

# Model validation

Simon og Emil

# Manuel Validation(Explicit)

- Nødvendigt ved brug af FormCollection

```
[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");
    }
}
```

# Property Validation Attributes

- [Required]
- [StringLength({Maks længde},{[Min længde]})]
- [Range({Start},{Slut})]
- [Compare({Til andet felt})]
- [RegularExpression({Regex streng})]

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

# Custom Property Validation Attributes

```
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; }  
}
```

# Custom Model Validation Attributes

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

```
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; }  
        }  
    }  
}
```

# Regular Expressions

Validering efter mønstre

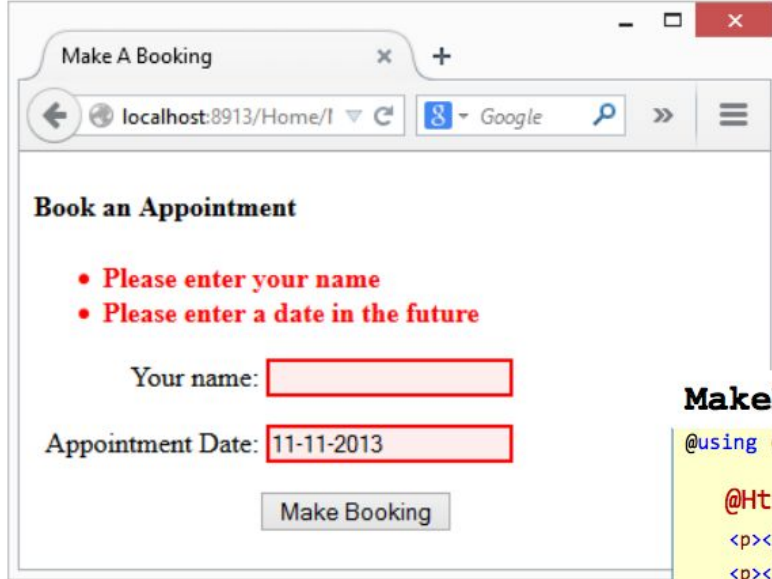
Field		
Email address	<code>\S+@\S+\.\S{2,3}</code>	Check for @ and . and allow only nonwhitespace characters
Password	<code>\w+</code>	Allow only word characters (letters, numbers and underscore)
Password	<code>\w{6,12}</code>	The same as previous but min. 6, max. 12 characters
Password	<code>[a-zA-Z]\w*\d+\w*</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

# Self-Validating Models

Automatisk ved model binding

```
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;  
    }  
}
```

# Displaying Validation Messages - Model level



The screenshot shows a web browser window titled 'Make A Booking'. The address bar shows 'localhost:8913/Home/1'. The page content includes a heading 'Book an Appointment' and two red bullet points: '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 'Appointment Date' field contains the text '11-11-2013'. Both input fields have red borders, indicating validation errors. A 'Make Booking' button is located at the bottom of the form.

## \_Layout.cshtml

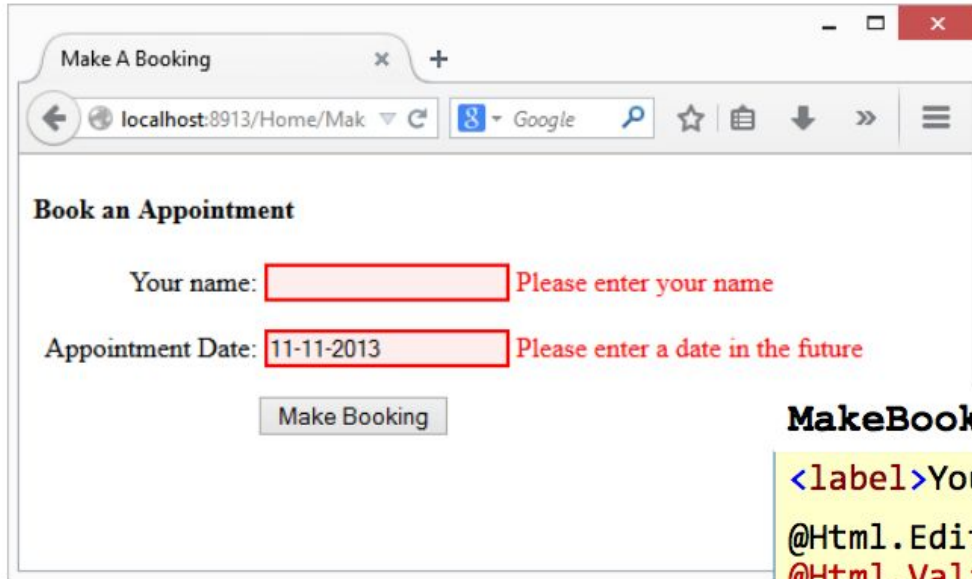
```
.validation-summary-errors {  
    color: #f00;  
    font-weight: bold;  
}
```

## 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 is empty, with a red border and a red validation message 'Please enter your name' to its right. The second is labeled 'Appointment Date:' and contains the text '11-11-2013', also with a red border and a red validation message 'Please enter a date in the future' to its right. At the bottom of the form is a 'Make Booking' button.

## Layout.cshtml

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

## MakeBooking.cshtml

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

# Client-Side Validation

- Javascript
- Kræver setup
- Ikke “rigtig” validering

## Unobtrusive JavaScript validation

- Graceful degradation

# Remote Validation

- Ajax forespørgsel
- Unik data