ESEMPI SOLUZIONI DA TRASFORMARE IN JSCRIPT (JavaScript)
Versione provvisoria

JS1

```
<html>
<head>
<base fptype="TRUE">
<meta http-equiv="Content-Language"
content="it">
<meta http-equiv="Content-Type"
content="text/html; charset=windows-1252">
<title>Primo esempio di script</title>
</head>

<body>
<h1> Primo esempio di script </h1>
<script language=javaScript>
window.alert "Ciao a tutti"
</script>
</body>
</html>
```

JS2 - file

```
<html>
<head>
<meta http-equiv="Content-Language" content="it">
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1252">
</head>
<body>
<h1 align="center">Secondo esempio di script</h1>
<script language=javaScript>
<!--
window.open
"Saluto.htm","","width=300,height=200,top=200,left=20
0"
-->
</script>
</body>
</html>
```

Saluto

```
<html>
<head>
<meta http-equiv="Content-Language" content="it">
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1252">
<title>Primo esempio di script</title>
</head>
<body>
<h1 align="center"> CIAO A TUTTI </h1>
</body>
</html>
```

JS3

```
<html>
<head>
<meta http-equiv="Content-Language" content="it">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Oggi è il</title>
<script id=clientEventHandlersVBS language=javaScript>
<!--
document.write "Oggi è il: " & date & " e sono le ore " & time()

Sub BtnInvia_onclick
Dim messaggio
messaggio = ""
if chk1.checked then
      messaggio = CHK1.value & chr(13) & chr(10)
end if
if chk2.checked then
      messaggio = messaggio &  chk2.value & chr(13) & chr(10)
end if
if chk3.checked then
      messaggio = messaggio &  chk3.value & chr(13) & chr(10)
end if
if chk4.checked then
      messaggio = messaggio & chk4.value & chr(13) & chr(10)
end if
if chk5.checked then
      messaggio = messaggio & chk5.value & chr(13) & chr(10)
end if
if messaggio = "" then
      msgbox "Non pratichi NESSUNO sport"
else
      msgbox "Gli sport praticati sono: " & messaggio
end if
End Sub
-->
</script>
</head>

<body>

<p><b><font size="4">Quali sport pratichi?
</font></b>
```

```
</p>
<p>
<INPUT type="checkbox" name="CHK1" id="CHK1" value="Calcio" >Calcio
<input type="checkbox" name="CHK2" value="Pallavolo">Pallavolo
<input type="checkbox" name="CHK3" value="Basket">Basket
<input type="checkbox" name="CHK4" value="Nuoto">Nuoto
<input type="checkbox" name="CHK5" value="Altri">Altri</p>
<input type="button" value="Invio" name="BtnInvia">
<input type="button" value="Annulla" name ="BtnAnnulla"></p>
</body>
</html>
```

JS4

```
<html>
<head>
<title>Acquisizione dati</title>
<meta http-equiv="Content-Language" content="it">
<meta name="GENERATOR" content="Microsoft FrontPage 6.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
</script>
<script id=clientEventHandlersVBS language=javaScript>
<!--
OPTION EXPLICIT
Dim  strTitolo
Dim bValido
Sub window_onload
      StrTitolo = "Analisi Mercato"
      btnNuovo_onclick
End Sub
Sub btnNuovo_onclick
      txtNome.value = ""
      txtIndirizzo.value =""
      txtCap.value =""
      txtCitta.value=""
      txtProvincia.value =""
End Sub
Sub btnInvia_onclick
bValido = true
VerificaCorrettezza txtNome.value, "indicare il Nome"
VerificaCorrettezza txtIndirizzo.value, "Indicare l'indirizzo"
VerificaCorrettezza TxtCitta.value , "Indicare la citta"
VerificaCorrettezza txtProvincia.value , "Indicare la provincia"
VerificaCorrettezza txtCAP.value , "Indicare il CAP"
End Sub

Sub VerificaCorrettezza (byval strFieldValue, byVal strMsg)
if strFieldValue = "" and bvalido = true then
      msgbox strMsg
      bVAlido = false
end if
end sub
--></script>
</head>
<body>
<h1>Acquisizione nominativi</h1>
<h3>Nominativo da segnalare</h3>
<pre>Cognome e Nome <input type="text" size="20" name="txtNome">
Indirizzo <input type="text" size="20" name="txtIndirizzo">
CAP <input type="text" size="20" name="txtCAP">
Città <input type="text" size="20" name="TxtCitta">
Provincia <input type="text" size="20" name="txtProvincia"></pre>
<h3>Nuovo nominativo o già conosciuto</h3>
<input type="radio" name="OptOccasion" value="Nuovo" checked>Nuovo
<input type="radio" name="OptOccasion" value="Conosciuto">Conosciuto
<input type="button" value="Invia" name="btnInvia" >
<input type="button" value="Annulla" name="BtnAnnulla">
<input type="button" value="Nuovo" name="btnNuovo">
</body>
</html>
</html>
```

