



# **Verwenden der Application- und -Session-Objekte in E-Commerce- Anwendungen**

Woche  
1

In der heutigen Lektion setzen wir unsere Besprechung der Programmierung von Active Server Pages fort. Der Schwerpunkt dieser Lektion liegt auf den Methoden, die Aktivitäten der Kunden zu verfolgen, die Ihre Website besuchen. Die Möglichkeit Kunden zu verfolgen, erlaubt es Ihnen auf den Kunden zugeschnittene Inhalte anbieten zu können. Die Möglichkeit Kundenaktivitäten zu verfolgen und Inhalte an den Kunden anzupassen ist wichtig, da dies Ihnen ermöglicht Ihren Umsatz zu steigern. Sie möchten, um einmal ein einfaches Beispiel zu nennen, unterschiedliche Werbung je nach den Interessen der einzelnen Kunden anzeigen. Wenn Sie herausgefunden haben, dass gewisse Kunden gerne die Seiten Ihrer Website besuchen, in denen Angelruten angeboten werden, dann können Sie dem Kunden automatisch mehr Werbung anzeigen, die sich auf Angelruten bezieht.

Heute werden Sie lernen:

- ▶ wie man Cookies im Browser des Kunden hinzufügt, um den Kunden automatisch identifizieren zu können; und zwar jederzeit, wenn er zu Ihrer Website zurückkehrt;
- ▶ wie man Session- und Application-Variablen einsetzt, um dauerhafte Informationen zu speichern;
- ▶ wie man die Global.asa-Datei einsetzt, um herauszufinden, wann die Kunden zum ersten Mal auf Ihre Website kommen und wann sie sie wieder verlassen.

## Kundeninformationen mit Hilfe von Cookies gewinnen

Cookies haben in letzter Zeit immer wieder die Aufmerksamkeit der Medien auf sich gezogen, da man Angst davor hat, dass sie eine Bedrohung für die Privatsphäre darstellen. Sie können ein Cookie verwenden, um Informationen auf einem PC eines Kunden zu speichern, sobald der Kunde Ihre Website besucht. Mithilfe dieser Information können Sie dann den Kunden identifizieren, wenn er das nächste Mal Ihre Website besucht.

Cookies sind von Netscape entwickelt worden, um das spürbare Defizit in der Interaktion zwischen Webserver und Webbrowser zu beheben. Ohne Cookies ist die Interaktion zwischen Webbrowser und Webserver zustandslos. Sie können denselben Benutzer auf Ihrer Website nicht mehr identifizieren, wenn er von einer Seite auf die nächste geht.



Woher kommt der Begriff »Cookie«? Lou Montulli, derjenige, der die Original-Cookie-Spezifikation für Netscape geschrieben hat, erklärt es folgendermaßen: »Ein Cookie ist ein weithin bekannter Fachbegriff der Computerwissenschaften. Dieser beschreibt ein abgeschlossenes Daten-

paket, das als Vermittler gespeichert wird. Der Begriff beschreibt die Verwendung von Cookies genau, es handelt sich dabei aber um einen Begriff, der nur in Computerkreisen bekannt ist.«

Die zustandslose Natur der Interaktion des Webserver und des Browsers bringt etliche Probleme für den Entwickler von Websites mit sich. Stellen Sie sich zum Beispiel vor, dass Sie einen bestimmten Bereich auf Ihrer Website eingerichtet haben, der Inhalte enthält, die nur für registrierte Besucher zugänglich sind. Ohne den Einsatz von Cookies ist schwer herauszufinden, ob es sich bei einem Benutzer um ein registriertes Mitglied handelt. Wenn der Benutzer sich auf einer Seite anmeldet, ist schwer zu ermitteln, ob es sich auf einer anderen Seite um den gleichen Benutzer handelt.



Eine gute Quelle, um Informationen über Cookies zu erhalten, ist die Cookie Central Website, die Sie unter folgender Internetadresse finden:  
<http://www.cookiecentral.com>.

Es gibt zwei Arten von Cookies: Session (Sitzungs-) Cookies und dauerhafte Cookies. Session Cookies werden im Speicher gespeichert. Sie bleiben nur solange auf dem Computer des Kunden, wie er Ihre Website besucht.

Ein dauerhafter Cookie auf der anderen Seite kann viele Monate, ja sogar Jahre dort gespeichert sein. Dauerhafte Cookies werden in einer Textdatei auf dem Computer des Kunden gespeichert. Diese Textdatei wird auf Windows-Computern »Cookie Datei« und auf Macintosh-Computern »Magic Cookie Datei« genannt.

Der Netscape Navigator und der Internet Explorer speichern dauerhafte Cookies ein wenig anders. Netscape speichert alle Cookies einer Website in einer Datei mit dem Namen Cookies.txt. Diese Datei finden Sie in dem Verzeichnis \Netscape oder \Netscape\User\Benutzername. Hier zum Beispiel die Inhalte der Cookie Datei auf meinem Computer:

```
# Netscape HTTP Cookie File
# http://www.netscape.com/newsref/std/cookie_spec.html
# This is a generated file! Do not edit.
.superexpert.com    TRUE    /    FALSE    965026643    u    steve
.superexpert.com    TRUE    /    FALSE    965026643    p    secret
www.webtrends.com   FALSE   /    FALSE    1293753685    WEBTRENDS    4MNF9Z98A
.flycast.com        TRUE    /    FALSE    1293753600    atf    1_4880095465
.doubleclick.net    TRUE    /    FALSE    1920499052    id    d6685383
```

Wie Sie sehen, enthält die Cookie-Datei fünf Cookies. Die ersten beiden sind von der superexpert-Website erstellt worden. Der erste Cookie heißt "u" (was für das englische Wort user, zu deutsch Benutzer, steht) und hat den Wert "steve". Der zweite Cookie heißt "p" (was für password, zu deutsch Passwort steht) und er beinhaltet mein Geheimwort bei supertext (na ja, nicht wirklich). Meine Cookie-Datei enthält auch

Cookies, die von Webtrends (ein Unternehmen, das ein populäres Loganalyse-Werkzeug für den Internet Information Server herstellt) und von den beiden Bannervermarktern Flycast und DoubleClick angelegt wurden.

Der Microsoft Internet Explorer erstellt eine getrennte Cookie-Datei für jede Website. Alle diese Dateien sind im Verzeichnis /Windows/Cookies gespeichert. Ich habe auf meinem Computer zum Beispiel eine Cookie-Datei mit dem Namen `administrator@amazon.txt`, die von der Amazon-Website erstellt worden ist.

Es ist wichtig zu wissen, dass eine Website nur die Cookies lesen kann, die sie selbst erstellt hat. Wenn Sie zum Beispiel beide Websites, die von Amazon und superexpert, besuchen und beide Seiten einen Cookie zu Ihrem Computer hinzufügen, kann Amazon nur seine eigenen Cookies lesen und nicht die von superexpert. Wenn Sie also einen Cookie zum Computer des Kunden hinzufügen, können nur Sie oder der Kunde den Inhalt des Cookies sehen.



