  
    [Required]  
    public string Manufacturer { get; set; }  
    [Required]  
    public string Color { get; set; }  
}
```

UX Problem ...?

- Model level validation are executed **AFTER** attribute level validation

1. Rent a Car

FullName: Feltet FullName skal udfyldes.
Manufacturer: Ford
Color: Red

Create

```
// exclude property errors
@Html.ValidationSummary(true, "", new { @class = "text-danger" })
```

2. Rent a Car

- The Manufacturer does not support this color**

FullName: Peter Thompson
Manufacturer: Ford
Color: Red

Create

Self validating models

An example

```
public class CarRent2 : IValidatableObject {  
    public string FullName { get; set; }  
    public string Manufacturer { get; set; }  
    public string Color { get; set; }  
  
    public IEnumerable<ValidationResult> Validate(ValidationContext validationContext) {  
        List<ValidationResult> errors = new List<ValidationResult>();  
        if (string.IsNullOrEmpty(FullName)) {  
            errors.Add(new ValidationResult("Please enter your name"));  
        }  
        if (string.IsNullOrEmpty(Manufacturer)) {  
            errors.Add(new ValidationResult("Please enter a manufacturer"));  
        }  
        if (Manufacturer == "Ford" && Color != "Blue") {  
            errors.Add(new ValidationResult("Ford only have blue cars"));  
        }  
        return errors;  
    }  
}
```

Model- and property-level validation are displayed together

Rent a Car

- Please enter your name
- Ford only have blue cars

FullName:

Manufacturer:

Ford

Color:

Red

Create

Client Side Validation

Install JavaScript

```
PM> Install-Package jQuery
```

```
PM> Install-Package jQuery.Validation
```

```
PM> Install-Package Microsoft.jQuery.Unobtrusive.Validation
```

Enable Client-Side validation (enabled by default)

web.config

```
<appSettings>  
  <add key="ClientValidationEnabled" value="true"/>  
  <add key="UnobtrusiveJavaScriptEnabled" value="true"/>  
</appSettings>
```

Reference JavaScript

```
36  <body>
37      @RenderBody()
38
39
40      <script src="/Scripts/jquery-1.8.0.js"></script>
41      <script src="/Scripts/jquery.validate.js"></script>
42      <script src="/Scripts/jquery.validate.unobtrusive.js"></script>
43
44  </body>
45  </html>
```

As Default JavaScript uses Single Property Validation

Unobtrusive JavaScript validation

```
<h4>Book an Appointment</h4>
<form action="/Home/MakeBooking" method="post">
<p><label>Your name:</label>
<input class="text-box single-line"
  data-val="true" <!-- Must be validated-->
  data-val-required="Feltet FullName skal udfyldes." <!-- Rule /
  ErrorMessage -->
  id="FullName" name="FullName"
  type="text" value="" />
...
<p><label> </label><input type="submit" value="Make Booking" /></p>
</form>
```

HTML5 Custom Data Attributes: **data-**

Custom data attributes are intended to store custom data private to the page or application, for which there are no more appropriate attributes or elements.

These attributes are not intended for use by software that is independent of the site that uses the attributes.

Every HTML element may have any number of custom data attributes specified, with any value.

W3C Specification

Enable and Disable Client-Side Validation in MVC

1. Application Level (**web.config**)

```
<appSettings>  
  <add key="ClientValidationEnabled" value="true"/>  
  <add key="UnobtrusiveJavaScriptEnabled" value="true"/>  
</appSettings>
```


2. Application Level (**global.asax**)

```
protected void Application_Start() {  
    HtmlHelper.ClientValidationEnabled = true;  
    HtmlHelper.UnobtrusiveJavaScriptEnabled = true;  
}
```

3. View Level

```
@model MvcApp.Models.Appointment
@{
    ViewBag.Title = "Make A Booking";
    HtmlHelper.ClientValidationEnabled = false;
}
```

4. Field Level

