

Hoofdstuk 8 : Validatie en Authenticatie/authorisatie

Validatie en Authenticatie/authorisatie

1. Inleiding
2. Display annotaties
3. Validatie
4. Authenticatie
 - Forms
 - OAuth en OpenId
5. Authorisatie
6. Encryptie

1. Inleiding

- ▶ We breiden de Bierhalle applicatie verder uit
 - Display annotaties
 - Validatie
 - Authenticatie en autorisatie
 - Authenticatie : Gebruiker moet aanmelden en wordt al dan niet geauthenticeerd.
 - Autorisatie : éénmaal de gebruiker geauthenticeerd is, krijgt de gebruiker bepaalde rechten in de applicatie afhankelijk van zijn rol.

2. Display/Edit Annotaties

- ▶ De presentatie in UI aanpassen dmv annotaties => AOP



- ▶ AOP

Overzicht

[\[bewerken\]](#)

Aspectoriëntatie is bedoeld als antwoord op een probleem waar alle "klassieke" paradigma's mee kampen, namelijk de vraag hoe om te gaan met de zogeheten *crosscutting concerns*: handelingen die door het hele programma heen uitgevoerd moeten worden. Typische voorbeelden hiervan zijn logging en beveiliging: op vrijwel ieder punt in een gemiddeld programma bestaat de behoefte aan de mogelijkheid om zaken naar een log weg te schrijven om fouten te traceren en zeer veel programma's moeten op verschillende punten controleren of de gebruiker van het programma wel gerechtigd is om de opgevraagde handeling uit te voeren. De "klassieke" paradigma's hebben hiervoor geen makkelijke oplossing en vallen daarom terug op het meerdere malen opnemen van precies dezelfde code op iedere plaats waar dezelfde handeling uitgevoerd wordt. Met als resultaat dat een verandering aan de uitvoering van een dergelijk crosscutting concern betekent dat door het hele programma heen code moet worden aangepast.

Als oplossing hiervoor biedt AOD (Aspect Oriented Development) de mogelijkheid een stukje code te schrijven dat op een groot aantal, door de programmeur te definiëren punten, ingevoegd wordt. Dit invoegen gebeurt zonder dat de geschreven code waarin ingevoegd wordt, aangepast wordt op de invoeging (dat er op een gegeven plaats iets ingevoegd wordt, kan men dus niet zien aan de programmacode). Om dit voor elkaar te krijgen, geeft de programmeur extern aan de programmacode een aantal punten op waarin bepaalde code ingevoegd dient te worden. Een apart programma neemt de code van de programmeur en zijn aanwijzingen over invoegen en stelt daarmee een nieuw programma samen waarin de juiste code op de juiste plaats ingevoegd is.

Er zijn in principe twee tijdstippen waarop het *weven* (het samenvoegen van code) kan plaatsvinden: tijdens de *compilatie* van code naar programma (dit wordt *statisch weven* genoemd) en tijdens het uitvoeren van het gecompileerde programma (*dynamisch weven*). In het eerste geval leeft de AOD-uitbreiding op de bestaande techniek in de *compiler*, in het tweede geval wordt de uitbreiding extern aangebracht via een *preprocessor*.

2. Display/Edit Annotations

- Namespace :
System.ComponentModel.DataAnnotations

Data Annotations Attribute	Effect
[DisplayColumn]	Determines which child property <code>Html.DisplayText()</code> and <code>Html.DisplayTextFor()</code> should use to generate a simple string representation of this item. Maps to <code>ModelMetadata</code> 's <code>SimpleDisplayText</code> property.
[UIHint]	Affects template selection when rendering displays or editors for this item. Maps to <code>ModelMetadata</code> 's <code>TemplateHint</code> property. Note that if a property has multiple <code>[UIHint]</code> attributes, the MVC Framework will give priority to one declared as <code>[UIHint(templateName, PresentationLayer="MVC")]</code> .
[DataType]	Affects template selection and how the built-in HTML helpers format the model value as text. Maps to <code>ModelMetadata</code> 's <code>DataTypeName</code> , <code>DisplayFormatString</code> , and <code>EditFormatString</code> properties. For example, <code>[DataType(DataType.Date)]</code> sets <code>DataTypeName</code> to "Date" and both format strings to {0:d}.
[ReadOnly]	Maps to <code>ModelMetadata</code> 's <code>IsReadOnly</code> property, although this doesn't affect any built-in templated view helper. ASP.NET MVC's <code>DefaultModelBinder</code> will notice a <code>[ReadOnly]</code> attribute (but not the <code>IsReadOnly</code> metadata property!) and will respond to this by not binding any new values for the associated property.

Ho

2. Display/Edit Annotations

[DisplayFormat]	Affects how the built-in HTML helpers represent the model value as text. For example, the format string {0:c} causes numerical values to be rendered as currency values. Maps to <code>ModelMetadata</code> 's <code>DisplayFormatString</code> and <code>EditFormatString</code> properties.
[ScaffoldColumn]	Controls whether the built-in Object templates should show this property. Maps to <code>ModelMetadata</code> 's <code>ShowForDisplay</code> and <code>ShowForEdit</code> properties.
[DisplayName]	Affects all built-in helpers that render property labels, including <code>Html.Label()</code> , the built-in Object template, and helpers that display validation messages. Maps to <code>ModelMetadata</code> 's <code>DisplayName</code> property.