ATTENZIONE E' DA MODIFICARE..E' IN 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. A mano a mano che si acquisirà maggiore familiarità, potrebbe risultare più produttivo utilizzare un editor con un supporto avanzato per ASP, quale Microsoft® Visual InterDev™.

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.

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 linguaggi di script

I linguaggi di programmazione quali Visual Basic, C++ e Java forniscono un accesso di basso livello alle risorse del computer e consentono di creare programmi complessi di grandi dimensioni. I linguaggi di script, invece, consentono di creare programmi con capacità limitate, denominati *script*, che eseguono funzioni del sito Web su un browser o un server Web. A differenza dei più complessi linguaggi di programmazione, i linguaggi di script vengono *interpretati*, il che significa che le istruzioni vengono eseguite in ordine sequenziale da un programma intermediario denominato interprete dei comandi. Nonostante l'interpretazione riduca l'efficienza di esecuzione, i linguaggi di script sono facili da apprendere e forniscono potenti funzionalità. Gli script possono essere incorporati nelle pagine HTML per formattarne il contenuto oppure possono essere utilizzati per implementare componenti COM che incapsulano la logica aziendale avanzata.

ASP consente agli sviluppatori Web di scrivere script eseguibili sul server in vari linguaggi di script. All'interno di un singolo file asp è infatti possibile utilizzare più linguaggi di script. Inoltre, poiché gli script vengono letti ed elaborati sul lato server, non è necessario che il browser che richiede il file asp supporti gli script.
È possibile utilizzare qualsiasi linguaggio di script per il quale è installato il modulo di script appropriato nel server Web. Un *modulo di script* è un programma che elabora i comandi scritti in un particolare linguaggio. Con ASP vengono forniti due moduli di script: Microsoft Visual Basic Scripting Edition (VBScript) e Microsoft JScript. È tuttavia possibile installare e utilizzare moduli di altri linguaggi di script, quali REXX, PERL e Python.
Gli sviluppatori Visual Basic potranno iniziare sin da subito a utilizzare VBScript, un sottoinsieme di Visual Basic. Gli sviluppatori Java, C o C++ noteranno similitudini con la sintassi JScript, sebbene JScript non sia direttamente correlato a Java o C.

Impostazione del linguaggio di script primario

Il *linguaggio di script primario* di ASP è il linguaggio utilizzato per elaborare i comandi tra i delimitatori <% e %>. Per impostazione predefinita il linguaggio di script primario è VBScript. È possibile utilizzare come linguaggio di script primario qualsiasi altro linguaggio di script per cui è disponibile un modulo di script. È inoltre possibile impostare il linguaggio di script primario per ogni singola pagina oppure per tutte le pagine di un'applicazione ASP.

Impostazione del linguaggio per un'applicazione

Per impostare il linguaggio di script primario per tutte le pagine di un'applicazione, impostare la proprietà **Linguaggio ASP predefinito** nella scheda **Opzioni applicazioni** dello snap-in Internet Information Services. Per ulteriori informazioni, vedere Configurazione di applicazioni ASP.

Impostazione del linguaggio per una pagina

Per impostare il linguaggio di script primario per una singola pagina, aggiungere l'istruzione `<%@ LANGUAGE %>` all'inizio del file asp. La sintassi di tale istruzione è la seguente:

```
<%@ LANGUAGE=LinguaggioDiScript %>
```

dove *LinguaggioDiScript* è il linguaggio di script primario che si desidera impostare per quella particolare pagina.
L'impostazione del linguaggio per una pagina ha priorità rispetto all'impostazione globale per tutte le pagine dell'applicazione.
Attenersi alle istruzioni per l'utilizzo di un'istruzione ASP. Per ulteriori informazioni, vedere Creazione di una pagina ASP.
Nota Per utilizzare come linguaggio di script primario un linguaggio che non supporta la sintassi **Oggetto.Metodo**, è prima necessario creare la chiave del Registro di sistema **LanguageEngines**. Per ulteriori informazioni, vedere Informazioni sul Registro di sistema.

