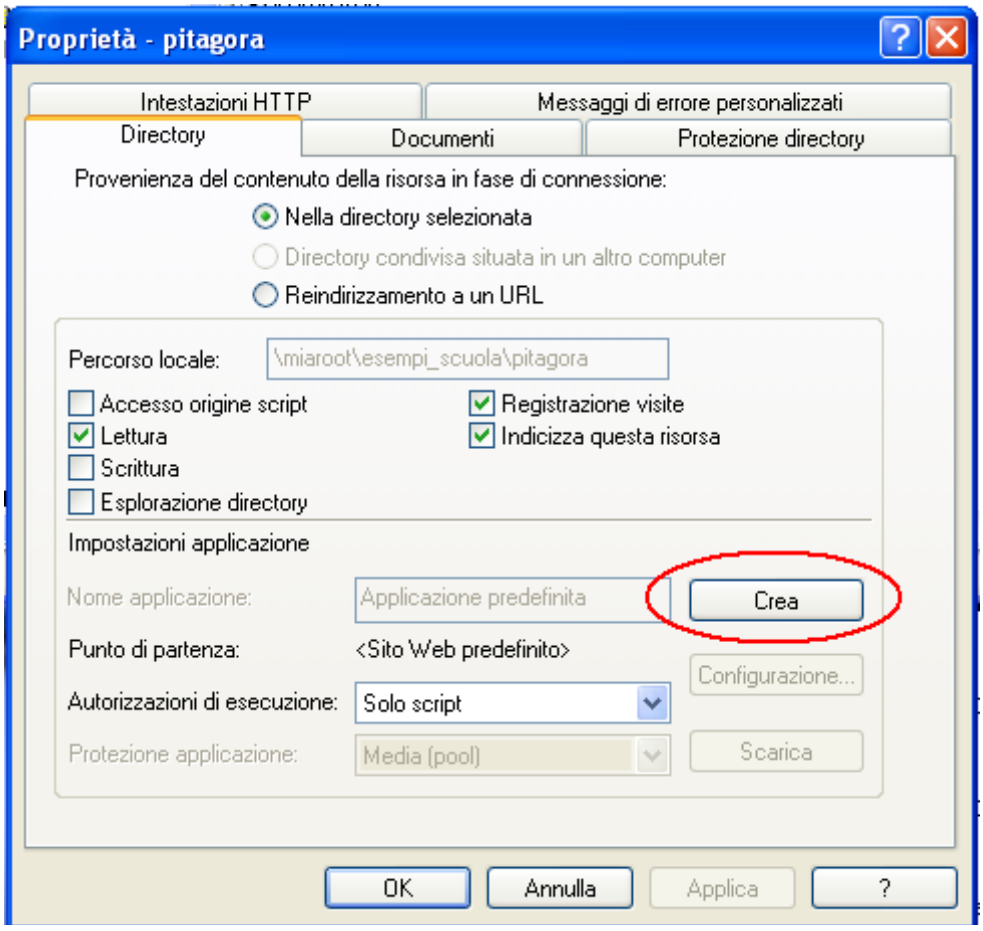
Classe 5a Abacus – Analisi ultimi progetti in ASP proposti in laboratorio

Sistema di rilevamento di utenti in linea su un certo sito

Il problema proposto è un classico problema più volte affrontato da siti Web che si occupano di programmazione Web ed in particolare programmazione in ASP. Con questo problema ci si pone l'obbiettivo di *determinare quanti utenti siano in quel momento on line sul sito considerato*. Ricordiamo che non è possibile individuare immediatamente gli utenti quando abbandonano, a tutti gli effetti, la loro sessione chiudendo semplicemente il browser, in quanto non essendo presente una connessione continua e permanente tra navigatore (client) e server Web, non è possibile individuare in modo immediato la chiusura del browser stesso. Diverso il caso che l'uscita dalla sessione sia effettuata attraverso l'apposito comando ASP Session.Abandon. In questo caso la sessione corrente viene immediatamente chiusa.

