24.6.6 Session

Bei einer Anfrage an einen HTTP-Server nutzt ein Client den angebotenen Dienst. Der besteht darin, dass der Server den HTML-Quelltext der im URL angegebenen Webseite an den Client zurück liefert. Es ist nach dem HTTP-Protokoll nicht vorgesehen den Benutzer zu erfassen, der die Anfrage gestellt hat, denn HTTP ist ein *zustandsloses* Protokoll. Wenn ein berechtigtes Interesse besteht, eine Anfrage genau einem Benutzer zuzuordnen – zum Beispiel bei einem Login in einen Admin-Bereich oder beim Füllen eines Warenkorbes oder beim Austausch personalisierter Daten – dann kann der Server die Verbindung zwischen dem HTTP-Server und einem HTTP-Client mit dem angemeldeten Benutzer verwalten. Man spricht davon, dass eine Sitzung oder eine Session begonnen wird. Unter den folgenden Links finden Sie kompakte Antworten auf die Frage, was eine Session in der Informatik ist:

- https://de.ryte.com/wiki/Session-ID
- http://de.wikipedia.org/wiki/Session-ID

24.6.6.1 Klasse Session

24.6.6.2 Eigenschaften

Die Klasse *Session* besitzt diese Eigenschaften:

Eigenschaft	Datentyp	Beschreibung
CookiePath	String	Gibt den Pfad zurück, in dem der Session-Cookie beim Client gespeichert ist oder setzt den Pfad. Wenn CookiePath leer ist, so wird CGI["SCRIPT_NAME"] dafür verwendet, mit der Ausnahme, dass "/." in diesem Fall durch "/" ersetzt wird.
Id	String	Gibt eine Session-ID als Zeichenkette zurück. Wenn keine aktuelle Session existiert, so wird NULL zurückgegeben. Das Format einer 'Session.Id' folgt der Syntax: <PREFIX>:<KEY>. <KEY> ist eine zufällige hexadezimale Zahl mit 24 Stellen. Wenn kein Präfix festgelegt wurde, so wird die IP-Adresse des Clients als Präfix verwendet. das ist der Standard.
Modified	Boolean	Gibt True zurück, wenn die Session modifiziert wurde oder setzt den Wahrheitswert.
Prefix	String	Gibt den Präfix-Teil der Session-ID zurück oder setzt ihn. Wird er nicht gesetzt, dann ist der Präfix die IP-Adresse des Clients.
Size	Long	Liest die Größe des Cookies aus.
Timeout	Float	Ermittelt oder legt die maximale Dauer einer Session in Sekunden fest. Der Standardwert ist 24*60*60 s = 86400 s, was einem Tag entspricht.
Type	String	Gibt den Typ der Datei zurück, in der die Session-Daten gespeichert werden oder setzt den Typ. Zulässig sind 'file' oder 'sqlite'. Wenn kein Typ angegeben wird, dann ist der Standard-Typ 'sqlite'. In diesem Fall erhält die Session-Daten-Datei automatisch die Extension .db.
Path	String	Pfad der Datei, in der Session-Daten auf dem Server gespeichert werden. Beispiel: /tmp/gambas.1000/session/27_0_0_1_724EC44AC512F3B5BAAC948C.db.
Unique	Boolean	Gibt True zurück, wenn die Session-ID als eindeutig deklariert wurde oder setzt den entsprechenden Wahrheitswert. Mit Session.Unique = True wird sichergestellt, dass es nur eine Session für den gleichen Präfix geben kann.
Keys	String[ ]	Gibt eine Liste aller Schlüssel als String-Array zurück, die mit einem Wert in der Sitzung verbunden sind.

Tabelle 24.6.6.2.1 : Eigenschaften der Klasse Session

24.6.6.3 Methoden

Über diese vier Methoden verfügt die Klasse *Session*:

Methode	Beschreibung
Load()	Lädt eine Session von der Festplatte des Servers. Entweder vom Standard-Pfad oder vom benutzer-definierten Pfad. Der Standard-Pfad ist /tmp/gambas.<user.id>/session.
Save()	Speichert eine Session auf der Festplatte des Servers. Entweder im Standard-Pfad oder im benutzer-definierten Pfad.