Utilizzo di VBScript e JScript sul server

Quando si utilizza VBScript sul server con ASP, vengono disattivate due funzioni VBScript. Poiché gli script scritti con ASP vengono eseguiti sul server, le istruzioni VBScript che visualizzano gli elementi dell'interfaccia utente, **InputBox** e **MsgBox**, non sono supportate.

Inserimento di commenti

Poiché in ASP l'elaborazione di tutti gli script viene eseguita sul lato server, non è necessario includere tag HTML di commento per nascondere gli script ai browser che non supportano gli script, come viene spesso fatto con gli script sul lato client. Tutti i comandi ASP vengono elaborati prima dell'invio del contenuto al browser. È possibile utilizzare i commenti HTML per aggiungere osservazioni a una pagina HTML. Tali commenti verranno restituiti al browser e potranno essere letti dall'utente visualizzando il codice HTML d'origine.

Commenti VBScript

In VBScript sono supportati i commenti introdotti dall'apostrofo. A differenza dei commenti HTML, questo tipo di commento viene rimosso durante l'elaborazione dello script e non è inviato al browser.

```
<%
  'Questa riga e le due seguenti sono commenti.
  'La funzione PrintTable stampa tutti
  'gli elementi di una matrice.
  PrintTable MyArray()
%>
```

Non è possibile inserire un commento in un'espressione di output. La prima riga riportata di seguito verrà ad esempio eseguita, mentre la seconda non produrrà alcun risultato perché inizia con <%=.

```
<% i = i +1 'Questa istruzione incrementa i. (Script corretto.) %>

<%= name 'Questa istruzione stampa il nome della variabile. (Script errato.) %>
```

Response.Write(x.toString())

Distinzione tra maiuscole e minuscole In VBScript non viene fatta distinzione tra maiuscole e minuscole. Per fare riferimento all'oggetto **Request** è ad esempio possibile specificare indifferentemente **Request** o **request**. Uno svantaggio della mancata distinzione tra maiuscole e minuscole è dato dal fatto che non si può utilizzare la formattazione maiuscola o minuscola per distinguere i nomi delle variabili. Non è ad esempio possibile creare due variabili distinte denominate Colore e colore.
In JScript *viene* invece fatta distinzione tra maiuscole e minuscole. Quando si utilizzano le parole chiave JScript negli script, è necessario digitare la parola chiave esattamente come indicato nella pagina di riferimento corrispondente. Se ad esempio si specifica **date** anziché **Date**, verrà generato un errore. La combinazione di maiuscole o minuscole indicata nella presente documentazione per gli oggetti ASP predefiniti è valida per i comandi JScript.

Utilizzo di variabili e costanti

Una *variabile* è un percorso di memorizzazione denominato presente nella memoria del computer e contenente dati, quali un numero o una stringa di testo. I dati contenuti in una variabile vengono denominati *valore* della variabile. Le variabili consentono di memorizzare, recuperare e modificare i valori tramite l'utilizzo di nomi che identificano la funzione svolta dallo script.

Dichiarazione e denominazione di variabili

Per la dichiarazione e la denominazione delle variabili, attenersi alle regole e alle linee guida del linguaggio di script in uso. Anche se non è necessario dichiarare una variabile prima di utilizzarla, è consigliabile eseguire abitualmente tale operazione per prevenire l'insorgenza di errori. *Dichiarare* una variabile significa indicare al modulo di script che una variabile con un nome particolare esiste, in modo che sia possibile inserire dei riferimenti alla variabile in qualsiasi punto di uno script.

