sintesi su LINGUAGGIO e OGGETTI ASP

Creazione di una pagina ASP

Un file di Active Server Pages (ASP) è un file di testo con estensione asp contenente una qualsiasi combinazione dei seguenti elementi:

- Testo
- Tag HTML
- Script sul lato server

Per creare rapidamente un file asp è possibile rinominare i file HTML sostituendo l'estensione del nome di file htm o html esistente con l'estensione asp. Se il file in questione non contiene funzionalità ASP, il server tralascerà la fase di elaborazione degli script ASP e invierà il file al client rapidamente. Ciò consente una notevole flessibilità agli sviluppatori Web, in quanto essi possono assegnare un'estensione asp ai file anche se prevedono di aggiungere le funzionalità ASP solo in un secondo momento.
Per pubblicare un file asp sul Web, salvare il nuovo file in una directory virtuale sul sito Web, assicurandosi che per tale directory sia stata impostata l'autorizzazione di esecuzione o creazione di script. Aprire quindi il file dal browser digitando il relativo URL. Si ricordi che le pagine ASP devono essere elaborate dal server e non è quindi possibile richiedere un file asp digitando il relativo percorso fisico. Dopo che il file è stato caricato nel browser, si noterà che il server ha restituito una pagina HTML. A prima vista ciò potrebbe sembrare strano, ma si tenga presente che il server analizza ed esegue tutti gli script sul lato server di ASP prima dell'invio del file. L'utente riceverà sempre la pagina in formato HTML standard. Per creare i file asp è possibile utilizzare qualsiasi editor di testo

Aggiunta di comandi script sul lato server

Uno script sul lato server è costituito da una serie di istruzioni che consentono di inoltrare i comandi al server Web in ordine sequenziale. Chi ha già sviluppato siti Web avrà probabilmente familiarità con gli script sul lato client, eseguiti nel browser. Nei file asp gli script sono distinti dal testo e dai tag HTML tramite delimitatori. Un *delimitatore* è un carattere o una sequenza di caratteri che contrassegna l'inizio o la fine di un'unità. Nel caso del linguaggio HTML tali delimitatori sono i simboli di minore (<) e maggiore (>), che racchiudono i tag HTML.
ASP utilizza i delimitatori <% e %> per racchiudere i comandi script. È possibile inserire tra i delimitatori qualsiasi comando valido per il linguaggio di script in uso. Nell'esempio seguente viene illustrata una semplice pagina HTML contenente un comando script:

```
<HTML>
  <BODY>
  Ultimo aggiornamento della pagina: <%= Now() %>.
  </BODY>
</HTML>
```

La funzione **Now()** di VBScript restituisce la data e ora correnti. Quando il server Web elabora questa pagina, sostituisce `<%= Now() %>` con la data e l'ora correnti e restituisce la pagina al browser con questo risultato:

```
Ultimo aggiornamento della pagina: 29/01/99 14.20.00.
```

I comandi inseriti tra i delimitatori sono denominati *comandi script primari*. Tali comandi vengono elaborati utilizzando il linguaggio di script primario. Tutti i comandi utilizzati all'interno di delimitatori di script devono essere validi per il linguaggio di script primario. Per impostazione predefinita il linguaggio di script primario è VBScript, ma è possibile impostare un linguaggio predefinito diverso. Per informazioni, vedere Utilizzo di linguaggi di script.
Se si ha già familiarità con gli script sul lato client, si conoscerà il tag HTML <SCRIPT> utilizzato per racchiudere i comandi script e le espressioni. Il tag <SCRIPT> consente inoltre di creare script sul lato server ogni volta che è necessario definire procedure in più linguaggi all'interno di un file asp. Per ulteriori informazioni, vedere Utilizzo di linguaggi di script.

