3. Управление на състоянието

Съхраняването на информацията за състоянието е въпрос с голяма важност за уеб приложенията. Например хиляди потребители може едновременно да изпълняват едно и също уеб приложение на един компютър (сървър), всеки от които комуникира през HTTP връзка, която не поддържа състояние. Разбирането за ограниченията за състоянието е ключа към създаване на ефективни уеб приложения.

3.1. Показване на състоянието

Показването на състоянието използва скрито поле, което ASP.NET автоматично вмъква накрая, обработвайки HTML кода на уеб страницата. Това поле е подходящо място за съхраняване на информация, която ще се използва за няколко връщания обратно на сървъра на една уеб страница.

Показването на състоянието не е лимитирано само до уеб контролите. Кодът на уеб страницата може да добавя малки части информация директно към полето за състояние, която да се извлече по-късно когато страницата се изпрати обратно. Типът на информацията, която може да се съхранява включва прости типове данни и собствени специфични обекти.

Колекция ViewState

Свойството **ViewState** на страницата осигурява текущата информация за състоянието. Това свойство осигурява инстанция на клас за колекция **StateBag** – колекция речник, в която всеки елемент се съхранява в отделен слот използвайки уникално стрингово име.

Пример за добавяне в състоянието на променлива съдържащата числова стойност:

```
Me.ViewState("Counter") = 1
```

Ключовата дума Me е обръщение към обекта текуща страница (опционална)

Добавя се променлива с име Counter и стойност 1 в колекцията ViewState. Когато няма елемент в колекцията с име Counter, той се добавя. Ако има такъв елемент неговата стойност ще се замени.

Извличането на стойността ще се извърши с използването на ключовото име. Стойността й трябва да се преобразува (cast) в подходящия тип. Колекцията ViewState изисква преобразуване защото тя съхранява всички елементи в базови обекти, което изисква те да се справят с много различни типове данни.

Код за извличане на Counter от ViewState и преобразуването му в integer.

```
Dim mycounter As Integer
mycounter = CType(Me.ViewState("Counter"), Integer)
```

Защитаване на състоянието

Информацията за състоянието се съхранява в един разбъркан низ.

```
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="dDw3NDg2NTI5MDg7Oz4=" />
```

Той не е кодиран и може да бъде преобразуван в Base64 стринг.

Защита от несанкционирано използване на състоянието

За защита на състоянието има две възможности.

Първо, може да се използва хеш код. Хеш кодът се описва с криптирани контролни суми. ASP.NET ще обработи данните с алгоритъма за хеширане.Той ще

създаде кратък сегмент от данни, които са хеш кода. Този код се добавя в края на данните за състоянието.

Скриване на състоянието

Включване на кодирането на за индивидуална страница се извършва с ViewStateEncryptionMode свойството на директивата Page.

```
<%@Page ViewStateEncryptionMode="Always" %>
```

Кодиране на всички страници на уеб сайта – настройва се атрибута viewStateEncryptionMode в конфигурационния файл (web.config).

```
<system.web>
<pages viewStateEncryptionMode="Always" />
...
</system.web>
```

Запазване на съставните променливи

Базовият принцип за запазване на състоянието на променливите от ASP.NET страницата е да се използва състоянието когато възниква събитието Page.PreRender и да се върнат когато възникне събитието Page.Load. Събитието Load възниква всеки път, когато страницата се създава. В случай на изпращане страницата обратно (postback), събитието Load възниква първо, следвано от другите събития на контролите.

Пример за използване на тази техника с една променлива (Contents) от страницата. Страницата съдържа една текстова кутия и два бутона. С натискане на бутон SaveContents се съхранява стринга съдържащ се в текстовата, а с натискане на бутон LoadContents може да се възстанови по-късно. За съхраняването и зареждането се използва стринговата променлива Contents. Манипулаторите на двата бутона не трябва да съхраняват и възстановяват в състоянието, защото манипулаторите PreRender и Load извършват действието, когато обработката на страницата започва и завършва.

Забележка: Манипулатори на събитията страница се добавят:

- като се посочи страницата в панел Solution, натисне се д.б. и се изпълни View Component Designer и в панел Properties се включва страницата на събитията;
- от лентата на .vb файла се избере отляво Page Events, а отдясно се избира събитието.