VBScript

Sebbene VBScript non richieda la dichiarazione delle variabili, è consigliabile dichiarare tutte le variabili prima di utilizzarle. Per dichiarare una variabile in VBScript, utilizzare l'istruzione **Dim**, **Public** o **Private**. Ad esempio:

```
<% Dim UserName %>
```

Per rendere obbligatoria la dichiarazione esplicita delle variabili tramite le istruzioni **Dim**, **Private**, **Public** e **ReDim**, è possibile utilizzare l'istruzione **Option Explicit** di VBScript in un file asp. L'istruzione **Option Explicit** deve essere inserita dopo tutte le istruzioni ASP e prima di qualsiasi testo HTML o comando script. Questa istruzione ha effetto solo sui comandi ASP scritti in VBScript, non sui comandi JScript.

```
<% Option Explicit %>
<HTML>
<%
  Dim strUserName
  Public lngAccountNumber
%>
```

.

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**.

È inoltre possibile creare istanze di oggetti con ambito a livello di sessione o applicazione. Per ulteriori informazioni, vedere Impostazione dell'ambito degli oggetti.

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.

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
Gli script di questo tipo possono essere estesi, ad esempio, per configurare un database sul lato client o uno script di personalizzazione DHTML. Questa tecnica innovativa consente inoltre di ridurre le connessioni ripetute client/server e l'elaborazione a carico del server.

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>
```

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.

Se tuttavia la convalida dei moduli richiede l'accesso a database, può essere utile ricorrere alla convalida sul lato server. Un sistema particolarmente vantaggioso per l'esecuzione di operazioni di convalida sul lato server consiste nel creare un modulo che invii le informazioni a se stesso. In tal caso il file asp contiene il modulo HTML che recupera l'input dell'utente. Si ricorda che per interagire con script sul lato client e codice HTML è possibile utilizzare ASP. Per ulteriori informazioni, vedere Interazione con script sul lato client. L'input viene restituito allo stesso file, che convalida quindi le informazioni e provvede a informare l'utente in caso di input non valido.
Questo metodo di elaborazione e convalida dell'input dell'utente consente un notevole miglioramento della semplicità di utilizzo e dei tempi di risposta dei moduli basati sul Web. Inserendo ad esempio le informazioni sugli errori accanto al campo del modulo in cui sono state immesse le informazioni non valide, l'utente potrà identificare l'origine dell'errore con maggiore facilità. (In genere i moduli basati sul Web inoltrano le richieste a una pagina Web separata contenente informazioni sugli errori. Gli utenti che non sono in grado interpretare immediatamente queste informazioni potrebbero scoraggiarsi.)
Lo script seguente determina ad esempio se un utente ha immesso un numero di conto valido inviando le informazioni a se stesso (il file Verify.asp) e richiamando una funzione definita dall'utente che esegue una query in un database:

```
<%
  strAcct = Request.Form("Account")
  If Not AccountValid(strAcct) Then
    ErrMsg = "<FONT COLOR=Red>Il numero di conto specificato non è
valido.</FONT>"
  Else
    Elabora l'input dell'utente
    .
    Server.Transfer("Complete.asp")
  End If

  Function AccountValid(strAcct)
    Posizione di una chiamata a uno script di connettività al database oppure
a un metodo del componente.
  End Function
%>