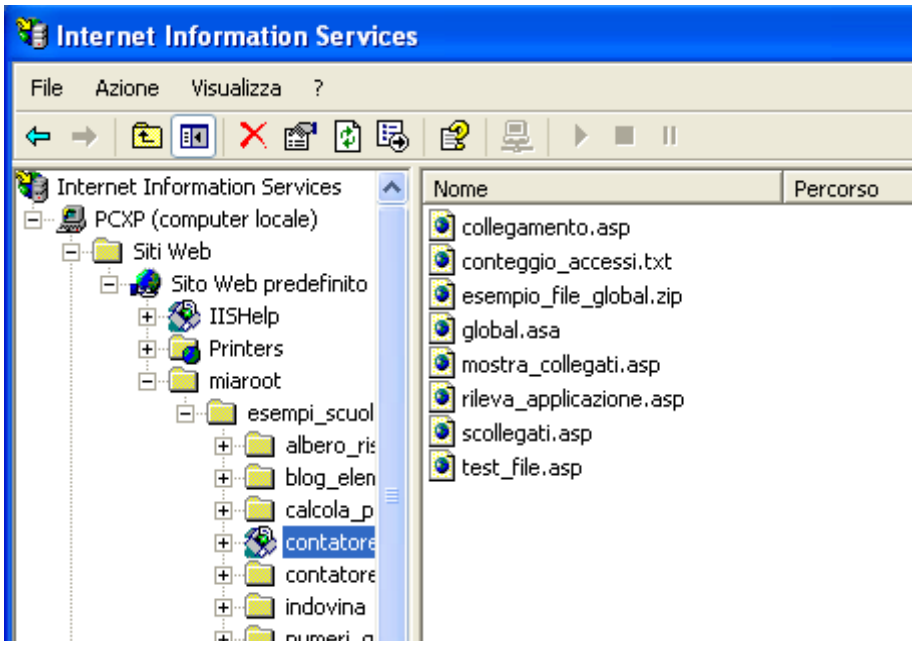
Per affrontare in modo efficace il problema proposto, si deve fare ricorso all'uso dello speciale file detto global.asa. Si tenga conto che usualmente un server Web ospita più siti in parallelo che devono funzionare in modo del tutto indipendente. Per questa ragione le stesse variabili Application, come riportato dal testo della HOEPLI[1] a pagina 163, non hanno tutte lo stesso valore se si attivano diverse directories virtuali, equivalenti ad attivare vari siti in modo indipendente. In pratica potremo avere che la variabile Application("pippo") vista da tutti gli utenti di sito 1 potrà non essere visibile da quelli di sito



1"I linguaggi del Web" - Camagni e Nikolassy - Ed. HOEPLI

2, oppure che il valore di una variabile Application omonima non sia vincolato ad essere lo stesso sui due siti. Proprio per questo e per poter utilizzare il file global.asa è importante rendere virtuale la sottodirectory di primo livello che contiene il sito di nostro interesse. Su IIS ciò può essere fatto molto semplicemente, premendo un apposito pulsante che rende la directory immediatamente virtuale, indicato dalle immagini indicata precedentemente

La finestra qui di seguito rappresentata si riferisce in particolare alla finestra che appare quando, utilizzando la console di gestione di IIS, si clicca su una clicca su una directory e si seleziona il menù proprietà. Per fare poi divenire la directory in questione virtuale, basta premere il pulsante cerchiato in rosso nella figura. L'icona della directory, nel programma di gestione di IIS cambierà forma e colore indicando la virtualizzazione della directory stessa. Da quel momento la directory e tutte le sue subdirectory presenti sono pronte a considerare quanto in questa directory in modo del tutto separato dai contenuti di altre directories.



A questo punto sarà possibile inserire un file global.asa[2] nella root della directory virtuale appena creata, e questo permette facilmente di ottenere operazioni altrimenti affatto banali nel loro svolgimento, utilizzando altri sistemi.

Nell'intero Web server ci potranno essere più global.asa, ma ognuno di essi sarà relativo ad una applicazione Web (ed in questo modo i valori e gli effetti delle variabili applicazione saranno distinti per i diversi siti).

Tutto ciò può essere facilmente constatato creando due directory virtuali e quindi potenzialmente due siti Web indipendenti. Scopo del file global.asa è attivare degli eventi utilizzati dal sito per svolgere delle funzionalità in automatico; esso, quando vuoto di codice da eseguire avrà la seguente forma e la seguente sintassi:

```
<script language="vbscript" runat="server">
Sub Application_OnStart()
```

2 Il nome global.asa non è tassativo e non può quindi essere cambiato.

```
End Sub
Sub Application_OnEnd()


End Sub
Sub Session_OnStart()


End Sub
Sub Session_OnEnd()


End Sub
</script>
```

come si può osservare bel file sono presenti 4 eventi[3] ognuno dei quali viene richiamato in un ben preciso momento:

- Application_OnStart( ) evento richiamato quando il Web server viene acceso
- Application_OnEnd( ) evento richiamato quando il Web server viene spento
- Session_OnStart( ) evento richiamato quando un nuovo browser si collega alla specifica sezione del sito (applicazione Web)
- Session_OnEnd( ) evento richiamato quando un nuovo browser si scollega dalla specifica sezione del sito (applicazione Web)

Grazie all'attivazione automatica di questi eventi è possibile facilmente registrare quanti utenti sono attualmente collegati alla specifica applicazione Web (sito).

