Hoofdstuk 3: Web forms

ASP.NET Web Forms

De <form> tag bakent op een webpagina een zone af die dient voor gebruikers input. Een formulier kan tekstvelden, checkboxen, radio-buttons en andere formulier elementen bevatten. Formulieren worden gebruikt om gebruikers-data naar een specifieke URL te sturen.

```
<form action="fomulier.aspx" method="get">
Voornaam:<input type="text" name="vnaam" value="Willy" /><br />
Achternaam:<input type="text" name="anaam" value="Wonka" /><br />
<input type="submit" value="Submit" />
</form>
```

Het action-attribuut geeft aan naar welke webpagina de gegevens worden doorgegeven. De methode kan zowel GET als POST zijn.

Als de GET methode wordt gebruikt dan zullen alle waarden die werden ingegeven in het formulier doorgegeven met behulp van de “querystring”. Dit is naam/waarde informatie die in de URL is opgenomen. Deze informatie is dan ook duidelijk te lezen in het adres van de web pagina, na de file naam.

http://www.google.be/search?q=ASP&ie=UTF-8&oe=UTF-8&hl=nl&btnG=Google+zoeken&meta=

Het blauwe gedeelte is de gewone naam van de file die wordt gezocht, alle rode tekst zijn telkens informatieparen die worden meegegeven met de querystring. Deze gegevens kunnen ook letterlijk achter de URL worden ingetypt zonder gebruik te maken van een formulier. De GET-methode is dus niet geschikt om gevoelige informatie te versturen. In praktijk zal je de GET-methode dan ook minder vaak gebruiken voor het versturen van formulieren.

De POST methode zorgt ervoor dat de ingegeven waarden worden doorgegeven aan het “form” object van de opgeroepen pagina. De naam/waarde informatie die wordt verzonden is opgenomen in de HTTP request body. Deze informatie is niet zichtbaar en kan enkel met een formulier worden gegenereerd. De POST-methode is in ASP.NET de standaardmethode. Deze methode zal automatisch toegepast worden, ook als je het action-attribuut niet hebt gespecificeerd.

In de ontvangende pagina kan men de ingegeven waarden verwerken door ze uit de respectievelijke querystring (GET) of form collectie (POST) te halen.

Onderstaand voorbeeld haalt alle parameters uit de QueryString:

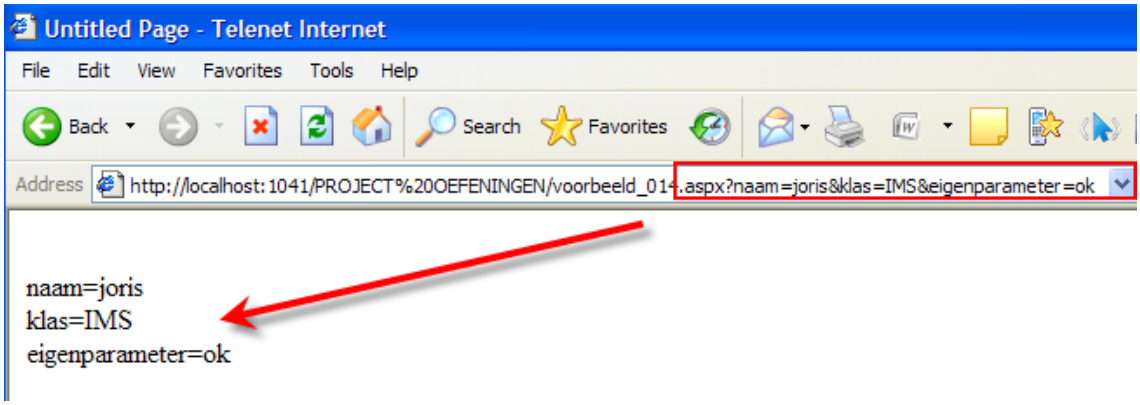
```
Dim tekst As String = ""
For Each key In Request.QueryString
        tekst = tekst & "<br/>" & key & "=" & Request.QueryString(key)
Next

Response.Write(tekst)
```

REFERENTIE: VOORBEELD 014.ASPX

Stel dat we een forumlier dat via de GET-methode werkt volgende querystring terugstuurt:

voorbeeld_014.aspx?naam=joris&klas=IMS&eigenparameter=ok

Dan heeft bovenstaand code-fragment volgende output als resultaat:



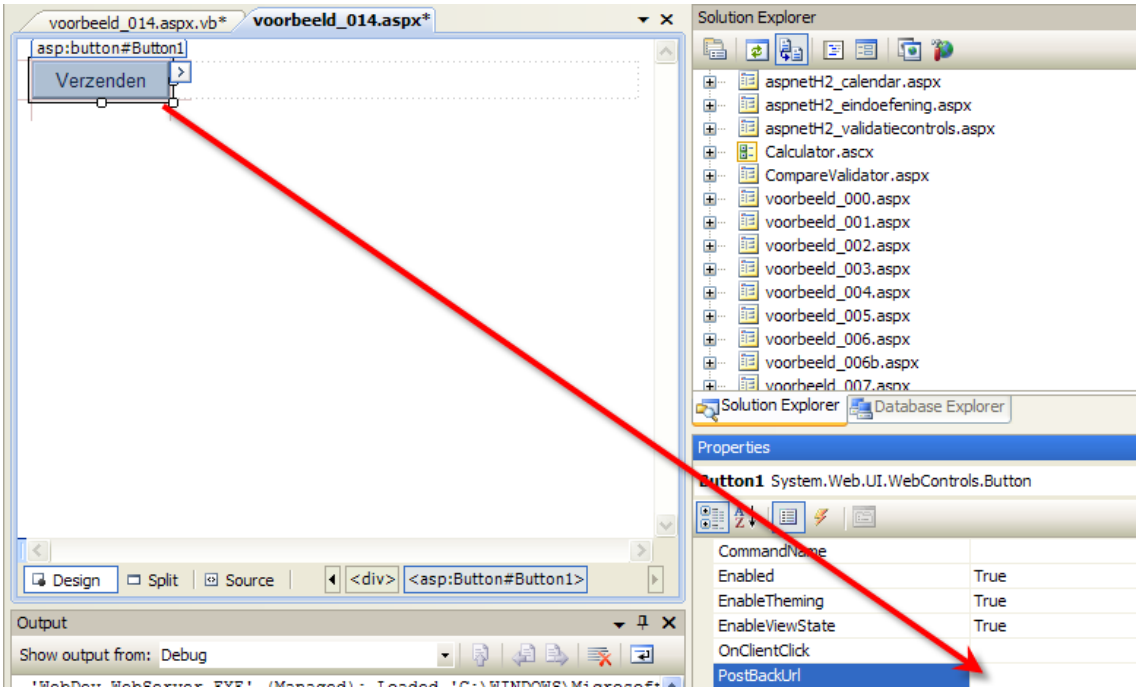
Bij ASP.NET introduceerde Microsoft het postback principe waarmee validatie en verwerking van gegevens uit formulieren een pak gemakkelijker gaat.

Door aan de <form> tag het runat="server" attribuut toe te voegen zal deze op de server verwerkt worden. Indien u server controls met server scripts wenst te wijzigen, dan MOETEN deze in een runat="server"-formulier worden opgenomen.

```
<form runat="server">
... server controls
</form>
```

Het formulier <form runat="server"> wordt altijd met een postback **naar zichzelf verstuurd**. Als er een action attribuut aanwezig is dan wordt dit volledig genegeerd. De name en id attributen worden door ASP.NET aangevuld indien deze niet werden meegegeven. Een .aspx pagina kan maar één <form runat="server"> formulier bevatten.

Het terugsturen van een postback naar zichzelf heeft het voordeel dat je alle code om het formulier te verwerken op dezelfde pagina kan schrijven, uiteraard in de code-behind file. Er kunnen echter uitzonderlijke situaties zijn wanneer je het formulier toch naar een andere pagina dan zichzelf wil verzenden via de POST-methode. Je kan dan de property PostBackUrl wijzigen van de control - vaak een button - die de submit van het formulier zal teweeg brengen.



Meer informatie hierover kan je vinden op http://msdn.microsoft.com/en-us/library/ms178140.aspx

Viewstate

In klassieke webapplicaties wordt alle formulierdata gewist nadat het formulier werd teruggestuurd. Om ervoor te zorgen dat de velden ingevuld bleven, om eventuele fouten uit de gegevens te halen na validatie, was heel wat programmeerwerk nodig.

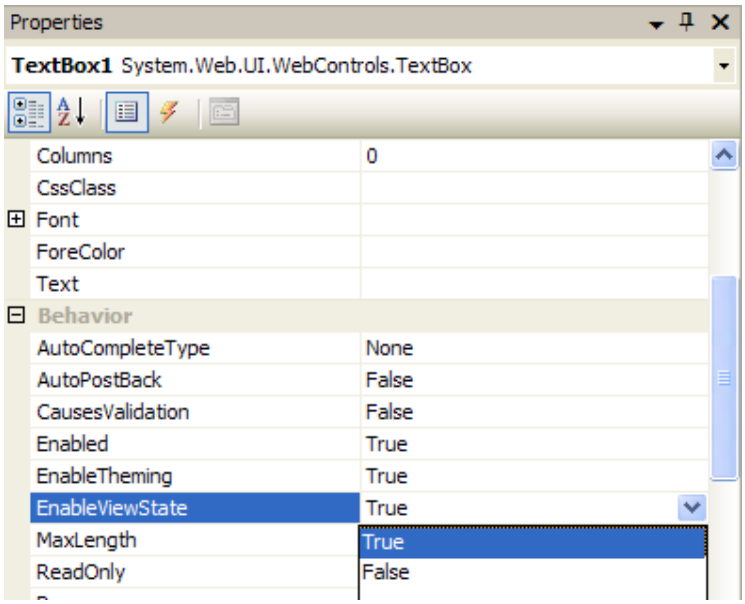
In ASP.NET worden waarden in formulieren bijgehouden in de VIEWSTATE zodanig dat men steeds kan terugkeren en de waarden eventueel kan aanpassen. De ViewState bewaart met andere woorden de status van het formulier op het moment dat dit wordt teruggestuurd naar de server. Dit gebeurt met behulp van een verborgen veld (__viewstate) dat in elk formulier wordt geplaatst met <form runat="server">. Omdat de ViewState gewoon in de pagina staat, blijft deze bruikbaar zolang de gebruiker de pagina open heeft in de webbrowser.

```
<input type="hidden" name="__VIEWSTATE"
value="dDwtNTI0ODU5MDE1Ozs+ZBCF2ryjMpeVgUrY2eTj79HNl4Q=" />
```

Indien u de ViewState van de volledige pagina niet wilt bijhouden (bijv. op een inlogscherm) dan kan u bovenaan de pagina volgende statement toevoegen

```
<%@ Page EnableViewState="false" %>
```

U kunt ook individuele controls (bijv. paswoord veld) uit de ViewState onttrekken door het attribuut EnableViewState="false" toe te voegen aan de bewuste control.



De ViewState houdt enkel de status bij van één pagina. Hiermee kunt u dus geen gegevens doorgeven tussen verschillende pagina's of eventueel gegevens bewaren voor andere gebruikers van de webapplicatie.

Applicatievariabelen, sessievariabelen en cookies

Naast de ViewState (status van één pagina) en formulieren (doorgeven gegevens tussen 2 pagina's) kan men ook gegevens bewaren voor een langere periode. Dit kan door middel van: applicatievariabelen, sessievariabelen en cookies.

Voorbeelden van situaties waarin het bijhouden van gegevens over meerdere pagina's en voor langere termijn nuttig kan zijn, zijn bijvoorbeeld de volgende:

- Onthouden of iemand is ingelogd en de ingelogde gebruikersnaam onthouden
- Een winkelwagentje opvullen en onthouden wat een gebruiker erin heeft gestoken
- Recente zoekopdrachten bewaren
- …

1. **Applicatievariabelen**: worden opgeslagen op de server en blijven zolang bewaard als dat de applicatie draait. Elke virtuele directory op de server wordt in ASP.NET beschouwd als 1 applicatie (mits nodige instellingen IIS). Een applicatie start op het moment dat de eerste bezoeker een .aspx pagina opent die zich in één van de folders van die applicatie bevindt. Met een applicatie variabele kunnen we bijvoorbeeld een teller maken die telt hoe vaak een pagina bezocht is, of men kan er de databankconnectie in opslaan die we door de hele applicatie nodig hebben.

```
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.Load
        Dim tekst As String = ""
        Application.Lock()
        Application("teller") = Application("teller") + 1
```

```
        tekst = "deze pagina werd reeds " & Application("teller") & " keer geopend
zinds de start van de applicatie"
        Application.UnLock()
        lblTeller.Text = tekst
    End Sub
```

REFERENTIE: VOORBEELD_015.ASPX

Aandachtspunt 1: applicatievariabelen bestaan zolang de applicatie draait. Vanaf dat we de webserver zouden herstarten, worden alle applicatievariabelen leeggemaakt.

Aandachtspunt 2: in het codevoorbeeld hierboven kan u opmerken dat we de methode lock() gebruiken. Wanneer u in uw webapplicatie de waarde van een applicatievariabele wil aanpassen, is het aangeraden om de applicatie eerst “op slot” te doen. Als u dat niet zou doen, dan kunnen meerdere pagina’s tegelijkertijd proberen dezelfde variabele aan te passen. Hierdoor kan het zijn dat bepaalde gegevens verloren gaan. Door de methode lock() te gebruiken reserveert u het object Application volledig voor uzelf. Het spreekt voor zich dat dit reserveren van een zo kort mogelijke duur moet zijn. Ander ontneemt u andere pagina’s of personen de kans de applicatievariabelen te gaan aanpassen. Zorg dus dat intensieve bewerkingen in een tijdelijke variabele gebeuren, alvorens u de application locked, om het locken zo snel mogelijk te laten gebeuren.

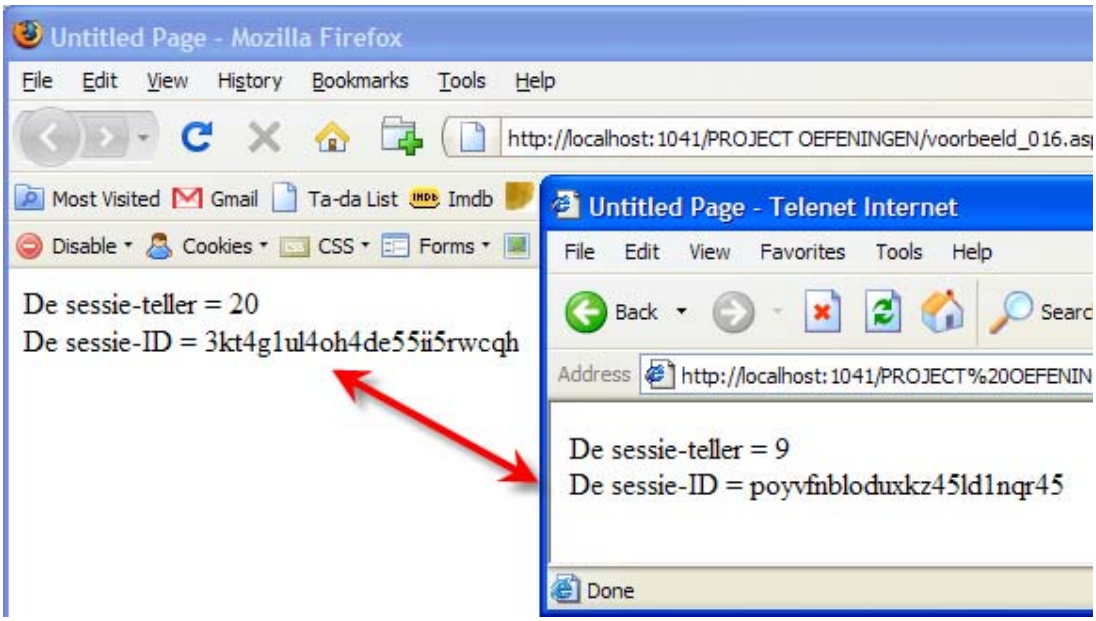
Wenst u applicatievariabelen te verwijderen dan kan dat met de methode *Remove*.

RemoveAll verwijdert alle applicatievariabelen. *Clear* heeft hetzelfde effect als *RemoveAll*.

2. **Sessie variabelen**: worden opgeslagen op de server. Een sessie start als een gebruiker voor het eerst een pagina van een bepaalde applicatie opvraagt. Een sessie stopt als de gebruiker voor een bepaalde tijd geen pagina’s behorende tot die applicatie meer heeft opgevraagd. De lengte van die periode waarin een sessie behouden blijft zonder activiteit wordt bepaald door het Session.Timeout attribuut. Zolang een sessie bestaat kunnen er in het Sessie-object verschillende variabelen worden opgeslagen.

   Als de sessie stopt zijn ook alle sessievariabelen verloren. Elke gebruiker van een webapplicatie heeft zijn eigen sessie, waardoor sessievariabelen individueel zijn. Om bij te houden welke sessie van een bepaalde gebruiker is, wordt er een cookie naar de browser gestuurd met een unieke waarde erin: de SessionID. Browsers die geen cookies accepteren, kunnen dus ook niet met een sessie werken. Binnen een sessie is het handig om gegevens te bewaren die relevant zijn voor de gebruiker. Bijvoorbeeld een gebruikersnaam en paswoord, zodat we weten of de gebruiker toegang heeft tot bepaalde gegevens.

   Zoals u in onderstaande schermafdruk al kan zien, zullen verschillende browsers op dezelfde client toch een aparte sessie toegewezen krijgen.




```
 Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load

Dim tekst As String = ""
Session("teller") = Session("teller") + 1
tekst = "De sessie-teller = " & Session("teller") & "<br />" & vbCrLf
tekst &= "De sessie-ID = " & Session.SessionID & vbCrLf
lblSessie.Text = tekst

End Sub
```

REFERENTIE: VOORBEELD_016.ASPX

3. **Cookies**: worden opgeslagen op de client. Er zijn persistente en niet-persistente (tijdelijke) cookies. Persistente cookies krijgen een vervaldatum mee met het Expires-attribuut. Een cookie wordt op de server aangemaakt en naar de client meegezonden met de eerst volgende response. Daar wordt deze bewaard en mee teruggezonden naar de server met de volgende request. Enkel die cookies die afkomstig zijn van de applicatie waar de nieuwe request naar gestuurd werd, worden mee teruggestuurd naar die applicatie.

   Wanneer u gebruik maakt van cookies dient u te beseffen dat niet alle (maar wel de meeste) gebruikers cookies toestaan in de browser. Om status te kunnen gaan bijhouden door middel van cookies, kan u gebruik maken van de *HttpCookie* klasse.

   U kan een nieuwe cookie aanmaken door eenvoudigweg een nieuwe instantie van de klasse HttpCookie aan te maken. Onderstaand codevoorbeeld toont u hoe:

```
Partial Class voorbeeld_017
    Inherits System.Web.UI.Page

    Dim myCookie As HttpCookie

    Protected Sub btnCheck_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnCheck.Click
        myCookie = Request.Cookies("mijncookie")
        If myCookie Is Nothing Then
            lblCookieStatus.Text = "Cookie bestaat nog niet, maar is nu aangemaakt"
            myCookie = New HttpCookie("mijncookie")
            myCookie.Expires = DateTime.Now.AddDays(10)
            Response.Cookies.Add(myCookie) ' cookie bewaren
        Else
            lblCookieStatus.Text = "Cookie bestaat reeds"
        End If

    End Sub

    Protected Sub btnRemoveCookie_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnRemoveCookie.Click
        myCookie = Request.Cookies("mijncookie")
        If Not myCookie Is Nothing Then
            myCookie.Expires = DateTime.Now.AddDays(-1) ' vervaldatum in verleden
            Response.Cookies.Add(myCookie) ' cookie bewaren
            lblCookieStatus.Text = "Cookie is weg"
        End If
    End Sub
End Class
```

REFERENTIE: VOORBEELD_017.ASPX

Voorbeeld: Status bijhouden van de laatst gezochte films

In onze voorbeeldapplicatie (Openmoviesets) kan het handig zijn om de laatst gezochte filmtitels bij te houden en deze te presenteren wanneer een gebruiker opnieuw op de website komt. Op die manier dient de gebruiker niet opnieuw de juiste filmtitel in te typen of na te denken over die ene toffe filmsetlocatie die hij of zij bij het vorig bezoek had gevonden.

We kiezen ervoor om de 5 meest recente gezochte filmtitels van een gebruiker bij te houden en te tonen op de voorpagina. We zouden kunnen werken met sessies, applicatievariabelen of zelfs onze databank. Het eenvoudigste en nuttigste in dit geval lijkt ons om te werken met cookies. Het gaat namelijk niet om gevoelige data die we gedurende enige tijd ter beschikking willen houden.

Een sessie vervalt mogelijk te snel, een applicatievariabele wordt niet onthouden bij een herstart van de server/applicatie en om met de databank te werken dienen we een extra tabel aan te maken en te onderhouden. Cookies zijn een eenvoudigere manier om toch deze termen te bewaren zolang we wensen. Uiteraard kan een gebruiker cookies wissen van zijn of haar computer. Ook kan het zijn dat een gebruiker niet graag heeft dat iemand ziet waar hij de laatste keer op gezocht heeft in onze applicatie.

Dit zijn zaken die je op voorhand moet afwegen of waar je de keuze aan de gebruiker overlaat. Je kunt een checkbox toevoegen aan de applicatie waarmee je aangeeft of een gebruiker al dan niet wil dat zijn zoektermen opgeslagen en getoond worden. Voor de eenvoud van deze les gaan we ervan uit dat een gebruiker er niet mee inzit dat zijn laatst gezochte termen getoond worden en zullen we ons enkel focussen op het aanmaken en uitlezen van de cookie.

Een cookie plaatsen

Er is meer dan één manier om een cookie te plaatsen.
Een zeer snelle manier is de volgende:

```
Response.Cookies("KiesEenNaam") = "waarde in de cookie"
```

Het is ook mogelijk om net zoals bij een array meerdere waarden in één cookie weg te schrijven.

```
Response.Cookies("LoginCookie")( "login") = "jhens"
Response.Cookies("LoginCookie ")( "logindatum") = "7 januari 2009"
```

In plaats van de cookie op deze manier rechtstreeks weg te schrijven kan je ook een Cookie object aanmaken, opvullen en pas nadien wegschrijven:

```
Dim CookieSearch As HttpCookie = New HttpCookie("CookieSearchTerms")
CookieSearch(0) = "joris"
' set expiration date manually to one month
Response.Cookies("CookieSearch").Expires = DateTime.Now.AddDays(30)
' store the cookie
Response.Cookies.Add(CookieSearch)
```

Een cookie uitlezen

```
Request.Cookies("LoginCookie")
```

Een cookie verwijderen

Een cookie kan verwijderd worden door de vervaldatum ervan in het verleden te plaatsen.

```
Dim mycookie As HttpCookie = New HttpCookie("CookieSearchTerms")
mycookie.Expires = DateTime.Now.AddDays(-1D)
Response.Cookies.Add(mycookie)
```

Extra informatie: http://msdn.microsoft.com/en-us/library/ms178194.aspx

Uitwerking in onze applicatie

We willen de vijf recentste zoekopdrachten in onze filmapplicatie bijhouden in één cookie. Net zoals in een array kunnen we een meerdimensionale cookie aanmaken met exact 5 posities om onze data in te plaatsen. Uiteraard bestaan er meerdere en wellicht beter manieren om dit op te lossen. In dit voorbeeld deden we het als volgt:

Achter onze zoekknop plaatsen we de opdracht om op zoek te gaan naar een eventuele cookie. Bestaat deze, dan gaan we kijken waar we de laatste zoekterm mogen plaatsen in de cookie. Zo zal bij een allereerste zoekopdracht enkel positie 0 in de cookie bezet zijn. Is alles reeds bezet (er zijn al 5 of meer zoekopdrachten geweest) dan zullen we ervoor moeten zorgen dat de allerlaatste zoekopdracht wordt toegevoegd (op positie 4) en dat de oudere zoektermen “opschuiven”. Op die manier zal de oudste zoekterm uit de cookie verwijderd worden.

Onderstaand voorbeeld dient enkel om aan te tonen hoe we cookies zouden kunnen gebruiken in onze filmset applicatie.

```
Protected Sub btnSearch_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnSearch.Click
        'check if cookie exist
        Dim CookieSearchTerms As HttpCookie =
Request.Cookies("CookieSearchTerms")

        If (CookieSearchTerms IsNot Nothing) Then
            ' exists, so just update values
            If (CookieSearchTerms(0) = "") Then
                CookieSearchTerms(0) = Me.txtSearchterm.Text.ToString
            End If

            If (CookieSearchTerms(1) = "") Then
                CookieSearchTerms(1) = Me.txtSearchterm.Text.ToString
            End If

            If (CookieSearchTerms(2) = "") Then
                CookieSearchTerms(2) = Me.txtSearchterm.Text.ToString
            End If

            If (CookieSearchTerms(3) = "") Then
                CookieSearchTerms(3) = Me.txtSearchterm.Text.ToString
            End If

            If (CookieSearchTerms(4) IsNot "") Then
                ' o-my! there is no more room in on of 5 spots inside our
cookie. Delete oldest entry and add new one!
```

```
                CookieSearchTerms(0) = CookieSearchTerms(1)
                CookieSearchTerms(1) = CookieSearchTerms(2)
                CookieSearchTerms(2) = CookieSearchTerms(3)
                CookieSearchTerms(3) = CookieSearchTerms(4)
                CookieSearchTerms(4) = Me.txtSearchterm.Text.ToString
                'litRecentSearchTerms.Text +=
Request.Cookies("CookieSearchTerms")(4).ToString
            Else
                CookieSearchTerms(4) = Me.txtSearchterm.Text.ToString
            End If

            Response.Cookies.Add(CookieSearchTerms) 'save changes to cookie
file

        Else
            ' does not yet exist, so create the cookie
            CookieSearchTerms = New HttpCookie("CookieSearchTerms")
            CookieSearchTerms(0) = Me.txtSearchterm.Text.ToString
            ' set expiration date manually to one month
            CookieSearchTerms.Expires = DateTime.Now.AddDays(30)
            ' store the cookie
            Response.Cookies.Add(CookieSearchTerms)
        End If

        readcookie() ' read all updated values and update label with latest
search terms

    End Sub
```

REFERENTIE IN PROJECT: DEFAULT.ASPX.VB

De subroutine “readcookie” doet niets meer dan heel onze cookie uitlezen en alles tonen op het scherm van de gebruiker. We hebben deze functionaliteit in een aparte subroutine geplaatst, omdat we zo deze routine op een andere plek nogmaals kunnen aanroepen, bijvoorbeeld bij het laden van onze pagina. We willen dat de cookie wordt uitgelezen bij een volgend bezoek, ook wanneer er nog niet op de zoekknop gedrukt werd.

```
Sub readcookie()
        Dim intLastPosition As Integer = 0
        Dim CookieSearchTerms As HttpCookie =
Request.Cookies("CookieSearchTerms")

        If (CookieSearchTerms IsNot Nothing) Then
            If (CookieSearchTerms(0) IsNot "") Then
                intLastPosition = 1
            End If

            If (CookieSearchTerms(1) IsNot "") Then
                intLastPosition = 2
            End If

            If (CookieSearchTerms(2) IsNot "") Then
                intLastPosition = 3
            End If

            If (CookieSearchTerms(3) IsNot "") Then
                intLastPosition = 4
            End If
```

```
            If (CookieSearchTerms(4) IsNot "") Then
                intLastPosition = 5
            End If

            Response.Cookies.Add(CookieSearchTerms)

        End If


        If intLastPosition > 0 Then
            ' there are entries inside our cookie, show them
            lblRecentStatus.Text = "You recently searched for: "
            litRecentSearchTerms.Text = ""
            For teller As Integer = (intLastPosition - 1) To 0 Step -1
                litRecentSearchTerms.Text &= CookieSearchTerms(teller)
                If teller <> 0 Then
                    litRecentSearchTerms.Text &= ", "
                End If
            Next

        End If

    End Sub


Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        If Not IsPostBack Then
            readcookie()
        End If
    End Sub
```

Hou in het achterhoofd dat de voorbeelden uit het filmproject geen best-practices zijn, maar wel voorbeelden van nuttige toepassingen van de geziene leerstof.

Oefeningen bij dit hoofdstuk

Oefening 1

1. Maak een ASP.NET pagina waar u in een formulier een paswoord kunt ingeven.

2. Bij de 'postback' van het formulier controleert u of het paswoord dat werd ingegeven weldegelijk "hamai" was.

   Als dit paswoord juist is zorgt u ervoor dat dit als sessievariabele wordt bijgehouden.
   Als het paswoord fout is schrijft u een cookie weg die de volgende dag vervalt waarin het aantal keer dat het paswoord fout was wordt opgeslagen. Na 3 keer een fout paswoord te hebben ingegeven, wordt de homepage van "GOOGLE" geopend en kan men de huidige pagina niet meer openen tot de cookie vervallen is (of verwijderd).

3. Maak een ASP.NET pagina die enkel opent als de sessievariabele met het juiste paswoord aanwezig is. Indien de sessievariabele niet aanwezig is wordt de pagina van oefening 1 geopend.
   Toon in deze nieuwe pagina de sessie-ID.

REFERENTIE: VOORBEELD_018.ASPX EN VOORBEELD_018B.ASPX