Methode	Beschreibung
Abandon()	Löscht die aktuelle Session. Die Session.Id ist dann Null. Die Session-Daten-Datei hat eine Lebensdauer, die durch die Eigenschaft Session.Timeout bestimmt wird.
Exist(Key As String)	Gibt True zurück, wenn ein Wert zu einem bestimmten Schlüssel in der aktuellen Sitzung existiert.

Tabelle 24.6.6.3.1 : Methoden der Klasse Session

24.6.6.4 Session

Die Klasse Session (gb.web) können Sie in einer Web-Anwendung einsetzen, um eine Session auf dem Server zu verwalten. Eine Session wird aber nicht automatisch angelegt, sondern nur dann, wenn eine interaktive Webseite Methoden bereitstellt, um eine Session anzulegen und zu verwalten.

Eine Session erzeugen

Eine neue Session wird erzeugt, sobald Sie das erste mal ein Session-Key-Wert-Paar (Session-Variable) speichern und aktuell keine Session existiert. Das können Sie prüfen, weil in diesem Fall die Eigenschaft Session.Id den Wert NULL hat.

```
If Not Session.Id Then Session["user1"] = webtxbUserName.Text
```

Hier ein Beispiel für eine Session-ID, wenn kein Präfix festgelegt wurde. In diesem Fall wird die IP-Adresse des Clients als Präfix verwendet: 10.254.3.151:CFA76E086C00E18CF07A8EC.

Wenn der Server eine neue Sitzungs-ID erzeugt und temporär abgespeichert hat, dann informiert er den Client über die Session-ID. Der Client muss dann bei jeder weiteren Anfrage (HTTP-Request) diese Session-ID an den Server mitschicken, um die Zuordnung Anfrage ↔ Client zu gewährleisten.

Eine Session speichern

- Eine vom Server erzeugte Session-ID wird an den Browser des Nutzers übertragen und dort gespeichert.
- Alle Daten, die mit einer Sitzung unter einer Session-ID verbunden sind, speichert der Webserver in einem extra dafür angelegten Verzeichnis auf dem Server. In der Regel ist das ein temporäres Verzeichnis "…/tmp". Die dort abgelegten Daten enthalten neben der Session-ID auch andere Inhalte, wie zum Beispiel eine Nutzer-ID oder Warenkorb-Daten.

Eine Session kann beim ersten Programmstart angelegt und bei einem weiteren Programmstart wieder hergestellt werden. Das setzt voraus, dass die Eigenschaft Session.Timeout in geeigneter Weise gesetzt wird. Der Standard sind 86400 Sekunden (24*60*60s) oder ein Tag.

Wenn die Eigenschaft *Cookie.Session* der Klasse Cookie (gb.qt4.webkit oder gb.qt5.webkit) auf den Wert *True* gesetzt wurde, dann wird ein (temporärer) Session-Cookie beim Client gesetzt. In diesem Cookie mit dem zugewiesenen Namen wird u.a. die *Session.Id* im gespeichert. Auch mit den beiden Methoden Response.SetCookie(...) und Response.RemoveCookie(...) der Klasse Response (gb.web) können Sie einen Cookie setzen oder löschen.

Session-Daten erzeugen, ändern und speichern

In einer Session mit einer eindeutigen Session-ID kann für jeden angegebenen Key ein Wert gespeichert werden. Sie können so viele Key-Wert-Paare speichern, wie Sie benötigen und auch deren Werte zu jeder Zeit der Sitzung ändern.

Für die Eingabe oder Änderung von Session-Daten wird häufig ein Web-Formular eingesetzt:

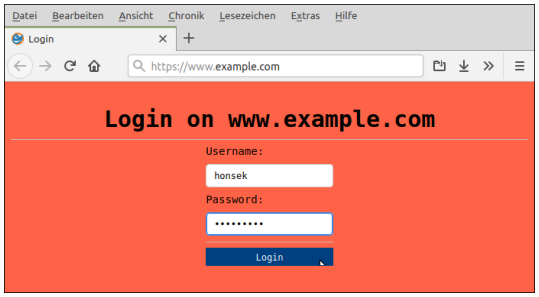

Abbildung 24.6.6.4.1: Login-Formular

Nur diese Datentypen sind für die Werte erlaubt: Array, native Datentypen und Collection:

```
Session["artikel1"] = "Tretroller"              ' Session-Variable → anlegen und mit einem Wert belegen
Session["preis1"] = 47.90                       ' Session-Variable → anlegen und mit einem Wert belegen
Session["artikel1"] = "Dreirad"                 ' Session-Variable → Wert ändern
Session["preis1"] = 30                          ' Session-Variable → Wert ändern
Session["liste1"] = ["Dreirad","rot","34€"]     ' Session-Variable → anlegen und mit Werten belegen
Session["artikel1"] = ""                        ' Session-Variable → löschen
```

Wenn Sie einem existierenden Key eine leere Zeichenkette übergeben, dann werden sowohl der Key als auch der zu diesem Key existierende Wert in der temporären Datei gelöscht!