Деклариране на променлива в класа на формата:

```
' Променлива, която ще бъде изчистена с всяко изпращане обратно
Private Contents As String
```

Код в манипулатора на събитието PreRender:

```
' Запазване на променливи
ViewState("Contents") = Contents
```

Код в манипулатора на събитието Load:

```
If Me.IsPostBack Then
    ' Възстановяване на променливи
    Contents = CType(ViewState("Contents"), String)
End If
```

Код в манипулатора на събитието Click на бутон SaveContetnts:

```
' Предаване на съдържанието в текстовата кутия в променливата
Contents = txtValue.Text
txtValue.Text = ""
```

Код в манипулатора на събитието Click на бутон LoadContents:

```
' Възстановяване на съдържанието на текстовата кутия от променливата
txtValue.Text = Contents
```

Логиката на събитията Load и PreRender позволява работа подобна на десктоп приложение. При използване на тази техника не трябва да се съхранява ненужна информация, защото тя ще увеличи размера на резултатната страница, което ще доведе до забавяне времето на нейното предаване. Друг недостатък, е че всяка част от данните трябва да бъде изрично съхранена и възстановена, което може да доведе до грешка, когато програмистът забрави да извърши някое от действията.

Препоръчва се подходът за съхраняване на променливите от страницата в състоянието да се използва *само при специални случаи*.

Съхраняване на специфични обекти

Могат да се съхраняват обекти дефинирани от потребителя в ViewState. За да бъдат съхранени те трябва да бъдат конвертирани в поток от байтове, за да бъдат добавени към скритото поле в страницата. Този процес се нарича сериализация (serialization). Ако обектът не е сериализиран се получава съобщение, когато се прави опит да се добави в ViewState.

За да се сериализира обекта, трябва да се добави атрибут Serializable преди декларацията на класа.

Пример за деклариране на сериализиран потребителски клас Customer:

```
<Serializable()> _
Public Class Customer

    private _firstName As String
    Public Property FirstName() As String
        Get
            Return _firstName
        End Get
    Set(ByVal Value As String)
        _firstName = Value
    End Set
    End Property

    private _lastName As String
    Public Property LastName() As String
        Get
            Return _lastName
        End Get
        Set(ByVal Value As String)
            _lastName = Value
        End Set
    End Property

    Public Sub New(ByVal firstName As String, ByVal lastName As String)
```

```
            Me.FirstName = firstName
            Me.LastName = lastName
        End Sub
End Class
```

Понеже клас Customer е маркиран като сериализиран, той може да бъде съхранен в ViewState:

```
'Съхраняване на Customer в ViewState
Dim cust As New Customer("Иван", "Петров")
ViewState("CurrentCustomer") = cust
```

Преди да се използва специфичния обект, той трябва да се конвертира (cast) преди извличане от ViewState:

```
'Извличане на Customer от ViewState
Dim cust As Customer
cust = CType(ViewState("CurrentCustomer"), Customer)
TextBox1.Text = cust.FirstName + " " + cust.LastName
```

3.2. Прехвърляне на информация между страниците

Сериозно ограничение на ViewState е, че то е тясно обвързано с определена страница. Ако потребителят навигира към друга страница тази информация се губи. Този проблем може да се реши, като най-добрите подходи са:

- изпращане на свързани страници;
- използване на параметризирани адреси.

Изпращане на свързани страници

Връщането обратно на свързани се страници е техника, която разширява механизма за изпращане обратно (postback), позволявайки една страница да изпрати потребителя към друга страница, пълна с цялата информация за тази страница.

Инфраструктурата, която поддържа изпращане обратно на свързани страници е свойството PostBackUrl, което е дефинирано от интерфейса IButtonControl и присъства в контроли-бутони като ImageButton, LinkButton и Button. За използване на тази техника е достатъчно да се зададе в PostBackUrl името на друга уеб форма. Когато потребителят натисне бутона, страницата ще бъде изпратена на този нов URL със стойностите на всички контроли от текущата страница. (Изпратената обратно информация включва скритото поле за състояние ViewState).

Пример. Страница (CrossPage1.aspx) с две текстови кутии (име и фамилия) и бутон ще изпрати информация на друга страница (CrossPage2.aspx), която ще покаже въведеното име и фамилия в етикет.

Създава се страница CrossPage1.aspx. В свойството PostBackUrl на бутона се въвежда URL – CrossPage2.aspx. Страницата CrossPage1.aspx не съдържа никакъв код.