Per far ciò basterà inserire il seguente codice nel file global.asa:

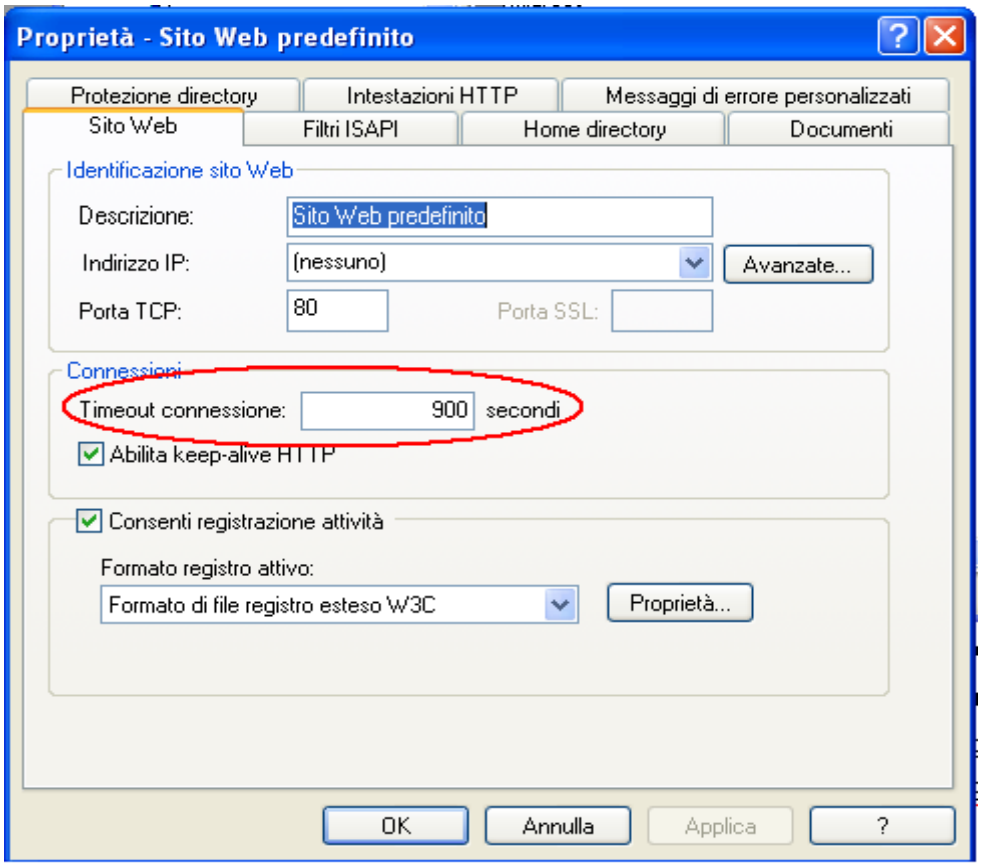
```
<script language="vbscript" runat="server">
Sub Application_OnStart()
      Application("Collegati") = 0
End Sub
Sub Application_OnEnd()

End Sub
Sub Session_OnStart()
    Application.Lock()
    Application("Collegati") = Application("Collegati") + 1
    Application.UnLock()
End Sub
Sub Session_OnEnd()
    Application.Lock()
    Application("Collegati") = Application("Collegati") - 1
    Application.UnLock()
End Sub
</script>
```

Questo codice svolge le seguenti operazioni ed ha la seguente logica:

1. Al momento dell'avvio del server, viene attivato l'evento Application_OnStart( ) che effettua una inizializzazione ovviamente a 0, ossia 0 utenti connessi, del valore della variabile di conteggio Application("Collegati")

3 Si noti che gli eventi sono qui implementati con delle subroutine evento, ossia che vengono chiamate direttamente dal sistema di gestione dell' ASP e non da codice ASP. Inoltre c'è da notare che la struttura di questo codice è del tutto simile a quella degli eventi definiti in Visual Basic, linguaggio da cui ASP deriva.