Die Daten einer Session werden automatisch in einer temporären SQLite-Datenbank-Datei mit folgendem Pfad auf dem Server gespeichert, wenn Sie die Eigenschaft 'CookiePath' *nicht* explizit setzen:

```
"/tmp/gambas" &/ System.User.Id &/ "session" &/ Session.Id
```

Eine Session löschen

Eine Session-ID, die in einem Session-Cookie im Browser des Nutzers gespeichert ist, wird automatisch gelöscht, sobald der Nutzer den Browser komplett schließt. In Gambas beendet die Methode Session.Abadon() die aktuelle Sitzung sofort.

Beachten Sie: Wenn Sie *Response.Begin()* zu früh aufrufen, kann die Session nicht angelegt werden, da die HTTP-Header bereits gesendet wurden. Sie haben dann zwei Möglichkeiten:

- Sie rufen *Response.Begin()* auf, wenn Sie sicher sind, dass die Session bereits erzeugt wurde.
- Setzen Sie *Response.Buffered* auf True, so dass die Header erst gesendet werden, wenn Sie *Response.End()* aufrufen.

24.6.6.5 Beispiel 1

Im Projekt SmallWiki 1.0.1 (Gambas Software-Farm) werden die Login-Daten in einer Datei *passwd* im Root-Verzeichnis hinterlegt und mit diesem Quelltext eine neue (Admin-)Session erzeugt:

```
If Request["login"] And If Request["password"] Then

'-- Eine bestehende Session wird beendet
    Session.Abandon()
    For Each sLine In Split(File.Load(Root &/ "passwd"), "\n")
      iPos = InStr(sLine, ": ")
      If iPos = 0 Then Continue
      sLogin = Trim(Left(sLine, iPos - 1))
      sPasswd = Trim(Mid$(sLine, iPos + 2))
      If sLogin = Request["login"] And If sPasswd = Request["password"] Then
     '-- Eine neue Session wird erzeugt, Daten werden abgespeichert und ein Session-Cookie gesetzt
         Session["login"] = sLogin
         Break
      Endif
    Next
Endif
```

Das sind die (formatierten) Antwort-Kopfzeilen (Header) nach dem Login mit dem Benutzernamen und dem Passwort. Diese Kopfzeilen sehen Sie zum Beispiel im WebBrowser Firefox nach CTRL+R und anschließendem SHIFT+F9 in der Rubrik 'Konsole':

```
[1]   HTTP/1.0 200 OK
[2]   127.0.0.1: D8053000E34641AA130A6A77
[3]   Set-Cookie: GBSESSIONID=;expires=Thu, 01 Jan 1970 00:00:00 GMT;path=/
[4]   Set-Cookie: GBSESSIONID=127.0.0.1:D8053000E34641AA130A6A77;path=/;httponly
[5]   Content-type: text/html;charset=utf-8
```

