Corso basi di dati
Mantenere dati persistenti

Gianluca Di Tomassi
Email: ditomass@dia.uniroma3.it

Università di Roma Tre

Problema ?

Internet è basata sul ***modello client/server***:

1. Il client apre un canale di comunicazione con il server e richiede la risorsa
2. Il server riceve la richiesta, trova la risorsa richiesta e la invia al client, chiudendo il canale di comunicazione

→ Trasmissione impersonale, il server non è interessato a sapere con chi ha parlato recentemente.

→ Quindi i siti Web sono stateless (senza stato)

PROBLEMA: In certe situazioni occorre mantenere lo stato

Soluzioni messe a disposizione nelle pagine ASP

Asp fornisce oggetti e collection per raggiungere l'obiettivo di mantenere lo stato dei dati:

1. Passare informazioni nella QueryString
2. La collection Cookies
3. L'oggetto Session
4. L'oggetto Application

Passare informazioni nella QueryString

RICORDA:

L'oggetto Request può elaborare informazioni prese dalla stringa d'interrogazione, se questa usa un formato con coppie nome/valore, con ogni nome e valore separato da un segno uguale (=) e ogni coppia nome/valore separata da una e commerciale (&)

Esempio: `?nome=Gianluca&cognome=DiTomassi`

NOTA: Questi dati devono essere passati mediante il metodo GET

Supponiamo di voler mantenere il nome e il cognome dell'utente connesso in tutte le pagine del nostro sito Web.

A ogni pagina deve essere passato il nome e il cognome attraverso la stringa d'interrogazione. Per assicurare ciò occorre aggiungere `?<%=Request.QueryString%>` a ogni collegamento ipertestuale su ogni pagina ASP

PROBLEMI:

- è un lavoro pesante
- Se l'utente accede ad una pagina del sito che ha precedentemente memorizzato tra i suoi bookmarks egli raggiunge la pagina senza aver passato dati persistenti
- Se lo sviluppatore dimentica di aggiungere la QueryString ad un link ipertestuale si perde lo stato
- Mantiene i dati solo mentre l'utente è nel sito

I cookie

- Sono delle piccole quantità di informazioni che sono memorizzate sul client per uno specifico periodo di tempo
- Quando un cookie è scaduto, viene rimosso automaticamente dal client
- Quando un sito Web scrive un cookie sul client, solo quel sito Web potrà leggere nuovamente il valore del cookie.

I cookie possono essere memorizzati sul client in 2 modi:

1. Utilizzare tanti cookie quante sono le informazioni da memorizzare
2. Utilizzare le chiavi

- Ogni cookie può avere un insieme di chiavi (key)
- Ogni chiave può essere usata per memorizzare un frammento di dati
- Ogni cookie può avere zero chiave o più chiavi
- L'utilizzo delle chiavi permette di creare cookie con una certa struttura logica

Esempio:

Se si vogliono usare dei cookie per salvare un'informazione dell'utente è più logico creare un singolo cookie chiamato ad esempio "UserInformation" con una chiave per ogni singolo dato da salvare, piuttosto che creare dei cookie diversi per ogni singolo dato

Come scrivere i cookie

I cookie sono scritti sul client utilizzando le intestazioni di risposta (response header)

Si utilizza la collection Cookies dell'oggetto Response

Esempio:

```
Response.Cookies("Nome") = "Gianluca"
Response.Cookies("Cognome") = "DiTomassi"
```

- I cookie possono contenere stringhe, numeri, date, valute e booleani
- I cookie non possono contenere vettori e oggetti

- Si possono creare dei cookie con chiavi

Esempio:

```
Response.Cookies("UserInformation")("Nome") = "Gianluca"
Response.Cookies("UserInformation")("Cognome") = "DiTomassi"
```

ATTENZIONE:

- Se si crea un cookie, gli si assegna un valore e poi si creano delle chiavi per quel cookie, il valore iniziale del cookie verrà eliminato.
- Un cookie non può contenere un valore e una chiave
- Le chiavi nei cookie hanno la precedenza sui valori dei cookie

La scadenza di un cookie

- I cookie scadono dopo un periodo di tempo determinato
- Quando un cookie scade viene cancellato dal client
- Quando si creano cookie è possibile specificare quando dovranno scadere utilizzando la property `Expires`
- Se non viene specificato il periodo di scadenza di un cookie, per default scade quando il client chiude il proprio browser

La property `Expires` riceve una data specifica della scadenza:

Esempio:

```
Response.Cookies("UserInformation").Expires = #01 june 2001#
```

In VBScript è possibile specificare una data nei seguenti modi:

```
#01 june 2001#
#1/6/2001#
CDate("01/6/2001")
DateSerial(2001, 6, 1)
```

Tutti questi metodi sono validi per impostare l'Expires di un cookie

Se vogliamo far scadere il cookie N giorni dopo la data attuale ?