Bannervermarkter wie Flycast und DoubleClick können diese Regel, dass Cookies nur von der Website gelesen werden können, die sie erstellt hat, umgehen. Dazu benutzen sie einen Trick. Wenn eine Website einen Werblock eines dieser Netzdienste anzeigt, wird die Werbung eigentlich vom Server des Dienstes geholt. Deshalb kann dieser Dienst ein Cookie von jeder Website aus, die seine Werbung anzeigt, hinterlegen und lesen. Das heißt, dass diese Dienste Besucher von einer Website zur nächsten verfolgen können.

Es ist ebenfalls wichtig zu wissen, dass nicht alle Browser Cookies unterstützen. Es gibt einige Gründe, warum Browser Cookies nicht unterstützen sollten. Zum einen mögen einige Leute aus Sorge um ihre Privatsphäre keine Cookies und haben daher Cookies auf ihrem Browser ausgeschaltet. Zum anderen haben Cookie-Dateien die Eigenart, aus dem einen oder anderen Grund zerstört zu werden.

Und schließlich, obwohl Netscape Navigator seit Version 1.0 Cookies unterstützt, gibt es aus unerfindlichen Gründen immer noch Browser, die keine Cookies unterstützen.

Sie sollten nicht davon ausgehen, dass ein Kunde Cookies auf seinem Browser aktiviert hat. Es gibt zum Beispiel eine vollkommen legitime Verwendung von Cookies, nämlich einen Kunden automatisch auf Ihre Website anzumelden. Wenn Sie dies jedoch tun, sollten Sie die Möglichkeit, sich anzumelden, auch für Benutzer einbauen, die Cookies nicht aktiviert haben.

## Cookies hinzufügen

Sie können ein Cookie im Browser des Kunden hinzufügen, indem Sie die Cookies-Sammlung des `Response`-Objekts verwenden. Stellen Sie sich zum Beispiel vor, dass Sie einen Cookie mit Namen `customerName` (Kundenname), der den Namen des Kunden beinhaltet, hinzufügen wollen. Um diesen Cookie hinzuzufügen, würden Sie die folgenden Befehle benutzen:

```
Response.Cookies( "customerName" ) = "Ruth Johnson"
```

Dieser Befehl fügt einen Cookie mit dem Namen "customerName" mit dem Wert "Ruth Johnson" hinzu. Bei dem so erstellten Cookie handelt es sich um einen Session Cookie. Er bleibt nur solange, wie der Kunde auf Ihrer Website ist.

Um einen dauerhaften Cookie zu erstellen, müssen Sie das Datum einfügen, an dem der Cookie auslaufen soll. Dazu verwenden Sie die `Expires`-(Auslaufen)Eigenschaft der Cookies-Aufzählung. Die folgenden zwei Befehle erstellen einen Cookie, der bis zum 4. Juli 2002 erhalten bleibt.

```
Response.Cookies( "customerName" ) = "Ruth Johnson"  
Response.Cookies( "customerName" ).Expires = "July 4, 2002"
```

Wenn Sie Cookies erstellen, müssen Sie sie erstellen, bevor ein Inhalt zum Browser gesendet wird. Ansonsten werden Sie die folgende Fehlermeldung erhalten:

Header Error

The HTTP headers are already written to the client browser. Any HTTP header modifications must be made before writing page content.

Wenn Sie diese Einschränkung umgehen wollen, können Sie Ihre ASP-Seite zwischenspeichern. Wenn Sie eine ASP-Seite zwischenspeichern, wird sie nicht sofort zum Browser gesendet. Sie bleibt im Speicher, bis die ganze Seite aufgebaut wurde. Um eine ASP-Seite zwischenzuspeichern, fügen Sie den folgenden Befehl am Anfang der Seite ein:

```
<% Response.Buffer = TRUE %>
```



Der Internet Information Server 5.0 speichert alle Seiten standardmäßig zwischen. Der Personal Web Server und die Versionen des Internet Information Servers vor der Version 5.0 speichern den Inhalt einer Seite nicht zwischen, es sei denn, dass die Eigenschaft explizit gesetzt wird.

Sie können jeden beliebigen Inhalt in einem Cookie speichern. Sie sollten jedoch einige Einschränkungen bei Cookies beachten. Entsprechend der originalen Cookie-Spezifikation (siehe [http://home.netscape.com/newsref/std/cookie\\_spec.html](http://home.netscape.com/newsref/std/cookie_spec.html)), kann ein Computer nur maximal 300 Cookies von allen Websites speichern. Darüber hinaus kann eine einzige Website nicht mehr als 20 Cookies zum Computer des Kunden hin-

zufügen. Und schließlich kann ein Cookie nur Daten von maximal 4 KB umfassen. Diese Begrenzung bezieht sich auf die Kombination der Länge des Namens des Cookies und die Größe der im Cookie enthaltenen Daten.

### Lesen der Cookies vom Kunden-Browser

Sie können einen Cookie, den Sie auf dem Computer des Kunden abgelegt haben lesen, indem Sie die `Cookies`-Aufzählung des `Request`-Objekts verwenden. Um zum Beispiel einen Cookie mit dem Namen `username` (Benutzername) zu holen und ihn der lokalen Variablen mit Namen `username` zuzuweisen, würden Sie den folgenden Befehl benutzen:

```
username = Request.Cookies( "username" )
```

Da es sich bei der `Cookies`-Sammlung um eine Sammlung des `Request`-Objekts handelt, können Sie auch mit folgendem Befehl arbeiten:

```
username = Request( "username" )
```

Wenn es jedoch einen URL-Parameter oder eine Formularvariable mit Namen `username` gibt, wird der vorangegangene Befehl den Wert des URL-Parameters oder der Formularvariablen zurückgeben und nicht den des Cookies. Wenn Sie das Objekt nicht ausdrücklich spezifizieren, wenn Sie mit dem `Request`-Objekt arbeiten, werden die Aufzählungen in der folgenden Reihenfolge durchsucht:

1. `QueryString`
2. `Form`
3. `Cookies`
4. `ClientCertificates`
5. `ServerVariables`

Sie können alle Cookies anzeigen, die von Ihrer Website gesetzt wurden, indem Sie die Inhalte der `Cookies`-Sammlung durchlaufen. Die ASP-Seite in Listing 3.1 zeigt zum Beispiel alle Cookies an, die es auf dem Computer des Kunden gibt.