- Uit de System.Web.Mvc namespace
 - [HiddenInput] => type="hidden"

2. Display/Edit Annotaties

```
public int BrouwerId { get; set; }
[Display(Name = "Naam brouwer")]
public string Naam {
    [DataType(DataType.MultilineText)]
    public string Beschrijving { get; set; }
    [Display(Name = "Email contact")]
    [DataType(DataType.EmailAddress)]
    public string ContactEmail { get; set; }
    [DataType(DataType.Date)]
    [Display(Name = "Datum oprichting")]
    [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
    public DateTime? OprichtingsDatum { get; set; }
    public string Straat { get; set; } // mag null zijn
    public virtual Gemeente Gemeente { get; set; } // mag null zijn
    [DisplayFormat(DataFormatString = "{0:F} euro", NullDisplayText = "omzet niet gekend")]
    public int? Omzet { get; set; }
```

Detail

Naam brouwer
Bavik
Straat
Rijksweg 33
Gemeente
8531 Bavikhove
Omzet
20000000,00 euro
Beschrijving
Naast de traditionele Pils, heeft Brouwerij Bavik nog meer in petto...
Er is het gamma tafelbieren Bavik dinner Beer, met zijn twee
Email contact
info@bavik.be
Datum oprichting
1950-01-01
AantalBieren
6
[Edit](#) | [Back to List](#)

Brouwers

[Creëer brouwer](#)

Naam brouwer	Straat	Gemeente	Omzet	AantalBieren	
Bavik	Rijksweg 33	8531 Bavikhove	20000000,00 euro	6	Detail Editeer Verwijder
De Graal			omzet niet gekend	0	Detail Editeer Verwijder
De Leeuw			omzet niet gekend	0	Detail Editeer Verwijder

Pag. 7

2. Display/Edit Annot

```
public int BrouwerId { get; set; }
[Display(Name = "Naam brouwer")]
public string Naam {
    [DataType(DataType.MultilineText)]
    public string Beschrijving { get; set; }
    [Display(Name = "Email contact")]
    [DataType(DataType.EmailAddress)]
    public string ContactEmail { get; set; }
    [DataType(DataType.Date)]
    [Display(Name = "Datum oprichting")]
    [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
    public DateTime? OprichtingsDatum { get; set; }
    public string Straat { get; set; } // mag null zijn
    public virtual Gemeente Gemeente { get; set; } // mag null zijn
    [DisplayFormat(DataFormatString = "{0:F} euro", NullDisplayText = "omzet niet gekend")]
    public int? Omzet { get; set; }
```

Editeer brouwer

Naam brouwer
Bavik
Straat
Rijksweg 33
Gemeente
Bavikhove
Omzet
20000000
Beschrijving
Naast de traditionele Pils, heeft Brouwerij Bavik nog meer in petto...
Er is het gamma tafelbieren Bavik dinner Beer, met zijn twee
Email contact
info@bavik.be
Datum oprichting
01/01/1950
[Bewaren](#)
[Back to List](#)


HoGent

Pag. 8

2. Display/Edit Annotaties

- ▶ Opmerking
 - [Display] : kan naast de displayname ook de volgorde van properties in UI bepalen
 - [Display(Name="First Name" , Order = 15000)]
 - [DisplayFormat]
 - DataFormatString : formatting
 - NullDisplayText : Tekst die getoond moet worden igv null
 - ApplyFormatInEditMode : default false
 - [DataType]
 - Bepaalt input type die wordt toegepast
 - DataType.Password => input type="password"
 - Voor DataType.Date geldt : in order to correctly display the date in Chrome,... , the value must be formatted as 2012-09-28. A valid full-date as defined in [RFC 3339], with the additional qualification that the year component is four or more digits representing a number greater than 0.

3. Validatie

- ▶ Data validatie kan op verschillende plaatsen gebeuren
 - Domein
 - GUI 
 - Database
- ▶ De validatie regels zijn dezelfde → duplicatie van code
- ▶ Microsoft oplossing : validatieregels via annotatie aan het model toevoegen
 - Het framework controleert de opgegeven regels zonder dat er code voor geschreven moet worden
 - Hoef je dan ook niet te testen. DataAnnotations = AOP style en werken enkel als volledig systeem functioneert. De defaultModelBinder zal deze aanroepen.
- ▶ We kijken hoe we resp. Model, Controller, View hiervoor moeten aanpassen

3. Validatie

► Client-side Validatie

- De validatie gebeurt tot nu toe aan de server side
- In web applicaties is het aangewezen de validatie op de client te doen, zodat een **round-trip naar de server** niet nodig is
- MVC zorgt voor de nodig JavaScript code voor de DataAnnotations
- DataAnnotations worden zowel aan de client als aan de server side gevalideerd (ook door EF Code first)
 - Blijft werken als gebruiker javascript heeft disabled in de browser

3. Validatie

► Het (View)Model

- Namespace : System.ComponentModel.DataAnnotations bevat annotaties voor een klasse in Models folder

Attribute	Meaning
[Range]	A numeric value (or any property type that implement IComparable) must not lie beyond the specified minimum and maximum values. To specify a boundary only on one side, use a MinValue or MaxValue constant—for example, [Range(int.MinValue, 50)].
[RegularExpression]	A string value must match the specified regular expression pattern. Note that your 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 (?i) modifier—that is, [RegularExpression("(?i)mypattern")].
[Required]	The value must not be empty or be a string consisting only of spaces. If you want to treat whitespace as valid, use [Required(AllowEmptyStrings = true)].
[StringLength]	A string value must not be longer than the specified maximum length. In .NET 4, you can also specify a minimum length.

3. Validatie

► Annotaties in Brouwer

- Foutmelding
 - [Required] : toont de defaultMessage : The naam field is required.
 - [Required(ErrorMessage="Naam is verplicht")] : toont deze foutmelding
 - [Required(ErrorMessage="{0} is verplicht")] : {0} wordt vervangen door de DisplayName van de property.
 - Ook mogelijk om multi-language te werken (zie laatste hoofdstuk)
 - [Required(ErrorMessageResourceName="Brouwer_Naam_Required" ErrorMessageResourceType=typeof(BrouwerErrorMessage))
 - BrouwerErrorMessage = de naam van de resource file
 - Brouwer_Naam_Required = naam van de fout in de resource file

3. Validatie

► Annotaties in Brouwer : opmerkingen

- StringLength : je kan ook minimum vermelden
 - [StringLength(160,MinimumLength=10)]
- Range
 - Range werkt enkel voor int en double (inclusief grenzen). Maar kan je ook laten werken op andere types
 - [Range(typeof(decimal),"0.00", "49.99"]
 - [Range(typeof(DateTime), "1/1/2012", "12/31/9999"]