Combinazione di testo HTML e comandi script

Tra i delimitatori ASP è possibile racchiudere qualsiasi istruzione, espressione, procedura o operatore valido per il linguaggio di script primario in uso. Un'*istruzione*, in VBScript e altri linguaggi di script, è un'unità completa da un punto di vista sintattico che esprime un tipo di azione, dichiarazione o definizione. L'istruzione condizionale **If...Then...Else** riportata di seguito è un'istruzione VBScript comune.

```
<%
  Dim dtmHour
  dtmHour = Hour(Now())
  If dtmHour < 12 Then
    strGreeting = "Buongiorno!"
  Else
    strGreeting = "Salve!"
  End If
```

```
%>

<%= strGreeting %>
```

In base all'ora corrente, questo script assegna il valore "Buongiorno!" o "Salve!" alla variabile stringa strGreeting. L'istruzione <%= strGreeting %> invia al browser il valore corrente della variabile.
Un utente che visualizza questo script prima di mezzogiorno (in base al fuso orario del server Web) vedrà quindi la riga di testo seguente:

```
Buongiorno!
```

Un utente che visualizza questo script a mezzogiorno o dopo vedrà la riga di testo seguente:

```
Salve!
```

È possibile includere testo HTML tra le sezioni di un'istruzione. Lo script seguente, che utilizza un testo HTML all'interno di un'istruzione **If...Then...Else**, genera lo stesso risultato dello script dell'esempio precedente:

```
<%
  Dim dtmHour
  dtmHour = Hour(Now())
  If dtmHour < 12 Then
%>
  Buongiorno!
<% Else %>
  Salve!
<% End If %>
```

Se la condizione è vera, ovvero se l'ora è precedente al mezzogiorno, il server Web invierà al browser il testo HTML che segue la condizione ("Buongiorno"). In caso contrario, invierà al browser il testo HTML che segue **Else** ("Salve!").
Questo tipo di combinazione di testo HTML e comandi script risulta utile per racchiudere in un'istruzione **If...Then...Else** più righe di testo HTML. L'esempio precedente risulta invece maggiormente utile quando si desidera visualizzare un messaggio di benvenuto in più punti della pagina Web. È possibile impostare il valore della variabile una sola volta e visualizzarlo ripetutamente.
Anziché combinare il testo HTML con comandi script, è possibile restituire il testo HTML al browser dall'interno di un comando script. Per restituire il testo al browser, utilizzare l'oggetto ASP predefinito **Response**. L'esempio seguente genera lo stesso risultato degli script precedenti:

```
<%
  Dim dtmHour

  dtmHour = Hour(Now())

  If dtmHour < 12 Then
    Response.Write "Buongiorno!"
  Else
    Response.Write "Salve!"
  End If
%>
```

Response.Write invia al browser il testo che segue. Utilizzare **Response.Write** all'interno di un'istruzione quando si desidera definire in modo dinamico il testo restituito al browser. Potrebbe ad esempio essere necessario creare una stringa contenente i valori di più variabili. Per ulteriori informazioni sull'oggetto **Response** e gli oggetti in generale, vedere Utilizzo di componenti e oggetti e Invio di contenuto al browser. Per il momento si tenga semplicemente presente che esistono vari modi per inserire comandi script in una pagina HTML.
Tra i delimitatori ASP è possibile includere procedure scritte nel linguaggio di script primario. Per ulteriori informazioni, vedere Utilizzo di linguaggi di script.
Oppure

Utilizzo di istruzioni ASP

ASP include istruzioni di output e di elaborazione che non fanno parte del linguaggio di script in uso.
L'*istruzione di output* <%= *espressione* %> visualizza il valore di un'espressione. Questa istruzione di output equivale all'utilizzo di **Response.Write** per la visualizzazione delle informazioni. L'espressione di output <%= città %> visualizza ad esempio Roma (il valore corrente della variabile) nel browser.
L'*istruzione di elaborazione* <%@ *parola chiave* %> fornisce ad ASP le informazioni necessarie per elaborare un file asp. L'istruzione seguente imposta ad esempio VBScript come linguaggio di script primario della pagina:

```
<%@ LANGUAGE=VBScript %>
```