#### **Listing 3.1: Ausgabe aller Cookies**

---

```
1 <HTML>
2 <HEAD><TITLE>All Cookies</TITLE></HEAD>
3 <BODY>
4
5 <%
6 FOR EACH cookie IN Request.Cookies
```

```
7 Response.Write cookie & "=" & Request.Cookies( cookie ) & "<BR>"
8 NEXT
9 %>
10
11 </BODY>
12 </HTML>
```



Eine VBScript FOR...EACH-Schleife wird verwendet, um durch die Inhalte der Cookies-Sammlung des Request-Objekts zu laufen. Der Name und der Wert von jedem Cookie wird angezeigt.

## Kundeninformationen mit Hilfe von Session-Variablen gewinnen

Sie können Session-Variablen als eine andere Methode verwenden, um Kundeninformationen zu verfolgen, wenn ein Kunde von einer Seite zur nächsten auf Ihrer Website wechselt. Session-Variablen sind eng mit Cookies verbunden. Tatsächlich sind Session-Variablen sogar auf Cookies angewiesen.

Wenn Sie entweder den Personal Web Server oder den Microsoft Internet Information Server verwenden, fügt der Webserver automatisch einen besonderen Cookie zum Browser jedes Kunden hinzu. Dieser Cookie heißt ASPSessionID-Cookie (wenn er zum Computer des Kunden hinzugefügt wird, werden aus Sicherheitsgründen andere zufällig erzeugte Zeichen zum Namen des Cookies hinzugefügt).

Der Webserver benutzt den ASPSessionID-Cookie, um Session-Variablen mit bestimmten Benutzern zu assoziieren. Session-Variablen werden im Speicher des Webserver gespeichert. Sie können eine Session-Variable verwenden, um jede Art von Informationen einschließlich Text, Nummern, Arrays und sogar ActiveX-Komponenten zu speichern.

Bevor Sie Session-Variablen verwenden, sollten Sie jedoch davor gewarnt werden, dass sie einige derselben Nachteile wie Cookies haben. Wenn der Kunde mit einem Browser arbeitet, der keine Cookies unterstützt, kann der Webserver das ASPSessionID-Cookie nicht erstellen. Ohne das ASPSessionID-Cookie können Session-Variablen nicht mit dem Kunden assoziiert werden, wenn er sich zwischen den Seiten bewegt. Daher ist es empfehlenswert, Session-Variablen so oft wie möglich zu vermeiden.



Die Verwendung von Session-Variablen in Ihrer ASP-Anwendung kann die Skalierbarkeit ihrer Anwendung beeinträchtigen. Jede Session-Variable benötigt Speicherplatz auf dem Server. Außerdem erschwert die Verwendung von Session-Variablen es, mehrere Webserver für eine Website zu verwenden (eine Server Farm), da die Session-Variablen auf einem individuellen Server erstellt werden.

Um eine Session-Variable zu erstellen, verwenden Sie das Session-Objekt. Die ASP-Seite in Listing 3.2 zum Beispiel erstellt eine Variable mit dem Namen "favoriteColor" (bevorzugte Farbe) und mit dem Wert "blue" (blau).

### **Listing 3.2: Anlegen einer Session Variablen**

```
1 <HTML>
2 <HEAD><BODY><TITLE>Session Variable</TITLE></HEAD>
3 <BODY>
4
5 <% Session( "favoriteColor" ) = "blue" %>
6
7 </BODY>
8 </HTML>
```



Die Session-Variable wird in Zeile 5 erstellt. Sie werden schnell feststellen, dass eine Session-Variable, im Gegensatz zu einem Cookie, überall auf einer ASP-Seite erstellt werden kann. Anders als bei einem Cookie müssen Sie eine Session-Variable nicht unbedingt erstellen, bevor der Inhalt an den Browser gesendet wird.

Nachdem die favoriteColor Session-Variable erstellt und ihr ein Wert zugewiesen wurde, behält sie diesen Wert während der ganzen Zeit, in der der Benutzer Ihre Website besucht. Die favoriteColor Session-Variable wird mit einem bestimmten Benutzer assoziiert, indem der ASPSessionID-Cookie verwendet wird.

Um eine Session-Variable abzurufen, nachdem sie erstellt worden ist, verwenden Sie ebenfalls das Session-Objekt. Die ASP-Seite in Listing 3.3 zeigt den Wert der in Listing 3.2 erstellten favoriteColor Session-Variablen an.

### **Listing 3.3: Anzeige einer Session-Variablen**

```
1 <HTML>
2 <HEAD><TITLE>Session Variable</TITLE></HEAD>
3 <BODY>
4
5 Your favorite color is <%=Session( "favoriteColor" )%>
6
7 </BODY>
8 </HTML>
```



Die Session-Variable wird in Zeile 5 angezeigt. Beachten Sie, dass der Session-Variablen kein Wert auf dieser Seite zugewiesen wird. Wenn die ASP-Seite in Listing 3.2 vor der ASP-Seite in Listing 3.3 aufgerufen wird, hat die `favoriteColor` Session-Variable einen Wert.

Es ist wichtig zu verstehen, dass die Session-Variablen bezogen auf bestimmte Benutzer erstellt werden. Stellen Sie sich vor, Ruth besucht Ihre Website und ruft eine Seite auf, die den Wert `blue` (blau) der Session-Variablen mit Namen `favoriteColor` zuordnet. Stellen Sie sich weiter vor, dass Andrew Ihre Website besucht und eine Seite aufruft, die den Wert `red` (rot) zu der Session-Variablen mit Namen `favoriteColor` zuordnet. Auch nachdem Andrew seine Seite aufgebaut hat, ändert sich der Wert der `favoriteColor`-Variablen für Ruth nicht. Jeder Besucher hat seine eigene Menge von Session-Variablen, die ihm zugeordnet sind.

Session-Variablen bleiben bestehen, bis ein Benutzer Ihre Website verlässt. Wie merkt der Webserver, wann das passiert? Normalerweise geht der Webserver davon aus, dass ein Benutzer, der länger als 20 Minuten keine Seite mehr aufgerufen hat, die Website verlassen hat. Dieses vorgegebene Verhalten können Sie mit Hilfe der `Timeout`-Eigenschaft des `Session`-Objekts verändern.

Wenn Sie zum Beispiel eine Website haben, die lange Produktbeschreibungen enthält, für die man einige Zeit braucht um sie zu lesen, dann möchten Sie vielleicht die `Timeout`-Eigenschaft abändern und zum Beispiel auf 60 Minuten hochsetzen. Dazu müssen Sie den folgenden Befehl an den Anfang der Seite setzen:

```
Session.Timeout = 60
```

Sie legen den Wert der `Timeout`-Eigenschaft in Minuten fest. Der neue `Timeout`-Wert wird für den Benutzer für den Rest der Benutzersitzung verwendet.

## Speichern von Arrays in Session-Variablen

Eine allgemeine Verwendung von Session-Variablen ist das Speichern des Warenkorbs des Kunden. Sie können einen Warenkorb erstellen, indem Sie ein Array zu der Session-Variablen hinzufügen. Die Elemente dieses Arrays beinhalten die einzelnen Produkte, die der Kunde in seinen Warenkorb eingefügt hat.