2. La variabile di conteggio è una variabile di tipo Application, ossia visibile a tutti gli utenti della applicazione Web e senza scadenza, in quanto deve contenere un valore globale a tutte le sessioni e lavora quindi ad un livello superiore alle sessioni stesse.
3. Appena un client (Web browser si connette con la specifica Web application, scatta l'evento Session_OnStart( ), che effettua un incremento di 1 del valore della variabile Application("Conteggio"). In tal modo si è registrato il collegamento del client e l'apertura di una sessione.
4. Si noti che precedentemente a ciò le variabili Applicazione del sito vengono bloccate (Application.Lock) (non è quindi più possibile per altre parti di codice accedervi in scrittura, ossia per modificare i valori di tutte le Application, e operazione analoga, ma inversa viene effettuata alla fine con il metodo Unlock.
5. Quando un client si scollega, immediatamente o dopo un certo tempo di timeout della sessione, viene richiamato l'evento Session_OnEnd( ). Tale evento non si attiva alla semplice chiusura del browser, ma viene chiamato al momento che scade il tempo di timeout della sessione[4].
   Viceversa l'abbandono della sessione diviene sincrono e immediato se viene eseguito del codice che contenga l'istruzione ASP Session.Abandon. All'interno dell'evento vi è il codice per ridurre di una unità il valore all'interno della variabile applicazione Application("Conteggio"), oltre alle solite istruzioni di blocco e sblocco delle modifiche delle variabili applicazione.

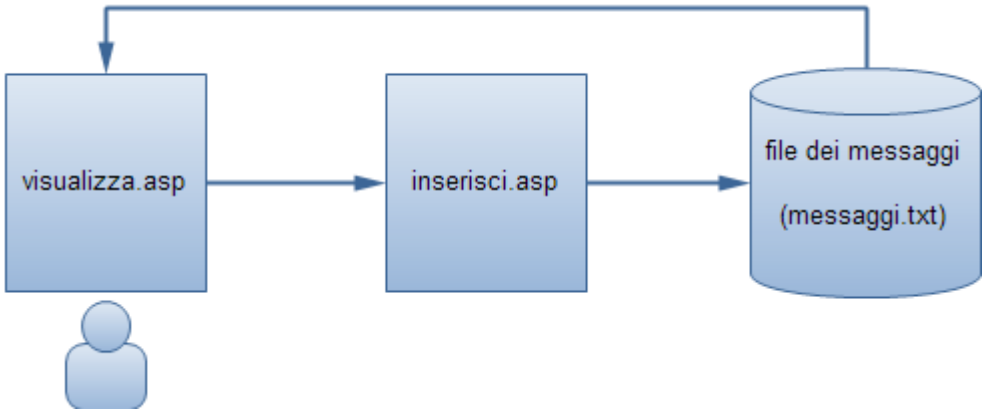


4 Abbiamo a questo proposito visto in laboratorio che il tempo di timeout di default è ricavabile tramite Session.Timeout

Operando nel modo sopra indicato è quindi possibile inserire tramite la semplice stampa della variabile applicazione "Conteggio", il numero di utenti attualmente collegati al sito. C'e' da osservare che questo dato non soffre di eventuali stop del server in quanto se il server viene fermato è chiaro che tutti gli utenti collegati risulteranno essersi scollegati, e quindi al suo riavvio la situazione partirà da uno stato ri-azzerato. Precedentemente ho riportato per completezza la schermata della finestra di proprietà riportante anche la regolazione del tempo di timeout della sessione sul server.

Realizzazione di un blog elementare

In questo esempio indichiamo come poter realizzare un blog elementare. Il blog è realizzato tramite la scrittura su un unico file comune, dove sono contenuti gli interventi risiedente sul server. Le pagine che operano sul file e che realizzano il piccolo progetto sono due, una che serve per la lettura dei messaggi ed ha alla fine un apposito form per inviare gli interventi, ed una (non visibile agli utenti) che implementa le operazioni di scrittura sul file, ossia scrive ed aggiunge il singolo intervento sul file.
Nel nostro caso, il blog è nel suo stato più semplice e non prevede neppure una autenticazione dell'utente, ma opera in modalità anonima per tutti gli utenti che intervengono sul blog stesso.
L'utente può quindi liberamente inserire un oggetto (titolo) ed il relativo messaggio e tale messaggio apparirà nella lista messaggi accodato a quelli precedenti.