CrossPage2.aspx може да взаимодейства с CrossPage1.aspx използвайки свойството Page.PreviousPage. Обработката се извършва в манипулатора на събитието **Load** на CrossPage2.aspx.

```
If Not Page.PreviousPage Is Nothing Then
    Dim tb1, tb2 As TextBox
    Dim str1 As String = ""

    tb1 = CType(PreviousPage.FindControl("TextBox1"), TextBox)
```

```
        If Not tb1 Is Nothing Then
            str1 = tb1.Text
        End If

        tb2 = CType(PreviousPage.FindControl("TextBox2"), TextBox)
        If Not tb2 Is Nothing Then
            str1 &= " " + tb2.Text
        End If

        Label1.Text = "Вашето име е: " + str1
    End If
```

Преди опита за достъп до предишната страница се проверява дали PreviousPage е недействителна връзка (Nothing), което е възможно когато CrossPage2.aspx е стартирана директно.

Методът FindControl се използва за намиране на контроли в текущия контейнер. Намерената стойност на контролата (TextBox1 или TextBox2) се записва в обект от същия тип (tb1 или tb2) чрез конвертиране с CType() (cast).

Използване на параметризирани адреси

Друг често използван подход за предаване на информация е чрез използване на стринг за заявка в URL. Този подход основно се използва в търсещите машини. При него потребителя ще се пренасочи към нов URL, който включва параметрите на заявката.

Например:
http://www.google.ca/search?q=Visual+Basic

Параметрите на заявката са частта след знака "?". В примера се съдържа една променлива "q", която съдържа стринговете Visual+Basic.

Предимства на подхода:

- Той е лек и не натоварва сървъра.

Недостатъци:

- Информацията е лимитирана до прости стрингове, които трябва да съдържат валидни URL символи;
- Информацията е ясно видима за потребителите и за тези, които подслушват Интернет;
- Някой потребител може да пробва да промени заявката, поставяйки нови стойности, които програмата и не очаква и срещу които не е защитена;
- Много браузъри имат ограничения за дължината на URL (обикновено от 1 KB до 2 KB), което не позволява поставяне на голямо количество информация в заявката.

За да се постави информация в стринга на заявката. Няма колекция, поради което потребителят трябва сам да вмъкне параметрите в URL.

Подаване на параметри с използване на метода Response.Redirect()

1) Преминаване към newpage.aspx изпращайки стринг с аргумент recordID задавайки му стойност 10

```
Response.Redirect("newpage.aspx?recordID=10")
```

2) Могат да се изпратят множество параметри, разделени с амперсанд (&)

```
Response.Redirect("newpage.aspx?recordID=10&mode=full")
```

Използване на параметрите от получаващата страница

Стойностите могат да се получат от колекцията речник QueryString достъпна чрез вградения обект Request.

Dim ID As String = Request.QueryString("recordID")

Забележка: Информацията винаги се извлича като стринг, който може да бъде конвертиран в друг прост тип данни. Стойностите в колекцията QueryString са индексирани по име на променливата. При опит да се извлече стойност, която не присъства в стринга на заявката се получава недействително обръщане (Nothing).

Пример: Подаване на двете имена въведени в страницата CrossPage1.aspx чрез параметризиран адрес.

На CrossPage1.aspx се добавя се нов бутон с текст QueryString.

Код на събитието Click на бутон QueryString:

```
Dim Url As String = "QueryStringRecipient.aspx?"
Url &= "FirstName=" & TextBox1.Text & "&"
Url &= "LastName=" & TextBox2.Text
Response.Redirect(Url)
```

Добавя се нова страница с име QueryStringRecipient.aspx. Върху нея се поставя етикет. В кода на събитието Load на страницата се изпълнява кода:

```
Dim str1 As String

str1 = Request.QueryString("FirstName")
str1 &= " " & Request.QueryString("LastName")

Label1.Text = "Вашето име е: " & str1
```

Кодиране на URL

Потенциален проблем може да възникне с стринга на заявката на параметризираните адреси, поради неразрешени символи в него. Поради това е добре да се изпълни кодиране на текстовите преди да се вмъкна в стринга с адреса. За кодиране се използва **UrlEncode()**, а за декодиране **UrlDecode()** методите на класа **HttpServerUtility**, които могат да се използва чрез свойството **Page.Server**.

Кодиране на първото име:

```
Url &= "FirstName=" & Server.UrlEncode(TextBox1.Text) & "&"
```

Забележка: Кодиране се извършва само на текстовите стойности, в които може да има неразрешени символи.