<FORM METHOD="Post"  ACTION="Verify.asp">
Account Number:  <INPUT TYPE="Text" NAME="Account"> <%= ErrMsg %> <BR>
<INPUT TYPE="Submit">
</FORM>
```

In questo esempio lo script è posizionato in un file denominato Verify.asp, lo stesso file contenente il modulo HTML. Lo script invia le informazioni a se stesso specificando Verify.asp nell'attributo ACTION.
Importante Se si utilizza JScript per la convalida sul lato server, assicurarsi di inserire una coppia di parentesi vuote dopo l'elemento dell'insieme **Request** (**QueryString** o **Form**) durante l'assegnazione dell'insieme a una variabile locale. Se non si specificano le parentesi, l'insieme restituirà un oggetto anziché una stringa.

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.

Proprietà

Buffer	Indica se l'output della pagina viene memorizzato nel buffer.
CacheControl	Crea un'intestazione che determina se i server proxy o agli altri sistemi di memorizzazione nella cache sono in grado di inserire nella cache l'output generato da ASP.
Charset	Aggiunge il nome del set di caratteri all'intestazione del tipo di contenuto. Il set di caratteri segnala al browser in quale modo devono essere visualizzati i caratteri.
CodePage	Imposta la tabella codici per i dati degli oggetti intrinseci per una risposta. La tabella codici segnala al server in quale modo devono essere codificati i caratteri di lingue diverse.
ContentType	Specifica il tipo di contenuto HTTP per la risposta.
Expires	Specifica il periodo di tempo che deve intercorrere prima della scadenza di una pagina memorizzata nella cache di un browser.
ExpiresAbsolute	Specifica la data e l'ora di scadenza di una pagina memorizzata nella cache di un browser.
IsClientConnected	Indica se il client è stato disconnesso dal server.
LCID	Imposta l'identificatore LCID dei dati per una risposta. L'identificatore definisce il formato di data, ora e valuta per una posizione geografica specifica.
Pics	Imposta il valore dell'intestazione della risposta dell'etichetta PICS, che indica le restrizioni PICS del contenuto.
Status	Il valore della riga di stato restituito dal server.

Metodi

AddHeader	Imposta il nome *(name)* dell'intestazione HTML su *value*.
AppendToLog	Aggiunge una stringa alla fine della voce del file registro del server Web relativa alla richiesta corrente.
BinaryWrite	Scrive le informazioni specificate nell'output HTTP corrente senza convertire i set di caratteri.
Clear	Cancella l'output HTML memorizzato nel buffer.
End	Interrompe l'elaborazione del file asp e restituisce il risultato corrente.
Flush	Invia immediatamente l'output memorizzato nel buffer.
Redirect	Invia un messaggio di reindirizzamento al browser che proverà a connettersi a un diverso URL.
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

ClientCertificate	I valori dei campi memorizzati nel certificato client inviato nella richiesta HTTP.
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.

Proprietà

TotalBytes	Sola lettura. Specifica il numero totale di byte inviati dal client nel corpo della richiesta.

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.

È possibile accedere direttamente a tutte le variabili richiamando l'oggetto **Request(*variable*)** senza indicare il nome dell'insieme. In questo caso, il server Web eseguirà la ricerca negli insiemi nel seguente ordine.

1. **QueryString**
2. **Form**
3. **Cookies**
4. **ClientCertificate**
5. **ServerVariables**

Se in più insiemi esiste una variabile con lo stesso nome, l'oggetto **Request** restituisce la prima istanza rilevata dall'oggetto.
Per fare riferimento ai membri di un insieme è consigliabile utilizzarne il nome completo. Ad esempio, utilizzare **Request.ServerVariables**(AUTH_USER) anzché **Request**.(AUTH_USER). In questo modo il server sarà in grado di individuare l'elemento in modo più rapido.

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**.
L'oggetto **Session** viene in genere utilizzato per memorizzare le preferenze dell'utente. Ad esempio, se un utente indica che preferisce non visualizzare la grafica, è possibile memorizzare tale informazione nell'oggetto **Session**. Per ulteriori informazioni sull'utilizzo dell'oggetto **Session**, vedere Gestione delle sessioni nella sezione Applicazioni ASP.
Nota Lo stato della sessione viene gestito solo per i browser che supportano i cookie.
Sintassi
Session.*collection*|*property*|*method*

Insiemi

Contents	Contiene gli elementi aggiunti alla sessione con i comandi di script.
StaticObjects	Contiene l'oggetto creato con il tag <OBJECT> e a cui è stata assegnato l'ambito a livello di sessione.

Proprietà

CodePage	Imposta la tabella codici per i dati degli oggetti intrinseci per un'intera sessione. La tabella codici segnala al server in quale modo devono essere codificati i caratteri di lingue diverse.
LCID	Imposta l'identificatore LCID dei dati per un'intera sessione. L'identificatore definisce il formato di data, ora e valuta per una posizione geografica specifica.
SessionID	Restituisce l'identificazione della sessione per l'utente corrente.
Timeout	Il periodo di timeout, espresso in minuti, per lo stato della sessione dell'applicazione corrente.

Metodi

Abandon	Elimina l'oggetto **Session** e rilascia le relative risorse.
Contents.Remove	Elimina un elemento dall'insieme **Contents**.
Contents.RemoveAll	Elimina tutti gli elementi dall'insieme **Contents**.

Eventi
Gli script relativi ai seguenti eventi sono dichiarati nel file Global.asa.
Session_OnEnd
Session_OnStart

Per ulteriori informazioni sugli eventi indicati e sul file Global.asa, vedere Informazioni di riferimento sul file Global.asa.
Osservazioni
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
%>
```