Wie Sie in der Zeile [3] und im Quelltext (Session.Abadon()) sehen, wird zuerst eine bestehende Session gelöscht, da das Datum einfach in die Vergangenheit gelegt wird. Dann wird eine neue Session angelegt und ein Session-Cookie gesetzt→ in der Zeile [4]. Mit diesem Quelltext:

```
<%If Session.Id Then%>
<% Print "<br>"; %>
<div class="box">
   <% Print "SESSION-ID = "; Session.Id; %>
   <% Print "<br>SESSION-TYP = "; Session.Type; %>
   <%
     Dim sPath As String
     Dim I As Integer
     Dim sChar As String
     Dim sName As String
     Dim sKey As String
     Dim aKeys As String[]
     Dim aVariant As Variant

     If Session.Id Then
        For I = 1 To Len(Session.Id)
          sChar = Mid(Session.Id, I, 1)
          If Not IsDigit(sChar) And If Not IsLetter(sChar) Then sChar = "_"
          sName &= sChar
        Next
     Endif
     Print "<br>";
     aKeys = Session.Keys
     For Each sKey In aKeys
       If Session.Exist(sKey) Then
          Print "SESSION.KEY.VALUE.PAIR: " & sKey & " => " & Session[sKey];
       Endif
     Next
     If Session.Type = "sqlite" Then
        Print "<br>SESSION-FILEPATH = "; Session.Path &/ sName & ".db";
     Else
        Print "<br>SESSION-FILEPATH = "; Session.Path &/ sName;
     Endif
   %>
   <% Print "<br>SESSION-SIZE = "; Session.Size; " Byte"; %>
   <% Print "<br>SESSION-TIMEOUT = "; Session.TimeOut; " s"; %>
   <%
     Print "<br>SESSION-COOKIE-PATH = "; Application.Root
     Print "<br>SESSION-COOKIE-PATH = "; CGI["SCRIPT_NAME"]
     Print "<br>SESSION-COOKIE-PATH = "; Session.CookiePath

     If Not Session.Prefix Then
        Print "<br>SESSION-PREFIX = Prefix is not set.";
     Else
        Print "<br>SESSION-PREFIX = "; Session.Prefix;
     Endif
   %>
</div>

<% Endif %>
```

wurden diese erweiterten Eigenschaften der Session ausgelesen:

```
[1]  SESSION-ID = 127.0.0.1:C8319E3CF5607465BA28B531
[2]  SESSION-TYP = sqlite
[3]  SESSION.KEY.VALUE.PAIR: login => admin
[4]  SESSION-FILEPATH = /tmp/gambas.1000/session/127_0_0_1_C8319E3CF5607465BA28B531.db
[5]  SESSION-SIZE = 20480 Byte
[6]  SESSION-TIMEOUT = 86400 s
[7]  SESSION-COOKIE-PATH =
[8]  SESSION-COOKIE-PATH = /.
[9]  SESSION-COOKIE-PATH =
[10] SESSION-PREFIX = Prefix is not set.
```

Interessant sind die beiden Zeilen 2 und 4, denn sie deuten darauf hin, das die Sitzungsdaten in einer (temporären) SQLite-Datenbank (mit den zwei Tabellen: config und values) gespeichert werden. Die unterschiedlichen Ausgaben in den Zeilen [7] bis [9] sind korrekt, denn

- Application.Root normalisiert CGI["SCRIPT_NAME"] und gibt "" zurück, wenn sein Wert "./" oder "/" ist.
- Wenn Session.CookiePath leer ist, so wird der Inhalt von CGI["SCRIPT_NAME"] verwendet, mit der Ausnahme, dass "/." in diesem Fall durch "/" ersetzt wird.

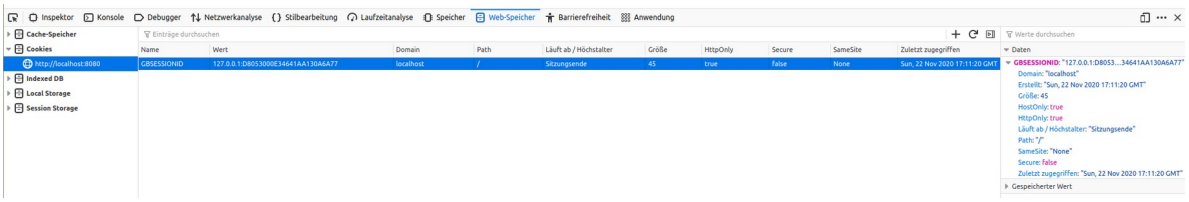

Abbildung 24.6.6.5.1: Ansicht der Cookie-Eigenschaften im Firefox – Aufruf mit F12

Die Anzeige der (Session-)Cookie-Eigenschaften im Firefox zeigt vor Allem im rechten Abschnitt interessante Details.

24.6.7 Beispiel 2

Das folgende Beispiel ist sehr einfach gehalten. Das berechtigte Interesse des Betreibers der Website besteht darin, einem angemeldeten Administrator mit dem Benutzer-Namen 'admin' eine andere Webseite anzuzeigen als einem Benutzer der Webseite mit dem Benutzernamen 'user1'.