Декодиране на първото име:

```
str1 = Server.UrlDecode(Request.QueryString("FirstName"))
```

3.3. Използване на бисквитките (Cookies)

Същност на бисквитките

Бисквитките (Cookies) са друг начин за съхраняване на информация за по-късно използване. Бисквитките са малки файлове, които се създават в паметта на уеб браузъра (ако са временни) или на хард диска на клиента (ако са постоянни). Предимство на бисквитките, е че те работят прозрачно, без потребителят да знае, че тази информация трябва да се съхрани. Те могат да се използват лесно от всяка

страница в приложението и дори да съхранят дългосрочно информация, която да се използва при следващи посещения.

Браузърът по подразбиране съхранява получената бисквитка и от него се очаква да я изпраща обратно към сървъра при всяка следваща заявка. Информацията в нея може да е произволна, стига цялостният размер на бисквитката (информацията и мета данни за самата бисквитка) да не надвишава 4 KB.

Те имат недостатъците, отнасящи се до стринговете на заявките – те са ограничени до проста текстова информация и лесно могат да се намерят и прочитат от потребителя отваряйки кореспондиращия файл. По тези причини те не са подходящи за съхраняване на сложна или лична информация и за големи обеми от данни. Някои потребители забраняват бисквитките на техните браузъри, което носи проблеми на уеб приложенията, които ги използват. Също така потребителите могат ръчно да изтрият бисквитките от хард диска. Бисквитките са широко разпространени и използвани от много уеб сайтове.

Свойства на бисквитките

По-важни свойства на бисквитките:

- **Expires** - указва кога изтича валидността на бисквитката. Ако не се укаже, бисквитката се запазва само в паметта. Ако това свойство се зададе, бисквитката се записва на твърдия диск и се пази за времето, което е указано. Когато браузърът изпраща дадена бисквитка, той проверява дали нейната валидност не е изтекла и ако така, той не я изпраща към сървъра, а я изтрива. Не трябва да забравяме, че потребителят може да изтрие бисквитката по всяко време.
- **Domain** – областта от интернет адреси, на които може да се праща бисквитката. По подразбиране това е адресът, от който е дошла, но може да се укаже и друго. Браузърът изпраща само бисквитките, предназначени за поискания интернет адрес.
- **Path** – път на адресите, до които може да се праща бисквитката. Бисквитката няма да се праща на адреси от по-високо ниво в дървото на директориите. Пример: ако пътят е /sites/stefan, тя няма да се прати на /sites/dido, нито на /sites, но ще се прати на /sites/stefan/pics. По подразбиране стойността на това свойство е виртуалната директория, от която първоначално е дошла бисквитката, но може и да се промени.

Работа с бисквитките

Двата обекта **Request** и **Response** (осигурени чрез свойствата на **Page**) осигуряват колекция Cookies:

1) Задаване на бисквитка се извършва със създаване на нов обект HttpCookie. След това той може да се запълни с текстова информация и да се прикрепи към текущия уеб отговор.

```
' Създаване на обект cookie
Dim cookie As New HttpCookie("Preferences")
' Задаване на стойност в него
cookie("LanguagePref") = "English"
' Добавяне на друга стройност
cookie("Country") = "US"
' Добавяне към текущия уеб отговор
Response.Cookies.Add(cookie)
```

Добавената бисквитка ще се запази докато потребителят затвори браузъра и ще бъде изпратена с всяка заявка. За създаване на бисквитка с по-продължително запазване трябва да се използва дата на изтичане (expiration date):

2) Извличане на бисквитките по нейното име се извършва с колекцията Request.Cookies.

```
Dim cookie As HttpCookie = Request.Cookies("Preferences")
' Проверка дали има бисквитка с търсеното име
Dim language As String
If cookie IsNot Nothing Then
    language = cookie("LanguagePref")
End If
```

Пример за типично използване на бисквитки:

1) При регистриране на потребителя записване на бисквитка с неговото име и държава с продължителност на запазване 1 г.

2) Разпознаване на потребителя при следващо посещение на уеб сайта и показване на неговото име и националност.

