A. Veneziani -

Analisi pagina ASP implementante il gioco di indovinare un numero tra 1 e 10

Il problema posto è far sì che una pagina ASP generi un numero intero e successivamente si disponga la pagina stessa di modo che possa assumere un input da parte dell'utente che cercherà di indovinare il numero stesso.

Possibili evoluzioni:

1. La pagina genera in modo incondizionato un numero random ad ogni nuovo caricamento della stessa e in essa a ciò segue del codice che recepisce l'input dell'utente e verifica se esso ha effettivamente indovinato il numero.
2. Possibile evoluzione: il numero non viene generato ad ogni caricamento, ma solo quando una apposita variabile di sessione risulta Empty. Da ciò risulta che:
   - La variabile di sessione viene "azzerata" ossia resa Empty solo quando il numero è stato indovinato o con apposito comando, che indica di inizializzare il gioco
   - E' possibile che vi siano quindi più tentativi durante i quali il numero da indovinare resta ovviamente lo stesso; esso permane sempre grazie alla variabile di sessione prescelta
3. Dopo lo step 2 è possibile pensare ad una ulteriore evoluzione che consenta di utilizzare un altra variabile come contatore dei tentativi fatti.

   Nota bene – E' opportuno che le operazioni di controllo ed eventuale incremento della variabile di conteggio siano effettuate solo nel caso che l'utente abbia dato un effettivo input e la pagina non sia semplicemente ricaricata.
4. Infine si potrebbe prevedere anche un limite di tentativi oltre ai quali il sistema potrebbe costringere l'utente ha partire con una nuova sessione di gioco in quanto non è riuscito ad indovinare il risultato nel numero di passi massimo previsto.
5. Altra evoluzione del semplice gioco potrebbe essere selezionare il numero da 1 a 10 da una combobox (box a discesa) che eviterà involontari errori nell'input (input vincolato).

In generale, relativamente al punto 1 la parte ASP si può suddividere in due parti: generazione del numero casuale e controllo di quanto inserito dall'utente.

La situazione 2 invece presuppone una serie di interventi e modifiche rispetto alla 1. Nella 2 il sistema non deve rigenerare un nuovo numero casuale se esso non è stato indovinato. Si può dire quindi che:

- Il numero casuale (anch'esso memorizzato in una variabile di sessione) non viene generato ad ogni passaggio. Se la apposita variabile di sessione indica che questa operazione non vada fatta il numero misterioso resta quello precedente.
- L'utente potrà più volte effettuare il controllo sullo stesso numero generato; il dato prodotto dall'utente viene recepito sulla stessa pagina e controllato.

Al punto 3 si ipotizza di iniziare a contare quante volte l'utente a cercato di indovinare il numero "misterioso". Tale numero di volte è inizializzato a 0 quando viene generato il numero casuale e

successivamente incrementato di una volta, ad ogni tentativo di indovinare il numero.

Dettagli sulla realizzazione del codice

Generatore casuale

Il generatore casuale si basa sulla funzione VbScriot **Rnd( )** e sulla Sub VbScript **Randomize( ).**

Rnd( ) genera i classici (per il BASIC) numeri casuali tra 0 e 1, tali numeri sono float, o meglio Single (il nome del tipo float a singola precisione del VbScript, ossia numeri con la virgola.

Essi quandi appaiono come 0,291192 0,0328337 0,583377 e così via.

Tali sequenze, come si è già constatato in laboratorio sono "casuali", ma ripetute, se non si inizializza il sistema generazione con un valore iniziale "casuale". Esso è spesso derivato dalla data di sistema o altro valore che possa essere il più possibile aleatorio.

Si è anche detto che in realtà tali numeri casuali sono una sequenza molto disordinata di numeri derivanti comunque da una specifica formula matematica ed è quindi per questo motivo che il computer è capace di riprodurre più volte tali sequenze nella stessa identica maniera. La casualità completa si ottiene grazie al comando Randomize che inserisce un valore di inizio, nel generatore, a sua volta "casuale" (a esempio una cifra derivata dal timer di sistema) e quindi permette di avere sequenze casuali ogni volta diverse.

Nel nostro caso la generazione di numeri da 1 a 10 si può ottenere con l'espressione:

```
Int(Rnd( ) * 10 ) + 1
```

La funzione Int effettua infatti un troncamento lasciando solo le cifre prima della virgola (cifre intere).

La moltiplicazione per 10 sposta le cifre rispetto alla virgola di un posto, ottenendo numeri da 0 a 9 compresi. A questo punto non resta che sommare 1 per ottenere numeri tra 1 e 10.