Das Script in Listing 3.4 beschreibt, wie Sie ein Array erstellen, Werte zweier seiner Elemente zuweisen und dann das Array einer Session-Variablen zuweisen.

### Listing 3.4: Anlegen eines Session-Arrays

---

```

1 <%
2 DIM ShoppingCart( 20 )
3 ShoppingCart( 0 ) = "toothpaste"
4 ShoppingCart( 1 ) = "comb"
5 Session( "ShoppingCart" ) = ShoppingCart
6 %>

```



Das ShoppingCart(Warenkorb)-Array wird in Zeile 2 erstellt. Das Array hat 20 Elemente. Als nächstes werden zwei der Array-Elemente ein Wert zugeordnet. Schließlich wird in Zeile 5, das Array der Session-Variablen mit dem Namen ShoppingCart zugewiesen.

Nachdem ein Array zu einer Session-Variablen hinzugefügt wurde, können Sie jedes Element eines Arrays unter Verwendung des Index anzeigen. Zum Beispiel zeigt der folgende Befehl das Element des Session-Arrays mit dem Index 1 an.

```
Response.Write Session( "ShoppingCart" )( 1 )
```

Wenn das Session-Array mit dem Script in Listing 3.4 erstellt worden wäre, würde das vorausgehende Programm den Wert "comb" (Kamm) anzeigen.

Sie können jedoch den Wert eines Elements im Session-Array nicht direkt verändern. Um einen Wert in einem Session-Array zu ändern, müssen Sie das Session-Array erst einem normalen VBScript-Array zuweisen. Dann können Sie die Änderung vornehmen und das Array wieder der Session-Variablen zuweisen.

Das Listing 3.5 zeigt zum Beispiel, wie man den Wert des zweiten Warenkorbelements von comb (Kamm) zu toothbrush (Zahnbürste) verändern kann.

### Listing 3.5: Ändern des Wertes in einem Session-Array

---

```

1 <%
2 ShoppingCart = Session( "ShoppingCart" )
3 ShoppingCart( 1 ) = "toothbrush"
4 Session( "ShoppingCart" ) = ShoppingCart
5 %>

```

Sie sind vielleicht versucht, die Änderung des Werts in einem Session-Array direkt vorzunehmen. Sie könnten zum Beispiel versuchen, den folgenden Befehl anzuwenden:

```
Session( "ShoppingCart" )( 1 ) = "toothbrush"
```

Dieser Befehl würde zwar keinen Fehler hervorrufen, hätte aber auch keine Wirkung. Sie können den Wert eines Session-Arrays direkt nicht ändern.

## Verfolgen einer Session mit der SessionID

Das `Session`-Objekt hat eine nützliche Eigenschaft, um Benutzer eindeutig zu identifizieren: die `SessionID`-Eigenschaft. Jedem Besucher Ihrer Website wird automatisch eine Nummer zugeordnet. Diese einmalige Nummer können Sie mit der `SessionID`-Eigenschaft abrufen.

Die ASP-Seite in Listing 3.6 zeigt zum Beispiel den Wert des `SessionID` für eine Person an, die die Seite aufgerufen hat.

### **Listing 3.6: Anzeigen der SessionID Eigenschaft**

```
1 <HTML>
2 <HEAD><BODY><TITLE>Session ID</TITLE></HEAD>
3 <BODY>
4
5 Your unique Session ID is <%=Session.SessionID%>
6
7 </BODY>
8 </HTML>
```

Ein `SessionID` ist einmalig für jeden Benutzer, der aktuell Ihre Website besucht. Dieselbe `SessionID` kann jedoch wieder verwendet werden, nachdem Ihr Webserver neu gestartet wurde. Das würde bedeuten, dass Sie nicht versuchen sollten, denselben Benutzer über diese Zeit hinaus mit Hilfe der `SessionID` zu verfolgen.

## Beenden einer Benutzersitzung

Standardmässig endet eine Benutzersitzung, nachdem der Benutzer über 20 Minuten lang keine Seite Ihrer Website mehr aufgerufen hat. Sie können eine Sitzung aber auch früher beenden, indem Sie die `Abandon`-Methode des `Session`-Objekts aufrufen. Mit dem Aufrufen dieser `Abandon`-Methode werden alle `Session`-Variablen, die mit dem Benutzer assoziiert sind, der die Seite aufgerufen hat, aus dem Speicher gelöscht.

Nachdem Sie die `Abandon`-Methode aufgerufen haben, endet die Benutzersitzung dann, wenn die aktuelle Seite komplett ausgeführt worden ist. Das bedeutet, dass alle Benutzer-`Session`-Variablen ihren Wert behalten, bis die Seite vollständig ausgeführt wurde. Außerdem behält die `SessionID`-Variable auf dieser Seite ihren Wert.

Betrachten Sie zum Beispiel die ASP-Seite in Listing 3.7.

### Listing 3.7: Aufrufen der Abandon-Methode

```

1 <HTML>
2 <HEAD><TITLE>Session Abandon</TITLE></HEAD>
3 <BODY>
4
5 <%
6 Session( "myVar" ) = "Hello World!"
7 %>
8 <p>The value of myVar is: <%=Session( "myVar" )%>
9 <%
10 Session.Abandon
11 %>
12 <p>The value of myVar is: <%=Session( "myVar" )%>
13
14 </BODY>
15 </HTML>

```



In Zeile 6 wird der Session-Variablen mit Namen "myVar" der Wert "Hello World!" zugewiesen. Diese Session-Variable wird in Zeile 8 angezeigt. Dann wird in Zeile 10 die Abandon-Methode des Session-Objekts aufgerufen. In Zeile 12 wird die "myVar" Session-Variable erneut angezeigt.

Die ASP-Seite in Listing 3.7 zeigt "Hello World!" zweimal an. Selbst wenn die Abandon-Methode aufgerufen wird, bevor die Session-Variable in Zeile 12 angezeigt wird, behält die Variable ihren Wert. Die Abandon-Methode beendet die Session nicht, solange die ganze Seite nicht vollständig übertragen wurde.

Die Abandon-Methode wird meistens zum Erstellen einer Abmelde-Seite verwendet. Sie können zum Beispiel den Benutzernamen des Kunden und das Passwort in Session-Variablen speichern, um den Kunden auf jeder Seite identifizieren zu können. Wenn der Kunde Ihre Website verlassen möchte, kann er zu einer Seite gehen, die die Abandon-Methode aufruft, um die Benutzersession zu beenden und den Benutzernamen und das Passwort aus dem Speicher zu löschen.

## Verwendung von Application-Variablen

Wie Session-Variablen können auch Application-Variablen dazu verwendet werden, Kundeninformationen über mehrere Seiten zu speichern. Im Gegensatz zu Session-Variablen werden Application-Variablen nicht mit einem besonderen Benutzer assoziiert. Die in der Application-Variablen gespeicherten Werte können jedem Kunden zugewiesen und von jedem Kunden Ihrer Website abgerufen werden.