Задаване на бисквитка с име и националност на потребителя. Записва се при натискане на бутон регистриране:

```
Dim cookie2 As HttpCookie = Request.Cookies("UserID")
If cookie2 Is Nothing Then
    cookie2 = New HttpCookie("UserID")
End If
cookie2("UserName") = TextBox1.Text
cookie2("UserCountry") = TextBox2.Text
cookie2.Expires = DateTime.Now.AddYears(1)
Response.Cookies.Add(cookie2)
```

Прочитане на бисквитката при зареждане на страницата. Ако няма бисквитка с това име се показва съобщение "Моля да се регистрирате". При откриване на бисквитката се прочитат име и държава и се отпечатва съобщение "Здравейте, потребител от държава". Кодът се включва в манипулатора на събитието Page.Load:

```
Dim CookieUserID As HttpCookie = Request.Cookies("UserID")
If CookieUserID Is Nothing Then
    ' Няма бисквитка с това име
    LabelUserID.Text = "Моля да се регистрирате"
Else
    ' Бисквитката е намерена
    LabelUserID.Text = "Здравейте, " & CookieUserID("UserName") & " от " &
CookieUserID("UserCountry")
End If
```

3.4. Състояние на сесиите (Session State)

Състоянието на сесията позволява да се съхрани всякакъв тип данни в паметта на сървъра. Тази информация е защитена, защото тя никога не се изпраща до клиента и е уникално обвързана с конкретната сесия. Всеки клиент, който има достъп до приложението има различна сесия и отделна колекция от информация. Състоянието на сесията е подходящо за съхраняване на информация, като елементите на текущата потребителска кошница, когато потребителят преглежда от една към друга страница.

Вградените в ASP.NET възможности за поддръжка на потребителска сесия (session state) ни позволяват да:

- Идентифицираме и класифицираме автоматично в логическа сесия всички заявки, идващи от един и същ браузър.
- Запазваме данни на сървъра за сесията, с цел използването им между множество отделни заявки.
- Да обработваме в кода ни събития, свързани със сесията (**Session_OnStart**, **Session_OnEnd**, и т.н.).
- Автоматично да бъдат освобождаване данните за сесията, ако в определен период от време не се получи заявка от браузъра.

Поддръжката на сесии в ASP.NET се характеризира с:

- Леснота за ползване.
- Надеждно запазване на данни, устойчиво на рестартиране на IIS или на работния процес на ASP.NET.
- Скалируемост в уеб ферма и уеб градина.
- Възможност за функциониране и без HTTP бисквитки.
- По-добро бързодействие спрямо класическото ASP.

Забележка: Състоянието на сесиите не се запазва извън границите на едно уеб приложение.

Проследяване на сесията

ASP.NET следи всяка сесия използвайки уникален 120 битов идентификатор SessionID. ASP.NET използва собствен алгоритъм за да генерира тази стойност, гарантирайки, че стойността е уникална. Идентификатора е съставен от ASCII символи и може да участва в URL адреси. Този идентификатор е само част от информацията свързана със сесията, която се предава между сървъра и клиента.

Когато клиентът представи идентификатора на сесията, ASP.NET търси кореспондиращата сесия, извлича обектите, които са съхранени преди това и ги поставя в специална колекция, така че да бъдат достъпни в кода. Този процес се извършва автоматично.

За да работи тази система, клиентът трябва да представи съответния идентификатор на сесията при всяка заявка. Това може да се извърши по два начина:

- ***Използване на бисквитки***: В този случай, идентификатора на сесията се предава чрез специална бисквитка (с име ASP.NET_SessionId), която се създава автоматично когато се използва колекцията на сесията. Това е по подразбиране.
- ***Използване на модифициран URL.*** В този случай, идентификаторът на сесията се предава в специално модифициран URL. Това предоставя възможност за създаване на приложения, които използват състояние на сесия с клиенти, които не поддържат бисквитки.

Състоянието на сесията натоварва сървъра. Той трябва да съхранява допълнителна информация в паметта. Въпреки, че информацията е малка, тя може бързо да нарасне, когато стотици или хиляди потребители използват сайта и да намали производителността.

Използване на състоянието на сесията

За използване на сесията може да се взаимодейства с клас System.Web.SessionState, който се предоставя от ASP.NET като вграден обект за сесия.