3. Validatie

► Annotaties in Brouwer

```
public int BrouwerId { get; set; }
[Display(Name = "Naam brouwer")]
[Required(ErrorMessage = "{0} is verplicht")]
[StringLength(50, ErrorMessage = "{0} is te lang.")]
public string Naam {
    [DataType(DataType.MultilineText)]
    public string Beschrijving { get; set; }
    [Display(Name = "Email contact")]
    [DataType(DataType.EmailAddress)]
    [RegularExpression(@"[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}", ErrorMessage = "Email is niet correct.")]
    public string ContactEmail { get; set; }
    [DataType(DataType.Date)]
    [Display(Name = "Datum oprichting")]
    [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
    public DateTime? OprichtingsDatum { get; set; }
    public string Straat { get; set; } // mag null zijn
    public virtual Gemeente Gemeente { get; set; } // mag null zijn
    [Range(0, Int32.MaxValue, ErrorMessage = "Omzet mag niet negatief zijn")]
    [DisplayFormat(DataFormatString = "{0:F} euro", NullDisplayText = "omzet niet gekend")]
    public int? Omzet { get; set; }
}
```

Pag. 15

3. Validatie

► 2 extra annotaties in System.Web.Mvc namespace

- [Compare]
 - 2 properties die dezelfde waarde moeten hebben.
 - Bvb property voor cornfirmatieWachtwoord
 - [Compare("Password")]
- [Remote]
 - Uitvoeren van client side validatie met een server callback
 - Voorbeeld : controleren van Username via CheckUserName in AccountController

```
[Remote("CheckUsername", "Account")]
public string Username { get; set; }
```

```
public JsonResult CheckUsername(string username)
{
    var result = Membership.FindUsersByName(username).Count==0;
    return Json(result, JsonRequestBehavior.AllowGet);
}
```


3. Validatie

► De Controller

- Uitvoeren van validatie logica gebeurt bij **model binding**
- Controllers beschikken over een **ModelState** property (type `ModelStateDictionary`).
 - Is een *bijproduct* van de model binding.
 - Adhv ModelState krijg je meer informatie over de model binding
 - Is al dan niet gelukt => `IsValid` property van de ModelState
 - En welke properties er problemen geven, en per property de bijhorende foutmelding. ModelState is dictionair. Key = naam property, value is de foutmelding.



HoGer

Pag. 17

3. Validatie

► De Controller

- Wie kan de ModelState aanpassen?
 1. **Model binders** = de ValueProviders die de Request gegevens naar parameters van action methode omzetten
 - Zij voeren de setters uit. Wordt er een exception gethrowed dan wordt een entry toegevoegd voor deze property aan de ModelState, waarde = exception message.
 - Zij controleren de DataAnnotations. Wordt er niet aan voldoen wordt een entry toegevoegd voor property, value is de error message.
 - Ook alle geposte waarden wordt opgeslaan
 - Stel brouwernaam is verplicht en niet ingevuld
 - `ModelState.IsValid = false;`
 - `ModelState.IsValidField("Naam") = false`
 - `ModelState["Naam"].Errors.Count > 0`
 - `string naamErrorMess = ModelState["Naam"].Errors[0].ErrorMessage;`

HoGent

Pag. 18

3. Validatie

► De Controller

- Wie kan de ModelState aanpassen?
 2. **UpdateModel()** : past eveneens de ModelState aan en throwt exception als het updaten van het model fouten geeft
 3. **TryUpdateModel()** = UpdateModel() maar retourneert boolean die aangeeft of er zich fouten hebben voorgedaan
 4. **De Controller zelf** kan ook fouten toevoegen

```
if (.....)
    ModelState.AddModelError("propertynaam", "boodschap");
```

- Als propertynaam = lege string => fout op model level! **Kan je ook gebruiken voor foutafhandeling in Controller (ipv TempData)**

3. Validatie

► De Controller

- De bijhorende test : terug default view renderen met brouwer

```
[TestMethod]
public void CreateWhenBrouwerContainsErrors()
{
    Brouwer newBrouwer = new Brouwer("Jan"){Omzet=-100};
    controller.ModelState.AddModelError("any key", "any error");
    mockBrouwerRepository.Setup(m => m.Add(newBrouwer));
    ViewResult result = controller.Create(newBrouwer, "3000") as ViewResult;
    Assert.IsInstanceOfType(result.Model, typeof(Brouwer));
    Assert.AreEqual(newBrouwer, result.Model);
}
```

3. Validatie

► De Controller

- Create : Igv fouten niet redirecten naar de Index, maar toon de Create View opnieuw met de foutmeldingen!

```
[HttpPost]
public ActionResult Create(Brouwer brouwer, string postcode)
{
    if (ModelState.IsValid)
    {
        brouwerRepository.Add(brouwer);
        brouwer.Gemeente = (String.IsNullOrEmpty(postcode) ? null : gemeenteRepository.FindBy(postcode));
        brouwerRepository.SaveChanges();
        return RedirectToAction("Index");
    }
    else
    {
        ViewBag.Postcode =
            new SelectList(gemeenteRepository.FindAll().OrderBy(g => g.Naam), "Postcode", "Naam", postcode);
        return View(brouwer);
    }
}
```

Als de validatie niet lukt, toon dan de Create view opnieuw

Pag. 21

3. Validatie

► De View

- Bevat Html Helpers voor weergeven van fouten in ModelState
- Run de applicatie : Maak eens een aantal fouten en klik op Save

The screenshot shows a web form titled "Brouwer" with the following fields and errors:

- Naam brouwer**: A text input field with a red error message "Naam brouwer is verplicht" (Name brouwer is required).
- Beschrijving**: A text area.
- Straat**: A text input field.
- Gemeente**: A dropdown menu with the text "--Selecteer gemeente--".
- Email contact**: A text input field containing "fvdsgfdsagé" with a red error message "Email is niet correct." (Email is not correct).
- Datum oprichting**: A text input field.
- Omzet**: A text input field containing "-200" with a red error message "Omzet mag niet negatief zijn" (Revenue may not be negative).