L'istruzione di elaborazione deve essere inserita nella prima riga di un file asp. Se si desidera aggiungere più istruzioni a una pagina, le istruzioni dovranno essere racchiuse entro lo stesso delimitatore

Utilizzo di variabili e costanti

Assegnazione dell'ambito a livello di sessione o applicazione

Le variabili globali sono accessibili in un singolo file asp. Se si desidera rendere accessibile una variabile anche all'esterno di una singola pagina, assegnare alla variabile l'ambito a livello di sessione o applicazione. Le variabili con ambito a livello di sessione sono disponibili per tutte le pagine di un'applicazione ASP richieste da un particolare utente. Le variabili con ambito a livello di applicazione sono disponibili per tutte le pagine di un'applicazione ASP richieste da qualsiasi utente. Le prime consentono di memorizzare informazioni relative a un singolo utente, quali le preferenze oppure il nome o l'identificazione dell'utente. Le seconde consentono di memorizzare informazioni relative a tutti gli utenti di una particolare applicazione, quali il messaggio visualizzato da un'applicazione all'accesso o i valori generali richiesti dall'applicazione.
ASP fornisce due oggetti predefiniti nei quali è possibile memorizzare le variabili, ovvero l'oggetto **Session** e l'oggetto **Application**.

Ambito a livello di sessione

Per assegnare a una variabile l'ambito a livello di sessione, memorizzarla nell'oggetto **Session** specificando un valore per la voce denominata dell'oggetto. I comandi seguenti memorizzano ad esempio due nuove variabili nell'oggetto **Session**:

```
<%
  Session("Nome") = "Giovanni"
  Session("Cognome") = "Giudici"
%>
```

Per recuperare informazioni dall'oggetto **Session**, accedere alla voce con nome utilizzando l'istruzione di output (<%=) o **Response.Write**. Nell'esempio seguente viene utilizzata l'istruzione di output per visualizzare il valore corrente di Session("Nome"):

```
Benvenuto <%= Session("Nome") %>
```

È possibile memorizzare le preferenze dell'utente nell'oggetto **Session**, quindi accedere a tali preferenze per determinare la pagina da restituire all'utente. È ad esempio possibile consentire a un utente di specificare una versione di solo testo del contenuto della prima pagina dell'applicazione e applicare tale scelta a tutte le pagine successive che l'utente visiterà in questa applicazione.

```
<%
  strScreenResolution = Session("ScreenResolution")
  If strScreenResolution = "Low" Then
%>
  Versione di solo testo della pagina.
<% Else %>
  Versione multimediale della pagina.
<% End If %>
```

Nota Se all'interno di uno script si fa più volte riferimento a una variabile con ambito a livello di sessione, è consigliabile ricorrere all'assegnazione a una variabile locale, come nell'esempio precedente, per ottenere un miglioramento delle prestazioni.

Ambito a livello di applicazione

Per assegnare a una variabile l'ambito a livello di applicazione, memorizzarla nell'oggetto **Application** specificando un valore per una voce denominata dell'oggetto. Il comando seguente memorizza ad esempio il messaggio visualizzato da un'applicazione nell'oggetto **Application**:

```
<% Application("Saluto") = "Benvenuti nel reparto vendite." %>
```

Per recuperare informazioni dall'oggetto **Application**, utilizzare l'istruzione di output (<%=) di ASP o **Response.Write** per accedere alla voce denominata da tutte le pagine successive dell'applicazione. Nell'esempio seguente viene utilizzata l'istruzione di output per visualizzare il valore di Application("Saluto"):

```
<%= Application("Saluto") %>
```

Anche in questo caso, se lo script fa più volte riferimento a una variabile con ambito a livello di applicazione, è consigliabile ricorrere all'assegnazione a una variabile locale per ottenere un miglioramento delle prestazioni.

Nell'esempio seguente `adOpenKeyset` e `adLockOptimistic` sono costanti ADO:

Interazione con script sul lato client

Utilizzando ASP per la generazione e la modifica di script sul lato client è possibile estendere l'efficacia di questo ambiente. È ad esempio possibile scrivere script sul lato server che assemblino gli script sul lato client in base a variabili specifiche del server, il tipo di browser dell'utente o i parametri delle richieste HTTP.