Добавяне на елемент към колекцията Session:

```
Session("FirstName") = "Иван"
```

Извличане на елемента се извършва с подходящо конвертиране:

```
Dim firstName As String = CType(Session.Item("FirstName"), String)
```

Състоянието за сесията е глобално за цялото приложение за текущия потребител. То се загубва при следните случаи:

- Ако потребителят затвори и рестартира браузъра
- Ако потребителя отвори същата страница от друг от друг прозорец на браузъра, тя е достъпна когато страницата се отвори от оригиналния прозорец на браузъра
- Ако времето на сесията изтече защото няма активност от страна на потребителя
- Ако кодът на уеб страницата завърши сесията чрез извикване на метода Session.Abandon()

Методи и свойства на класа HttpSessionState

Count	Осигурява броя на елементите в текущата колекция на сесията
IsCookieless	Идентифицира дали сесията се осъществява с бисквитки или с модифицирани URL адреси
IsNewSession	Идентифицира дали сесията е създадена само за текущата заявка
Keys	Дава колекция от всички ключове на сесията, които се използват за съхраняване на елементи в колекцията на сесията
Mode	Осигурява стойности в изброяване, които обясняват как е съхранена информацията в сесията
SessionID	Показва стринг с уникалния идентификатор на сесията за текущия клиент
Timeout	Показва броя на минутите, които ще изминат преди текущата сесия да бъде прекратена
Abandon()	Завършва веднага текущата сесия и освобождава паметта, която тя заема
Clear()	Изтрива всички елементи на сесията, но без да изтрива идентификатора на текущата сесия

Пример: Използване на състоянието на сесията за съхраняване на няколко обекта DigitalCamera.

Обектът DigitalCamera комбинира няколко свързани променливи и използва конструктор за да ги създаде и инициализира.

Създава се нов клас с име DigitalCamera:

```
Public Class DigitalCamera

    Public Model As String
    Public Color As String
    Public Price As Double

    Public Sub New(ByVal model As String,
     ByVal color As String, ByVal price As Double)
        Me.Model = model
        Me.Color = color
        Me.Price = price
    End Sub

End Class
```

Три обекта DigitalCamera се създават, когато страницата се зарежда и се съхраняват в състоянието на сесията. След това потребителят може да избира от списък името на цифровия фотоапарат. Когато бъде направи избора кореспондиращия обект ще бъде извлечен и информацията ще бъде показана.

На събитието Page.Load се въвежда кода:

```
If Me.IsPostBack = False Then

    ' Създаване на 3 обекта DigitalCamera
    Dim Item1 As New DigitalCamera("Canon PowerShot A810", "Сребърен", 129.35)
    Dim Item2 As New DigitalCamera("Canon PowerShot A2200", "Черен", 179.85)
    Dim Item3 As New DigitalCamera("Canon PowerShot A2300", "Червен.", 189.99)

    ' Добавяне на обектите към сесията
    Session("DigitalCamera 1") = Item1
    Session("DigitalCamera 2") = Item2
    Session("DigitalCamera 3") = Item3

    ' Добавяне на редовете към визуалната контрола DropDownList1
    DropDownList1.Items.Add(Item1.Model)
    DropDownList1.Items.Add(Item2.Model)
    DropDownList1.Items.Add(Item3.Model)
End If
```

На събитието Button1.Click се въвежда кода:

```
If DropDownList1.SelectedIndex = -1 Then
    Label2.Text = "Няма избран фотоапарат!"
    Label3.Text = ""
    Label4.Text = ""
Else
    ' Конструиране на ключово име базирано на индекс
    ' (DigitalCamera1, DigitalaCamera2, .....)
    Dim Key As String
    Key = "DigitalCamera" & (DropDownList1.SelectedIndex + 1).ToString()
    ' Извличане на обекта DigitalCamera от състоянието на сесията
    Dim Item As DigitalCamera = CType(Session(Key), DigitalCamera)
    ' Показване на информация за обекта
    Label2.Text = "Модел: " & Item.Model
    Label3.Text = "Цвят: " & Item.Color
    Label4.Text = "Цена: " & Item.Price.ToString("c")
End If
```

3.5. Конфигурация на сесията

Конфигурирането на сесията се извършва с файла web.config за текущото приложение. Конфигурационният файл позволява да се зададат допълнителни опции.

Настройката се извършва в секцията на елемента <sessionState>. Някои от настройките се прилагат само за някои видове състояние на сесията. Най-важните настройки са показани в следния пример:

```
<?xml version="1.0"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=169433
  -->
<configuration>
  <system.web>
    <compilation debug="true" strict="false" explicit="true" targetFramework="4.0"/>
    <sessionState
```

```
        cookieless="UseCookies"
        cookieName="ASP.NET_SessionId"
        regenerateExpiredSessionId="false"
        timeout="20"
        mode="InProc"
        stateConnectionString="tcpip=127.0.0.1:42424"
        stateNetworkTimeout="10"
        sqlConnectionString="data source=127.0.0.1;Integrated Security=SSPI"
        sqlCommandTimeout="30"
        allowCustomSqlDatabase="false"
        customProvider=""
        compressionEnabled="false" />
  </system.web>
</configuration>
```