Um eine Application-Variable zu erstellen, verwenden Sie das Application-Objekt. Zum Erstellen einer Application-Variablen namens "myVar", würden Sie zum Beispiel den folgenden Befehl verwenden:

```
Application( "myVar" ) = "Hello World"
```

Um eine Application-Variable abzurufen, können Sie ebenfalls das Application-Objekt verwenden. Der folgende Befehl zeigt die Inhalte der Application-Variablen namens "myVar" an:

```
Response.Write Application( "myVar" )
```

Wenn der Wert der Application-Variablen geändert wird, ändert er sich für alle Benutzer Ihrer Website. Stellen Sie sich vor, Ruth ruft eine Seite von Ihrer Website auf, die den Wert blau der Application-Variablen mit Namen `favoriteColor` zuweist. Stellen Sie sich weiter vor, Andrew würde dann eine Seite aufrufen, die den Wert red zu der Application-Variablen `favoriteColor` zuweist. Nachdem Andrew den Wert der `favoriteColor` Application-Variablen geändert hat, verändert sich der Wert für alle Benutzer. Nachdem Andrew die Seite aufgerufen hat, hat die `favoriteColor`-Variable auch für Ruth den Wert red.

Da dieselbe Application-Variable von verschiedenen Benutzern Ihrer Website verändert werden kann, können Konflikte auftreten. Ein übliche Verwendung von Application-Variablen ist es, herauszufinden, wie oft eine Seite besucht wurde. Die ASP-Seite in Listing 3.8 zeigt einen einfachen Zugriffszähler (siehe Abbildung 3.1).

### **Listing 3.8: Ein einfacher Zugriffszähler**

```
1 <%
2 Application( "counter" ) = Application( "counter" ) + 1
3 %>
4 <HTML>
5 <HEAD><TITLE>Page Counter</TITLE></HEAD>
6 <BODY>
7
8 This page has been viewed
9 <%=Application( "counter"%> times.
10
11 </BODY>
12 </HTML>
```



Die ASP-Seite in Listing 3.8 verwendet eine Application-Variable namens "counter", um zu verfolgen, wie oft eine Seite besucht wird. Die Application-Variable wird in Zeile 2 erhöht. Der aktuelle Wert der Application-Variablen wird in Zeile 9 angezeigt.

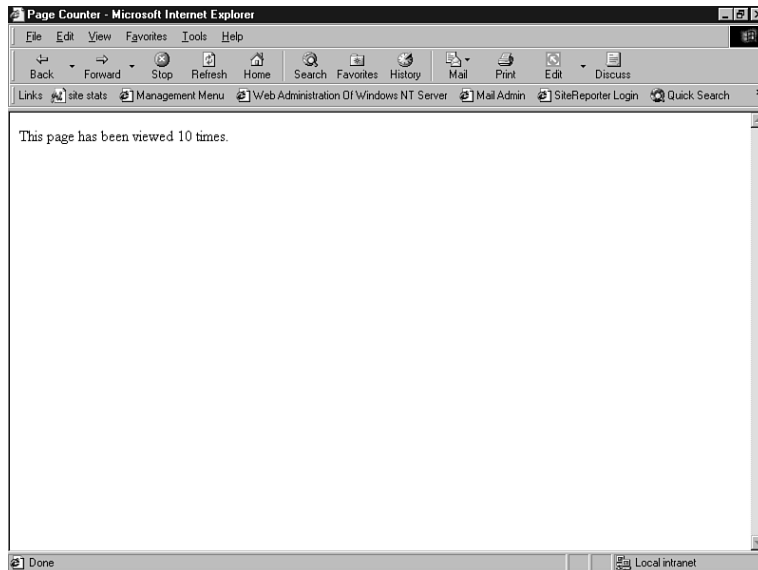


Abbildung 3.1:  
Ein einfacher  
Zugriffszähler

Mit der ASP-Seite in Listing 3.8 gibt es ein gravierendes Problem. Stellen Sie sich vor, zwei Leute rufen gleichzeitig eine Seite auf. Ruth ruft die Seite auf und die Application-Variable `counter` hat den Wert 345. Zur gleichen Zeit ruft Andrew die Seite auf und die Application-Variable hat den Wert 345. Wenn beide Besucher die Seite aufgerufen haben, hat die Application-Variable den Wert 346. Es haben jedoch zwei Personen die Seite aufgerufen, sodass der Wert eigentlich 347 betragen müsste.

Es gibt glücklicherweise eine einfache Möglichkeit, um dieses Problem zu vermeiden. Das Application-Objekt verfügt über zwei Methoden namens `Lock` und `Unlock`. Die `Lock` Methode sperrt alle Application-Variablen und verhindert, dass außer dem aktuellen Benutzer ein anderer diese liest oder verändert. Die `Unlock` Methode gibt die Application-Variablen wieder frei.

Die ASP-Seite in Listing 3.9 enthält eine verbesserte Version des Zugriffszählers:

#### **Listing 3.9: Ein besserer Zugriffszähler**

```

1  <%
2  Application.Lock
3  Application( "counter" ) = Application( "counter" ) + 1
4  Application.Unlock
5  %>
6  <HTML>
7  <HEAD><TITLE>Page Counter</TITLE></HEAD>

```

```
8 <BODY>
9
10 This page has been viewed
11 <%=Application( "counter" )%> times.
12
13 </BODY>
14 </HTML>
```



Die ASP-Seite in Listing 3.8 ist die gleiche wie in Listing 3.9 mit dem Unterschied, dass sowohl die `Lock`- als auch die `Unlock`-Methode des `Application`-Objekts aufgerufen werden. Die `Lock`-Methode wird in Zeile 2 aufgerufen. Dies verhindert, dass ein anderer die `Application`-Variable lesen oder ändern kann. Nachdem die `Application`-Variable in Zeile 3 modifiziert wurde, wird in Zeile 4 die `Unlock`-Methode aufgerufen, um die `Application`-Variablen wieder freizugeben.

Es ist wichtig zu wissen, dass das Aufrufen der `Lock`-Methode alle gespeicherten `Application`-Variablen sperrt. Es ist nicht möglich, nur einzelne `Application`-Variablen zu sperren.

Nachdem Sie die `Lock`-Methode aufgerufen haben, bleiben alle `Application`-Variablen solange gesperrt, bis die `Unlock`-Methode aufgerufen wird oder die Seite vollständig aufgebaut ist. Das bedeutet, dass Sie nicht aus Versehen für alle Zeit die `Application`-Variablen über ein ASP-Script sperren können.