Integrando istruzioni di script sul lato server all'interno di script sul lato client (racchiusi tra tag HTML <SCRIPT>), come indicato nel modello di esempio seguente, è possibile inizializzare e modificare gli script sul lato client in modo dinamico al momento della richiesta:

```
<SCRIPT LANGUAGE="VBScript">
<!--

variabile = <%=valore definito dal server %>
.
.
script sul lato client

<% script sul lato server utilizzato per generare un'istruzione sul lato
client %>

script sul lato client
.
.
-->
</SCRIPT>
```

L'incorporamento di tali funzionalità consente utili e interessanti applicazioni. Di seguito è ad esempio riportato un

Elaborazione dell'input dell'utente

L'oggetto **Request** di ASP consente di creare script semplici ma potenti per la raccolta e l'elaborazione dei dati raccolti tramite moduli HTML. In questo argomento, non solo si imparerà a creare script semplici per l'elaborazione dei moduli, ma si apprenderanno anche le tecniche necessarie per la convalida delle informazioni dei moduli, sia sul server Web che nel browser dell'utente.

Informazioni sui moduli HTML

I moduli HTML, il metodo più diffuso per la raccolta di informazioni basate sul Web, sono costituiti da tag HTML speciali opportunamente disposti che visualizzano elementi dell'interfaccia utente in una pagina Web. Le caselle di testo, i pulsanti e le caselle di controllo sono tutti esempi di elementi che permettono agli utenti di interagire con una pagina Web e inviare informazioni a un server Web.
I tag HTML seguenti generano ad esempio un modulo in cui un utente può immettere il nome, il cognome e l'età e contenente un pulsante per l'invio delle informazioni a un server Web. Questo modulo contiene anche un tag di input nascosto, non visualizzato dal browser, che consente di trasmettere ulteriori informazioni a un server Web.

```
<FORM METHOD="Post" ACTION="Profile.asp">
<INPUT TYPE="Text" NAME="FirstName">
<INPUT TYPE="Text" NAME="LastName">
<INPUT TYPE="Text" NAME="Age">
<INPUT TYPE="Hidden" NAME="UserStatus" VALUE="Nuovo">
<INPUT TYPE="Submit" VALUE="Invia">
</FORM>
```

Un'analisi completa dei tag dei moduli HTML esula dagli obiettivi di questo argomento, ma esistono numerose fonti di informazioni a cui è possibile attingere per imparare a creare moduli HTML utili ed efficaci. La funzione del browser per la visualizzazione del codice sorgente consente ad esempio di esaminare i moduli HTML creati in altri siti Web.

Elaborazione degli input dei moduli con ASP

Dopo aver creato un modulo HTML, è necessario elaborare l'input dell'utente, ossia inviare le informazioni a un file asp per l'analisi e la modifica. Esaminare ancora una volta il codice HTML dell'esempio precedente. Si noti che l'attributo ACTION del tag <FORM> fa riferimento a un file denominato Profile.asp. Quando l'utente invia le informazioni in formato HTML, il browser utilizza il metodo POST per inviare tali informazioni a un file asp sul server, in questo caso il file Profile.asp. Il file asp può contenere script che elaborano le informazioni e interagiscono con altri script, componenti COM o risorse, quali un database.
Esistono tre modi principali per raccogliere le informazioni dai moduli HTML tramite ASP:

- Un file htm statico può contenere un modulo che invia i relativi valori a un file asp.
- Un file asp può creare un modulo che invia le informazioni a un altro file asp.
- Un file asp può creare un modulo che invia le informazioni a se stesso, ossia al file asp contenente il modulo.

I primi due metodi indicati funzionano allo stesso modo dei moduli che interagiscono con programmi di altri server Web, fatta eccezione per il fatto che in ASP le operazioni di raccolta ed elaborazione delle informazioni dei moduli sono notevolmente semplificate. Il terzo metodo è particolarmente utile e verrà descritto nella sezione Convalida dell'input dei moduli.