3. Validatie

► De View

- De View bevat Validation HTML Helper methodes

```
@using (Html.BeginForm()) {  
    @Html.ValidationSummary(true)  
    <fieldset>  
        <legend>Brouwer</legend>  
  
        <div class="editor-label">  
            @Html.LabelFor(model => model.Naam)  
        </div>  
        <div class="editor-field">  
            @Html.EditorFor(model => model.Naam)  
            @Html.ValidationMessageFor(model => model.Naam)  
        </div>  
    </fieldset>  
}
```

3. Validatie

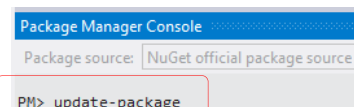
- Client-side validation en gebruik van unobtrusive javascript wordt ingesteld in de web.config.

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

- Client side validatie gebeurt hierdoor met jQuery validate plugin
- HTML5 validatie nog niet in alle browsers ingebouwd, gebruik jQuery validate

- **GEBRUIK MINSTENS jquery.1.9 anders problemen met validatie email**

- TOOLS > Library Package Manager > Package Manager Console.
- Geef update-package in : upgrade bibliotheken naar laatste versie



3. Validatie

- ▶ Als je een Edit/Create view creëert, vink “Reference script libraries” aan

☒ Reference script libraries

- Voegt toe aan de View :

```
@section Scripts {  
    @Scripts.Render("~/bundles/jqueryval")  
}
```



```
bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(  
    "~/Scripts/jquery.unobtrusive*",  
    "~/Scripts/jquery.validate*"));
```

3. Validatie

- ▶ De View
 - Bekijk de gegenereerde HTML code. Paginabron weergeven
 - Bevat veel attributen in elke form control startend met **data-**. D.i. **unobtrusive javascript** in action. Deze attributen zijn een feature in HTML5, maar volledig backward compatible met alle modern browsers (including IE6).

```
<div class="editor-label">  
    <label for="Naam">Naam browser</label>  
</div>  
<div class="editor-field">  
    <input class="text-box single-line" data-val="true" data-val-length="Naam browser is te lang." data-val-length-max="100" data-val-required="Naam browser is verplicht" value="" type="text" />  
    <span class="field-validation-valid" data-valmsg-for="Naam" data-valmsg-replace="true"></span>  
</div>
```

- 2 libraries worden hiervoor gebruikt

```
<script src="/Scripts/jquery.validate.min.js" type="text/javascript"></script>  
<script src="/Scripts/jquery.validate.unobtrusive.min.js" type="text/javascript"></script>
```

The jQuery Validation library: jquery.Validate.js. Meer uitleg op <http://docs.jquery.com/Plugins/Validation>

De MVC wrapper rond de jQuery Validation library : jquery.validate.unobtrusive.js library

3. Validatie

- ▶ De View
 - De attributen

<code>data-val</code>	indicates that the field contains validation data and should be processed by the unobtrusive adapters script.
<code>data-val-{validator name}</code>	e.g. <code>data-val-length</code> - contains the error message for the validator.
<code>data-val-{validator name}-{argument name}</code>	e.g. <code>data-val-length-min</code> - zero or more arguments necessary for performing validation.

The data- prefix is defined by the HTML5 standard. The specification states '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'. In the past, people tended to use hacks such as embedding data in CSS classes to add such data, but thankfully this is no longer necessary.

3. Validatie

- ▶ De View
 - De attributen

```
[Range(0, Int32.MaxValue, ErrorMessage = "Omzet mag niet negatief zijn")]  
public int? Omzet { get; set; }
```

```
<input data-val="true"  
  data-val-number="The field Omzet must be a number."  
  data-val-range="Omzet mag niet negatief zijn"  
  data-val-range-max="2147483647"  
  data-val-range-min="0"  
  id="Omzet" name="Omzet" type="number" value="" />
```

De annotaties worden
vertaald naar data-val-xxx
attributen

3. Validatie

- ▶ De View
 - De ValidationMessageFor

```
@Html.ValidationMessageFor(model => model.Omzet)
```



```
<span class="field-validation-valid" data-valmsg-for="Omzet" data-valmsg-replace="true"></span>
```

- Data-valmsg-for = name van input veld waarvoor fout geldt
- Data-valmsg-replace = true als boodschap daar moet getoond worden

3. Validatie

- ▶ De View
 - De weergave van de fouten kan je aanpassen in de css

```
/* Styles for validation helpers .....*/  
  
.field-validation-error {  
    color: #ff0000;  
}  
  
.field-validation-valid {  
    display: none;  
}  
  
.input-validation-error {  
    border: 1px solid #ff0000;  
    background-color: #ffebee;  
}  
  
.validation-summary-errors {  
    font-weight: bold;  
    color: #ff0000;  
}  
  
.validation-summary-valid {  
    display: none;  
}
```

3. Validatie

► Html.ValidationMessageFor

- Laat ook een 2^{de} parameter toe, die de foutmelding in de ModelState overschrijft

```
@Html.ValidationMessageFor(model => model.Naam, "!=")
```

► Html.ValidationSummary

- Rendert alle fouten in de ModelState als een lijst

```
@Html.ValidationSummary(true)
```

- 2^{de} overload : 1 par van type bool : Exclude property errors. De fouten die op property niveau getoond worden worden niet hier getoond
- 3^{de} overload : summary error message (string)

3. Validatie

► Pas de view als volgt aan en run opnieuw

```
@using (Html.BeginForm()) {  
    @Html.ValidationSummary("Gelieve onderstaande fouten te verbeteren :")  
    <fieldset>  
        <legend>Brouwer</legend>  
  
        <div class="editor-label">  
            @Html.LabelFor(model => model.Naam)  
        </div>  
        <div class="editor-field">  
            @Html.EditorFor(model => model.Naam)  
            @Html.ValidationMessageFor(model => model.Naam, "!=")  
        </div>  
    </fieldset>  
}
```

Please correct the following errors :

- ♦ Naam brouwer is verplicht

Brouwer

Naam brouwer

3. Validatie

► Client-side Validatie

- Als een knop de client side validatie niet mag triggeren
- Voeg de knop toe

```
<input id="submitBooking" type="submit" value="Place booking" />
```