Sie sollten ebenfalls beachten, dass das Sperren von `Application`-Variablen andere Benutzer nicht daran hindert, `Application`-Variablen zu modifizieren. Wenn eine Anzahl von Benutzern alle zur gleichen Zeit versuchen eine `Application`-Variable zu ändern und jeder Benutzer eine Seite aufruft, die die `Lock`-Methode aufruft, werden alle Änderungen erfolgen. Aber die Veränderungen werden nacheinander und nicht gleichzeitig erfolgen.

## Speichern von Arrays in Application-Variablen

Eine häufige Verwendung von `Application`-Variablen ist das Speichern von häufig benötigten aber nicht so häufig modifizierten Daten. Das Abrufen von Daten kann ein langsamer Vorgang sein. Wenn sich die Daten nicht oft ändern, empfehle ich, dass Sie die Daten nur einmal abrufen und sie in einem `Application`-Array speichern. Auf diese Weise können die Verzeichnisse das nächste Mal, wenn Sie aufgerufen werden, schnell vom `Application`-Array abgerufen werden.



In der Lektion des 5. Tags werden Sie lernen, wie man derartige Daten abrufen.

Das Script in Listing 3.10 zeigt, wie Sie ein Array zu einer Application-Variablen mit Namen `myArray` hinzufügen.

#### **Listing 3.10: Anlegen eines Application-Arrays**

```
1 <%
2 DIM myArray( 10 )
3 myArray( 0 ) = "Hello World!"
4 Application( "myArray" ) = myArray
5 %>
```



Das Script in Listing 3.10 erstellt ein Array mit Namen `myArray` und weist es der Application-Variablen mit Namen `myArray` zu. Das Array wird in Zeile 2 erstellt. Danach wird der Wert in Zeile 3 einem Element des Arrays zugewiesen. Schließlich wird in Zeile 4 das lokale Array der Application-Variablen zugewiesen.

Sie können den Wert direkt von einem Application-Array abrufen und anzeigen. Der folgende Befehl zeigt den Wert des Elements des Application-Arrays mit dem Index 0 an:

```
Response.Write Application( "myArray" )( 0 )
```

Obwohl Sie den Wert des Elements, der in dem Application-Array beinhaltet ist, direkt lesen können, können Sie ihn nicht verändern. Der folgende Befehl zum Beispiel zeigt keine Wirkung:

```
Application( "myArray" )(2) = "Goodbye!"
```

Wenn Sie den Wert eines Elements in einem Application-Array verändern wollen, müssen Sie das Application-Array erst einem lokalen Array zuweisen. Das Script in Listing 3.11 zum Beispiel verändert den Wert des im Application-Array enthaltenen Elements richtig.

#### **Listing 3.11: Ändern eines Elements in einem Application-Array**

```
1 <%
2 Application.Lock
3 myArray = Application( "myArray" )
4 myArray( 0 ) = "Goodbye!"
5 Application( "myArray" ) = myArray
6 Application( "myArray" ).Unlock
7 %>
```



Das Script in Listing 3.11 verändert ein Element des Application-Arrays. In Zeile 3 wird das Application-Array mit dem Namen myArray dem lokalen Array mit demselben Namen zugewiesen. Danach wird in Zeile 4 ein Element des lokalen Arrays verändert. Schließlich wird in Zeile 5 das lokale Array wieder dem Application-Array zugewiesen.

## Löschen von Application-Variablen aus dem Speicher

Beim Erstellen von Application-Variablen sollten Sie vorsichtig sein. Application-Variablen nehmen viel Speicherplatz in Anspruch. Anders als Session-Variablen, werden Application-Variablen nicht automatisch aus dem Speicher gelöscht.

Vor der in Windows 2000 enthaltenen Version von Active Server Pages war es nicht möglich, eine Application-Variable innerhalb eines ASP-Scripts aus dem Speicher zu löschen. Application-Variablen blieben im Speicher, bis der Webdienst beendet wurde, die Global.asa-Datei geändert wurde oder Ihre ASP-Anwendung entladen wurde.

Die neue Version der Active Server Pages, die in Windows 2000 enthalten ist, umfasst zwei neue Methoden, die Sie verwenden können, um Application-Variablen aus dem Speicher zu löschen: die Remove()- und die RemoveAll()-Methode. Die Remove()-Methode löscht eine bestimmte Application-Variable aus dem Speicher. Die RemoveAll()-Methode löscht alle Application-Variablen aus dem Speicher.

Das Script in Listing 3.12 zum Beispiel erstellt zwei Application-Variablen und löscht dann eine von ihnen wieder.

### Listing 3.12: Verwenden der Remove() Methode

```
1 <%  
2 Application( "myvar1" ) = "Red"  
3 Application( "myvar2" ) = "Blue"  
4 Application.Contents.Remove( "myvar1" )  
5 %>
```



In den Zeilen 2 und 3 werden zwei Application-Variablen erstellt. In Zeile 4 wird die Remove()-Methode angewandt, um die in Zeile 2 erstellte Application-Variable zu löschen.

Um alle Application-Variablen aus dem Speicher zu löschen, können Sie die RemoveAll()-Methode verwenden. Das Script in Listing 3.13 zeigt, wie diese Methode angewandt werden kann.

### Listing 3.13: Verwenden der RemoveAll() Methode

```
1 <%
2 Application( "myvar1" ) = "Red"
3 Application( "myvar2" ) = "Blue"
4 Application.Contents.RemoveAll()
5 %>
```



In den Zeilen 2 und 3 werden zwei Application-Variablen erstellt. Wenn in Zeile 4 die RemoveAll()-Methode aufgerufen wird, werden alle Application-Variablen aus dem Speicher gelöscht, auch die beiden Application-Variablen, die in diesem Script erstellt wurden.

## Verwendung der Global.asa-Datei

In diesem Abschnitt werden Sie lernen, wie man eine spezielle Datei mit dem Namen Global.asa verwendet. Die Global.asa-Datei kann ASP-Scripts beinhalten. Anders als bei einer normalen ASP-Seite wird die Global.asa-Datei nicht dazu verwendet, um Inhalte anzuzeigen. Stattdessen wird sie verwendet, um globale Ereignisse in ihrer Anwendung zu behandeln.

Bevor Sie die Global.asa-Datei verwenden können, müssen Sie zuerst eine ASP-Anwendung erstellen. Um dies mit Ihrem Personal Web Server durchführen zu können, folgen Sie bitte den aufgeführten Schritten:

1. Starten Sie den Personal Web Manager.
2. Klicken Sie das Feld »Erweitert« an.
3. Wählen Sie Ihr Home Verzeichnis aus und klicken Sie auf Bearbeiten Eigenschaften.
4. Kreuzen Sie das Kästchen Ausführen an.
5. Starten Sie Ihren Computer neu.