```
<div class="editor-field">  
    @{Html.EnableClientValidation(false);}  
    @Html.TextBoxFor(m => m.PersId) // validation disabled  
    @{ Html.EnableClientValidation(true);}  
</div>
```

Remote Validation

Remote Validation

- Sends a Ajax request to the server (controller / action method) with the method **GET**
- The Action Method must:
 - Have a parameter with the same name as the field to be validated
 - Returns a JsonResult object

A future-date example

```
public class Appointment {  
  
    [Required]  
    public string FullName { get; set; }  
  
    [Remote("ValidateDate", "Home")]  
    public DateTime Date { get; set; }  
  
}
```

Controller & Action Method

```
public JsonResult ValidateDate(string Date) {  
    DateTime parsedDate;  
    if (!DateTime.TryParse(Date, out parsedDate)) {  
        return Json("Please enter a valid date (dd-mm-yyyy)", JsonRequestBehavior.AllowGet);  
    }  
    else if (DateTime.Now > parsedDate) {  
        return Json("Please enter a date in the future", JsonRequestBehavior.AllowGet);  
    }  
    else {  
        return Json(true, JsonRequestBehavior.AllowGet);  
    }  
}
```

Book an Appointment

✓	Metode	Fil	Domæne	Type	Størrelse		Netværk	Headers	Cookies	Parametre	Response	Timings
● 200	GET	MakeBooking	localhost:8913	html	2.28 KB	→ 4 ms		Request-URL: http://localhost:8913/Home/ValidateDate?Date=11-11-2014				
▲ 304	GET	jquery-1.8.0.js	localhost:8913	js	253.74 KB	→ 3 ms		Request-metode: GET				
▲ 304	GET	jquery.validate.js	localhost:8913	js	41.52 KB	→ 4 ms		Statuskode: ● 200 OK				Rediger og send igen
▲ 304	GET	jquery.validate.unobtrusive.js	localhost:8913	js	18.62 KB	→ 11 ms		Filter headers				
▲ 304	GET	globalize.js	localhost:8913	js	46.18 KB	→ 16 ms		Response-headers (0.431 KB)				
▲ 304	GET	globalize.culture.da-DK.js	localhost:8913	js	1.71 KB	→ 11 ms		Cache-Control: "private"				
▲ 304	GET	jquery.validate.globalize.js	localhost:8913	js	1.59 KB	→ 17 ms		Content-Length: "35"				
● 200	GET	ValidateDate?Date=11-11-2014	localhost:8913	json	0.03 KB	→ 5 ms		Content-Type: "application/json; charset=utf-8"				
								Date: "Tue, 11 Nov 2014 13:17:01 GMT"				
								Server: "Microsoft-IIS/8.0"				
								X-AspNet-Version: "4.0.30319"				
								X-AspNetMvc-Version: "5.2"				
								X-Powered-By: "ASP.NET"				
								X-SourceFiles: "=?UTF-8?B?QzpcVXNlcnNcamVhc...XEhvbWVcVmFsaWRhdGVEYXRI?="				
								Request-headers (0.528 KB)				
								Host: "localhost:8913"				
								User-Agent: "Mozilla/5.0 (Windows NT 6.3; WOW64...33.0) Gecko/20100101 Firefox/33.0"				
								Accept: "application/json, text/javascript, */*; q=0.01"				
								Accept-Language: "da,en-us;q=0.7,en;q=0.3"				
								Accept-Encoding: "gzip, deflate"				
								X-Requested-With: "XMLHttpRequest"				
								Referer: "http://localhost:8913/Home/MakeBooking"				
								Cookie: "__RequestVerificationToken=Hblu...mdknWks-K7sUTv1PWW9P-XlvWdCaKE1"				
								Connection: "keep-alive"				

Remember always:

Client & Server side validation

```
public class FutureDateAttribute : ValidationAttribute {  
    public override bool IsValid(object value) {  
        DateTime dt;  
        return (DateTime.TryParse(value.ToString(), out dt) &&  
            (DateTime)value > DateTime.Now);  
    }  
}
```

```
public class Appointment {  
    ...  
    [DataType(DataType.Date)]  
    [Remote("ValidateDate", "Home")] //Client  
    [FutureDate(ErrorMessage="Please enter a date in the future")] //Server  
    public DateTime Date { get; set; }  
}
```

Exercises