Controllo di coincidenza

E' una operazione piuttosto semplice, che viene fatta se esiste un input dell'utente (altrimenti non ha senso). Fatto l'opportuno controllo del tipo:

nel caso 1 il controlla farà uso di una variabile comune

```
...
If NumMist = CInt(Request.Form("n"))  Then
...
```

viceversa nel caso 2 e successivi, si deve utilizzare una variabile di sessione per mantenere il valore tra i vari ricaricamenti di pagina.

```
...
If Session("NumMist") = CInt(Request.Form("n"))  Then
...
```

Si verifica se il numero dato dall'utente (e opportunamente convertito in valore numerico) ed il numero casuale tra 1 e 10 memorizzato nella variabile di sessione NumMist (numero mistrerioso) coincidono

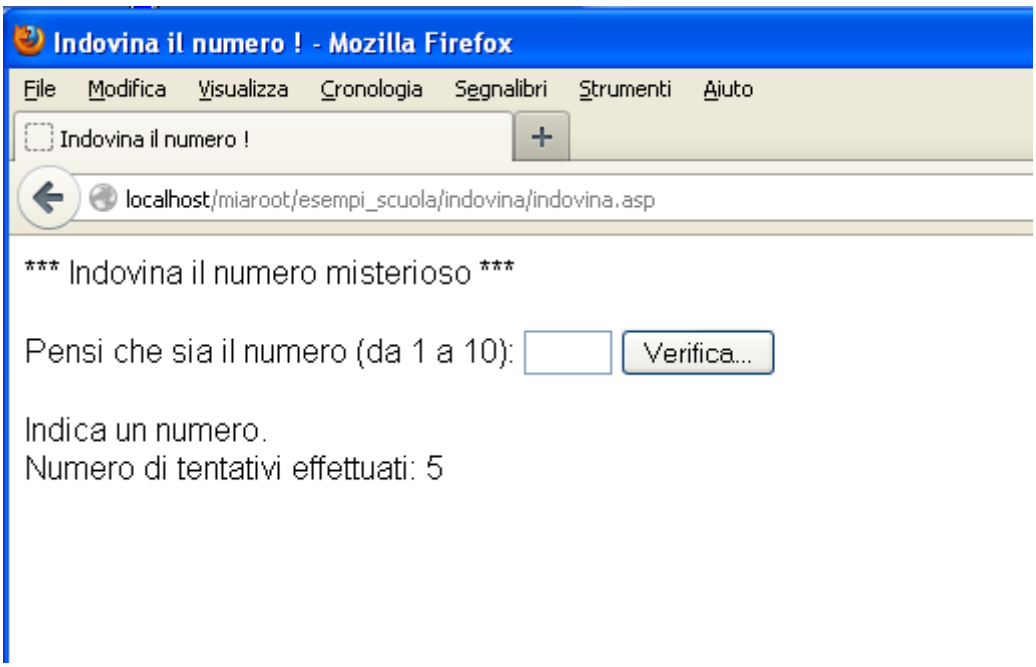
allora si segnala la cosa.
Inoltre, nel caso la variabile NumMist sia di sessione, si "azzera" la variabile stessa (di modo da forzare la rigenerazione di un nuovo numero casuale. Infatti nel caso 2 e successivi abbiamo detto che la generazione del numero casuale deve essere di nuovo sollecitata quando il precedente numero sia stato indovinato o in altri casi equivalenti. Questo può essere fatto mettendo ad Empty il valore della variabile di sessione Session("NumMist").
Volendo contare le volte necessarie ad indovinare il numero misterioso, si può inserire nello script un altra variabile di sessione (potremmo chiamarla Conta_Prove) che conta i test che sono stati fatti.
Un accortezza è inserire l'incremento della variabile stessa Session("Conta_Prove") nella condizione che il valore letto sia <> da "", ossia da stringa nulla. Questo si traduce in una istruzione del tipo:

```
...
If Request.Form("n") = "" Then
          <controllo di coincidenza del valore >
           Session("Conta_Prove") = Session("Conta_Prove") + 1
     End  If
...
```

Questo conteggio si azzera al momento di creare un nuovo numero da indovinare.
Il caso 4 può essere semplicemente implementato inserendo una condizione che produca una scritta di sconfitta nel caso il contatore dei tentativi abbia raggiunto un valore limite; questo ovviamente accade, per quanto detto in precedenza, solo se vengono falliti certo numero di tentativi.
Quindi sarà inserita la condizione:

```
...
   If Session("Conta_Prove") = 5 Then
         Response.Write "Hai perso.  Numero prove esaurito. <br><br>"
         Session("NumMist") = Empty
       Session("Conta_Prove") = 0
   End If
...
```

come sempre una condizione di terminazione del gioco (l'utente ha perso) comporta che le due variabili di sessione che contengono la situazione di gioco siano riportate ai valori iniziali.
Qui di seguito una possibile schermata della pagina in questione:


Indovina il numero ! - Mozilla Firefox
File Modifica Visualizza Cronologia Segnalibri Strumenti Aiuto
Indovina il numero !
localhost/miaroot/esempi_scuola/indovina/indovina.asp
*** Indovina il numero misterioso ***
Pensi che sia il numero (da 1 a 10):
Verifica...
Indica un numero.
Numero di tentativi effettuati: 5