Um eine ASP-Anwendung mit dem Internet Information Server zu erstellen, gehen Sie folgendermaßen vor:

1. Starten Sie den Internet Service Manager.
2. Klicken Sie mit der rechten Maustaste auf Default Website und klicken Sie auf Eigenschaften. Dies öffnet einen Eigenschaftsdialog.
3. Wählen Sie den Reiter Homeverzeichnis.

4. Im Abschnitt Anwendungs-Einstellungen klicken Sie auf Erstellen (Wenn Sie nur das Feld Löschen sehen, ist die Anwendung bereits erstellt).

Nachdem Sie die ASP-Anwendung erstellt haben, können Sie die `Global.asa`-Datei in das Verzeichnis Ihrer Anwendung hinzufügen. In der Regel fügen Sie die `Global.asa`-Datei dem Verzeichnis `wwwroot` hinzu. Sie können die `Global.asa`-Datei wie eine normale ASP-Seite mit einem Text-Editor erstellen.

In der `Global.asa`-Datei können Sie Unterprogramme erstellen, die im Falle von vier Ereignissen ausgeführt werden. Hier ist eine Liste dieser Ereignisse:

1. Das `Session_OnStart`-Ereignis – Dieses Ereignis wird ausgelöst, wenn ein Kunde Ihre Website zum ersten Mal besucht. Dieses Ereignis wird sofort ausgeführt, wenn der Kunde die erste Seite aufruft.
2. Das `Session_OnEnd`-Ereignis – Dieses Ereignis wird ausgelöst, wenn der Benutzer die Sitzung beendet. Es wird ausgeführt, wenn die Benutzersitzung ausläuft oder wenn die `Abandon()`-Methode des `Session`-Objekts aufgerufen wird.
3. Das `Application_OnStart`-Ereignis – Dieses Ereignis wird ausgelöst, wenn das erste Mal eine Seite von Ihrer Website aufgerufen wird, nachdem der Server gestartet wurde. Das Ereignis wird immer vor dem `Session_OnStart`-Ereignis ausgeführt.
4. Das `Application_OnEnd`-Ereignis – Dieses Ereignis wird ausgelöst, wenn der Server beendet wird. Es läuft immer nach dem `Session_OnEnd`-Ereignis ab.

Nehmen Sie zum Beispiel einmal an, dass Sie die Zahl der Besucher, die sich aktuell in Ihrem Store befinden, anzeigen möchten. Dazu können Sie das Ereignis `Session_OnStart`, das Ereignis `Session_OnEnd` und das `Application_OnStart`-Ereignis verwenden (siehe Listing 3.14).

---

**Listing 3.14: Zählen von Kunden**

---

```
1 <SCRIPT LANGUAGE="VBScript" RUNAT="Server">
2
3 Sub Session_OnStart
4 Application.Lock
5 Application( "customerCount" ) = Application( "customerCount" ) + 1
6 Application.Unlock
7 End Sub
8
9 Sub Session_OnEnd
10 Application.Lock
11 Application( "customerCount" ) = Application( "customerCount" ) - 1
12 Application.Unlock
13 End Sub
14
```

```
15 Sub Application_OnStart
16 Application( "customerCount" ) = 0
17 End Sub
18
19 Sub Application_OnEnd
20 End Sub
21
22 </SCRIPT>
```



Die Global.asa-Datei in Listing 3.14 verwendet drei Ereignisse. Die Zeilen 3-7 beinhalten eine Routine, die das Session\_OnStart-Ereignis behandelt. Wann immer ein Kunde Ihre Website besucht, erhöht diese Routine die aktuelle Zahl der Kunden um 1. Die Zeilen 9-13 enthalten eine Routine, die das Session\_OnEnd-Ereignis behandelt. Wenn ein Kunde die Sitzung beendet, wird die Zahl der aktuellen Kunden um 1 verringert. Schließlich wird in den Zeilen 15-17 das Application\_OnStart-Ereignis verwendet, um die customerCount-Variable vorzubelegen.

Sie sollten beachten, dass die Script-Trennzeichen <% und %> in der Global.asa-Datei nicht verwendet werden. Stattdessen sind der Anfang und das Ende des Scripts durch das HTML-Schlüsselwort HTML <SCRIPT> gekennzeichnet (siehe Zeile 1 und 22). Dem RUNAT-Attribut des <SCRIPT> Schlüsselworts wird der Wert SERVER gegeben, um anzuzeigen, dass es sich eher um ein Server-seitiges Script als um ein Client-seitiges Script handelt.

Die Global.asa-Datei in Listing 3.14 gibt keinen Inhalt aus. Um die aktuelle Kundenzahl anzuzeigen, müssen Sie die Application-Variable namens customerCount auf einer Seite anzeigen. Dies wird auf der Seite in Listing 3.15 dargestellt.

#### **Listing 3.15: Anzeige der Anzahl der Benutzer**

```
1 <HTML>
2 <HEAD><TITLE>Welcome</TITLE></HEAD>
3 <BODY>
4
5 Welcome to our store!
6 <p>There are currently
7 <%=Application( "customerCount" )%>
8 customers actively browsing our store.
9
10 </BODY>
11 </HTML>
```



Diese Seite zeigt die Zahl der aktiven Kunden an. Sie zeigt einfach den Wert der Application-Variablen mit dem Namen "customerCount" an (siehe Abbildung 3.2).

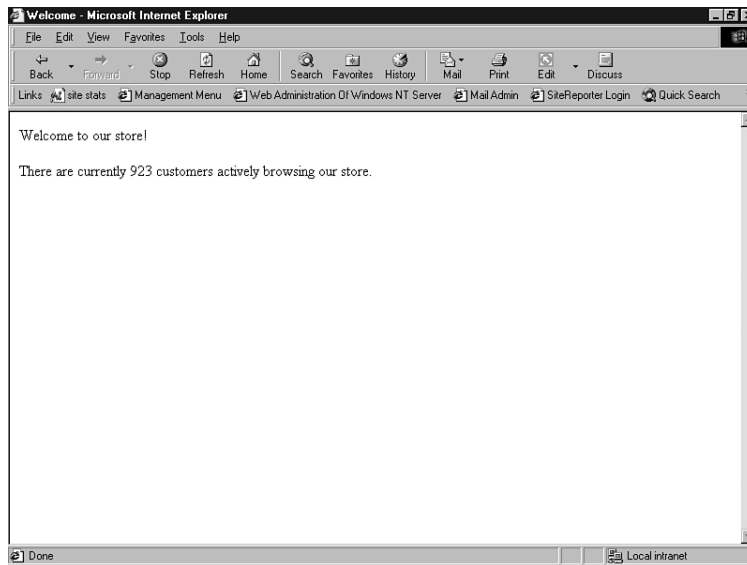


Abbildung 3.2:  
Anzeige von akti-  
ven Kunden

Sie sollten wissen, dass Sie eine große Anzahl der Standard Active Server Page-Objekte in der Global.asa-Datei nicht verwenden können. In den Application\_OnStart und Application\_OnEnd-Routinen können Sie nur die Server- und Application-Objekte verwenden. In der Session\_OnStart-Routine können Sie jedes eingebaute ASP-Objekt verwenden. In der Session\_OnEnd können Sie nur die Application-, Server- und Session-Objekte verwenden.

## Zusammenfassung

In der heutigen Lektion haben Sie mehrere Methoden kennen gelernt, Kundeninformationen zu verfolgen, wenn der Kunde von einer Seite auf die nächste auf Ihrer Website wechselt. Im ersten Abschnitt haben Sie erfahren, wie man Cookies zum Computer des Kunden hinzufügt. Sie wissen jetzt, wie man Session-Cookies und dauerhafte Cookies erstellt.

Im zweiten Abschnitt haben Sie gelernt, wie man Session-Variablen erstellt und liest. Sie wissen jetzt, wie man Session-Variablen verwendet, um beständige Informationen

über den Kunden zu speichern, wie man eine Benutzersitzung mit der `Abandon()`-Methode beendet und wie man einen Kunden mit Hilfe des `SessionID` verfolgt. Schließlich haben Sie gelernt, wie man Session-Arrays erstellt.

Im dritten Abschnitt haben Sie die Verwendung von Application-Variablen kennen gelernt. Sie haben gelernt, wie man einen einfachen Seitenzähler mit der Application-Variablen erstellt und wie man mit Application-Arrays arbeitet.

Abschließend, im letzten Abschnitt der heutigen Lektion, haben Sie gelernt, wie man die `Global.asa`-Datei verwendet, wie man Unterprogramme erstellt und wie man mit den Ereignissen `Session_OnStart`, `Session_OnEnd`, `Application_OnStart` und `Application_OnEnd` arbeitet. Sie haben auch erfahren, wie man eine `Global.asa`-Datei benutzt, um die Zahl der aktiven Kunden auf Ihrer Website anzuzeigen.

## Fragen und Antworten

### **F Sollte ich Cookies oder Session-Variablen verwenden, wenn ich eine kommerzielle Website erstelle?**

**A** *Es gibt viele erfolgreiche Websites im Internet, die es erfordern, dass Kunden Cookies aktiviert haben. Aus schmerzvoller persönlicher Erfahrung heraus kann ich sagen, dass Cookies und Session-Variablen bei einer überraschend hohen Zahl von Browsern nicht funktionieren. Wenn Sie eine Website erstellen möchten, die für möglichst viele Kunden zugänglich ist, empfehle ich Ihnen, keine Cookies oder Session-Variablen zu verwenden. Andererseits, wenn Sie schnell eine kommerzielle Website entwickeln müssen, kann die Verwendung von Session-Variablen bei der Entwicklung einer Website sehr zeitsparend sein.*

### **F Wie kann ich es vermeiden, Cookies und Session-Variablen einzusetzen?**

**A** *Wenn Sie Kundeninformationen von Kunden, die auf Ihrer Website von Seite zu Seite gehen, verfolgen möchten, können Sie anstelle von Cookies oder Session-Variablen URL-Parameter und versteckte Formularfelder verwenden. Wenn Sie zum Beispiel einen Kunden mit einer ID-Nummer verfolgen möchten, müssen Sie die ID-Nummer in jede Anfrage-Zeichenfolge und jedes HTML-Formular integrieren. Die folgende ASP-Seite beschreibt, wie Sie dies tun können:*

```
<%  
' Auslesen der Kunden ID  
cid = Request( "cid" )  
%>
```

```
<HTML>
<HEAD><TITLE>No Cookies</TITLE></HEAD>
<BODY>
<a href="nextpage.asp?cid=<%=cid%>">Next Page</a>
<p>
<FORM method="post" ACTION="nextpage.asp">
<input name="cid" type="hidden" value="<%=cid%>">
<input type="submit" value="Next Page">
</FORM>
</BODY>
</HTML>
```

*Die vorherige Seite ruft die Kunden-ID vom `Request`-Objekt ab und übergibt sie zur nächsten Seite, sowohl in Form einer Anfrage-Zeichenfolge wie auch in Form eines versteckten Formularfeldes. Beachten Sie, dass die Verwendung von `Request( "cid" )` die Kunden-ID abrufen, egal ob sie in der `QueryString` oder in der Form-Aufzählung des `Request`-Objekts enthalten ist.*

## Workshop

Die Verständnisfragen dienen dazu, Ihr Wissen bezüglich des in diesem Kapitel behandelten Stoffs zu testen. Die Antworten finden Sie in Anhang A, Antworten zu den Verständnisfragen.

### Verständnisfragen

1. Nehmen Sie an, Sie möchten ein Cookie erstellen, das länger als eine bestimmte Benutzersitzung bestehen bleibt. Welche Eigenschaft der `Cookies`-Sammlung müssen Sie einsetzen, um ein Cookie zu erstellen, das bis zu einem bestimmten Datum bestehen bleibt?
2. Angenommen, Andrew ruft eine ASP-Seite auf, die den Wert `red` (rot) einer `Session`-Variablen mit Namen `color` (Farbe) zuweist. Weiter angenommen, Ruth ruft eine ASP-Seite auf, die den Wert `blue` (blau) der `Session`-Variablen namens `color` (Farbe) zuweist. Wenn Andrew die ASP-Seite aufruft, die den Wert der `color` `Session`-Variablen ausgibt, welcher Wert wird dann angezeigt?
3. Wie können Sie alle `Session`-Variablen, die sich auf einen bestimmten Benutzer beziehen, aus dem Speicher löschen?
4. Angenommen, Andrew ruft eine ASP-Seite auf, die den Wert `red` einer `Application`-Variablen mit Namen `color` zuweist. Weiter angenommen, Ruth ruft eine

ASP-Seite auf, die den Wert `blue` der Application-Variablen namens `color` zuweist. Wenn Andrew dann eine ASP-Seite aufruft, die den Wert der `color` Application-Variablen ausgibt, welcher Wert wird dann angezeigt?

5. Welcher Fehler steckt in der folgenden `Global.asa`-Datei?

```
<%
Sub Session_OnStart
    Application.Lock
    Application( "customerCount" ) = Application( "customerCount" ) + 1
    Application.Unlock
End Sub
Sub Session_OnEnd
    Application.Lock
    Application( "customerCount" ) = Application( "customerCount" ) - 1
    Application.Unlock
End Sub
Sub Application_OnStart
    Application( "customerCount" ) = 0
End Sub
%>
```

## Übung

Erstellen Sie eine ASP-Seite, die die `SessionID` und die Eintrittszeit aller Kunden auflistet, die Ihre Website besucht haben. Dazu erstellen Sie eine `Global.asa`-Datei, um herauszufinden, wann der Kunde auf Ihrer Website ankommt, und eine ASP-Seite, um die Liste der `SessionIDs` und der Eintrittszeiten anzuzeigen.