- Voeg een javascript toe voor de knop

```
<script type="text/javascript">  
    document.getElementById("submitBooking").disableValidation = true;  
</script>
```

3. Validatie

► Foutmeldingen

- Je kan ook zelf via de code fouten toevoegen in de Controller
 - ModelState.AddModelError("any key", "any error");
 - Op property-niveau, dan als key de naam van de property meegeven
 - Op model niveau, dan als fout "" meegeven
- Je kan ook met TempData werken. Voorbeeld delete

```
[HttpPost, ActionName("Delete")]  
public ActionResult DeleteConfirmed(int id)  
{  
    Brouwer brouwer = brouwerRepository.FindBy(id);  
    try  
    {  
        brouwerRepository.Delete(brouwer);  
        brouwerRepository.SaveChanges();  
        return RedirectToAction("Index");  
    }  
    catch (Exception ex)  
    {  
        TempData["message"] = "Verwijderen brouwer mislukt : " + ex.Message;  
        // ModelState.AddModelError("", "Verwijderen brouwer mislukt : " + ex.Message);  
        return View(brouwer);  
    }  
}
```

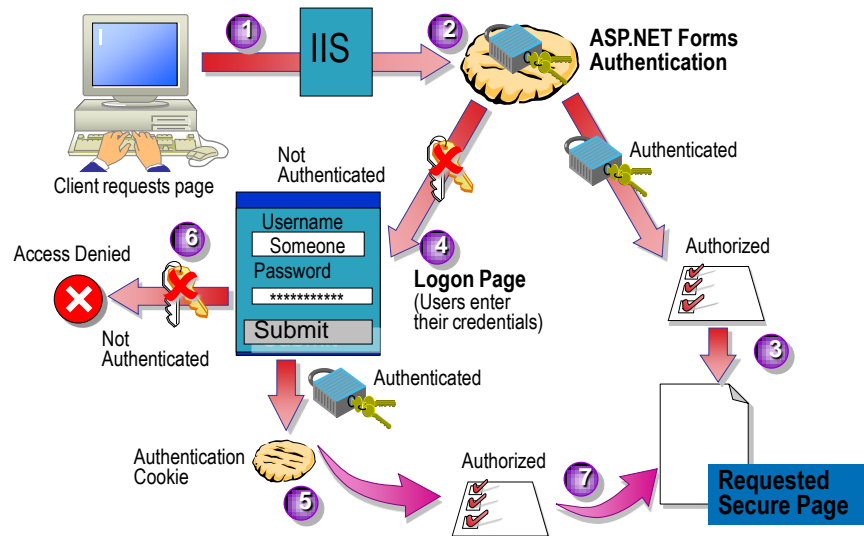
3. Validate

- ▶ Oefening
 - Pas de Edit methode aan in BrouwerController
 - Maak ook een bijhorende unit test aan

4. Authenticatie

- ▶ Beveiliging?
 - Authenticatie (verificatie) : wie is de gebruiker en heeft hij/zij toegang tot de applicatie? Gebruiker geeft identificatie in en deze wordt gevalideerd. Mogelijkheden binnen ASP.NET :
 - Forms : op formulieren gebaseerde verificatie met cookieondersteuning. Gebruiker heeft identiteit in, in aanlog formulier, en na verificatie wordt een geëncrypteerd authenticatie cookie gegenereerd. Niet geverifieerde verzoeken worden automatisch omgeleid naar een aanlog formulier.
 - OAuth en OpenID : externe login via 3th party (Facebook,...)
 - Windows authenticatie : voor Intranet. Single Sign on via Active Directory
 - Authorisatie : O.b.v. de identificatiegegevens van gebruiker wordt nagegaan wat een gebruiker mag doen binnen de applicatie. Kan worden toegekend per gebruiker of per rol

4. Authenticatie : Forms



HoGent

Pag. 37

4. Authenticatie : Forms

► Werkwijze

1. Stel Form based authentication mode in in de Web.config van hoofdmap.
2. Stel de autorisatie in via Authorize attribuut in de Controller
3. Creëer een XML file, database,... met de namen en paswoorden van de gebruikers
4. Maak login en logout pagina met code om gebruikers te authenticeren. Maak eventueel registratie pagina om nieuwe gebruikers toe te voegen. Maak eventueel een wachtwoord aanpassen en wachtwoord vergeten pagina
5. Opvragen identificatie gegevens gebruiker

Reeds
voorzien

HoGent

Pag. 38

4. Authenticatie : Forms

► 1. Web.config in de root

```
<authentication mode="Forms">
  <forms loginUrl="~/Account/Login" timeout="2880" />
</authentication>
```

- Mode = Forms (indien aanloggen via Forms authenticatie). Kan ook Windows zijn (kies dan voor een MVC4 intranet applicatie) of None (geen authenticatie)
- loginUrl = de actie waarnaar wordt omgeleid indien de gebruiker nog niet is aangemeld en aanloggen is vereist.
- Is voorzien als je Internet MVC4 applicatie aanmaakt

4. Authenticatie : Forms

► 2. Controller

- Maak gebruik van de filter [Authorize]
 - Een filter plukt extra functionaliteit in de request handling pipeline
 - Controleert of gebruiker is aangemeld alvorens actie wordt uitgevoerd. Indien niet wordt de gebruiker omgeleid naar /Account/Login (zie web.config)
 - Voert OnAuthorization methode uit in IAuthorizationFilter interface. Dit bevat volgend stukje code

```
IPrincipal user = HttpContext.User;
If (!user.Identity.IsAuthenticated) return false;
```

4. Authenticatie : Forms

► 2. Controller

- Maak gebruik van de filter [Authorize]
 - Indien authenticatie faalt => HttpUnauthorizedResult (genereert een HTTP 401). Dit wordt opgevangen door FormsAuthenticationModule OnLeave methode die dan redirect naar Account/Login (zie web.config). Redirect url bevat ook de return Url, zodat na aanloggen terug deze actie kan worden uitgevoerd
- Plaats [Authorize] boven een actie of controller (geldt dan voor alle acties in Controller, anders enkel voor de actie)

```
[Authorize]  
public class BrouwerController : Controller
```

4. Authenticatie : Forms

► 3. Beheer gebruikers : ASP.NET Membership

- Reeds voorzien in MVC4 Internet applicatie
 - AccountController/Views
 - Voorziet in login, registreer, logout, manage account
 - Gebruikt hiervoor WebMatrix.Security
 - Is een wrapper rond SimpleMembershipProvider
 - SimpleMembershipProvider : Provides support for website membership tasks, such as creating accounts, deleting accounts, managing passwords.
 - Erft van de abstracte klasse System.Web.Security.MembershipProvider
 - Werkt met SQL Server : maakt tabellen aan voor opslag usernames/pwd
 - => Bij runnen wordt een aparte LocalDb database gecreëerd (mdf file in App_Data folder), zie web.config defaultConnection
- We gaan dit verder customiseren!