Per far funzionare correttamente un simile sistema di comunicazioni è necessario il file abbia un preciso formato dei messaggi che permetta di determinare e mantenere un sincronismo di quanto è stato letto; nel nostro caso i messaggi interni al file sono formattati nel seguente formato:

```
<Data / ora del messaggio>
Anonimo     (costante stringa)
<Soggetto>
<Corpo del messaggio
...   ...
(n linee)>
----
(messaggio successivo)
```

in pratica si effettuerà la lettura su tre linee (tre operazioni di lettura) + n altre operazioni in ciclo, fino a

quando si incontrerà il separatore ----. A questo punto il sistema di lettura riparte da capo, ciclicamente nella lettura di un nuovo intervento, oppure rileva la presenza della fine del file[5].
Sono quindi da prevedere 2 cicli, uno esterno (messaggi) ed uno interno (righe di testo del messaggio).
Analizzata la metodica realizzativa possiamo passare ad osservare come in concreto è possibile scrivere il codice HTML e ASP nelle due pagine considerate.
La pagina visualizza.asp è la più complessa. Essa mixa codice HTML ad ASP e genera codice HTML in modo dinamico[6]. Il codice della pagina può essere suddiviso in due parti: una la parte di lettura e report dei messaggi già presenti, e la seconda la conformazione del form per inviare altri interventi al blog:

```
<!--
==========================================================================================
     A. Veneziani - Esempio di blog elementare basato su memorizzazione su file
                    tramite utilizzo della libreria  Scripting.FileSystemObject
==========================================================================================
-->

<!DOCTYPE      html      PUBLIC      "-//W3C//DTD      XHTML      1.0      Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Blog elementare</title>
<style type="text/css">
<!--
body {
       background-color: #FFFF99;
}
.Stile7  {font-family:  Verdana,  Arial,  Helvetica,  sans-serif;  font-size:  12px;  font-
weight: bold; }
.Stile10 {font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 12px; }
-->
</style></head>

<body>
<%
    Set Fso = Server.CreateObject("Scripting.FileSystemObject")

       If Fso.FileExists(Server.MapPath(".") & "\messaggi.txt") Then
              Set Flo = Fso.OpenTextFile(Server.MapPath(".") & "\messaggi.txt")

              Do While Not Flo.AtEndOfStream

                     OraData = Flo.ReadLine
                     Utente = Flo.ReadLine
                     Oggetto = Flo.ReadLine

                     Messaggio = ""
                     Riga = " "
                     Do While (Not Riga = "----") And (Not Flo.AtEndOfStream)
                            Riga = Flo.ReadLine
                     If Not Riga = "----" Then Messaggio = Messaggio & Riga

               Loop
%>

<table width="451" border="0" align="center">
  <tr>
```

5 Si ricorda che l'oggetto TextStream, possiede la proprietà AtEndOfStream che indica se si è raggiunta la fine del file.
6 Tecnica che abbiamo già visto e sperimentato in qualche esercitazione in laboratorio.

```
                    <td      width="127"      height="33"      bgcolor="#FFFF66"><span
class="Stile7">Data/Ora:</span></td>
    <td width="314" bgcolor="#FFFF66"><span class="Stile10"><%= OraData %></span></td>
  </tr>
  <tr>
    <td height="31" bgcolor="#FFFF66"><span class="Stile7">Utente:</span></td>
    <td bgcolor="#FFFF66"><span class="Stile10"><%= Utente %></span></td>
  </tr>
  <tr>
    <td height="27" bgcolor="#FFFF66"><span class="Stile7">Oggetto:</span></td>
    <td bgcolor="#FFFF66"><span class="Stile10"><%= Oggetto %></span></td>
  </tr>
  <tr>
    <td height="52" bgcolor="#FFFF66"><span class="Stile7">Messaggio:</span></td>
    <td bgcolor="#FFFF66"><span class="Stile10"><%= Messaggio %></span></td>
  </tr>
</table>
<p>
  <% Loop

  End If
  %></p>
```

questa prima parte della pagina, come si può vedere contiene istruzioni per aprire il file di testo messaggi.txt. Il codice prevede tra l'altro l'evenienza che il file non esista, in questoso caso l'operazione di lettura non ha luogo e questo viene permesso dal metodo FileExists(...) che controlla se il file con path passato come parametro esiste effettivamente o meno.