Si utilizza la funzione `Date()` che permette di aggiungere un numero per incrementare o decrementare il valore della data

Esempio:

```
Date() + 5.Date()   'cinque giorni a partire dalla data attuale
Date() - 5.Date()   'cinque giorni indietro a partire dalla
                    'data attuale
```


Ad esempio impostando

```
Response.Cookies("UserInformation").Expires = Date() + 7
```

Vogliamo che il cookie UserInformation scada 7 giorni dopo la data attuale

Se la proprietà `Expires` viene impostata a una data precedente alla data attuale, il cookie scadrà quando l'utente chiude il proprio browser

I cookie possono essere letti solo da uno specifico sito Web

Come leggere i cookie

I cookie sono letti e memorizzati usando le intestazioni HTTP

Per vedere un elenco di tutti i cookie sul client che sono stati creati dal nostro sito Web:

```
<%=Request.ServerVariables("HTTP_COOKIE")%>
```

L'elenco è composto da coppie nome/valore

- Il nome e il valore sono separati dal simbolo "="
- Le coppie nome/valore sono separate dal simbolo ";"

L'oggetto Request contiene la collection **Cookies** utilizzata per leggere i valori del cookie secondo la sintassi:

```
Request.Cookies(NomeCookie)[(NomeChiave)]
```


Per determinare se un cookie ha delle chiavi si utilizza il metodo HasKeys della collection Cookies

Esempio:

Se sul client è memorizzato un cookie denominato "UserInformation" creato dal nostro sito Web

```
<%
  If Request.Cookies("UserInformation").HasKeys then
    Response.Write "UserInformation ha chiavi"
  Else
    Response.Write "UserInformation non ha chiavi"
  End If
%>
```

Se il cookie "UserInformation" non esiste sul client allora HasKeys restituirà false

Se un cookie ha delle chiavi, è possibile accedere alle chiavi specificando prima il nome del cookie e aggiungendo poi (*NomeChiave*)

Esempio:

```
<%
  Response.Write "Ultima visita al sito: " & _
    Request.Cookies("UserInformation")("UltimaVisita")
  Response.Write "Il tuo nome è: " & _
    Request.Cookies("UserInformation")("Nome")
%>
```

Quindi per leggere i valori delle chiavi si specifica il nome del cookie seguito dal nome della chiave

```
Request.Cookies(NomeCookie)(NomeChiave)
```


Dato che Request.Cookies è una collection si può scandire l'intera collection di cookie un elemento alla volta

Esempio:

```
 1:   <%@ Language=VBScript %>
 2:  <% Option Explicit %>
 3:  <%
 4:  Dim strCookieName, strKeyName
 5: For Each strCookieName in Request.Cookies
 6:  'Se il cookie attuale ha delle chiavi, bisogna scorrere
 7:  'le chiavi, usando il ciclo For…Next
 8:   If Request.Cookies(strCookieName).HasKeys then
 9:    For Each strKeyName in Request.Cookies(strCookieName)
10:       Response.Write strCookieName & "(" & _
11:               strKeyName & ") = " & _
12:               Request.Cookies(strCookieName)(strKeyName)
```

```
13:          Response.Write "<BR>"
14:      Next
15:    Else
16:     'Non vi sono chiavi per questo cookie, quindi si
17:     'visualizza il valore del cookie
18:     Response.Write strCookieName & " = " & _
19:               Request.Cookies(strCookieName)
20:    End If
21:    Response.Write "<BR>"
22:  Next
23:%>
```

Si determina se il cookie corrente ha delle chiavi (riga 8). Se ne ha si scorre la collection (riga 9). Se il cookie non ha nessuna chiave allora HasKeys è false e viene visualizzato il nome e il valore del cookie. Se non è stato creato nessun cookie sul client non si visualizza niente

Property della collection *Response.Cookies*

Nella collection **Response.Cookies** si possono impostare tre property:

1. **Domain** che per default è uguale al nome del dominio del proprio sito Web, permette di inserire un diverso nome di dominio

2. **Path** determina come possono essere letti i cookie. Può essere impostata per permettere ai cookie di essere letti solo da pagine ASP in una certa directory. Per default è impostata la directory radice del proprio sito Web (in modo da permettere la lettura dei cookie, creati da una pagina ASP, da parte di una qualsiasi pagina ASP esistente in una qualsiasi directory del file system

3. **Secure** property con valore booleano di sola scrittura determina se un cookie sarà inviato attraverso un protocollo non sicuro:
   - Se **True** allora il cookie sarà inviato solo se si accede alla pagina mediante protocollo HTTPS
   - Se **False** (valore di default) il cookie si invierà sia su HTTP che su HTTPS

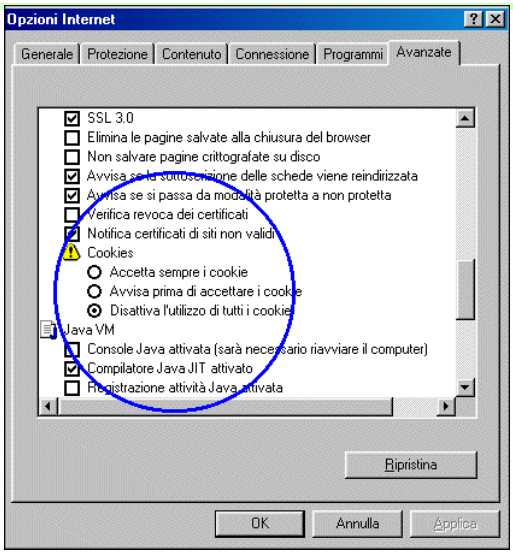
CONSIGLIO: Non modificare le property **Domain** e **path** altrimenti, se impostate male, si rischia di non essere in grado di leggere i cookie creati sul client

In definitiva:

Response.Cookies	scrive i cookie sul client
Request.Cookies	legge i cookie dal client

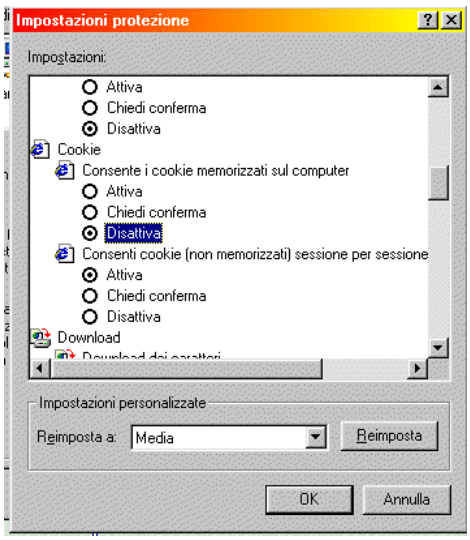
Come disabilitare i COOKIE con IE 4.0

Se volete disabilitare i cookies con Internet Explorer 4.0 andate in Visualizza-Opzioni Internet e poi Avanzate…



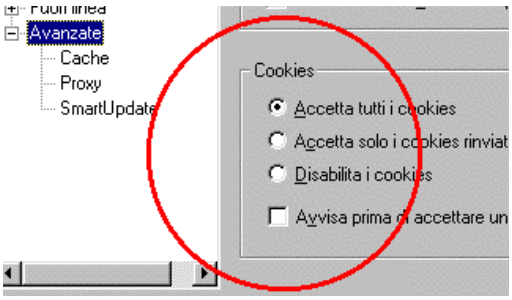
Come disabilitare i COOKIE con IE 5.0

In Internet Explore 5 per accedere alla manutenzione dei cookies occorre andare in STRUMENTI-OPZIONI INTERNET e poi su PROTEZIONE. Cliccando su PERSONALIZZA LIVELLO potrete accedere ad una altra finestra con comandi + specifici, tra cui quelli dei cookies



Come disabilitare i COOKIE con Netscape

In Netscape si va nel comando Modifica e poi in Preferenze, e da Avanzate puoi scegliere come regolarti



Vantaggi e svantaggi nell'utilizzo dei cookie

VANTAGGI:

1. Risiedendo sul client non deve essere allocato dello spazio sul server Web per memorizzare dati sugli utenti
2. Possono memorizzare delle piccole quantità di dati per lunghi periodi di tempo: settimane, mesi o anni
3. Possono essere usati per personalizzare la visita di un utente a un sito Web

SVANTAGGI:

1. Gli utenti possono scegliere di non accettare i cookie sui propri browser
2. I cookie non sono in grado di memorizzare oggetti, array ma possono memorizzare solo stringhe date e numerici

Usare l'oggetto Session

- Può memorizzare qualsiasi tipo di dato
- È usato per mantenere lo stato nell'ambito della durata della visita del sito per ogni singolo utente

1. Arriva un nuovo utente
2. Sul server Web viene allocata la memoria necessaria a memorizzare l'oggetto Session per quell'utente

La stessa memoria viene liberata se l'utente non richiede pagine al di sotto di un certo periodo di tempo (per default 10 minuti)

Sintassi per creare variabili di sessione:

```
Session(nomeVariabileSessione) = valore
```

Esempio:

```
Session("Nome")      = "Gianluca"
Session("Cognome")   = "Di Tomassi"
Session("DataOggi") = Date()
```

Occorre creare una variabile di sessione per ogni informazione che deve essere mantenuta

Sintassi per leggere le variabili di sessione:

```
Variabile = Session(nomeVariabileSessione)
```

Esempio:

```
<%
  Dim strNome, strCognome
  strNome = Session("Nome")
  strCognome = Session("Cognome")
  Response.Write "Benvenuto "&strNome&" "&strCognome
%>
```

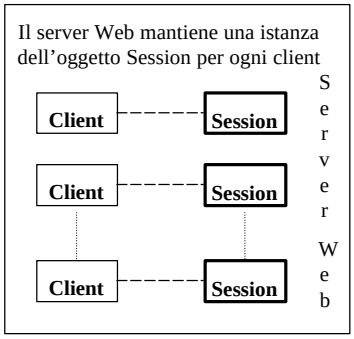
ATTENZIONE:

- L'oggetto Session identifica in maniera univoca i visitatori attraverso i cookie.
- Le variabili di sessione non vengono mantenute tra pagine ASP diverse se l'utente ha disabilitato i cookie

In dettaglio sull'oggetto Session

Progettato per mantenere lo stato per i singoli utenti

Analogia: L'oggetto Session è come un magazzino. Quando un nuovo utente arriva al sito, riceve il proprio magazzino. Ogni pagina ASP del sito può mettere o rimuovere dati nel magazzino dell'utente (Es. Carrello della spesa)



Ad ogni utente che naviga il sito viene assegnata una nuova istanza dell'oggetto Session

- Poiché ad ogni utente viene assegnato il proprio Session, ogni istanza deve essere univocamente identificabile
- Per identificare una particolare Session appartenente ad un utente si utilizza un ID numerico, il **SessionID**
- Il SessionID è un valore numerico che identifica univocamente ogni Session
- Per conoscere il SessionID assegnato alla sessione di un utente:

```
<% Session.SessionID %>
```


ATTENZIONE: Viene garantito che il SessionID sia unico per ogni singola sessione fino a quando il server Web è in esecuzione. Se il server Web viene riavviato, i nuovi SessionID possono essere dei duplicati dei vecchi SessionID. Pertanto non utilizzare il SessionID come identificatore unico per una tabella del database

- *Il SessionID viene memorizzato* in 2 parti; *sul server Web e sul client*.
  - Ogni Session gestita dal server Web contiene il proprio SessionID.
  - Il SessionID viene memorizzato anche sul client sotto forma di cookie
- Per questo motivo il server Web può stabilire quale Session appartiene ad un client

Esempio:

```
<% Response.write "Ciao " & Session("Nome") %>
```

Ogni volta che viene richiesta una pagina Web vengono inviate alcune intestazioni HTTP. Una di queste è l'intestazione Cookie che contiene tutti i cookie sul client che sono stati creati dal sito Web.

Se nel sito Web vengono usate le variabili di sessione, uno di questi cookie contiene il SessionID associato ad una particolare Session sul server Web. Questo cookie permette di riconoscere la corretta Session e di visualizzare la variabile "Nome"

Dato che l'oggetto Session viene gestito nella memoria del server web occorre liberare questa memoria quando l'utente abbandona il sito

Problema:

Il modello client/server non permette di determinare quando l'utente non è più presente

Soluzione:

Si può verificare l'ultimo accesso di un utente

Il periodo di tempo che passa prima che il Session di un utente sia liberato viene chiamato timeout della sessione

- Per default 10 minuti con IIS 5.0
- Per default 20 minuti con IIS 4.0 e Windows 2000 release candidate 2

Per impostare il timeout della sessione si utilizza la property Timeout dell'oggetto Session

Esempio: L'oggetto Session scade dopo cinque minuti

```
<% Session.Timeout = 5 %>
```

Per distruggere esplicitamente il Session di un utente, prima dello scadere del Timeout della sessione, si utilizza il metodo Abandon

Esempio:

```
<% Session.Abandon %>
```

NOTA: E' utile prevedere nel proprio sito un bottone Logout che quando premuto rimuove ogni informazione salvata, come i cookie e le variabili di sessione

L'oggetto Session permette di memorizzare qualsiasi tipo di variabile anche array (i cookie solo semplici tipi di dati sul client)