4. Authenticatie : Forms

► 3. Beheer gebruikers : ASP.NET Membership

- Beter : verder customizeren
 - Filters folder > InitializeSimpleMembershipProvider.cs
 - Kopieer onderstaande uit deze file naar global.asax, Application_Start: dit stelt connectie met db in en creëert de nodige tabellen
 - WebSecurity.InitializeDatabaseConnection("DefaultConnection", "UserProfile", "UserId", "UserName", autoCreateTables: true);
 - Par 1 : Connectionstring uit web.config
 - Par 2 : Naam tabel met users
 - Par 3 : Naam Kolom met userId
 - Par 4 : Naam Kolom met username
 - Verwijder de filter folder

4. Authenticatie : Forms

► 3. Beheer gebruikers : ASP.NET Membership

- Global.asax – Application_Start event

```
Database.SetInitializer<BierhalleContext>(new BierhalleInitializer());  
new BierhalleContext().Brouwers.ToList();  
WebSecurity.InitializeDatabaseConnection("DefaultConnection", "UserProfile", "UserId", "UserName", autoCreateTables: true);
```

- Brouwers.ToList()=> forceert uitvoering seeding methode
- Pas web.config aan "DefaultConnection"

```
<connectionStrings>  
<add name="DefaultConnection" connectionString="Data Source=localhost\\sqlexpress;Initial Catalog=Bierhalle;Integrated Security=True"  
providerName="System.Data.SqlClient" />  
</connectionStrings>
```

- Opvragen connectieString : View> Server Explorer > connect to database : servername : localhost\\sqlexpress, database "Bierhalle". 1 maal toegevoegd, rechtsklik de connectie > Properties. Connectionstring bevat de string

4. Authenticatie : Forms

► 3. Beheer gebruikers : ASP.NET Membership

- Models > AccountModel
 - Verwijder UserContext => anders wordt een aparte database aangemaakt. We gaan gebruik maken van BierhalleContext.
 - UserProfile tabel moet nu gecreëerd worden in de Bierhalle database
 - Maak er een aparte klasse UserProfile van in Models > Domain, verplaats de code
 - Plaats StringLength attribute boven UserName
 - Indien gewenst kan je extra properties toevoegen

```
namespace Bierhalle.Models.Domain
{
    [Table("UserProfile")]
    public class UserProfile
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int UserId { get; set; }
        [StringLength(50)]
        public string UserName { get; set; }
    }
}
```

4. Authenticatie : Forms

► 3. Beheer gebruikers : ASP.NET Membership

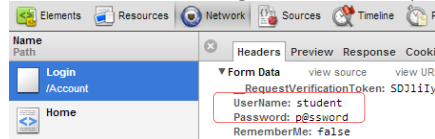
- DAL > BierhalleContext
 - Voeg DbSet<UserProfile> toe
- AccountController :
 - Verwijder using Bierhalle.Filters
 - Delete [InitializeSimpleMembership], gebeurt nu in de global.asax
 - Vervang UsersContext door BierhalleContext. Beter nog is om hiervoor ook Repositories aan te maken en de code in die zin aan te passen (doen we voorlopig niet, zie de completed applicatie)

```
public DbSet<UserProfile> UserProfiles { get; set; }
```

4. Authenticatie : Forms

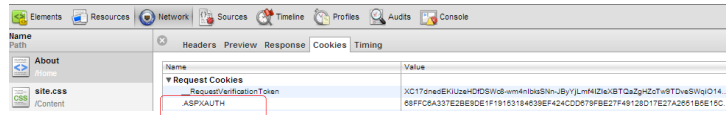
3. Beheer gebruikers : ASP.NET Membership

- Run, ga naar brouwers
 - Je komt automatisch op login scherm. Url bevat in querystring returnUrl, zodat na inloggen geredirect wordt naar de brouwers pagina
 - Registreer gebruiker, bekijk request in Browser Developer tools > Network
 - username en wachtwoord als tekst doorgestuurd in de http Header



• **Gebruik HTTPS en SSL!!!**

- Vanaf nu wordt bij elke request een cookie .ASPXAUTH meegestuurd



HoGent

Pag. 47

4. Authenticatie : Forms

3. Beheer gebruikers : ASP.NET Membership

- Run, ga naar brouwers
 - Eénmaal aangemeld, kan je je account beheren door te klikken op je username bovenaan.

Hello, docent!

Manage Account.

You're logged in as **docent**.