Raccolta dell'input dei moduli

L'oggetto **Request** di ASP fornisce due insiemi che semplificano la raccolta delle informazioni dei moduli inviate sotto forma di richiesta di URL.

L'insieme QueryString

L'insieme **QueryString** recupera i valori dei moduli passati al server Web sotto forma di testo posizionato dopo un punto di domanda nell'URL richiesto. I valori del modulo possono essere aggiunti all'URL richiesto utilizzando il metodo HTTP GET oppure manualmente.

Se ad esempio il modulo dell'esempio precedente utilizzasse il metodo GET (METHOD="GET") e l'utente digitasse *Giovanni*, *Giudici* e *30*, al server verrebbe inviata la richiesta di URL seguente:

```
http://Reskit/Workshop1/Painting/Profile.asp?FirstName=Giovanni&LastName=Giudici&Age=30&UserStatus=New
```

Profile.asp potrebbe contenere il seguente script per l'elaborazione dei moduli:

```
Buongiorno <%= Request.QueryString("FirstName") %> <%=
Request.QueryString("LastName") %>.
Tu hai <%= Request.QueryString("Age") %> anni.

<%
  If Request.QueryString("UserStatus") = "New" Then
    Response.Write "Questa è la tua prima visita al sito."
  End if
%>
```

In tal caso il server Web restituirebbe il testo seguente al browser dell'utente:

```
Buongiorno Giovanni Giudici. Tu hai 30 anni. Questa è la tua prima visita al sito.
```

L'insieme **QueryString** offre inoltre un parametro facoltativo che consente di accedere a uno dei vari valori visualizzati nella richiesta di URL (utilizzando il metodo GET). È inoltre possibile utilizzare la proprietà **Count** per contare il numero di volte per cui viene visualizzato un tipo specifico di valore.

Un modulo contenente una casella di riepilogo con vari elementi potrebbe ad esempio generare la richiesta seguente:

```
http://Reskit/CibiOrganici/elenco.asp?Food=Mele&Food=Olive&Food=Pane
```

Per contare più valori si potrebbe utilizzare il comando seguente:

```
Request.QueryString("Food").Count
```

Per visualizzare vari tipi di valori, Elenco.asp potrebbe contenere lo script seguente:

```
<%
  lngTotal = Request.QueryString("Food").Count
  For i = 1 To lngTotal
    Response.Write Request.QueryString("Food")(i) & "<BR>"
  Next
%>
```

Lo script precedente visualizzerebbe:

```
Mele
Olive
Pane
```

È inoltre possibile visualizzare l'intero elenco di valori sotto forma di stringa delimitata da virgole utilizzando:

```
<% Response.Write Request.QueryString("Item") %>
```

In tal caso verrebbe visualizzata la seguente stringa:

```
Mele, Olive, Pane
```

Raccolta di moduli

Quando si utilizza il metodo HTTP GET per passare a un server Web valori di moduli lunghi e complessi, si corre il rischio di perdere parte delle informazioni. Alcuni server Web tendono a limitare le dimensioni della stringa della query dell'URL, in modo che i valori di moduli lunghi passati con il metodo GET possano essere troncati. Se è necessario inviare una grande quantità di informazioni da un modulo a un server Web, utilizzare il metodo HTTP POST. Questo metodo, che invia i dati dei moduli nel corpo della richiesta HTTP, è in grado di trasmettere a un server un numero illimitato di caratteri. Per recuperare i valori inviati con il metodo POST, è possibile utilizzare l'insieme **Form** dell'oggetto **Request** di ASP.

L'insieme **Form** memorizza i valori in maniera analoga all'insieme **QueryString**. Se, ad esempio, un utente ha compilato un modulo specificando un lungo elenco di nomi, per recuperare i nomi è possibile utilizzare lo script seguente:

```
<%
  lngTotal = Request.Form("Food").Count
```

```
  For i = 1 To lngTotal
   Response.Write Request.Form("Food")(i) & "<BR>"
  Next
%>
```

Convalida dell'input dei moduli

Un modulo Web ben progettato spesso include uno script sul lato client che convalida l'input dell'utente prima dell'invio delle informazioni al server. Gli *script di convalida* sono ad esempio in grado di verificare se l'utente ha immesso un numero valido o se una casella di testo è stata lasciata vuota. Se, ad esempio, il sito Web contiene un modulo che consente agli utenti di calcolare il tasso di rendimento di un investimento, sarà necessario verificare che l'utente abbia inserito i valori numerici o testuali nei campi appropriati per evitare che vengano inviate al server informazioni non valide.
In genere è preferibile eseguire tutte le operazioni di convalida possibili sul lato client. Oltre a informare gli utenti più rapidamente degli errori di input, la convalida sul lato client favorisce tempi di risposta migliori, riduce il carico di lavoro dei server e rende disponibile larghezza di banda per altre applicazioni.
Lo script sul lato client seguente convalida l'input dell'utente, verificando che il numero di conto immesso dall'utente sia effettivamente un numero, prima di inviare le informazioni al server:

Oggetto Response

È possibile utilizzare l'oggetto **Response** per l'invio di output al client.
Sintassi
Response.*collection*|*property*|*method*

Insiemi

Cookies	Specifica i valori dei cookie. Tali valori possono essere impostati utilizzando questo insieme.

Metodi

Write	Scrive una variabile o un testo nell'output HTTP corrente in forma di stringa.

Oggetto Request

L'oggetto **Request** recupera i valori passati dal browser client al server durante una richiesta HTTP.
Sintassi
Request[.*collection*|*property*|*method*](*variable*)

Insiemi

Cookies	I valori dei cookie inviati nella richiesta HTTP.
Form	I valori degli elementi dei moduli inclusi nel corpo della richiesta HTTP.
QueryString	I valori delle variabili incluse nella stringa di richiesta HTTP.
ServerVariables	I valori delle variabili di ambiente predeterminate.

Metodi

BinaryRead	Recupera i dati inviati dal client al server nell'ambito di una richiesta POST.

I parametri delle variabili sono stringhe che specificano l'elemento da recuperare da un insieme o da utilizzare come input per un metodo o una proprietà. Per ulteriori informazioni sul parametro *variable*, vedere le descrizioni dei singoli insiemi.
Osservazioni
Se la variabile specificata non è inclusa in nessuno dei cinque insiemi precedenti, l'oggetto **Request** restituisce EMPTY.

Se in più insiemi esiste una variabile con lo stesso nome, l'oggetto **Request** restituisce la prima istanza rilevata dall'oggetto.

Oggetto Session

È possibile utilizzare l'oggetto **Session** per memorizzare le informazioni necessarie per una particolare sessione utente. Le variabili memorizzate nell'oggetto **Session** non vengono eliminate quando l'utente passa tra pagine diverse dell'applicazione. Tali variabili, infatti, sono permanenti per l'intera sessione utente.
Quando viene richiesta una pagina Web dell'applicazione da un utente che non ha ancora aperto una sessione, il server Web crea automaticamente un oggetto **Session**. Quando la sessione scade o viene terminata, il server elimina l'oggetto **Session**.

È possibile memorizzare valori nell'oggetto **Session**. Le informazioni memorizzate nell'oggetto **Session** sono disponibili per tutta la sessione e hanno ambito a livello di sessione. Lo script seguente illustra la memorizzazione di due tipi di variabili.

```
<%
Session("username") = "Janine"
Session("age") = 24
%>
```

Oggetto Server

L'oggetto **Server** consente di accedere ai metodi e alle proprietà del server, la maggior parte dei quali vengono utilizzati come funzioni di utilità.

Sintassi

Server.*property*|*method*

Proprietà

ScriptTimeout	Il tempo di esecuzione di uno script prima del timeout.

Metodi

CreateObject	Crea un'istanza di un componente del server.
Execute	Esegue un file asp.