Se il file esiste, esso viene aperto in lettura e inizia la lettura dei suoi contenuti. Si noti come esistano due cicli Do While, uno appunto per ciclare sui singoli interventi e l'altro che legge le molteplici 8ed in numero non definito) righe di testo di un singolo messaggio.

Per ogni messaggio vengono effettuate tre operazioni di lettura di una intera linea (data/ora, Anonimo – rimpiazzabile con nome o identificatore dell'utente connesso, se si usa l'autenticazione, e oggetto).

Susseccivamente si legge tutto il testo del messaggio che viene opportunamente cumulato nella variabile Messaggio. Anche i valori letti dalle prime tre righe sono cumulati in tre opportune variabili.

Si noti che la condizione del ciclo più interno indica proprio che esso viene interrotto quando si incontri la sequenza ---- Letti tutti i dati si tratta poi di esporre in HTML quanto letto in ASP. Questo è fatto all'interno del ciclo Do While esterno, infatti si noti che la chiusura del primo ciclo Do While è sottostante a tutta una serie di tag, che essendo all'interno di un ciclo ASP, vengono ripetuti n volte. Questi tag definiscono una tabella di 4 righe in cui vengono posti i dati letti, relativamente alle indicazioni di cui si è già parlato.

Opportuni tag ASP (<%= %>) affogati nell' HTML permettono di emettere in modo opportuno i valori nei punti necessari ad una presentazione corretta.

Al seguito di questo codice e della chiusura del ciclo Do While e dell' If che controlla l'esistenza del file messaggi.txt, c'è il codice HTML che definisce la form di invio degli interventi.

```
<form action="inserisci.asp" method="POST">
<table   width="460"   height="330"   border="0"   align="center"   cellspacing="3"
bgcolor="#FFFF99">
  <tr>
                    <td     width="137"     height="56"     bgcolor="#FFFF66"><span
class="Stile7">Oggetto:</span></td>
```

```
      <td width="313" bgcolor="#FFFF66">

        <div align="left">
          <input name="oggetto" type="text" id="oggetto" size="40" />
        </div></td>
    </tr>
    <tr>
      <td height="208" bgcolor="#FFFF66"><span class="Stile7">Messaggio:</span></td>
      <td bgcolor="#FFFF66">
        <div align="left">
          <textarea name="messaggio" cols="40" rows="10" id="messaggio"></textarea>
        </div></td>
    </tr>
    <tr>
      <td height="54" colspan="2" bgcolor="#FFFF66"><div align="center">
        <input type="submit" name="button" id="button" value="Invia" />
      </div></td>
    </tr>
  </table>
  </form>
```

In questo form si richiama la pagina inserisci.asp per ricevere i dati dovuti all'inserimento e inserirli effettivamente nel file messaggi.txt, con il solito opportuno formato.

Si noti quindi l'attributo action="..", del tag <form...

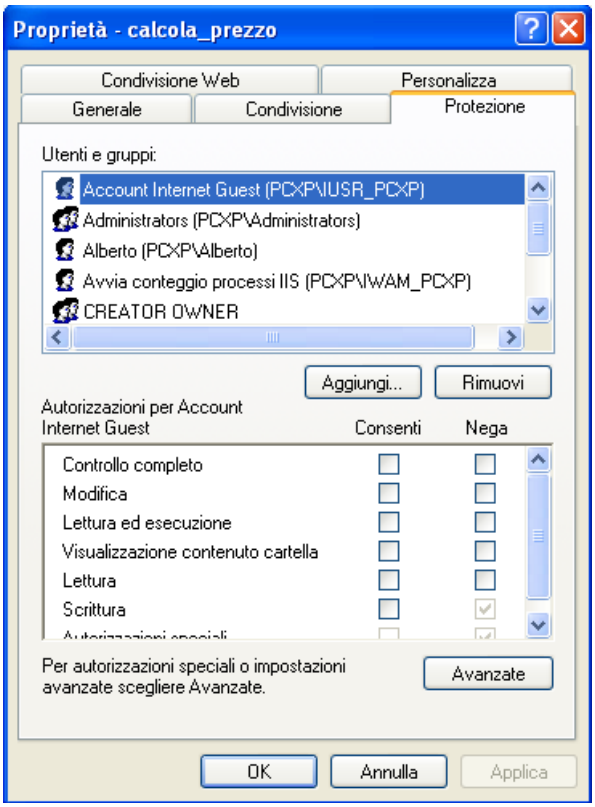
Infine la inserisci.asp, essendo una pagina in puro codice ASP è più leggibile e chiara:

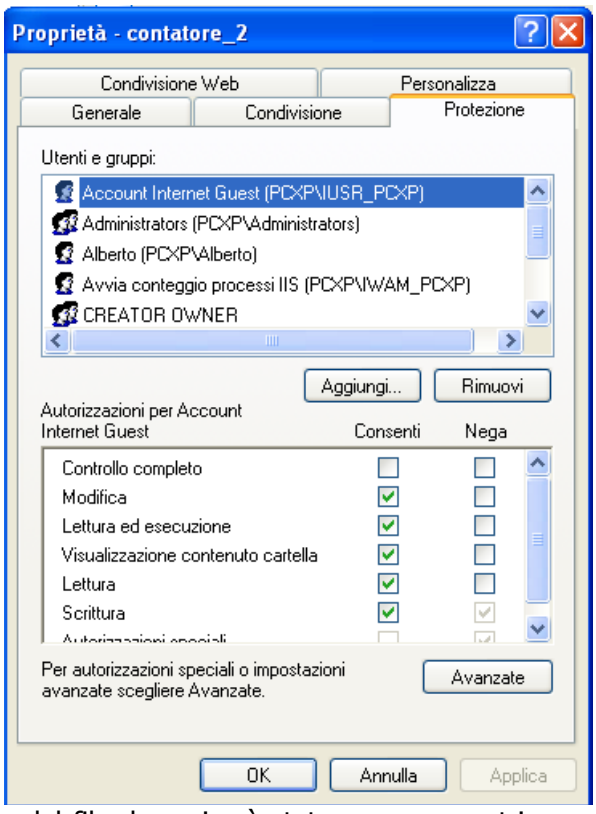
```
DataOra = Now()
Utente = "Anonimo"
Oggetto = Request.Form("Oggetto")

Messaggio = Request.Form("Messaggio")

Set Fso = Server.CreateObject("Scripting.FileSystemObject")
Set Flo = Fso.OpenTextFile(Server.MapPath(".") & "\messaggi.txt",8,True)

Flo.WriteLine DataOra
Flo.WriteLine Utente
Flo.WriteLine Oggetto
Flo.WriteLine Messaggio
Flo.WriteLine "----"

Flo.Close()

Response.Redirect "visualizza.asp"
```

Si tratta del codice per effettuare la lettura opportunamente fatta del file messaggi.txt; si noti anche che il codice effettua la scrittura in append, ossia in aggiunta del contenuto, altrimenti ogni intervento porterebbe a perdere tutti i precedenti interventi fatti. Il parametro finale della OpenTextFile indica che il file indicato, se non fosse esistente, và creato.

Si osservi anche che ricavare la data e ora corrente è facile in ASP, basta invocare la funzione Now che produce una stringa indicante data ed ora attuali (ossia data ed ora precise del momento della chiamata).

Anche in questo caso si preparano una serie di variabili i cui valori ci si predispone ad inserire ne file messaggi.txt; questi valori sono quasi tutti prelevati dal form della pagina visualizza.asp e vengono passati alle istruzioni di scrittura sul file. Seguono 4 istruzioni di scrittura (l'ultim delle quali effettua una scrittura multilinea sul file con una sola istruzione, successivamente avviene la chiusura dello stream di testo, ed infine il controllo viene "catapultato" sulla pagina ASP visualizza.asp, grazie ad un





redirect[7] opportuno. Si noti anche che la posizione del file da aprire è stata resa parametrica grazie all'istruzione MapPath(".") che ricaverà il path della directory corrente.

Un accorgimento da tenere in conto è che questo codice di solito non funziona (!). Infatti tutte le directory sono regolate con permessi in lettura per l'utente IUSR_<nome macchina>, ma non in scrittura (si osservi la prima a sinistra delle form di regolazione.