Change password

Current password

New password

Confirm new password

Change password

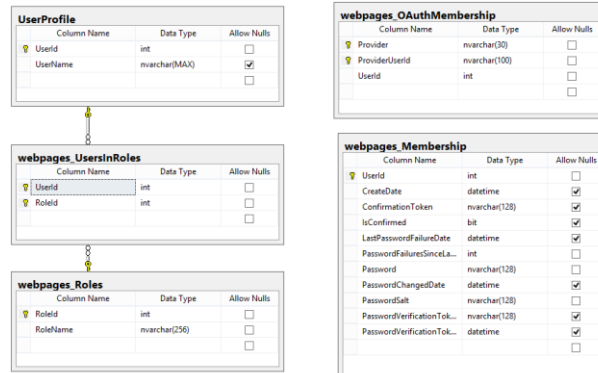
HoGent

Pag. 48

4. Authenticatie : Forms

► 3. Beheer gebruikers : ASP.NET Membership/Roles

- Bierhalle database : bevat nu extra tabellen
 - Webpages_ : tabellen gebruikt door WebSecurity
 - Webpages_Membership : bevat 1 record/userprofile : bevat o.a. geëncrypteerd wachtwoord (hashed)



HoGent

Pag. 49

4. Authenticatie : Forms

► 3. Beheer gebruikers : ASP.NET Membership

- Seeding de database (accounts en rollen)
 - Hiervoor werken we met de ASP.NET Membership en Roles Provider. De implementatie die we gebruiken is de SimpleMembershipProvider/SimpleRoleProvider. Meer info op [http://msdn.microsoft.com/en-us/library/webmatrix.webdata.simplemembershipprovider\(v=vs.111\).aspx](http://msdn.microsoft.com/en-us/library/webmatrix.webdata.simplemembershipprovider(v=vs.111).aspx) en [http://msdn.microsoft.com/en-us/library/webmatrix.webdata.simpleroleprovider\(v=vs.111\).aspx](http://msdn.microsoft.com/en-us/library/webmatrix.webdata.simpleroleprovider(v=vs.111).aspx)
 - Voeg een private methode SeedMembership toe in de Initializer en roep deze aan vanuit de Seed methode zodat ook de tabellen voor beheer users/rollen gecreëerd worden
 - Opm : daar we hier werken met DropCreateDatabaseAlways
 - In global.asax : verwijder WebSecurity.InitializeDatabaseConnection want deze methode mag maar 1 keer in de applicatie worden uitgevoerd.
 - Voeg deze code toe aan de nieuwe methode SeedMembership

HoGent

Pag. 50

4. Authenticatie : Forms

- ▶ 3. Beheer gebruikers : ASP.NET Membership and Roles
 - Seeding de database (accounts en rollen) (een beetje afhankelijk van je initialisatie strategy)

```
private void SeedMembership()  
{  
    WebSecurity.InitializeDatabaseConnection("DefaultConnection", "UserProfile", "UserId", "UserName",  
        autoCreateTables: true);  
    var roles = (SimpleRoleProvider) Roles.Provider;  
    var membership = (SimpleMembershipProvider) Membership.Provider;  
  
    roles.CreateRole("admin");  
    roles.CreateRole("student");  
    membership.CreateUserAndAccount("docent", "p@ssword");  
    membership.CreateUserAndAccount("student", "p@ssword");  
    roles.AddUsersToRoles(new string[] { "docent" }, new string[] { "admin" });  
    roles.AddUsersToRoles(new string[] { "student" }, new string[] { "student" });  
}
```

HoGent

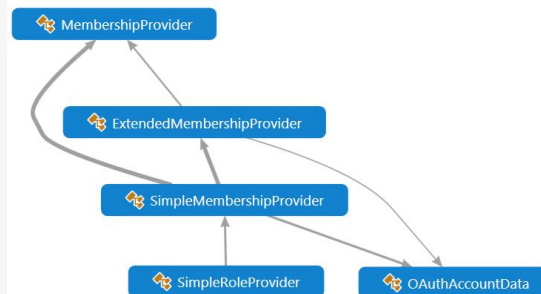
Pag. 51

public abstract class **ExtendedMembershipProvider** : [System.Web.Security.MembershipProvider](#)
Member of [WebMatrix.WebData](#)

Summary:

Represents an abstract class that is used to extend the membership system that is provided by the [System.Web.Security.MembershipProvider](#) class.

```
▶ {} Microsoft.Internal.Web.Utils  
▲ {} WebMatrix.WebData  
    ▲ ExtendedMembershipProvider  
        ◊ ConfirmAccount(string)  
        ◊ ConfirmAccount(string, string)  
        ◊ CreateAccount(string, string, bool)  
        ◊ CreateAccount(string, string)  
        ◊ CreateOrUpdateOAuthAccount(string, string, string)  
        ◊ CreateUserAndAccount(string, string, bool, System.Collections.Generic.IDictionary<string,object>)  
        ◊ CreateUserAndAccount(string, string, System.Collections.Generic.IDictionary<string,object>)  
        ◊ CreateUserAndAccount(string, string, bool)  
        ◊ CreateUserAndAccount(string, string)  
        ◊ DeleteAccount(string)  
        ◊ DeleteOAuthAccount(string, string)  
        ◊ DeleteOAuthToken(string)  
        ◊ ExtendedMembershipProvider()  
        ◊ GeneratePasswordResetToken(string, int)  
        ◊ GeneratePasswordResetToken(string)  
        ◊ GetAccountsForUser(string)  
        ◊ GetCreateDate(string)  
        ◊ GetLastPasswordFailureDate(string)  
        ◊ GetOAuthTokenSecret(string)  
        ◊ GetPasswordChangedDate(string)  
        ◊ GetPasswordFailuresSinceLastSuccess(string)  
        ◊ GetUserIdFromOAuth(string, string)  
        ◊ GetUserIdFromPasswordResetToken(string)  
        ◊ GetUserNameFromId(int)  
        ◊ HasLocalAccount(int)  
        ◊ IsConfirmed(string)  
        ◊ ReplaceOAuthRequestTokenWithAccessToken(string, string, string)  
        ◊ ResetPasswordWithToken(string, string)  
        ◊ StoreOAuthRequestToken(string, string)
```



Pag. 52

4. Authenticatie : Forms

- ▶ 4. Login, logout, registreer pagina
 - Bekijk de code in de AccountController/Views
 - WebSecurity is wrapper round ASP.NET Membership functionaliteit
 - CRUD accounts, encryptie,...
 - Login(username, password, persistCookie): aanmelden van gebruiker en aanmaken van een cookie (.ASPXAUTH (al dan niet persistent))
 - CreateUserAndAccount(userName, password) : creatie user

4. Authenticatie

- ▶ 5. Opvragen identificatie gegevens van gebruiker
 - Opvragen gegevens van de huidige gebruiker
 - In Controller actie methode via User.Identity property van de HttpContext.
 - IsAuthenticated = true als de gebruiker is aangemeld, anders false
 - Name : username van aangemelde gebruiker
 - Of in view (zie bvb _LoginPartial.cshtml)

```
@if (Request.IsAuthenticated) {  
    Hello, @Html.ActionLink(User.Identity.Name, "Manage", "Account",  
    routeValues: null, htmlAttributes: new { @class = "username", title = "Manage" })!  
}
```

 - Klik op de naam brengt je naar het beheer van je account
 - _LoginPartial.cshtml maakt ook gebruik van AntiForgeryToken. Zoek dit eens op.

4. Authenticatie

▶ 6. Mailen gegevens

- Indien gewenst kan je bij registratie ook mail versturen
- Configureer de SMTP server in global.asax

```
WebMail.SmtpServer = "mailserver.example.com";  
WebMail.EnableSsl = true;  
WebMail.UserName = "username@example.com";  
WebMail.Password = "your-password";  
WebMail.From = "your-name-here@example.com";
```

- Pas Register methode aan

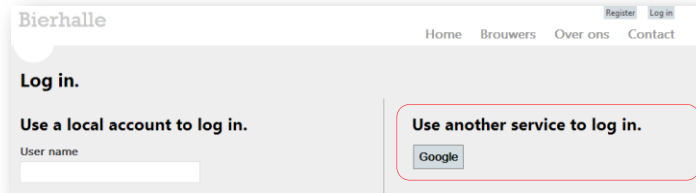
```
WebMail.Send(  
    to: model.Email,  
    subject: "username created",  
    body: "blabla"
```

4. Authenticatie : OAuth en OpenId

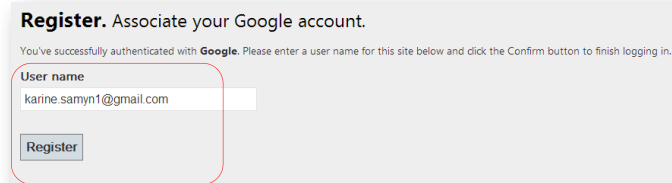
- ▶ Authenticatie via externe sites (Facebook, Twitter,...)
- ▶ Meer op
 - OpenId : <http://openid.net/>
 - OAuth : <http://oauth.net/>
 - DotNetOpenAuth (open source gebruikt in .Net) :
<http://www.dotnetopenauth.net/>
- ▶ Configuratie
 - Registreer eerst je site op Facebook,.... => je krijgt een key en een secret. Meer op
<http://go.microsoft.com/fwlink/?LinkID=252166>
 - Voor Google is dit niet nodig
 - App_Start folder > AuthConfig.cs. Plaats vereiste code in RegisterAuth methode uit commentaar. Hier bvb het google gedeelte

4. Authenticatie : OAuth en OpenId

- Run de applicatie
 - Nu heb je de keuze.

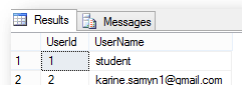


- Kies Google. Geef gmail account in en druk Register



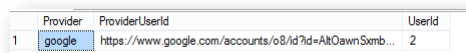
4. Authenticatie : OAuth en OpenId

- De informatie wordt bijgehouden in de membership tables.
 - UserProfiles table



	UserId	UserName
1	1	student
2	2	karine.samyn1@gmail.com

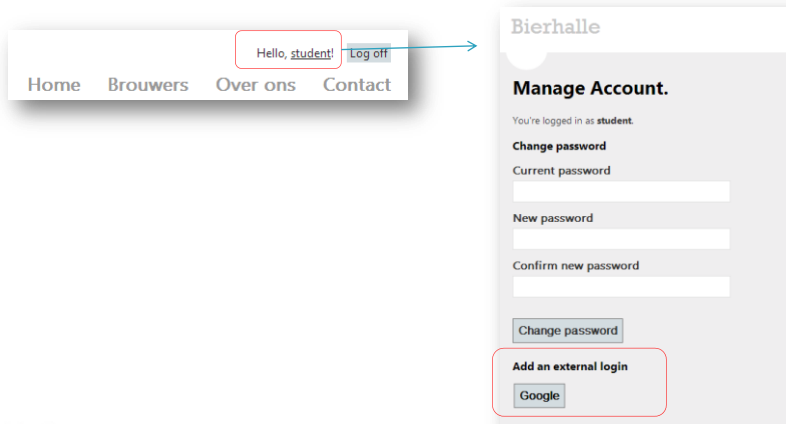
- Webpage_OAuthMembership table



	Provider	ProviderUserId	UserId
1	google	https://www.google.com/accounts/o8/id?id=AltOawnSymb...	2

4. Authenticatie : OAuth en OpenId

- Als je inlogt met een local account, en je klikt bovenaan op Hello, xxxxx dan kan je je user profile aanpassen en linken met een external login.



5. Authorisatie

- Eénmaal aangemeld, krijgt de gebruiker bepaalde machtigingen binnen de applicatie afhankelijk van zijn role
 - Zie seeding voor creatie roles
 - Boven Controller of actie methode :
[Authorize(Roles="admin")]
 - Of in view
 - @if(User.IsInRole("admin")) {
 - @Html.ActionLink("Create",...) }

Appendix Encryptie

► C# bevat bibliotheken voor encryptie

```
/// <summary>
/// Encrypts a string using the SHA512 (Secure Hash Algorithm) algorithm.
/// This works in the same manner as MD5, providing 512bit encryption.
/// </summary>
/// <param name="password">A string containing the data to encrypt.</param>
/// <returns>A string containing the string, encrypted with the SHA512 algorithm.</returns>
public static string Encrypt(string password)
{
    System.Security.Cryptography.SHA256Managed sha = new System.Security.Cryptography.SHA256Managed();
    byte[] hash = sha.ComputeHash(Encoding.ASCII.GetBytes(password));
    StringBuilder stringBuilder = new StringBuilder();
    foreach (byte b in hash)
    {
        stringBuilder.AppendFormat("{0:x2}", b);
    }
    return stringBuilder.ToString();
}
```