Cookieless - без бисквитки

Може да се използват настройки без бисквитки дефинирани от изброяването HttpCookieMode, кокто е показано в таблицата:

Стойности на HttpCookieMode

UseCookies	Бисквитките винаги се използват, дори в случаите когато браузъра не поддържа бисквитки или те са забранени. Ако браузъра не поддържа бисквитки информацията за сесията ще бъде загубена, защото при следващите заявки, всяка заявка ще получи нов идентификатор.
UseUri	Бисквитките никога не се използват, независимо от възможностите на браузъра. Вместо това идентификационния номер на сесията се съхранява в URL.
UseDeviceProfile	ASP.NET избира дали ще използва сесия без бисквитки чрез изследване на възможностите на обекта BrowserCapabilities. Недостатъкът е, че този обект индикира какво устройството може да поддържа, не взима под внимание, че потребителят може да е забранил използването на бисквитки, въпреки, че устройството ги поддържа.
AutoDetect	ASP.NET опитва да разбере, дали браузъът поддържа бисквитки чрез опитване да зададе и прочете бисквитка (техника, която често се използва в уеб). Тази техника може коректно да определи дали браузъра поддържа бисквитки, но те са забранени.

Timeout – време за прекратяване на сесията

Определя броя минути, които ASP.NET ще изчака, без получаване на заявка преди да прекрати сесията.
<sessionState timeout="20" ... />

Тази настройка е много важна за състоянието на сесията. Разликата в минутите има много голям ефект върху натоварването на сървъра и производителността на приложението. При по-кратък брой минути сървърът ще освободи по-бързо ценна памет, но ще затрудни потребителя, ако е направил по-голяма пауза преди да продължи.

Времето за продължение на сесията може да се настрои и чрез кода в страницата, чрез смяна на свойството Timeout:
Session.Timeout = 10

Mode – режими на използване

В зависимост от използваните режими ASP.NET може да използва различни услуги при управление на състоянието в зависимост от използваните режими:

InProc. Подразбиращият се режим. Подходящ е за малки уеб сайтове. При него информацията се съхранява в същия процес, в който работи нишката на ASP.NET. Той осигурява добро производителност, но малка продължителност. При рестартиране на сървъра информацията за състоянието ще се загуби.

Off. Тази настройка забранява управлението на състоянието на сесията за всяка страница в приложението. Тя може да осигури малко подобряване на производителността за уеб сайтове, които не използват състояние на сесията.

StateServer. С тази настройка ASP.NET използва отделна услуга за управление на състоянието. Тази услуга ътябва да работи на същия уеб сървър, но извън главния процес на ASP.NET.

Когато се използва StateServer трябва да се зададе стойност на настройката stateConnectionString.

SQLServer. Тази настройка инструктира ASP.NET да използва БД на SQL Server за да съхранява информацията, която е идентифицирана с атрибут sqlConnectionString.

Custom. При използване на този режим, трябва да се индикира кой доставчик за съхраняване на състоянието ще се използва чрез атрибута customProvider. Атрибутът customProvider индикира името на клас. Класът може да бъде част от уеб приложението (като в този случай трябва да бъде разположен във вложената папка Code), или може да бъде асембли използвано от уеб приложението (в този случай компилираното асембли трябва да бъде разположено във вложената папка Bin).

Compression. Когато се зададе enableCompression на true, данните на сесията се компресират преди да се изпратят извън процеса. Включването на настройката enableCompression има ефект само когато се използва съхранява извън процеса, защото това е само в положение, когато данните са сериализирани.

3.6. Състояние на приложението (Application State)

Състоянието на приложението (Application State) позволява да се съхраняват глобални обекти, които могат да бъдат достъпни от всеки клиент. Състоянието на приложението е базирано на класа System.Web.HttpApplicationState, който се осигурява във всички уеб страници чрез вградения обект Application.

Състоянието на приложението е същото както състоянието на сесията. То поддържа същите типове обекти, запазва информация на сървъра и използва същия синтаксис базиран на речник. Типични примери на използване на състоянието на приложението е глобален брояч, който проследява колко пъти дадена операция е изпълнена от всички клиенти на уеб сайта. При тези случаи се създава манипулатор на събитие в global.asax, който проследява колко на брой сесии са създадени и колко заявки са получени от приложението.

Пример за брояч проследяващ колко пъти определена страница е изискана от различни клиенти.

В манипулатора на събитието Page.Load се поставя кода:

```
' Извличане на текущата стойност брояча.
Dim Count As Integer = CType(Application("HitCounterForOrderPage"), Integer)
' Увеличаване на брояча
Count += 1
' Съхраняване на текущата стойност на брояча
Application("HitCounterForOrderPage") = Count
Label1.Text = Count.ToString()
```

Състоянието на приложението не се използва често, защото то като цяло е неефективно. В предишния пример, брояча няма да работи точно, при натоварен трафик. Ако двама клиенти едновременно са поискали една и съща страница по едно и също време брояча няма да работи точно.

За да се избегне този проблем трябва да се използват методите Lock() и Unlock(), които позволяват само на един потребител достъп до колекцията на приложението по едно и също време.

```
' Изисква се достъп само на един потребител
Application.Lock()

' Извличане на текущата стойност брояча.
Dim Count As Integer = CType(Application("HitCounterForOrderPage"), Integer)
' Увеличаване на брояча
Count += 1
' Съхраняване на текущата стойност на брояча
Application("HitCounterForOrderPage") = Count

' Освобождаване на единствения достъп
Application.UnLock()

Label1.Text = Count.ToString()
```

При използването на подхода изискващ заключване на състоянието приложението, всеки клиент, който изисква страницата ще бъде спрян, докато не бъде освободен достъпа. По тази причина, често променяни стойности не са подходящи за съхраняване в състоянието на приложението. Те може да бъдат разположени в таблица на БД.

3.7. Сравнение на възможностите поддържащи състоянието

Всяка възможност за избор от състоянието на приложението има различна продължителност на използване, обхват, производителност, използвани ресурси и нива на поддържане.

	Показване на състоянието (ViewState)	**Стринг на заявка в параметризирани адреси (Query String)**	**Специфични бисквитки (Custom Cookies)**
Позволени типове данни	Всички сериализирани .NET типове	Лимитиран обем от стрингови данни	Стрингови данни
Местоположение на съхраняване	Скрито поле в текущата уеб страница	Стринг в URL на браузъра	На клиентския компютър (в паметта или в малък текстов файл, в зависимост от продължителността на използване)
Продължителност на живота	Запазва се постоянно за изпращане обратно в текущата страница	Загубват се когато потребителят въведе нов URL	Задава се от програмиста. Може да бъдат

		или затвори браузъра	използвани в много страници и може да бъдат запазени между посещенията.
Обхват	Ограничено до текущата страница	Ограничено до целевата страница	За цялото ASP.NET приложение
Сигурност	Лесни за четене. Кодиране може да се извърши със свойството ViewStateEncryptionMode на директивата Page.	Лесно разбираеми и лесни за модифициране от потребителя	Несигурни и може да бъдат модифицирани от потребителя
Последици при изпълнение	Бавни, ако голям обем информация се съхранява, но не влияят на сървъра	Няма, защото количеството от данни е ограничено	Няма, защото количеството от данни е ограничено
Типично използване	Специфични настройки на страницата	Изпращане на идентификатор на продукт от страница-каталог към детайлна страница	Персонализация на предпочитанията на даден уеб сайт

	Състояние на сесията (Session State)	**Състояние на приложението (Application State)**
Позволени типове данни	Всички .NET типове данни за подразбиращия се режим на съхранение в процеса	Всички .NET типове данни
Местоположение на съхраняване	Паметта на сървъра, услуги на състоянието или SQL сървър в зависимост от използвания режим	Паметта на сървъра
Време на използване	Времето изтича след предварително дефинирания период (обикновено 20 минути, но може да бъде променен глобално или програмно)	Времето на използване на приложението (обикновено докато сървърът не се рестартира)
Обхват	Цялото ASP.NET приложение	Цялото ASP.NET приложение. За разлика от други методи и данни на приложението е глобално за всички потребители
Сигурност	Много сигурни, защото данните никога не се изпращат до клиента	Много сигурни, защото данните никога не се изпращат до клиента
Последици при изпълнение	Бавни, когато се съхранява голям обем информация,	Бавни, когато се съхранява голям обем информация,

	специално ако има много потребители едновременно, защото всеки потребител ще има свое копие от данните	защото тези данни никога няма да бъдат премахнати
Типично използване	Съхраняване на елементи в кошница на покупки	Съхраняване на различни типове глобални данни