Tuttavia, se si memorizza un oggetto nell'oggetto **Session** e si utilizza VBScript come linguaggio di script primario, è necessario utilizzare la parola chiave **Set**, come illustrato nell'esempio seguente.

```
<% Set Session("Obj1") = Server.CreateObject("MyComponent.class1") %>
```

A questo punto è possibile richiamare i metodi e le proprietà esposte da `MyComponent.class1` nelle successive pagine Web, utilizzando la seguente sintassi:

```
<% Session("Obj1").MyMethod %>
```

oppure estraendo una copia locale dell'oggetto e utilizzando la seguente sintassi:

```
<%
Set MyLocalObj1 = Session("Obj1")
MyLocalObj1.MyObjMethod
%>
```

Per creare oggetti con ambito a livello di sessione è inoltre possibile utilizzare i tag <OBJECT> nel file Global.asa.
Non è tuttavia possibile memorizzare un oggetto predefinito in un oggetto **Session**. Ad esempio, ciascuna delle righe di codice seguenti restituisce un errore.

```
<%
Set Session("var1") = Session
Set Session("var2") = Request
Set Session("var3") = Response
Set Session("var4") = Server
Set Session("var5") = Application
%>
```

Prima di memorizzare un oggetto nell'oggetto **Session**, è necessario conoscere il modello di threading che utilizza. Solo gli oggetti impostati per l'utilizzo del modello di threading Both possono essere memorizzati nell'oggetto **Session** senza bloccare la sessione a un solo thread. Per ulteriori informazioni, vedere Platform SDK.
Se si memorizza una matrice in un oggetto **Session**, non è possibile modificare direttamente gli elementi della matrice memorizzata. Ad esempio, lo script seguente non funziona:

```
<% Session("StoredArray")(3) = "new value" %>
```

Questo succede perché l'oggetto **Session** viene implementato come insieme. L'elemento della matrice `StoredArray(3)` non riceverà il nuovo valore. Il valore viene invece indicizzato nell'insieme sovrascrivendo qualsiasi informazione memorizzata in tale posizione.
Se si memorizza una matrice nell'oggetto **Session**, è consigliabile recuperarne una copia prima di recuperare o modificare qualsiasi elemento al suo interno. Dopo aver modificato la matrice, è necessario memorizzarla nuovamente nell'oggetto **Session** in modo che le modifiche vengano salvate, come illustrato nell'esempio seguente:

```
---file1.asp---
<%
'Creating and initializing the array
Dim MyArray()
Redim MyArray(5)
MyArray(0) = "hello"
MyArray(1) = "some other string"

'Storing the array in the Session object.
Session("StoredArray") = MyArray

Response.Redirect("file2.asp")
%>

---file2.asp---
<%
'Retrieving the array from the Session Object
'and modifying its second element.
LocalArray = Session("StoredArray")
LocalArray(1) = " there"

'Printing out the string "hello there."
Response.Write(LocalArray(0)&LocalArray(1))

'Re-storing the array in the Session object.
'This overwrites the values in StoredArray with the new values.
Session("StoredArray") = LocalArray
%>
```

Esempio

Il codice seguente assegna la stringa `MyName` alla variabile di sessione denominata `name`, assegna un valore alla variabile di sessione `year` e assegna un'istanza del componente `someObj` alla variabile `myObj`.

```
Session("name") = "MyName"
Session("year") = 96
Set Session("myObj") = Server.CreateObject("someObj")
 %>
```

Per ulteriori informazioni, vedere Gestione delle sessioni.

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.
GetLastError	Restituisce un oggetto **ASPError** che descrive la condizione di errore.
HTMLEncode	Applica la codifica HTML alla stringa specificata.
MapPath	Esegue il mapping del percorso virtuale specificato (che può essere il percorso assoluto nel server corrente o il percorso relativo alla pagina corrente) con un percorso fisico.
Transfer	Invia tutte le informazioni correnti sullo stato a un altro file asp per l'elaborazione.
URLEncode	Applica alla stringa le regole di codifica degli URL, compresi i caratteri di escape.