Hier ein Ausschnitt aus dem Quelltext für das Beispiel:

```
Public Sub webbtnLogin_Click()

    If Not Trim(webtxbUserName.Text) Then
       Message.Warning("Attention!<br>The username is empty.")
       webtxbUserName.SetFocus(True)
    Else
       If webtxbUserName.Text = "user1" Then
          Session["user1"] = webtxbUserName.Text
          WebForm.Startup = "WebFormMain"
          WebformLogin.Reload()
       Else If webtxbUserName.Text = "admin" Then
          Session["admin"] = webtxbUserName.Text
          WebForm.Startup = "WebFormMain"
          WebformLogin.Reload()
       Else
          webtxbUserName.SetFocus(True)
       Endif
    Endif

End
```

So zeigt sich die Login-Seite in einem Webbrowser:

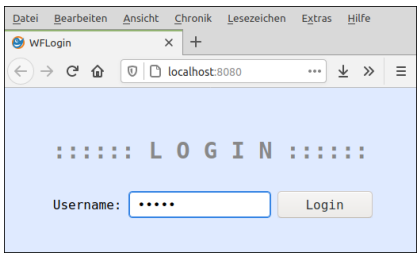

Abbildung 24.6.7.1: Login-Seite

Nach einem erfolgreichen Login wird eine weitere Webseite geöffnet. Deren Inhalt ist abhängig von der erzeugten Session:

```
Public Sub WebForm_Open()

  WebTimer1.Start()

  If Session["admin"] Then
     WebHtml1.Text = "<h3>Hello " & Session["admin"] & "!</h3>"
     WebHtml1.Text &= "<br><p>Your service starts now ...</p>"
  Endif
```

```
  If Session["user1"] Then
     WebHtml1.Text = "<h3>Hello " & Session["user1"] & "!</h3>"
     WebHtml1.Text &= "<br><p>Note: The course 'Web Programming' starts next Friday at 14:30 (UTC) in the
C-Building in room 123.</p><br>"
  Endif

End

Public Sub WebTimer1_Timer()

  If Session["admin"] Then weblblTime.Text = "Current time: " & Format(Now(), "hh:nn:ss")
  If Session["user1"] Then weblblTime.Text = "Current time: " & Format(DateAdd(Now(), gb.Second, -3600),
"dd.mm.yyyy | hh:nn:ss (UTC)")

End


Public Sub webbtnLogout_Click()
  Session.Abandon()
  WebformMain.Reload()
End
```

Hinweis:

Die Methode Session.Abadon() beendet die aktuelle Sitzung mit dem Logout. Der gesetzte Session-Cookie wird gelöscht. Die Session-Datei wie 127_0_0_1_724EC44AC512F3B5BAAC948C.db im Pfad /tmp/gambas.1000/session dagegen existiert noch, da ihre (Standard-)Lebensdauer 1 Tag beträgt.

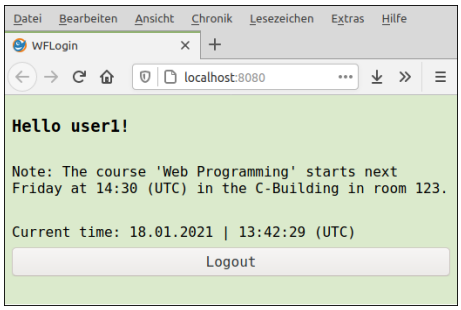

Abbildung 24.6.7.2: User-Seite

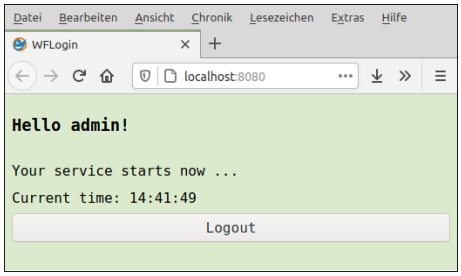

Abbildung 24.6.7.3: Admin-Seite

In der Praxis werden die zulässigen Benutzer oft über ein Web-Formular zur Registrierung auf der Website erfasst. Die erfassten Daten (Login-Name und Passwort) werden häufig in einer Datenbank in einer Datenbank-Tabelle gespeichert und mit weiteren Daten ergänzt, die auf der Registrierungsseite wieder mit einem Formular abgefragt werden.

Abbildung 24.6.7.4: Webformular zur Anmeldung oder zur Registrierung