Usare l'oggetto Session per memorizzare gli array

```
 1:<% 'Creazione di un array
 2:   Dim aSentence(2)
 3:   aSentence(0) = "Memorizzazione "
 4:   aSentence(1) = "dei valori "
 5:   aSentence(2) = "in un array "
 6:   Dim iLoop
 7:   For iLoop = LBound(aSentence) to UBound(aSentence)
 8:      Response.Write aSentence(iLoop)
 9:   Next
10:    'Memorizzazione dell'array nell'oggetto Session
11:    Session("Sentence") = aSentence
12:%>
```

SPIEGAZIONE:

Viene creato l'array `aSentence` (riga 2.) che contiene 3 elementi, da 0 a 2. Le righe dalla 3. Alla 5. impostano i valori di ognuno degli elementi che poi vengono visualizzati. Si utilizzano le funzioni `Lbound()` e `Ubound()` per scorrere l'array un elemento alla volta (riga 7.) all'interno di un ciclo for. La riga 8. Visualizza l'elemento corrente per ogni iterazione del ciclo. La riga 11. Crea una variabile di sessione `Sentence` e la imposta uguale all'array `aSentence`

RICORDA: `Lbound()` restituisce l'indice iniziale di un array
`Ubound()` restituisce l'indice finale di un array

Visualizzare il contenuto della variabile di sessione array

```
1:<%  Dim iLoop
   'Ci assicuriamo che la variabile di sessione è un array
2:  If IsArray(Session("Sentence")) then
     'Stampiamo tutti gli elementi dell'array
3:    For iLoop = LBound(Session("Sentence")) to _
                UBound(Session("Sentence"))
4:       Response.Write Session("Sentence")(iLoop)
5:    Next
6:   Else
     'Se Session("Sentence") non è un array
7:    Response.Write "ERRORE NON E' POSSIBILE PROCEDERE!"
8:  End If
%>
```

SPIEGAZIONE:

Prima di provare a leggere da Session("Sentence"), la riga 2. controlla che Sentence sia un array valido usando la funzione IsArray() (Se la variabile Sentence non è stata ancora creata restituirà false). Se Sentence è un array, il ciclo di riga 3. scorre ogni elemento dell'array, visualizzandone il contenuto riga 4.

Sintassi per riferirsi ad un elemento in una var sessione array

Con un array si fa riferimento a una specifica variabile con

`NomeArray(indice)`

Quando si memorizza un array nell'oggetto Session, il NomeArray è

`Session(`*`nomeVariabileSessione`*`)`

Quindi per leggere un elemento da una variabile di sessione array, la sintassi è

`Session(`*`nomeVariabileSessione`*`)(indice)`

L'oggetto Session fornisce 2 collection:

1. **`Contents`** contiene le variabili di sessione che non sono oggetti
2. **`StaticObject`** contiene le variabili di sessione oggetti

IMP: Per creare e per memorizzare una nuova istanza di un oggetto in una variabile deve essere usata la parola chiave Set (anche quando si memorizza un oggetto in Session)

`Set Session(`*`nomeVariabileSessione`*`) = ObjectInstance`

Esempio utilizzo Contents:

```
 1: <% Response.Write "Ci sono " & Session.Contents.Count & _
                      " variabili di sessione<P>"
 2: Dim strName, iLoop
 3: For Each strName in Session.Contents
 4:    If IsArray(Session(strName)) then
 5:       For iLoop = LBound(Session(strName)) to _
                      UBound(Session(strName))
 6:        Response.Write strName & "(" & iLoop & ") - " & _
              Session(strName)(iLoop) & "<BR>"
 7:       Next
 8:    Else
 9:      Response.Write strName & " - " & _
                        Session.Contents(strName) & "<BR>"
       End If
10: Next%>
```


SPIEGAZIONE:

La proprietà `Count` (riga 1.) della collection `Contents` si utilizza per visualizzare il numero delle variabili di sessione. La collection `Contents` contiene tutte le variabili di sessione non oggetti, pertanto anche le variabili di sessione array, pertanto prima di visualizzare una variabile di sessione, occorre determinare se questa è un array (riga 4.). Se la variabile di sessione è un array allora attraverso un ciclo for (riga 5.) si scorre ogni elemento dell'array della var. di sessione e si visualizzano (riga 6.). Se la var. di sessione non è un array (riga 8.) viene visualizzato il nome della var. di sessione e il suo valore (riga 9.)

In output vengono visualizzate tutte le variabili di sessione dell'utente e i loro valori.

Problemi nell'uso delle variabili di sessione

Evitare di:

1. Mettere oggetti in una Session di un utente
2. Impostare il Timeout a un valore non ottimale
3. Creare variabili di sessione non necessarie

Porsi queste domande prima di utilizzare un oggetto Session:

1. Quanti utenti potrò avere contemporaneamente sul mio sito ?
2. Quante variabili di sessione penso di utilizzare ?
3. Posso mantenere lo stato usando un approccio alternativo ?

Sviluppando applicazioni ASP, se il numero di utenti contemporanei stimati è > 50 allora cercare di evitare le variabili di sessione

Variabili di sessione senza cookie

Ogni utente ha il proprio Session, quindi come fa il server Web a collegare un utente ad una istanza dell'oggetto Session ?

→ Attraverso un unico `SessionID` che collega il Session e l'utente

Il `SessionID` si trova nell'istanza dell'oggetto Session e sul client, nella forma di cookie.

Quando l'utente richiede una pagina ASP che usa le var. di sessione, il cookie del client può essere letto e fatto corrispondere al Session esistente in memoria.

PROBLEMA: Se l'utente ha configurato il browser in modo da non accettare i cookie non si è in grado di mantenere lo stato per questo utente

Internet Server Application Programming Interface

SOLUZIONE:

Microsoft ha creato un filtro ISAPI che simula i cookie per gli utenti che hanno configurato il proprio browser per non accettare i cookie

- Un filtro ISAPI (detto *Cookie Munger*) è un piccolo programma di basso livello che permette di aggiungere funzionalità ad un server Web
- Scaricabile dal sito http://www.microsoft.com
- Con questo filtro sul server Web anche per i client che non accettano i cookie si possono utilizzare le var. di sessione

Funzionamento del filtro ISAPI

Quando un utente che accetta i cookie visita il sito, il *Cookie Munger* non fa nulla, al contrario se l'utente non accetta i cookie allora

1. Quando viene richiesta una pagina, il *Cookie Munger* genera un SessionID per quel particolare client
2. La pagina ASP prende quel SessionID e può farlo corrispondere a una particolare istanza dell'oggetto Session
3. Quando una pagina ASP viene inviata al client che non accetta i cookie, il *Cookie Munger* esamina il codice HTML per i collegamenti ipertestuali. Alla fine dell'URL del link viene aggiunto il SessionID per il client
4. Quando un client richiede un URL che ha il SessionID aggiunto alla fine, il *Cookie Munger* rimuove il SessionID aggiunto, chiama la pagina ASP richiesta e passa il SessionID alla pagina ASP.

Svantaggio nell'utilizzo del filtro ISAPI

Visto che il Cookie Munger deve elaborare ogni pagina ASP che è stata richiesta e deve elaborare ogni risposta ASP in uscita, tutte queste operazioni possono essere un grosso lavoro per il server Web (specialmente per siti con tanti visitatori)

ATTENZIONE:

ASP non invia i cookie di sessione sotto le seguenti condizioni:

1. se disabilitiamo i cookie
2. se nelle pagine ASP è contenuto:

```
<%@ EnableSessionState=False %>
```


Usare l'oggetto Application

- mantiene lo stato globalmente sull'intero sito Web

Differenza tra gli oggetti Session e Application

Mentre l'oggetto Session mantiene lo stato per ogni singolo utente, l'oggetto Application mantiene lo stato globalmente sull'intero sito Web

Qualsiasi utente può accedere a qualsiasi variabile dell'applicazione da una pagina ASP qualsiasi, quindi:

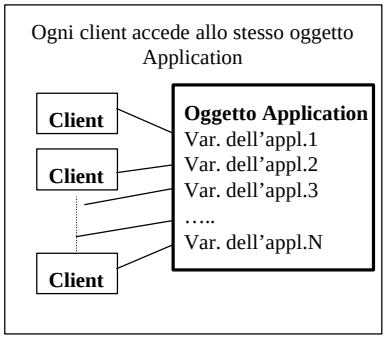
Variabili dell'applicazione = variabili globali

IMP: Esiste una sola istanza dell'oggetto Application per il proprio sito Web. Ogni singolo utente ha accesso allo stesso insieme di variabili dell'applicazione

In dettaglio sull'oggetto Application

Progettato per mantenere informazioni globali per tutti gli utenti per l'intero sito

Analogia: L'oggetto Application è come un magazzino. Però a differenza dell'oggetto Session ne esiste uno solo per tutti gli utenti indipendentemente dal loro numero.



Esiste un solo oggetto Application per una applicazione Web

Sintassi per creare variabili dell'applicazione :

```
Application(nomeVariabileApplicazione) = valore
```

Sintassi per leggere le variabili dell'applicazione:

```
Variabile = Application(nomeVariabileApplicazione)
```

L'oggetto Application (come l'oggetto Session) esiste nella memoria del server Web

Poiché esiste un solo oggetto Application condiviso tra tutti gli utenti, occorre fare attenzione negli aggiornamenti delle var. dell'applicazione.

Possibilità di avere N utenti collegati contemporaneamente al sito

→ Quando si aggiorna una var. dell'applicazione la modifica deve riguardare un solo utente alla volta

L'oggetto Application fornisce due metodi:

1. **Lock** che deve essere chiamato prima di modificare una qualsiasi var. dell'applicazione.
2. **UnLock** permette il rilascio esclusivo della var. dell'applicazione

Quando invocato il Lock nessun altro utente può aggiornare alcuna variabile dell'applicazione

Esempio:

```
<%
  Application.Lock
  Application("Contatore") = Application("Contatore") + 1
  Application.UnLock
%>
<%= Application("Contatore") %>
```


Problemi nell'uso delle variabili dell'applicazione

Valgono le stesse considerazioni fatte nelle slide 36, 37, 38 con l'accortezza di sostituire Session con Application

Al fine di non diminuire le prestazioni sul server evitare di:

1. Mettere oggetti nell'oggetto Application a meno che non sia necessario
2. Creare variabili dell'applicazione che non sono necessarie

Esempio:

- Le costanti che servono per tutto il sito (es. email webmaster, la barra di navigazione comune a tutte le pagine) è meglio includerle attraverso un file statico
- Se i dati che si devono memorizzare cambiano rapidamente come la data dell'ultimo messaggio (es. News) allora l'informazione si deve memorizzare in una var. dell'applicazione

Quale scegliere come approccio ?

Stringa d'interrogazione

Assicurarsi che ogni link ipertestuale passi la stringa d'interrogazione corrente alla pagina ASP successiva.

Svantaggio: Aumenta la difficoltà di manutenzione del sito

Vantaggio: non richiede che l'utente abbia abilitato i cookie

Quando usarla

Se occorre mantenere dei semplici tipi di dato per la sola durata della visita dell'utente e se è fondamentale mantenere lo stato anche per gli utenti che hanno disabilitato i cookie

Cookie

Utile per mantenere lo stato per periodi più lunghi della durata della visita dell'utente (unica soluzione)

Svantaggio: Possono memorizzare solo dei semplici tipi di dati (stringhe, numeri, date) e possono essere rifiutati dal client

Vantaggio: non provocano nessun sovraccarico sul server Web perché memorizzati sul client

Quando usarli

Se le prestazioni sono un grosso problema e se c'è bisogno di mantenere dati per giorni, settimane o mesi

Variabili di sessione

Se occorre mantenere lo stato solo per la durata della visita dell'utente

Svantaggio: Risiede sul server Web, pertanto se si ricevono molti utenti contemporaneamente oppure se si mette una grande quantità di dati nell'oggetto Session, le prestazioni del server Web diminuiranno

Vantaggio: A differenza dei cookie può memorizzare qualsiasi tipo di variabile

Quando usarle

Quando c'e' bisogno di mantenere lo stato solo per il solo periodo della visita dell'utente

Variabili dell'applicazione

Usate per mantenere dati globali per l'intero sito Web

Non dovrebbe essere utilizzato per mantenere lo stato per ogni singolo utente.

Quando usarle

Quando si ha a che fare con dati che cambiano spesso e che sono globali all'intero sito Web (contatore degli accessi all'intero sito Web)

Inizializzare le variabili di sessione e dell'applicazione

Gli oggetti Session e Application si utilizzano come magazzino.

Quando l'utente visita per la prima volta il sito viene creato un nuovo magazzino Session che per default è vuoto

Entrambi gli oggetti posseggono un evento che consente di inizializzare i contenuti della Sessione dell'utente e dell'Application del sito

Il file *global.asa*

Un'applicazione ASP è costituita dall'insieme dei file contenuti in una directory virtuale del Web server e nelle sue subdirectory

La gestione delle informazioni di stato avviene (abbiamo visto) prevalentemente tramite gli oggetti Application e Session

L'esecuzione di script in corrispondenza dell'avvio e della terminazione di un'applicazione e/o di una sessione prevede, se presente, l'esecuzione di un file denominato *Global.asa* , contenente gli script opportuni

- Il file la cui estensione ".asa" sta per **A**ctive **S**erver **A**pplication, viene letto dal motore ASP all'avvio dell'applicazione e all'inizio di ciascuna sessione utente
- Il file global.asa deve risiedere nella directory radice del sito Web

Struttura del file *global.asa*

```
<SCRIPT LANGUAGE=VBScript RUNAT=Server>

 Sub Application_OnStart()
 End Sub

 Sub Application_OnEnd()
 End Sub

 Sub Session_OnStart()
 End Sub

 Sub Session_OnEnd()
 End Sub

</SCRIPT>
```


Eventi Application

Un'applicazione inizia appena l'utente richiede una delle pagine Web del sito e termina in corrispondenza dello shut down del server

L'applicazione ha due eventi:

1. l'evento `Application_OnStart`
2. l'evento `Application_OnEnd`

Possiamo realizzare script per questi eventi nel file Global.asa

- Quando l'applicazione parte il server controlla nel Global.asa e processa gli script contenuti in `Application_OnStart`
- Quando si effettua lo shut down del server, vengono processati gli script di `Application_OnEnd`

Eventi Session

Il server Web crea automaticamente una nuova sessione quando un utente, richiede una pagina Web del sito e la distrugge quando scade un timeout associato alla sessione dell'utente, oppure quando invoca il metodo Abandon

Una sessione ha due eventi associati:

1. L'evento `Session_OnStart`
2. L'evento `Session_OnEnd`

- L'evento `Session_OnStart` occorre quando il server crea una nuova sessione. Questo evento rappresenta il momento giusto per definire tutte le variabili di sessione che verranno utilizzate durante la sua visita
- L'evento `Session_OnEnd` occorre quando una sessione viene terminata o scade il relativo timeout

- Nell'evento OnEnd dell'oggetto Session sono disponibili solo gli oggetti Session, Application e Server
- Nell'evento OnEnd dell'oggetto Application sono disponibili solo gli oggetti Application e Server

ATTENZIONE:

I gestori degli eventi OnEnd sono usati **solo** per liberare esplicitamente gli oggetti. Dato che il gestore dell'evento Session viene eseguito un attimo dopo che l'utente ha lasciato il sito (a seconda del Session.Timeout) non è possibile visualizzare un messaggio all'utente oppure portarlo ad un'altra pagina.

→ I gestori degli eventi OnEnd non dovrebbero fare nient'altro che eliminare gli oggetti creati dai gestori degli eventi OnStart.

Concludendo sul file *global.asa*

***Usare** il file global.asa per:*

Creare e inizializzare tutte le variabili di sessione e dell'applicazione con i gestori degli eventi OnStart.

Il file dovrebbe anche contenere i gestori degli eventi OnEnd per liberare esplicitamente, ogni oggetto variabile di sessione o dell'applicazione

***Non usare** il file global.asa per*:

Tentare di portare gli utenti a un'altra pagina o per visualizzare dei messaggi

Esempio di file *global.asa*

```
1. <SCRIPT LANGUAGE=VBScript RUNAT=Server>
2.  Sub Application_OnStart()
3.    'Crea un oggetto nella variabile application
4.    Set Application("MyDict") =
            Server.CreateObject("Scripting.Dictionary")
5.  End Sub

6.  Sub Application_OnEnd()
7.    'Libera esplicitamente l'oggetto della variabile
      'dell'applicazione
8.    Set Application("MyDict") = Nothing
9.  End Sub
```

```
10.  Sub Session_OnStart()
11.     'Crea l'oggetto nella variabile di session.
        'NON E' corretto ma serve per far vedere come usare
        ‘Session_OnEnd()
12.   Set Session("MyObject") = Server.CreateObject("My.Obj")
13.  End Sub

14.   Sub Session_OnEnd()
15.     'Libera esplicitamente l'oggetto della variabile di
        'sessione
16.     Set Session("MyObject") = Nothing
17.   End Sub
18. </SCRIPT>
```


Spiegazione esempio

Il file inizia con il gestore dell’evento OnStart (riga 2) per l’oggetto Application. In questo gestore dell’evento, viene creata una variabile dell’applicazione chiamata MyDict che contiene una istanza dell’oggetto Dictionary (riga 4). Nella riga 6, viene messo l’evento OnEnd per l’oggetto Application. La riga 8 libera esplicitamente la variabile dell’applicazione MyDict impostandola uguale a Nothing.

Il gestore dell’evento OnStart dell’oggetto Session è codificato nella riga 10.

Quando viene creato un nuovo Session, viene creata una variabile di sessione chiamata MyObj e viene posta uguale a una istanza dell’oggetto My.Obj (riga 12). *Mettendo un oggetto nel Session, naturalmente, si possono avere delle diminuzioni di prestazioni su siti Web con un alto numero di utenti*. Nella riga 14 inizia il gestore degli eventi OnEnd per l’oggetto Session. Infine nella riga 16, la variabile di sessione MyObject viene esplicitamente distrutta impostandola uguale a Nothing.