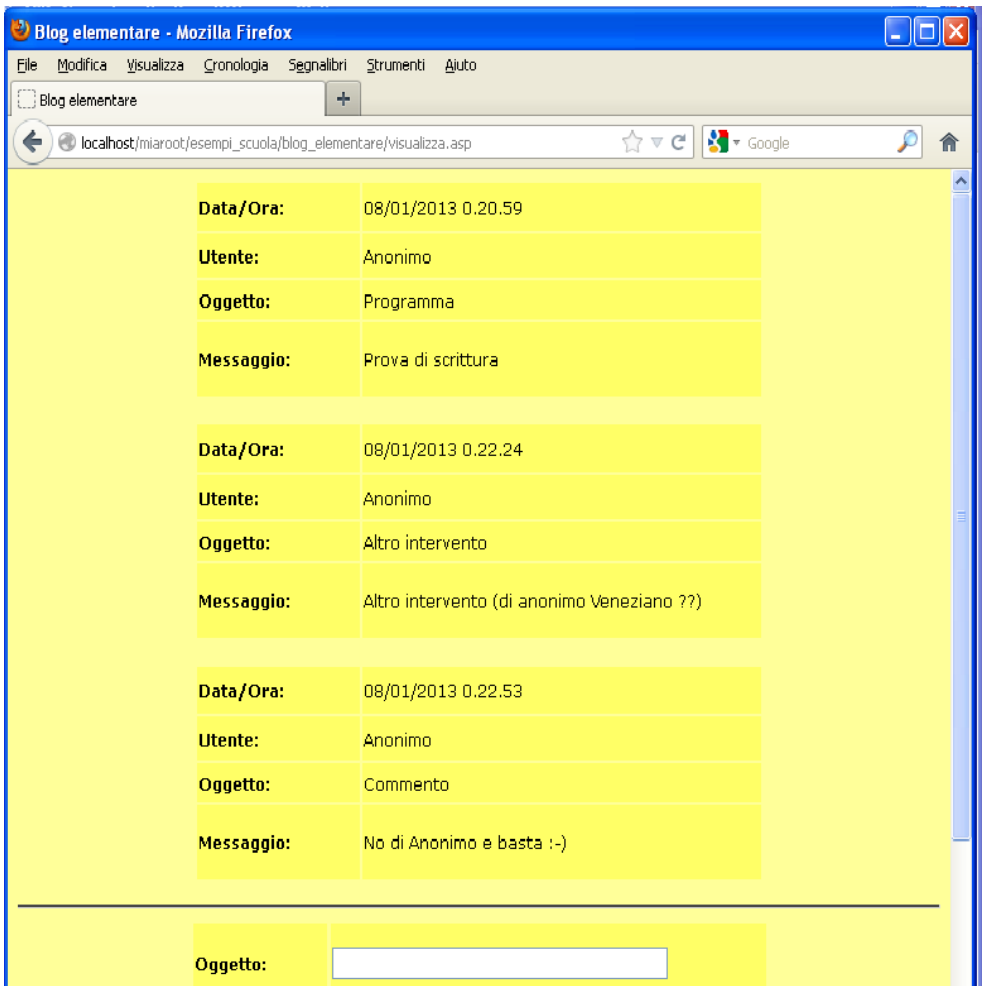
L'utente IUSR_ <nome macchina> rappresenta l'utente Web anonimo che opera attraverso ASP sulla macchina. Per effettuare l'allargamento dei permessi anche alle operazioni in lettura bisogna regolare opportunamente i permessi di IUSR_<nome macchina> (e quindi avere account che offrano questa possibilità).

Alla fine, effettuata questa regolazione, come indicata nella seconda immagine della form, a destra, sarà possibile scrivere file tramite ASP nella directory dove la regolazione è stata effettuata.

Alla fine la pagina del blog apparirà come in figura, con gli interventi su file e quindi al sicuro da spegnimenti della macchina server o altri problemi.

Riportiamo qui sotto una immagine della pagina del blog durante il funzionamento:

7 Utilizzando l'istruzione Response.Redirect(...)


Blog elementare - Mozilla Firefox
File Modifica Visualizza Cronologia Segnalibri Strumenti Aiuto
Blog elementare
localhost/miaroot/esempi_scuola/blog_elementare/visualizza.asp
Google
Data/Ora: 08/01/2013 0.20.59
Utente: Anonimo
Oggetto: Programma
Messaggio: Prova di scrittura
Data/Ora: 08/01/2013 0.22.24
Utente: Anonimo
Oggetto: Altro intervento
Messaggio: Altro intervento (di anonimo Veneziano ??)
Data/Ora: 08/01/2013 0.22.53
Utente: Anonimo
Oggetto: Commento
Messaggio: No di Anonimo e basta :-)
Oggetto: